ISSN0869-0510

радио любитель

4
1996

Ваш компьютер

ВК

Учредитель: НТК "Инфотех"

ЧИТАЙТЕ В НОМЕРЕ:

радиолюбитель

Приложение: Ваш компьютер
Издается с сентября 1995 г.

Главный редактор
Валентин БЕНЗАРЬ

Зам. гл. редактора
Иван БЕЛЬСКИЙ

Выпускающий редактор
Елена ЛЕВИТМАН

Редакторы разделов:
Виктор ЕРМОЛЕНКО,
Анатолий КОРБИТ

Янина БЕЛЬСКАЯ — компьютерная верстка
Оксана НАЙДОВИЧ,
Ольга КРИВЕЛЬ — компьютерный набор

Оформление обложки —
Надежда БОГОМОЛОВА,
Наталья БЕЛЬСКАЯ
Татьяна БЕЛЬСКАЯ — техническая графика

Отдел экспедирования и рассылки журналов — **Наталья ПАСЫНКОВА**, тел. (0172) 22-14-34.
Адрес для писем: 220050, г.Минск-50, а/я 41.
E-mail: *rl@rl.belpak.minsk.by*
Адрес редакции: 220099, Минск, ул. Авакяна, 30-1-2.
Тел./Факс: (0172) 22-14-34.

Расчетный счет 3012202650014 в Октябрьском РКЦ Ленинского отделения Белбизнесбанка в г.Минске МФО 153001763 код 763, для НТК "Инфотех". Корр. счет 700161963 в Главном управлении Национального банка по г.Минску и Минской обл. (адрес банка: 220099, Беларусь, Минск, ул. Казинца, 21, корп. 3).

Журнал зарегистрирован Министерством информации Республики Беларусь 13.11.95г. (рег. удост. N1240).

Подписано к печати 15.3.96. Формат 60 x 84 1/8. Печать офсетная. 4,5 печ. л. Зак. 004.

Отпечатано с оригинал-макета, изготовленного редакцией журнала, в МУ НТК "Инфотех".

М.РЕВОТЮК, Б.КИСЕЛЕВ,
г.Минск, БГУИР, ФИТиУ,
каф."Информационные технологии
автоматизированных систем".

ПРОЦЕДУРЫ ПОИСКА ДАННЫХ

Задача поиска, наряду с задачей сортировки, наиболее часто встречается при обработке данных [1,2]. Для введения основных понятий рассмотрим пример. Список всех книг библиотеки помещен в файл, который представляет собой множество (массив) записей. Каждая запись соответствует книге и содержит поля с информацией об авторах, названии книги, издательстве, годе издания, месте хранения и т.п. Возможные обращения к списку книг:

а) Найти книгу Иванова С.А.;

б) Отобрать книги прошлого века;

в) Отобрать книги Петрова, изданные в 1973 году и т.д.

Поиск данных — процесс выделения из некоторого множества записей определенного подмножества, элементы которого удовлетворяют заранее заданному условию поиска. В нашем примере множество — записи всего файла; подмножество — первая встретившаяся книга Иванова С.А. (случай "а"), книги прошлого века (случай "б") и т.д.; условие поиска — совпадение при просмотре списка авторов фамилии Иванов С.А. с такой же в списке (случай "а"), попадание года издания книги в диапазон 1800-1899 (случай "б"), совпадение фамилии автора и года издания (случай "в"). Любую из разновидностей задач поиска (например поиск по совпадению ("а"), интервалу ("б"), сложному условию ("в") и т.п.) можно свести к задаче поиска по совпадению некоторого эталонного значения (ключа поиска) со значением ключа записи. В примере ключами поиска являются: Иванов С.А., 1800 и 1899, Петров и 1973, а ключами записи — соответственно фамилия автора, год издания, фамилия автора и год издания.

Эффективность методов поиска оценивается числом сравнений пар признаков, проведенных до момента окончания поиска. При сравнении различных методов поиска моментом его окончания принято считать момент обнаружения первой требуемой записи.

В общем случае эффективность поиска зависит от организации записей в массиве и метода доступа к отдельным записям. Для последовательно организованных массивов наилучшие условия поиска достигаются в случае упорядоченности записей по возрастанию либо убыванию ключевого признака.

Представляемые далее примеры процедур поиска написаны на языке С, их параметрами является указатель массива целых чисел a, количество его элементов n и искомое (эталонное) значение b.

Возвращаемое значение — индекс элемента массива x, совпадающего с искомым значением b, а в случае неудачного поиска — значение -1.

```
/*
 *  Пример процедуры последовательного поиска по
совпадению в неупорядоченном линейном массиве
 */
   int psearch(int *a, int n, int b) {
    int i;
    for (i=0; i<n; i++)
     if (a[i]==b) return(i);
     return(-1);
    }
```

Наиболее эффективным методом поиска произвольных данных в последовательно организованных упорядоченных массивах является дихотомический метод (метод двоичного поиска или метод деления пополам) [2].

Рассмотрим идею алгоритма поиска методом дихотомии в упорядоченном по возрастанию массиве.

Шаг 1. Выделяется текущий интервал записей, который включает искомую запись (сначала — весь массив).

Шаг 2. Ключ средней записи текущего интервала сравнивается с ключом поиска, чтобы определить где расположена искомая запись — до центра, в центре либо после центра интервала.

Шаг 3. Если искомая запись обнаружена в центре текущего интервала, поиск завершен. В противном случае в качестве нового интервала рассматривается та часть текущего интервала, в которую попал ключ поиска, затем выполняется возврат к шагу 2.

Процедура дихотомического поиска должна включать обработку ситуаций, когда искомых данных нет.

```
/*
 *  Пример процедуры дихотомического поиска по
совпадению в упорядоченном линейном массиве
 */
   int dsearch(int *a, int n, int b) {
      int i,j,k;
      if (n<1) return(-1);
      j=0, k=n-1;
      while (j<k) {
       i=(j+k)/2;
       if (a[i]<b) j=i+1;
       else k=i;
      }
      return((a[j]!=b)? -1:j);
     }
```

Вычислительная трудоемкость процедур дихотомического поиска — $n*\log_2(n)$ сравнений.

Рассмотрим другой метод поиска в упорядоченном массиве — последовательный перебор с некоторым шагом.

Вначале здесь выбирается некоторый шаг просмотра m, задающий величину интервала записей. Далее последовательно анализируются ключи записей с номерами 1, m+1, 2m+1,... n. При обнаружении интервала с требуемой записью организуется последовательный поиск внутри этого интервала.

Оптимальная величина шага просмотра n^(1/2) [2].

Вычислительная трудоемкость процедуры последовательного поиска с равномерным оптимальным шагом — n^(1/2) сравнений.

Наиболее эффективный поиск реализуется на основе адресной функции вида i=f(p), где i — номер или адрес записи, p — значение ключа поиска. Вид функции f зависит от конкретной задачи.

Например пусть ключи записей принимают значения в ин-

тервале [a,b]. Очевидно, что количество записей здесь не превышает значения (b-a+1).

Предполагая, что для хранения записей используется массив x[0...b-a], можно предложить для адресации записей зависимость i=(p-a).

Использование адресной функции предоставляет возможность доступа к требуемой записи за один шаг, однако на практике не всегда удается найти подходящий вид зависимости f(p).

Случай, когда адресная функция i=p, соответствует понятию инвертированного массива. Значения инвертированного массива определяются как y[x[i]]=i, т.е. значение элемента массива x определяет индекс массива y и наоборот. Размерность инвертированного массива обычно больше размерности исходного, поэтому "лишним" элементам присваивают какое-либо значение (для удобства использования признака неудачи поиска). Рассмотрим пример построения и использования инвертированного массива.

```
#include <stdio.h>
#include <stdlib.h>
/* Пример исходного массива целых чисел */
     int x[]={3,5,1,7,9};
/* Размерность исходного массива */
     int n=sizeof(x)/sizeof(*x);
/* Указатель инвертированного массива */
     int *y;
/* Размерность инвертированного массива */
     int m;
/*
 *  Создание инвертированного массива
 */
   int *prepar(int *x, int n) {
     int i, *y;
/* Оценка размерности инвертированного массива */
     for (m=i=0; i<n; i++) if (x[i]>m) m=x[i];
 /* Захват памяти для размещения инвертированного
массива */
     if ((y=malloc(++m*sizeof(*y)))!=NULL) {
/* Инициализация инвертированного массива */
      for (i=0; i<m; i++) y[i]=-1;
/* Окончательное формирование инвертированного
массива */
      for (i=0; i<n; i++) y[x[i]]=i;
     }
     return y;
    }
/*  Процедура поиска в инвертированном массиве */
    int invsrc(int *y, int m, int b) {
     return((b<m)? y[b]: -1 );
    }
 void main(){
   int i,b;
    y=prepar(x, n);
    if (y!=NULL) {
     while (printf("\nЭлемент - ? "),
scanf("%d",&b)>0) {
      printf("\nЭлемент %d",b);
     i=invsrc(y, m, b);
      printf(" имеет индекс %d",i);
      if(i<0) printf(" элемент не найден");
     }
    } else printf("\n Нет памяти...");
   }
```

Рассмотренные здесь процедуры поиска не учитывают частоту выборки конкретных данных. Очевидно, что возможно размещение данных с учетом частоты их использования, при котором среднее время поиска будет минимальным. Например данные можно разместить так, чтобы сначала поиск проводился среди наиболее часто встречающихся элементов [1].

Стандартные процедуры сортировки и поиска данных

Рассмотрим набор библиотечных процедур поиска и сортировки данных в системе программирования Turbo-C. Представляемые процедуры предназначены для обработки массивов последовательно размещенных в памяти одинаковых по размеру элементов данных. Внутреннее представление данных может быть произвольным, но программист должен определить понятие ключа сортировки либо поиска посредством задания процедуры-функции проверки порядка следования любой пары элементов массива.

Обозначения аргументов библиотечных процедур-функций сортировки и поиска данных:

base — указатель последовательного массива данных в памяти;

nelem — количество элементов данных;

width — длина элемента данных в байтах;

fcmp — указатель функции сравнения элементов данных;

x,y — указатели аргументов функции fcmp;

key — эталонный элемент процедур поиска данных;

pnelem — указатель поля, содержащего количество элементов данных.

Декларация функций поиска и сортировки данных:

```
/* Процедура-функция сортировки линейного массива по
возрастанию значений ключевых элементов */
     void qsort(void *base, int nelem, int width,
                int (*fcmp)(void *x, void *y));
/* Процедура-функция дихотомического поиска в
линейном массиве, упорядоченном по возрастанию
ключевых элементов */
     void *bsearch(void *key, void *base, int nelen,
        int width, int (*fcmp)(void *x, void *y));
/* Процедура-функция последовательного поиска в
линейном неупорядоченном массиве. Если элемент не
найден, он добавляется в массив */
     void *lsearch(void *key, void *base, int
*pnelem,                int width, int (*fcmp)(void
*x, void *y));
/* Процедура-функция последовательного поиска в
линейном неупорядоченном массиве */
     void *lfind(void *key, void *base, int *pnelem,
        int width, int (*fcmp)(void *x, void *y));
```

Особенности использования функций поиска и сортировки:

- процедура-функция **fcmp* должна возвращать целочисленное значение K(*x)-K(*y), где K(z) — значение ключа элемента данных z (абсолютная величина получаемой разности принципиального значения не имеет, учитывается лишь ее знак для установления относительного порядка следования элементов данных *x и *y при условии упорядочения массива по возрастанию значений ключей — знак '-' означает, что элемент *x предшествует элементу *y, знак '+' — элемент *x следует за элементом *y, а нуль означает совпадение ключей элементов *x и *y);

- процедуры поиска *bsearch* и *lfind* возвращают указатель на первый найденный элемент данных или 0 (пустой указатель NULL) в случае неудачи;

- процедура поиска *lsearch* возвращает указатель на первый найденный элемент, а в случае неудачи — на место размещения добавленного в конец массива элемента;

- поиск второго и последующих элементов данных с совпадающими значениями ключа возможен посредством ре-

куррентного обращения к функции поиска с коррекцией на каждом шаге значений указателя массива и количества оставшихся элементов.

```
 /* Примеры использования процедур поиска и
сортировки */
#include <stdio.h>
#include <stdlib.h>
/* Тестовый массив целых чисел */
    int x[]={0,8,3,2,1,9,4,6,7};
/* Размерность тестового массива */
    int nx=sizeof(x)/sizeof(*x);
/*
 *  Функция сравнения элементов
 */
    int compare(int *x,int *y) {
     return(*x-*y);
    }
 /* Функция тестирования принадлежности элементов
множеств */
    main () {
     int i,*d;
/* Упорядочение элементов множества */
     qsort(x,nx,sizeof(*x),compare);
/* Ввод и тестирование принадлежности элементов */
     while (printf("\nЭлемент - ? "),
        scanf("%d",&i)>0) {
      d=bsearch(&i,x,nx,sizeof(*x),compare);
      printf("\nЭлемент %d",i);
      if(d!=NULL) printf(" имеет индекс %d",d-x);
      else printf(" не найден");
     }
    }
```

Вариант построения процедуры тестирования на основе инвертированного массива:

```
#include <stdio.h>
#include <stdlib.h>
/* Тестовый массив целых чисел */
   int x[]={0,8,3,2,1,9,4,6,7};
   int n; /* Размерность тестового массива */
   void main () {
    int i;
    int *d; /* Указатель инвертированного массива */
    int m; /* Размерность инвертированного массива */
    n=sizeof(x)/sizeof(*x);
/*
 *  Создание инвертированного массива
 */
/* Оценка размерности инвертированного массива */
     for (i=m=0; i<n; i++) if (x[i]>m) m=x[i];
/* Захват памяти для размещения инвертированного
массива */
     if((d=malloc(++m*sizeof(*d)))==NULL) return;
/* Инициализация инвертированного массива */
     for (i=0; i<m; i++) d[i]=n;
     for (i=0; i<n; i++) d[x[i]]=i;
/*  Ввод и тестирование принадлежности элементов */
     while (printf("\nЭлемент - ? "),
        scanf("%d",&i)>0) {
      printf("\nЭлемент %d",i);
      if((i<m)&&(d[i]<n)) printf(" имеет индекс
                %d",d[i]);
      else printf(" не найден");
     }
     free(d);
    }
```

Литература:

1. Кнут Д. Искусство программирования для ЭВМ. Т.3: Сортировка и поиск /Пер. с англ. —М.: Мир, 1978. — 844 с.

2. Вирт Н. Алгоритмы и структуры данных: Пер. с англ. —М.: Мир, 1989. — 360 с.

А.ИВАНЧИКОВ,
БГУИР. ФИТиУ. каф. "Информационные технологии автоматизированных систем",
г.Минск, тел. 39-88-23.

ВНИМАНИЕ, СТРУКТУРЫ!

Структуры являются встроенным типом данных в СИ, поэтому язык поддерживает для работы с объектами данного типа некоторый, хотя и очень ограниченный, набор операций. Операция присваивания для структур появилась в синтаксисе языка СИ не сразу, но ее появление внесло в язык дополнительные удобство и гибкость. Однако пользоваться операцией присваивания для структур стоит с опаской.

Операция присваивания для структур поэлементно копирует одну в другую, и если члены структур являются указателями, при таком копировании возникают проблемы, связанные с тем, что фактически указатель состоит из двух элементов — переменной-указателя (в ней хранится адрес используемой памяти) и непосредственно самой этой памяти, а при копировании переносится лишь адрес вектора. В результате члены двух структур будут ссылаться на одну и ту же память.

Выход из положения состоит в использовании вместо операций присваивания для подобных структур (имеющих члены-указатели) специальных функций. Ниже показано определение и использование подобной функции.

```
#include <string.h>
#include <stdlib.h>

typedef struct InformBook
 {
  char *Name;
  char *Autor;
  float Price;
 } book;

 /* Функция возвращает правильную копию структуры _s
*/
book copybook(book _s)
{
  book tmp;

  tmp.Name = malloc(strlen(_s.Name)+1);
  strcpy(tmp.Name,_s.Name);
  tmp.Autor = malloc(strlen(_s.Autor)+1);
  strcpy(tmp.Autor,_s.Autor);
  tmp.Price = _s.Price;

  return tmp;
}

void main(void)
{
  book s1, s2;
  . . .
  s2 = s1;                    /* неверно   */
  s2 = copybook(s1);          /* правильно */
  . . .
}
```

С.ВИЦАН,
194358, г.С.-Петербург, ул.Композиторов, 33/5 — 553.

РУСИФИКАЦИЯ ИМПОРТНЫХ ПРИНТЕРОВ

Если вы пользуетесь отечественным принтером, проблемы вывода на печать русского текста у вас не возникает. В последнее время в продажу поступают и русифицированные принтеры импортного производства. Но как быть, если ваш импортный принтер не русифицирован?

Есть несколько способов решения этой проблемы. Один из них — вывод русского текста в графическом режиме. Этот способ используется в некоторых текстовых редакторах, но для "Спектрума" не годится: 32 знака в строке, форма букв шрифта оставляет желать лучшего, скорость печати также невелика.

Другой способ — найти (у друзей, на работе или еще где-нибудь) такой же как у вас, но русифицированный принтер, скопировать с помощью программатора его ПЗУ и установить вместо имеющегося. Но этот способ еще менее приемлем, так как далеко не каждый владелец дорогостоящего устройства позволяет забраться в него.

Есть еще третий способ, который мы и рассмотрим. Если ваш принтер позволяет загружать внешние шрифты, можно создать свой шрифт на "Спектруме" и загружать его в ОЗУ принтера по мере необходимости. Этот способ также имеет ряд недостатков, но, на мой взгляд, является простым и удобным.

Мы рассмотрим вариант русификации EPSON-совместимого принтера, который подключен к "Спектруму" через интерфейс "ZX Lprint-III". Если у вас принтер с другой системой команд или другой тип интерфейса, сначала уточните формат команд для вашего принтера и интерфейса и измените предлагаемую программу в соответствии с ними.

1.Конструирование символов

Рассмотрим конструкцию символов принтера в режиме "черновой печати". Каждый знак, печатаемый принтером, состоит из 11 столбцов по 9 точек в каждом. 9-я (верхняя или нижняя) иголка печатающей головки как правило не используется. Какая конкретно — указывается в специальной команде (см. ниже).

Каждая точка, начиная с нижней, имеет свой код: 1, 2, 4, 8, 16, 32, 64, 128. Вы рисуете на бумаге форму буквы, которая вам нравится, а затем суммируете коды соответствующих точек в каждом столбце, начиная с левого. В приведенном на рис.1 примере буквы "Ж" в первом столбце точки имеют коды 2 и 128, таким образом, n1=130. Во втором столбце — 4,8,32,64. Сложив эти числа, получаем n2=108. В третьем — n3=16. В четвертом — точек нет, поэтому n4=0, и так далее. В результате вы получаете 11 десятичных чисел, с помощью которых кодируется буква "Ж": 130, 108, 16, 0, 254, 0, 16, 108, 130, 0, 0.

Аналогично кодируются все остальные символы вашего будущего шрифта. Подобным образом можно закодировать символы и для режима "высококачественной печати", но для описания каждого символа вам понадобится не 11, а 36...46 чисел. Кроме того, некоторые типы принтеров позволяют загружать шрифты только в режиме "черновой печати". Те же, которые позволяют загружать шрифты в режиме "высококачественной печати", могут иметь ограничения на количество символов.

2.Таблица символов

В настоящее время их существует несколько: ASCII, КОИ-7, КОИ-8. Мы рассмотрим первую. В этих таблицах каждому символу присваивается определенный код, который компьютер выдает на принтер. Принтер в соответствии с этим кодом печатает соответствующий этому коду символ. Коды с 0 по 31 являются управляющими, с 32 по 127 — стандартными для данной таблицы символов, а с 128 по 255 отводятся под нужды пользователя. Вам нужно решить, какому символу стандартной кодовой таблицы вы присвоите тот или иной символ вашего шрифта.

В принципе, вы можете составить свою кодовую таблицу и расположить в ней символы как вам удобно. Нужно помнить только две особенности:

- во-первых, символы с 0 по 31 трогать нельзя, так как они являются управляющими;
- во-вторых, некоторые типы принтеров позволяют загружать внешний шрифт только в ту часть таблицы символов, которая находится между 32 и 127. Именно это создает основную трудность в одновременном использовании русского и латинского шрифтов при печати.

С помощью простой программы вы можете вывести на печать символы кодовой таблицы и соответствующие им десятичные номера (см. программу 1).

Программа 1:

```
10 REM Program "Print CHR$"
20 FOR k=32 TO 79
30 LPRINT k;" = ";CHR$ (k);"     ";k+48;" = ";CHR$
(k+48)
40 NEXT k
```

3.Программа ввода символов "EPSONrus"

Итак, вы подготовили все коды вашего будущего шрифта и решили как они будут располагаться в кодовой таблице. Теперь осталось ввести их в ОЗУ принтера. Как это сделать? Если вы посмотрите инструкцию к принтеру, то в разделе "Контрольные коды" можете найти примерно такую надпись:

<ESC> & 0 c1 c2 a n1...n11

Что обозначает эта запись? Первые три символа — это, собственно, команда загрузки внешнего символа в ОЗУ принтера; c1 и c2 указывают номер символа в таблице, ко-

Рис. 1

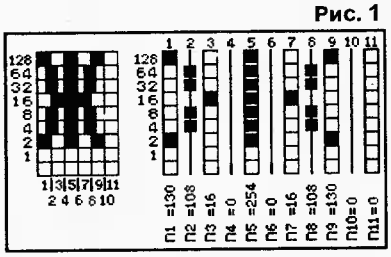


Рис. 2

1	2	3	4	5	6	7	8	9	0
	Ю								Ъ
Q	W	E	R	T	Y	U	I	O	P
Я	В	Е	Р	Т	Ы Ш	У Щ	И	О	П
A	S	D	F	G	H	J	K	L	EN
Ач	Сэ	Д Э	Ф Ш	Г Щ	Х Ч	Й	К	Л	
CS	Z	X	C	V	B	N	M	SS	SP
	З	Ь Ю	Ц	Ж	Б	Н	М		

торый будет заменен новым; а — указывает принтеру, какая из 9 иголок (верхняя или нижняя) не используется в данном символе; n1...n11 — это те одиннадцать чисел, которые вы определили.

Теперь осталось записать эту строку в формате, понятном для вашего интерфейса и принтера. В рассматриваемом примере в начале каждой строки добавлена команда для интерфейса, которая выключает знаки "Спектрума" и все байты, следующие за ней, интерпретируются как ASCII коды (LPRINT CHR$ 5).

Предлагаемая вашему вниманию программа формирует маленькие и большие русские буквы в стандарте "ЯВЕРТЫ" вместо букв латинского алфавита. Цифры и основные знаки препинания остаются на своих местах. Вместо больших латинских букв формируются маленькие русские и наоборот. Расположение русских символов на клавиатуре при этом полностью соответствует одному из вариантов русификации текстового редактора "The Last Word" [2] и приведено на рис.2. Символы, выделенные жирным шрифтом, набираются с одновременным нажатием клавиши "Symbol Shift".

Вкратце о самой программе.

Строка 10 — переводит принтер в режим "черновой печати" (т.н. "DRAFT").

Строка 20 — команда копирования стандартных символов из ПЗУ принтера в его ОЗУ. Это делается для того, чтобы не создавать заново те символы, которые мы не меняем (например цифры).

Строки 30...650 — это, собственно, данные загрузки вашего шрифта.

Строка 690 — переключает принтер на загруженный шрифт.

Строка 700 — возвращает интерфейс в режим выдачи текстовой информации.

Рассмотрим подробнее пересылку данных из строк 30...650.

CHR$(5) — переводит интерфейс в режим выдачи кодов управления;

CHR$(27); CHR$(38); CHR$(0) — команда загрузки внешнего символа в ОЗУ принтера;

CHR$(123); CHR$(123) — указывает номер символа, который будет заменен;

CHR$(11) или CHR$(139) — указывает принтеру, какая из 9-ти иголок (верхняя или нижняя соответственно) не используется в данном символе;

CHR$(254);... CHR$(0) — остальные 11 параметров содержат описание нового символа.

4.Работа с программой

Несмотря на то, что программа получается довольно большой, работает она довольно быстро. Итак, инициализируем принтер командой LPRINT CHR$ 13. Загружаем программу с внешнего носителя и даем RUN. При выдаче программы на принтер идет холостой прогон примерно одного листа бумаги формата А4. В результате в памяти принтера будет загружен ваш новый шрифт. Компьютер после этого можно сбросить и загрузить, например, текстовый редактор. Теперь ваш принтер будет печатать по-русски, если ввести следующую команду:

```
LPRINT CHR$ 1;"4";CHR$(27);CHR(37);CHR$(1);CHR$(0)
```

или по-английски, если ввести другую команду:

```
LPRINT CHR$ 1;"4";CHR$(27);CHR$(37);CHR$(0);CHR$(0).
```

Для работы с текстовым редактором необходимо ввести эти коды в РСТ-таблицу, а значение префикса изменить на 5.

```
1 REM Program "EPSONrus"
10 LPRINT CHR$(5); CHR$(27); CHR$(120); CHR$(0)
20 LPRINT CHR$(5): CHR$(27); CHR$(58); CHR$(0); CHR$(0); CHR$(0)
30 DATA 102, 139, 120, 132, 0, 132, 122, 132, 0, 132, 120, 0, 0
40 DATA 105, 139, 254, 0, 4, 8, 16, 32, 64, 0, 254, 0, 0
50 DATA 119, 139, 130, 124, 130, 16, 130, 16, 130, 16, 108, 0, 0
60 DATA 117, 139, 130, 64, 34, 20, 8, 16, 32, 64, 128, 0, 0
70 DATA 97, 139, 30, 32, 72, 128, 8, 128, 72, 32, 30, 0, 0
80 DATA 112, 139, 254, 0, 128, 0, 128, 0, 128, 0, 254, 0, 0
90 DATA 114, 139, 254, 0, 144, 0, 144, 0, 144, 0, 96, 0, 0
100 DATA 123, 139, 254, 0, 2, 0, 254, 0, 2, 0, 254, 0, 0
110 DATA 111, 139, 124, 130, 0, 130, 0, 130, 0, 130, 124, 0, 0
120 DATA 108, 139, 2, 4, 136, 112, 128, 0, 128, 0, 254, 0, 0
130 DATA 100, 139, 3, 0, 134, 120, 130, 0, 130, 0, 254, 0, 3
140 DATA 120, 139, 0, 254, 0, 18, 0, 18, 0, 18, 0, 12, 0
150 DATA 116, 139, 128, 0, 128, 0, 254, 0, 128, 0, 128, 0, 0
160 DATA 125, 139, 254, 0, 2, 0, 254, 0, 2, 0, 254, 0, 3
170 DATA 122, 139, 68, 130, 0, 130, 0, 146, 0, 146, 108, 0, 0
180 DATA 106, 139, 254, 0, 4, 72, 144, 32, 64, 0, 254, 0, 0
190 DATA 107, 139, 254, 0, 16, 0, 16, 0, 40, 68, 130, 0, 0
200 DATA 121, 139, 254, 0, 18, 0, 18, 0, 18, 12, 0, 254, 0
210 DATA 101, 139, 254, 0, 146, 0, 146, 0, 146, 0, 130, 0, 0
220 DATA 103, 139, 254, 0, 128, 0, 128, 0, 128, 0, 192, 0, 0
230 DATA 109, 139, 254, 0, 64, 32, 16, 32, 64, 0, 254, 0, 0
240 DATA 99, 139, 254, 0, 2, 0, 2, 0, 2, 0, 254, 0, 3
250 DATA 126, 139, 224, 16, 0, 16, 0, 16, 0, 254, 0, 0, 0
260 DATA 110, 139, 254, 0, 16, 0, 16, 0, 16, 0, 254, 0, 0
270 DATA 113, 139, 98, 4, 152, 0, 144, 0, 144, 0, 254, 0, 0
280 DATA 70, 11, 56, 0, 68, 0, 254, 0, 68, 0, 56, 0, 0
290 DATA 73, 11, 124, 0, 8, 0, 16, 0, 32, 0, 124, 0, 0
300 DATA 87, 11, 124, 0, 84, 0, 84, 0, 84, 40, 0, 0, 0
310 DATA 85, 11, 65, 32, 17, 10, 4, 8, 16, 32, 64, 0, 0
320 DATA 65, 11, 8, 20, 64, 20, 64, 20, 64, 56, 4, 0, 0
330 DATA 80, 11, 64, 60, 64, 0, 64, 0, 64, 0, 124, 0, 0
340 DATA 82, 11, 127, 0, 68, 0, 68, 0, 68, 56, 0, 0, 0
350 DATA 91, 11, 124, 0, 4, 0, 124, 0, 4, 0, 124, 0, 0
360 DATA 79, 11, 56, 68, 0, 68, 0, 68, 0, 68, 56, 0, 0
370 DATA 76, 11, 0, 4, 72, 48, 64, 0, 64, 0, 124, 0, 0
380 DATA 68, 11, 6, 72, 52, 64, 4, 64, 4, 120, 6, 0, 0
390 DATA 88, 11, 0, 124, 0, 20, 0, 20, 0, 20, 8, 0, 0
400 DATA 84, 11, 64, 0, 64, 0, 124, 0, 64, 0, 64, 0, 0
410 DATA 93, 11, 124, 0, 4, 0, 124, 0, 4, 0, 124, 0, 6
420 DATA 90, 11, 40, 68, 0, 84, 0, 84, 0, 84, 40, 0, 0
430 DATA 74, 11, 124, 0, 8, 128, 16, 128, 32, 0, 124, 0, 0
440 DATA 75, 11, 0, 124, 0, 16, 0, 16, 0, 40, 68, 0, 0
450 DATA 89, 11, 124, 0, 36, 0, 36, 24, 0, 0, 124, 0, 0
460 DATA 69, 11, 56, 68, 16, 68, 16, 68, 16, 68, 48, 0, 0
470 DATA 71, 11, 0, 124, 0, 64, 0, 64, 0, 96, 0, 0, 0
480 DATA 77, 11, 124, 0, 32, 16, 8, 16, 32, 0, 124, 0, 0
490 DATA 67, 11, 120, 4, 0, 4, 0, 4, 0, 124, 0, 6, 0
500 DATA 94, 11, 96, 16, 0, 16, 0, 16, 0, 124, 0, 0, 0
510 DATA 78, 11, 124, 0, 16, 0, 16, 0, 16, 0, 124, 0, 0
520 DATA 81, 11, 32, 84, 8, 80, 0, 80, 0, 124, 0, 0, 0
530 DATA 118, 139, 130, 1081, 16, 0, 254, 0, 16, 108, 130, 0, 0
540 DATA 86, 11, 68, 40, 16, 0, 124, 0, 16, 40, 68, 0, 0
550 DATA 124, 139, 68, 0, 130, 0, 146, 0, 146, 68, 56, 0, 0
560 DATA 92, 11, 68, 0, 68, 16, 68, 16, 68, 56, 0, 0, 0
570 DATA 104, 139, 130, 68, 40, 16, 0, 16, 40, 68, 130, 0, 0
580 DATA 72, 11, 68, 40, 16, 0, 16, 40, 68, 0, 0, 0, 0
590 DATA 98, 139, 254, 0, 146, 0, 146, 0, 146, 0, 12, 0, 0
600 DATA 66, 11, 124, 0, 84, 0, 84, 0, 84, 0, 8, 0, 0
610 DATA 96, 139, 254, 0, 16, 0, 124, 0, 130, 0, 130, 0, 124
620 DATA 64, 11, 124, 0, 16, 0, 56, 0, 68, 0, 68, 56, 0
630 DATA 95, 11, 64, 0, 124, 0, 20, 0, 20, 0, 8, 0, 0
640 DATA 115, 139, 124, 130, 130, 130, 130, 130, 130, 130, 68, 0, 0
650 DATA 83, 11, 56, 68, 68, 68, 68, 68, 68, 68, 68, 0, 0, 0
660 FOR i=30 TO 650 STEP 10: LPRINT CHR$(5); CHR$(27); CHR$(38); CHR$(0);: READ x: LPRINT CHR$(x); CHR$(x);
670 FOR j=0 TO 11: READ x: LPRINT CHR$(x);: NEXT j
680 LPRINT: NEXT i
690 LPRINT CHR$(5); CHR$(27): CHR$(37); CHR$(1); CHR$(0)
700 COPY
```

Литература

1. Принтер DT-130. Инструкция пользователя.

2. Ларченко А. Редактор текстов TLW2m. — СПб: МОА, 1992.

3. Поплавский А. Первый превосходный//Компьютер. — 1991. — 1(4).

А.ЕРЕМЕНКО,
г.Минск, МГВРТК, 1 курс,
член ШЮП при кафедре ВМиП БГУИР,
220023, пр.Ф.Скорины, 102 — 10, тел.63-63-52.

ЕЩЕ РАЗ О ЗАЩИТЕ ДИСКОВ ОТ ЗАПИСИ

Уже предлагалась программа, защищающая диск компьютера от записи [1]. Вот более совершенный вариант, способный автоматически защитить сразу все диски.

После запуска программы нажатием на клавиши Right Shift + Ctrl + <5> (пятерка на цифровой клавиатуре) вы можете включить или выключить защиту дисков компьютера. После запуска программы включается "защищающий" режим: если в течение минуты не было нажатий на клавиши или обращения к дисководам — защита включается сама.

После запуска программа проверяет свое наличие в памяти. Если программы в памяти нет, она перехватывает 13h прерывание (обслуживание дисков) и 09h прерывание (клавиатура). При появлении кода клавиш Right Shift + Ctrl + <5> программа инвертирует переменную OnOff, которая указывает, включен или выключен режим защиты. Если 13h прерывание вызвано с одной из функций записи или форматирования сектора, то программа блокирует эту функцию и устанавливает флаг CF — перенос, который указывает на ошибку, а в регистр AH вносится код ошибки — 3. Таким образом диск оказывается защищенным от записи.

Программа создана на языке Turbo Assembler.

```
prog    segment 'code'
assume  CS:prog,DS:prog
        org     100h
WRTLN   macro   char_string
local   string,endwrt
        push    DS
        push    CS
        pop     DS
        lea     DX,string
        mov     AH,9
        int     21h
        pop     DS
        jmp     endwrt
string  db      '&char_string',10,13,'$'
endwrt: endm
MAIN    proc
        jmp     init
Old13h  dd      0
Old09h  dd      0
Old08h  dd      0
OnOff   db      1
AutoOn  dw      0
NEW13H  proc
        mov     CS:AutoOn,0
        cmp     AH,50h
        je      pres
        jmp     going
pres:   mov     AH,096h
        jmp     pres1
going:  cmp     CS:OnOff,0
        je      no_op
        cmp     AH,03h
        je      stop
        cmp     AH,05h
        je      stop
        cmp     AH,06h
        je      stop
        cmp     AH,07h
        je      stop
        cmp     AH,0Bh
        je      stop
        cmp     AH,1Ah
        je      stop
        jmp     no_op
stop:   pop     DX
        pop     CX
        popf
        stc
        pushf
        push    CX
        push    DX
        mov     AH,03h
        mov     AL,0
        jmp     pres1
no_op:  jmp     CS:Old13h
pres1:  iret
NEW13H  endp
NEW09H  proc
        mov     CS:AutoOn,0
        push    AX
        push    CX
        push    ES
        in      AL,60h
        cmp     AL,4Ch
        jne     go
        mov     AX,040h
        mov     ES,AX
        test    byte ptr ES:[17h],4
        je      go
        test    byte ptr ES:[17h],1
        je      go
        in      AL,61h
        or      AL,80h
        out     61h,AL
        and     AL,7Fh
        out     61h,AL
        mov     AL,20h
        out     20h,AL
        cmp     CS:OnOff,1
        je      on
        mov     CS:OnOff,1
        mov     AH,50h
        jmp     beep
on:     mov     CS:OnOff,0
        mov     AH,4h
beep:   mov     CX,7000h
        in      AL,61h
        or      AL,3
        out     61h,AL
        mov     AL,0B6h
        out     43h,AL
        mov     AL,0
        out     42h,AL
        mov     AL,AH
        out     42h,AL
        loop    $
        in      AL,61h
        and     AL,0FCh
        out     61h,AL
        pop     ES
        pop     CX
        pop     AX
        jmp     theend
go:     pop     ES
        pop     CX
        pop     AX
        jmp     CS:Old09h
theend: iret
NEW09H  endp
NEW08H  proc
        cli
        inc     CS:AutoOn
        cmp     CS:AutoOn,1092
        je      auto
```

```
        jmp     CS:Old08h
auto:   mov     CS:AutoOn,0
        cmp     CS:OnOff,1
        je      noauto
        mov     CS:OnOff,1
        push    AX
        push    CX
        mov     AH,50h
        mov     CX,7000h
        in      AL,61h
        or      AL,3
        out     61h,AL
        mov     AL,0B6h
        out     43h,AL
        mov     AL,0
        out     42h,AL
        mov     AL,AH
        out     42h,AL
        loop    $
        in      AL,61h
        and     AL,0FCh
        out     61h,AL
        pop     CX
        pop     AX
noauto: jmp     CS:Old08h
NEW08H  endp
MAIN    endp
INIT    proc
        mov     AH,50h
        int     13h
        cmp     AH,096h
        je      error
        jmp     noerr
error:  jmp     errmsg
noerr:  mov     AX,3513h
        int     21h
        mov     word ptr Old13h,BX
        mov     word ptr Old13h+2,ES
        mov     AX,2513h
        mov     DX,offset New13h
        int     21h
        mov     AX,3509h
        int     21h
        mov     word ptr Old09h,BX
        mov     word ptr Old09h+2,ES
        mov     AX,2509h
        mov     DX,offset New09h
        int     21h
        mov     AX,3508h
        int     21h
        mov     word ptr Old08h,BX
        mov     word ptr Old08h+2,ES
        mov     AX,2508h
        mov     DX,offset New08h
        int     21h
        wrtln   <Резидентная защита всех дисков
                от записи и форматирования.>
        wrtln   <Shift + Ctrl + [5] - включить
                или выключить защиту дисков.>
        wrtln   <После минутного простоя компью-
                тера защита включается сама.>
        wrtln   <Copyright by Yeremenko Andrey,
                Minsk, 1996.>
        mov     AX,3100h
        mov     DX,(Init-Main+10Fh)/16
        int     21h
errmsg: wrtln   <Резидентная защита дисков уже
                загружена...>
        mov     AH,4Ch
        mov     AL,00h
        int     21h
INIT    endp
prog    ends
        end     MAIN
```

Литература

1. Радиолюбитель. Ваш компьютер. — 1996. — N 1. — С.3.

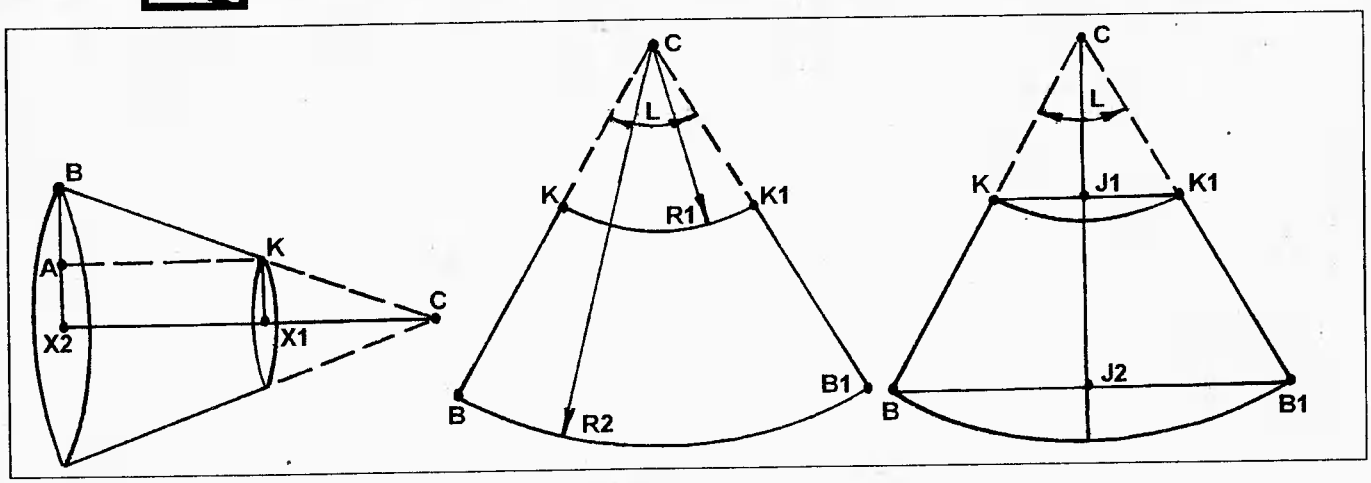
В.УСОВ,
220046, г.Минск, ул.Холмогорская, 20 — 1.

ГЕНЕРАТОР УЗОРОВ И РАСЧЕТ РАЗВЕРТКИ КОНУСА НА БЕЙСИКЕ

Предлагаю две небольшие, но интересные программы. Написаны они для компьютера "Вектор-06Ц". Язык программ максимально доведен до стандартного подмножества для облегчения адаптации на других версиях Бейсика. Думаю, что пользователи, набирая эти программы, оформят их по своему вкусу.

Примечания к рисунку: диаметр малого основания конуса — D1=KX1*2; диаметр большого основания конуса — D2=BX2*2; развертка конуса — фигура KBB1K1.

```
10 CLS: SCREEN2,15:COLOR8,240,0
20 REM ..............................
30 REM   *** ГЕНЕРАТОР УЗОРОВ ***
40 REM ..............................
50 REM
60 REM 220046 МИНСК ХОЛМОГОРСКАЯ 20/1
70 REM УСОВ ВИКТОР ГЕОРГИЕВИЧ 1993
80 REM
90 REM НАЗНАЧЕНИЕ: ГЕНЕРАЦИЯ УЗОРОВ ДЛЯ
100 REM ВЯЗАНИЯ, ВЫШИВАНИЯ И Т.Д.
110 REM
120 CUR 10,24:PRINT "ГЕНЕРАТОР УЗОРОВ": PRINT:
PRINT
130 INPUT "ЗАДАЙТЕ РАЗМЕР МАССИВА: H-ГОРИЗОНТАЛЬ
(2<=H<=12), V-ВЕРТИКАЛЬ (2<=V<=12)",H,V
140 X=100: Y=200:K=2:T=25:DIM P(12,12)
150 FOR J=1 TO V/2
160 FOR W=1 TO H/2
170 D(1)=INT(RND(1)+0.5)
180 D(2)=INT(RND(1)+0.5)
190 D(3)=INT(RND(1)+0.5)
200 D(4)=INT(RND(1)+0.5)
210 Z=W*2-1:C=J*2-1
220 P(Z,C)=D(1)
230 P(W*2,J*2-1)=D(2)
240 P(W*2-1,J*2)=D(3)
250 P(W*2,J*2)=D(4)
260 NEXT W
270 NEXT J
280 CLS
290 FOR J=1 TO V
300 FOR W=1 TO H
310 IF P(W,J)=1 THEN COLOR 8: GOTO 330
320 COLOR 7
330 PLOT X+W*2,Y-J*2,1
340 PLOT X+W*2+1,Y-J*2,1
350 PLOT X+W*2,Y-J*2-1,1
360 PLOT X+W*2+1,Y-J*2-1,1
370 NEXT W
380 NEXT J
390 PRINT
400 INPUT "ВЫВЕСТИ УЗОР D/N", A$
410 IF A$="D" THEN GOTO 430
420 IF A$="N" THEN GOTO 120
430 FOR Q=1 TO 3
440 FOR G=1 TO 4
450 FOR J=1 TO V
460 FOR W=1 TO H
```




```
470 IF P(W,J)=1 THEN COLOR 8: GOTO 490
480 COLOR 7
490 PLOT X+50+W*2,Y-J*2,1
500 PLOT X+50+W*2+1,Y-J*2,1
510 PLOT X+50+W*2,Y-J*2-1,1
520 PLOT X+50+W*2+1,Y-J*2-1,1
530 NEXT W
540 NEXT J
550 X=X+2*H
560 NEXT G
570 X=X-8*H
580 Y=Y-2*V
590 NEXT Q
600 INPUT "ЕЩЕ?",A$
610 IF A$="D" THEN GOTO 120
620 IF A$="N" THEN GOTO 630
630 REM ——— END ———
```

```
10 CLS:SCREEN 2,15: COLOR 8,240,0
20 REM .........................
30 REM      <<- KONUS ->>
40 REM .........................
50 REM
60 REM 220046  МИНСК  ХОЛМОГОРСКАЯ 20/1
70 REM УСОВ  ВИКТОР  ГЕОРГИЕВИЧ  1994
80 REM
90 REM НАЗНАЧЕНИЕ: РАСЧЕТ РАЗВЕРТКИ КОНУСА ИЛИ
УСЕЧЕННОГО КОНУСА
100 REM ПРИМЕНЕНИЕ: СОЕДИНЕНИЕ ТРУБ РАЗ-
110 REM ЛИЧНОГО ДИАМЕТРА, ИЗГОТОВЛЕНИЕ
120 REM ВОРОНОК, КОЛПАКОВ И Т.Д.
130 CUR 13,24:PRINT"——KONUS——":PRINT
135 PRINT:PRINT "РАЗМЕРНОСТЬ: D1, D2, BK —
(MM)":PRINT
140 INPUT "ВВЕДИТЕ МЕНЬШИЙ ДИАМЕТР D1 (ЕСЛИ КОНУС
НЕ УСЕЧЕННЫЙ, ТО D1=0,1) - D1=",D1
150 INPUT "ВВЕДИТЕ БОЛЬШОЙ ДИАМЕТР D2=",D2
160 PRINT "ВВЕДИТЕ  РАССТОЯНИЕ МЕЖДУ ДИАМЕТРАМИ ПО
ОБРАЗУЮЩЕЙ КОНУСА — BK="
170 INPUT BK
180 REM РАСЧЕТ РАЗВЕРТКИ
190 KX1=B1/2:BX2=D2/2
200 KC=BK/(BX2/KX1-1):R1=KC
210 BC=BK+KC:R2=BC
220 L=BX2*360/BC
230 AK=SQR(BK^2-(BX-KX1)^2)
240 CJ1=KC*COS((L/2)/180*PI)
250 KJ1=SQR(KC^2-CJ1^2)
260 KK1=KJ1*2
270 BB1=KK1*BC/KC
280 REM ВЫВОД РЕЗУЛЬТАТОВ
290 PRINT
300 PRINT "МЕНЬШИЙ РАДИУС РАЗВЕРТКИ - R1=";R1;"MM"
310 PRINT "БОЛЬШИЙ РАДИУС РАЗВЕРТКИ - R2=";R2;"MM"
320 PRINT "УГОЛ СЕКТОРА РАЗВЕРТКИ - L=";L;"GR"
330 PRINT "ВЫСОТА КОНУСА - AK=";AK;"MM"
340 PRINT
350 PRINT "ЕСЛИ НЕТ ТРАНСПОРТИРА, ТО РАЗВЕРТКУ
МОЖНО ПОСТРОИТЬ ПО ХОРДАМ РАДИУСОВ РАЗВЕРТКИ:"
360 PRINT
370 PRINT "ХОРДА МАЛОГО РАДИУСА - KK1=";KK1;"MM"
380 PRINT "ХОРДА БОЛЬШОГО  РАДИУСА -
BB1=";BB1;"MM"
```

ПОПРАВКИ

В статье "Выбор тактовой частоты для музыкального сопроцессора" ("РЛ. Ваш компьютер" 1/1996, с.23-26) при публикации допущено несколько опечаток.

В листинге 1:

- в строке 50 в кавычках перед фразой A#A должен быть пробел;
- правильное написание строки 70:

70 OUT r,0: LET t=IN r: OUT r,1: LET g=IN r

В листинге 2 в строке 130 должно быть: ... INT((max+min)/2);...

В листинге 4 должно быть: 30 OUT r,7: OUT w,8 ...

В формуле на с.25 должен быть знак возведения в степень:

Ki(opt)=CINT(K1/2^((i-1)/12)).

С.РЮМИК,
250033, г.Чернигов, а/я 1772.

В программе "Крокодил" ("РЛ. Ваш Компьютер" 3/1996, с.32-33) допущены опечатки:

- в строке 6 после числа 197 необходимо вставить 34,192, ;
- в строке 360 должно быть ... PLOT 5,125,1 ... ;
- в строке 380 — PLOT 10,210,2: LINE ... ;
- в строках 500...520 и 550...570 после надписи в операторе PRINT (в кавычках) нужно добавить пробел;
- в строке 936 должно быть ... GOTO 2000 ;
- в строке 2350 перед IF нужно добавить N$=INKEY$: .

Редакция

С.КУЗНЕЦОВ,
222310, г.Молодечно, ул.Ф.Скорины, 33 — 19. Тел.6-40-56.

АВТОМАТИЧЕСКАЯ ТРАССИРОВКА ПЕЧАТНЫХ ПЛАТ В ПАКЕТЕ PCAD

(Продолжение. Начало в NN1-3/96)

Следующий шаг — переход в подменю >Define detailed parameters<.

Здесь можно переустановить шаг координатной сетки или задать несколько сеток трассировки, если это требуется. Подобная необходимость возникает в случаях применения компонентов с различным или нестандартным шагом выводов. Далее определяется тип переходных отверстий. Они могут быть сквозными (Through — на всех типах печатных плат), межслойными (Interstitial) и смешанными (Mixed — только для многослойных печатных плат). После выбора типа переходных отверстий выбирается способ установки их на плате. Это либо наиболее часто применяемое Via sites All grid points — во всех точках координатной сетки, либо по специально заданной сетке Via Lattice, либо на ограниченной области Via lattice region. В последующих строках определяется количество проходов трассировки и расстояние до края платы. Так как все параметры не помещаются на экране, последняя строка — >Continue detailed parametr menu< (Продолжение). Здесь определяется цена переходного отверстия Through via cost, значение которой по умолчанию установлено 30, а вообще может иметь значение от 1 до 999. Значение 30 достаточно оптимально, значение менее 10 может привести к некачественной трассировке, а более 50 — к значительному увеличению времени. Bevel size — параметр размера изломов диагональных проводов, который желательно установить равным 0 для исключения изломов типа "пила".

Следующая строка — определение ориентации компонентов. Этот параметр определяет основные направления трассировки. Может быть: Horizontal (горизонтально), Vertical (вертикально), Unspecified (не определено).

Одним из важных параметров этого меню является генерация переходных отверстий от планарных выводов SMD компонентов (Generate stringers). В процессе трассировки программа резервирует пространство под эти переходные отверстия и впоследствии убирает ненужные.

Нажав два раза клавишу ESC, возвращайтесь в первое меню.

Следующее меню — Edit ripup parametrs — определяет типы проходов трассировки и их количество.

В версиях PCAD 4.XX появилось мощное средство доводки трассировки до максимально возможного процентного значения. Это так называемый многослойный Rip-N-Route алгоритм. В процессе работы этого алгоритма пересматриваются все проведенные связи на плате и переходные отверстия, проводники могут переноситься со слоя на слой, изменяется количество переходных отверстий, перемещаются переходные отверстия. Процесс Ripup может занимать значительное время.

Первая группа меню — количество проходов трассировки.

Number of Maze Routing Passes — количество проходов:

1) Normal: количество проходов основного алгоритма Maze Routing Passes — в нашем случае Steiner. Обычно задается 2 прохода.

2) Ripup: обычно 1 из-за времени исполнения.

3) Optimize: проход оптимизации проводов и переходных отверстий. Обычно задают один проход, но многие предпочитают оптимизировать плату вручную, тем более что, если не достигнута стопроцентная трассировка, после оптимизации очень трудно провести на плате проводник.

Параметр Penalize corner — скругление углов — рекомендуется устанавливать в положение NO, т.к. отечественные и основная масса импортных фотопостроителей не могут рисовать скругленные углы.

Нажатием клавиши ESC возвращаемся в основное меню.

Переходим к пункту меню >Edit pad description< (корректировка площадок). Это очень важная операция, т.к. если ваши контактные площадки отсутствуют в стратегии, программа останавливается и не трассирует плату. Если вы не описали площадки, конечная информация будет искажена. Первичная информация берется из таблицы типов контактных площадок, описанной ранее. В меню корректировки 3 параметра: тип площадки, форма и размеры.

Например в исходном файле стратегии вы видите Pin type: 0, Pad shape: Circle, Diameter: 40. В нашей таблице диаметр контактной площадки типа 0 (переходное отверстие) должен быть 1,5 мм, т.е. 60 милс. Изменяем размер на 60 и переходим к следующему типу. Выбираем курсором или клавишей Enter Pin type: 0 и вводим 1. Тип 1 — это квадрат со стороной 1,5 мм. Переходим к строке Pad shape и пробелом выбираем Square и нажимаем Enter. В третьей строке появляется вместо Diameter Side, вводим размер стороны квадрата 60. Таким образом необходимо пересмотреть все площадки. На планарные микросхемы, например серий 133, 564, 533, тип контактной площадки — 25, это прямоугольник со сторонами 33 на 50 милс.

Клавишей ESC вернитесь в основное меню.

Следующий пункт меню — >Edit wiring rules<:

Табл. 2

Pad/Pad Clearance:12			
Net	Trace width	Line/line Clearance	Line/pad Clearance
DEFAULT	16	12	12
PWR	40	12	12
IGNORE	20	12	12

После нажатия клавиши Home можно изменить минимальный зазор между площадками. В самой таблице (табл.2) определяют имена цепей, ширину проводников, зазоры между линиями и между линией и площадкой. Цепи, которые проведены до начала трассировки в редакторе PCCARDS и имеют ширину, отличную от предполагаемой, должны быть внесены в таблицу, либо в таблицу записывается тип цепи IGNORE с конкретной шириной проводников.

Следующий пункт меню — >Edit net class definition< — определяет класс цепей и их приоритеты. Выбрав класс цепи, например PWR, программа переходит к запросам о ширине проводника (предлагает ту, что внесена в таблицу предыдущего пункта) и приоритете (None, Ignore, Low, Medium, High). Если выбрать высший приоритет (High), эти цепи трассируются в первую очередь. Если выбрать Ignore, эти цепи не будут трассироваться вообще. В строке Nets определяют,

какие цепи принадлежат к данному классу. Например класс PWR обычно определяет цепи питания. Можно ввести 5V, 12V, 0V, -12V. Эти цепи трассируются по закону класса PWR.

Последний пункт меню — >Edit layer description< — описывает направление трассировки проводников различных слоев. При этом приводится количество слоев, цена переходного отверстия и таблица слоев (табл.3):

Табл. 3.

	Name	Type	N/S	E/W
1	COMP	EAST/WEST	4	1
2	SOLDER	NORTH/SOUTH	1	4

Направление предпочтительной прокладки проводов можно изменить.

Нажав ESC, возвратитесь в меню. Повторно нажмите ESC — и программа переходит в меню записи стратегии трассировки. Здесь предлагаются 3 варианта. Можно записать файл под тем же именем, что и до корректировки — >Save under current name<. Если корректировалась стратегия PCAD1, файл не перезапишется, о чем появляется соответствующее предупреждение. Такая стратегия записывается под новым именем — >Save under new name< . Введите новое имя файла стратегии трассировки и нажмите Enter. И последний вариант — выйти из меню корректировки стратегии без изменений (>Do not use or save<).

Выйдите из меню корректировки, при этом программа переходит в главном меню на строку выбора стратегии трассировки. Пробелом необходимо выбрать стратегию, записанную нами, и перейти к пункту >Route<. Здесь имеется 4 запроса и пункт Start. Запросы следующие:

Extract data	Yes
Route	New
Create routed database	Yes
Database name	TX.PCB

Изменение первых двух запросов связано с тем, что программа формирует файл отчета хода трассировки и прерванную в любой момент трассировку можно возобновить с момента прерывания. Если на третий запрос ответ отрицательный, после окончания трассировки не будет записываться на диск файл готовой платы. Это делается в том случае, когда надо только проверить возможность трассировки той или иной платы на промежуточном этапе проектирования.

После ответа на все запросы переводите курсор на строку Start и нажимайте Enter. Трассировка началась.

Сначала программа прочитывает данные из файла TX.PLC, определяет границы платы, считает все цепи, сравнивает данные со стратегией трассировки и, при положительном результате анализа, начинает трассировать плату. Если в базе данных имеются несоответствия стратегии трассировки, программа останавливается и предлагает прочитать файл ошибки TX.REP, где указано, что именно не соответствует. Как правило, это либо отсутствие в стратегии контактных площадок, либо в файле TX.PLC проведены линии, не определенные стратегией.

Если все в порядке, на экране сначала появляется счетчик времени, а затем программа переходит в экран трассировки. В правой части экрана расположен так называемый Status, где отображается информация об имени базы данных, имени стратегии трассировки, текущее время трассировки, количество слоев, количество субцепей. Кроме текущего времени, все это — постоянные данные. Далее следует изменяющаяся информация: количество переходных отверстий на плате, максимальное количество оттрассированных субцепей, текущая цепь на трассировке, процент выполненного задания, название типа прохода, имя трассируемого слоя, имя текущей цепи.

После завершения трассировки программа дает сообщение Route complete (CR). При нажатии Enter, либо автоматически, через 3...5 минут: программа создает файл TX.PCB.

В процессе трассировки можно вносить коррективы в ход действия, нажав клавишу ESC. При этом можно прервать трассировку, пропустить цепь и перейти к трассировке следующей, пропустить проход.

Если программа PCROUTE страссировала плату на 100% и записала файл базы данных, можете себя поздравить — вы практически освоили весь цикл проектирования. Остальная часть пакета посвящена связи конструкторской разработки с технологией изготовления печатных плат.

Корректировка печатных плат

Загрузите файл TX.PCB в редактор PCCARDS. Вы видите печатную плату, которую сами разработали и оттрассировали буквально за несколько часов. Кстати сказать, у автора на разработку этого примера от рисования электрической схемы до корректировки и вплоть до подготовки материалов в формате HPGL для публикации ушло около часа времени. Поэтому не расстраивайтесь, если на первые схемы и платы у вас уйдет много времени. Вы быстро освоитесь в системе и с каждым разом будете делать все намного быстрее.

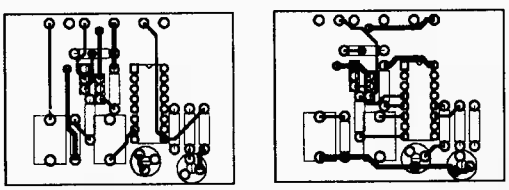


Рис. 1

Не думаю, что полученная сразу после трассировки топология (рис.1) удовлетворит вас.

Вы наверняка заметите, что в некоторых местах проводники "необоснованно" выходят на слой COMP, а в некоторых местах есть лишние переходные отверстия. Есть места, где проводник мог быть короче и без "забора". Для "прически" платы используйте команды редактирования PCCARDS, такие как EDIT/DELS для удаления лишних сегментов проводников (ни в коем случае при этом не пользуйтесь командой DEL, так как она удалит целиком цепь и восстановить ее будет весьма затруднительно), EDIT/MOVV, EDIT/MOVA для перемещения проводников и вершин, EDIT/ADDV для формирования изломов, EDIT/DELV для удаления изломов и т.д. Попробуйте корректировать плату. Чтобы не было неприятностей, сделайте резервное копирование исходного файла, и вы всегда сможете вернуться в начало.

После корректировки топологии слои могут выглядеть примерно так, как показано на рис.2.

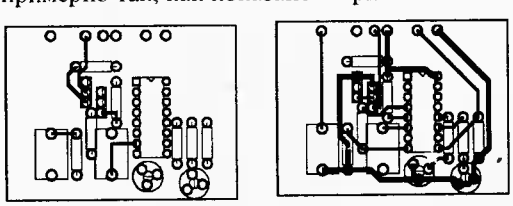


Рис. 2

Как вы заметили, слой SOLDER изображен на рисунке зеркально — так, как он видится в редакторе PCCARDS. При подготовке для чертежа или производства плат слой обрабатывается программами под зеркальное изображение.

А.ИНОЗЕМЦЕВ,
141980, Московская обл., г.Дубна, ул.Моховая, 4 — 3.

ПРОГРАММАТОР РПЗУ ДЛЯ IBM PC

В последнее время стали очень популярны радиолюбительские конструкции, в основе которых лежат элементы микропроцессорной техники. Как правило, их неотъемлемой частью является РПЗУ — постоянное запоминающее устройство со стиранием информации ультрафиолетовым излучением и с возможностью многократного программирования. В отличие от остальных элементов микропроцессорной техники, РПЗУ нуждается в программировании, которое из-за трудоемкости часто выполняется с помощью ЭВМ специальным устройством — программатором РПЗУ.

Как правило, программатор — дорогостоящее устройство. Поэтому если перед конструктором стоит задача однократного изготовления и настройки прибора, использующего в своем составе РПЗУ, имеет смысл не приобретать программатор, а собрать его самостоятельно.

Вниманию читателей предлагается простой программатор РПЗУ. В отличие от описанного Ю.Власовым [1], он содержит меньшее число активных элементов, прост в управлении, требует небольшого количества действий оператора при работе с ним и обеспечивает программирование большинства отечественных и зарубежных БИС РПЗУ.

Подключение программатора непосредственно к системной магистрали компьютера позволяет без изменений в устройстве последнего производить программирование БИС РПЗУ с максимальной быстротой, определяемой в основном алгоритмом их программирования.

Для работы программатора необходимы:

- любой компьютер, совместимый с IBM PC, имеющий в своем составе как минимум 256 Кб ОЗУ, один НГМД или НЖМД. Версия ДОС должна быть не младше 3.30;
- источник питания (можно нестабилизированный) с выходным напряжением 27...35 В при токе нагрузки не менее 50 мА.

Программатор обеспечивает выполнение следующих функций:

- программирование всех отечественных и зарубежных БИС РПЗУ с объемами памяти от 2К х 8 до 64К х 8 включительно и имеющих напряжения программирования от 12.5 В до 26 В;
- чтение и запись информации из РПЗУ в файл на диске и наоборот;
- проверку на качество стирания РПЗУ перед записью в него информации;
- автоматическое включение и выключение напряжения программирования (Upr).

Схема электрическая принципиальная программатора условно разделена на две части: аналоговую и цифровую. Они показаны на рис.1 и 2 соответственно (рис.2 будет приведен в продолжении статьи, см. "ВК" N5/96).

Цифровая часть программатора работает следующим образом.

Все необходимые сигналы из ПК поступают в программатор через разъем X1.1, ответная часть которого находится на системной плате ПК. Каждой функциональной плате, которая подключается к системной магистрали ПК, соответствует набор адресов ввода-вывода [2]. Для плат, разработанных пользователем, отведены несколько интервалов адресов. За программатором закреплен один из них, включающий адреса с 300H по 31FH. Эта информация положена в основу работы узла декодирования адреса, построенного на элементах DD1 и DD2.4.

Рассматриваемый узел вырабатывает на выводе 8 DD1.3 сигнал CS выбора БИС DD3 в случае одновременного появления любого числа из интервала 300H...31FH на адресной шине ПК и сигнала разрешения адреса AEN.

Связь программатора с ПК осуществляется с помощью БИС DD3 — универсального программируемого интерфейса КР580ВВ55А. Три его восьмиразрядных порта ввода-вывода A, B и C работают в двух режимах: A, B, C — на вывод; A — на ввод, B, C — на вывод информации. Управляющий регистр интерфейса находится по адресу 30CH, порт A — по адресу 300H, порт B — по адресу 304H и порт C — по адресу 308H.

Для дальнейшего описания работы программатора удобно рассмотреть три основные его шины: двунаправленную шину данных, формируемую портом A БИС DD3, и однонаправленные шины адреса и управления, формируемые портами B и C БИС DD3 совместно с регистром DD5.

Восьмиразрядная шина данных обеспечивает обмен данными при чтении и записи между БИС РПЗУ и ПК. Сигналы на ней формируются ПК при записи в РПЗУ и РПЗУ при чтении из него.

Шестнадцатиразрядная шина адреса предназначена для передачи адреса в РПЗУ. Адрес на ней формируется следующим образом.

Адреса A0...A7 поступают на шину с порта B БИС DD3. Адреса A8...A12 формируются портом C БИС DD3. Адреса A13...A15 формируются с помощью разрядов B0...B2 порта B регистром DD5.

Для более четкого понимания процесса формирования старших трех разрядов адреса и некоторых сигналов управления далее рассматривается процесс записи в регистр DD5. Запись в него происходит следующим образом. Сначала в порт B выводится информация, которую необходимо записать в регистр DD5.

Затем на выводе 10 DD3 (C7) формируется отрицательный перепад напряжения. При последней операции происходит запись информации из порта B в регистр DD5.

Шина управления предназначена для передачи сигналов управления и напряжения Upr на БИС РПЗУ при чтении-записи. Все сигналы на нее поступают с узла формирования программирующего импульса и с выходов Q3...Q5 регистра DD5.

Узел формирования программирующего импульса выполнен на одновибраторе DD4.2 и элементах R4, R3 и C3. Рассматриваемый узел формирует импульс с длительностью t1 = 1 мс на выходах DD4.2 при отрицательном перепаде напряжения на выводе 11 DD3 (C6). В течение t1 происходит однократная запись одного байта в БИС РПЗУ.

Управляющие сигналы

С помощью сигналов с выходов Q3 DD5 и Q2 DD4.2 элементами DD2.2 и DD2.3 формируется сигнал CS для БИС РПЗУ 2732, 27256 и 27512. На выходе Q4 DD5 формируется сигнал OE для всех БИС РПЗУ, кроме 2732 и 27512. На выходе Q5 DD5 формируется сигнал управления напряжением Upr. На выходе Q2 DD4.2 формируется сигнал CS БИС РПЗУ 2716 (КР573РФ5). Он также используется для формирования сигнала OE БИС РПЗУ 2732 и 27512. На выходе Q2 DD4.2 формируется сигнал PGM для БИС РПЗУ 2764 и 27128. Временные диаграммы управляющих сигналов аналогичны приведенным в [1]. Переключатели JP1... JP7 (табл.1, см. "ВК" N5/96) служат для выбора типа БИС РПЗУ.

(Продолжение следует)

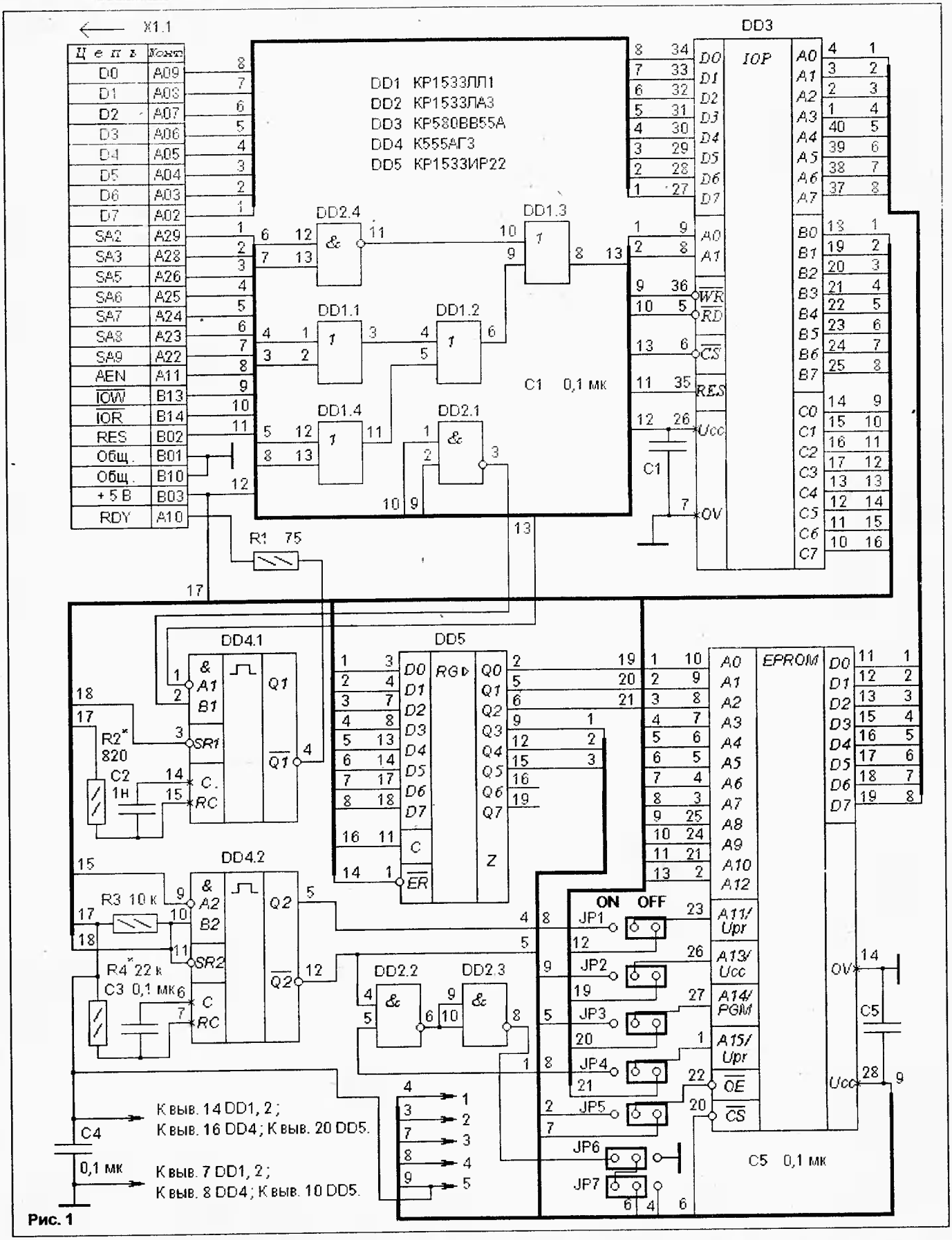

X1.1
Цепь | Конт
D0 A09
D1 A08
D2 A07
D3 A06
D4 A05
D5 A04
D6 A03
D7 A02
SA2 A29
SA3 A28
SA5 A26
SA6 A25
SA7 A24
SA8 A23
SA9 A22
AEN A11
IOW B13
IOR B14
RES B02
Общ. B01
Общ. B10
+5 В B03
RDY A10
DD1 КР1533ЛП1
DD2 КР1533ЛА3
DD3 КР580ВВ55А
DD4 К555АГ3
DD5 КР1533ИР22
DD2.4
DD1.3
DD1.1
DD1.2
DD1.4
DD2.1
C1 0,1 мк
DD3
IOP
R1 75
DD4.1
DD5
R2* 820
C2 1н
DD4.2
R3 10 к
R4* 22 к
C3 0,1 мк
DD2.2
DD2.3
EPROM
ON OFF
JP1
JP2
JP3
JP4
JP5
JP6
JP7
C4 0,1 мк
C5 0,1 мк
К выв. 14 DD1, 2;
К выв. 16 DD4; К выв. 20 DD5.
К выв. 7 DD1, 2;
К выв. 8 DD4; К выв. 10 DD5.

Рис. 1

В.ФРОЛОВ,
г.Кишинев.
тел.(0422) 73-88-64.

ЭМУЛЯТОР "ZX-SPECTRUM" НА "ВЕКТОРЕ-06Ц"

Предлагаемый адаптер Z80 для ПК "Вектор-06Ц" позволяет работать на "Векторе" непосредственно с синклеровскими дискетами и лентой, а также дает возможность включения принципиально новых режимов работы. Кратко о новых возможностях и преимуществах:

1. Дополнительные команды Z80, более быстрое выполнение старых команд.

2. Возможность гибкого режима с квазидиском через регистр (HL).

3. Активное использование всего ОЗУ электронного диска, не отличающееся от использования собственного ОЗУ "Вектора". Удобство перехода между ОЗУ.

4. "Sinclair", экран и возможность практически полной эмуляции "Sinclair-48" (впоследствии — 128).

5. Благоприятный режим работы микросхем памяти. С точки зрения программиста, новые режимы задаются записью в порт 80h числа, биты которого соответственно равны:

0 — включение синклеровского экрана;

1 — инверсия старшего бита адреса ОЗУ;

2 — обращение к квазидиску в операциях с (HL);

4 — немаскируемые прерывания по незнакомым портам;

5 — имитация ПЗУ для адресов 0...3FFFh, блокировка электронного диска при обращении к адресам 4000h...5FFFh;

6 — увеличение быстродействия некоторых операций;

7 — блокировка обращения к квазидиску в стековых операциях.

После выключения все биты устанавливаются в "0", что соответствует режиму обычного "Вектора".

Специально отмечу, что есть небольшие различия в исполнении команд 580ВМ80 Z80-м процессором, в частности, формирование флага паритета/четности в операциях сложения, вычитания и др.

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

С устройством работает абсолютное большинство "векторовских" программ. Исключение составляют программы, для которых существенно вышеупомянутое отличие в исполнении команд. Ко времени написания статьи создана уже третья версия программы ZX-эмуляции работы "Spectrum". В отличие от первой версии, программа практически полностью эмулирует работу "Spectrum-48" (клавиатура, цвет при наличии квазидиска, звук, загрузка-выгрузка с магнитофона, один или несколько дисководов, программируемый джойстик и клавиши) без каких-либо доработок в периферии "Вектора" и без переработок синклеровских программ (загрузка непосредственно с синклеровской дискеты или кассеты). В следующей версии планируется предусмотреть работу с принтером и музыкальным сопроцессором, обращение к электронному диску как к одному из дисководов. Широкие возможности открывает адаптация "ямаховского" MSX-DOS.

ПОДКЛЮЧЕНИЕ

Подключение адаптера не является даже незначительной переработкой "Вектора". В простейшем случае (герконовая клавиатура) специальный разъем вставляется прямо в панельку, где стоял 580-й процессор. Еще три сигнала берутся с платы "Вектора". Есть возможность обратной замены.

ПРОГРАММИРОВАНИЕ АДАПТЕРА

Процессор Z80 обладает несравненно более обширной системой команд по сравнению с 580-м процессором. За малым исключением, Z80 исполняет одинаково все команды 580-го. Исключение составляют формирование флага паритета/четности при сложении и вычитании и некоторые другие.

Другой ряд новых возможностей адаптера связан со встроенным портом вывода. Адрес этого порта — 80h (точнее, 10*0*** в двоичном коде). После сброса все биты порта устанавливаются в "0", что соответствует обычному режиму "Вектора". Новый режим работы задается путем записи в порт какого-либо числа, которое запоминается до следующего обращения к порту или сброса. Каждый бит числа, записанного в порт, отвечает за какой-либо режим работы.

Краткое назначение битов порта 80h описано выше, теперь разберемся подробнее. Биты числа, записанного в порт 80h, обозначены как O0, O1...O7 и, разумеется, могут принимать значение "0" и "1". При обращении к ОЗУ "Вектора" или квазидиска 16 бит адреса обозначены как A0, A1...A9, AA, AB, AC, AD, AE, AF.

БЫСТРОДЕЙСТВИЕ

Для выполнения ряда общих команд Z80 и i8080 требуется различное время. Так, операции MOV R1, R2; INR R; DCR R (R, R1, R2 отличны от M) выполняются в два раза быстрее, что может сказаться на программной совместимости, например при программировании палитры цветов. С целью более полной совместимости в адаптер введена возможность задержки. Так, при O6=0 вышеперечисленные операции выполняются за то же время, а при O6=1 — вдвое быстрее.

НЕМАСКИРУЕМОЕ ПРЕРЫВАНИЕ

Адаптер эмулирует работу "не-векторовских" программ. При O4=1 происходит немаскируемое прерывание после обращения (чтения или записи) к порту с двоичным адресом ***1****. По адресу 66h, к которому происходит переход после прерывания, должна находиться подпрограмма, идентифицирующая устройство, к которому идет обращение, и заменяющая его на обращение к соответствующим портам "Вектора". При O4=1 такое прерывание не происходит.

ФОРМИРОВАНИЕ СИГНАЛА STEC

В "Векторе" этот сигнал играет важную роль при обращении к электронному диску. 580-й процессор формирует этот сигнал при записи в стек или чтении из него. В "Векторе" обращение к квазидиску через сигнал STEC происходит, если в 4-й бит порта 10h записана "1". Отметим, что в адаптере формирование сигнала STEC не эквивалентно обращению к стековому адресу, как с 580-м. Адаптер формирует этот сигнал более сложным образом, благодаря чему возможны принципиально новые режимы работы с электронным диском. Промежуточный сигнал

STECB=[[обращение к (HL) в MOV, INR, DCR] and O3] or [[обращение к стеку в операциях POP, PUSH] and/O7],

где / обозначает логическую инверсию: /X=not X. Окончательный сигнал:

STEC=[STECB xor O2] and /[/AF and AE and /AD and O5].

Кратко поясним, как можно использовать некоторые новые режимы, чтобы обращаться к электронному диску, ис-

пользуя сигнал STEC. Запишем в порт 10h число 0001NN00 (двоичный код), где NN — страница электронного диска. При O7=1 исключается чтение и запись на квазидиске при стековых операциях. При O3=1 обращение к электронному диску происходит при некоторых операциях с регистром M, что гораздо мобильнее обращения через стек. Бит O2=1 инвертирует сигнал STEC, при этом ОЗУ "Вектора" и электронного диска как бы меняются местами: выполняемая программа находится на электронном диске, а к ОЗУ "Вектора" происходит обращение как к электронному диску.

При включении этого бита нужно соблюдать определенную осторожность. Об этом ниже.

Бит O5 используется для блокировки сигнала STEC при обращении по адресам 4000...5FFFh, и блокировки записи по адресам 0...3FFFh. Этот бит используется для эмуляции работы Sinclair, у которого по адресам 0...3FFFh находится ПЗУ, а внутри области 4000...5FFFh находится экран. Отметим, что адаптер не интерпретирует команду XTHL как стековую, кроме того, при работе с квазидиском через стек (четвертый бит порта 10h равен "1") не регламентируется работа с условными и безусловными командами CALL, RET, RST.

Практически это не ведет к каким-либо ограничениям, так как для работы с квазидиском используются исключительно команды PUSH и POP. Также не регламентируются работы новых команд Z80, начинающихся DDh, EDh, FDh, CBh при O4=1 и работе с RAM диском.

ИЗМЕНЕНИЕ АДРЕСНОГО ПРОСТРАНСТВА

Важную роль играет промежуточный сигнал SINC, формируемый как SINC=O0 and AF2 and /STEC,

где AF2=AF xor O1.

При обращении МП к ОЗУ с двоичным адресом AF AE AD AC AB AA A9 A8 A7 A6 A5 A4 A3 A2 A1 A0 реально происходит обращение по адресу AF2 AE AD AC AB AA A9 A8 A7 A6 A5 A4 A3 A2 A1 A0 при SINC=0, и к AF2 AE AD AC A4 A3 A2 A1 A0 /AC /AB /A7 /A6 /A5 /AA/ A9 /A8 при SINC=1.

Изменение адресного пространства никак не сказывается на формировании сигнала STEC и применяется для формирования "синклеровского" экрана, отличного от "векторовского". При O0, O1, O5=1 при работе "синклеровских" программ, графика заносится на первую экранную плоскость "Вектора" по координатам, соответствующим положению на "синклеровском" экране. Таким образом можно получить черно-белое изображение без каких-либо переделок в программе.

ПЕРЕКЛЮЧЕНИЕ БИТОВ O1 И O5

Предположим, мы захотели изменить один из этих битов. Если адрес команды OUT 80h — MET, то следующий адрес, запрашиваемый МП после выполнения OUT 80h — MET2. После изменения режимов работы, связанных O1 и O5, реальное считывание происходит либо по преобразованному адресу (бит O1), либо происходит переход с ОЗУ "Вектора" на ОЗУ квазидиска, или наоборот (бит O5). В новом месте должен быть подготовлен текст продолжения программы. Если бы соответствующие режимы переключались сразу же после OUT 80h, пришлось бы подготавливать продолжение программы в строго определенном месте. Это неудобно. Аппаратно найдено другое решение — переключение происходит только после извлечения кода следующей за OUT команды. Рекомендуется задавать команду PCHL.

Пример перехода из ОЗУ "Вектора" на нулевую плоскость электронного диска:

```
   MVI A, 10H
   OUT 10H
   LXI H, MET1
   MVI A, 84H
   OUT 80H
   PCHL
; переход по адресу MET1 на квазидиск.
   ...
;текст, заранее перенесенный на квазидиск
   .PHASE ...
   ...
;продолжение программы на квазидиске
MET1: ...
   ...
   .DEPHASE
   END
```

Пример изменения старшего бита адреса:

```
   LXI H, MET3
   MVI A, 2
   OUT 80H
   PCHL
;переход по адресу MET3 и изменение адр. пространства
MET2: .PHASE 8000H+MET2
   ...
;продолжение программы
MET3: ...
   ...
   .DEPHASE
   END
```

ПРИНЦИП РАБОТЫ

Сигналы адаптера условно разделены на четыре части: ШИНА I8080, ШИНА Z80, СИНХРОНИЗИРУЮЩИЕ СИГНАЛЫ и ВНУТРЕННИЕ СИГНАЛЫ адаптера.

В ШИНЕ I8080 присутствуют сигналы, необходимые для эмуляции работы микросхемы. В ШИНЕ Z80 — сигналы нового процессора. Обозначения сигналов обеих шин общепринятые.

Триггеры D23 и D25, тактируемые кварцевым генератором, синхронизируют работу "Вектора" и адаптера. Каждый выходной сигнал микросхем T1...TF 8 тактов кварцевого генератора подряд принимает значение логической "1", а остальные 8 — "0". Сигналы T с соседними номерами сдвинуты друг относительно друга на такт: T2 задерживается на такт относительно T1, T3 — относительно T2 и т.д. Предложенное решение позволяет получить синхронные с "Вектором" сигналы любой формы с периодичностью 16 тактов генератора, дополнительно используя лишь элементарную логику. Так, сигнал WE формируется элементом D26.1, SYNC — D22.1, WAIT — D22.2 и т.д.

Более сложный тактовый сигнал процессора D5 F формируется D26. Микросхемы D18.2, D18.3, D12.3 задерживают задний фронт тактового сигнала процессора F. Этот сигнал не является строго периодичным с частотой 3 МГц, как у процессора I8080 в "непеределанном" "Векторе". Более высокая тактовая частота применяемого процессора Z80A (B) — 4(6) МГц по сравнению с 3 МГц I8080 позволяет отдельные такты процессора, в данном случае связанные с обращением к ОЗУ, несколько удлинить.

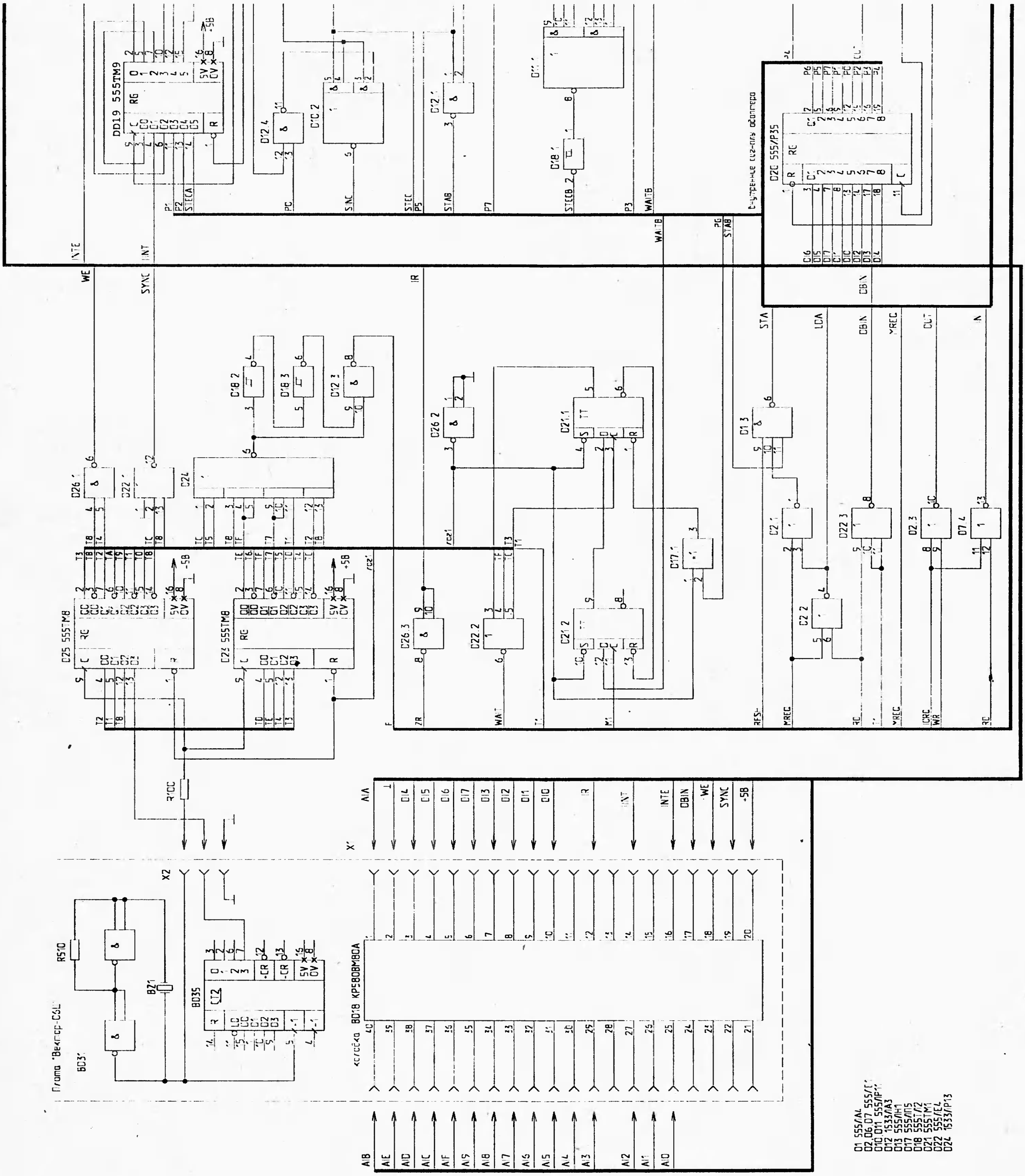
Мультиплексирование данных и слова состояния, применяемое на I8080, выполнено на D14, D15.

Элементарная логика D1.3, D2.1, D2.2, D2.3, D7.4 формирует необходимое слово состояния (сигналы STA, STEC, LDA, OUT, IN, MREQ). Регистр D16 обеспечивает прием данных в циклах чтения. Адрес, передаваемый на колодку I8080 (сигналы AI0...AIF) элементами D3, D4, D8, D9, D11.2, D13.2, существенно отличается от адреса Z80 (сигналы AZ0...AZF). Можно выделить три основных случая адресации:

1. Обращение к памяти в режиме “Вектора” (IORQ=1 SINC=1). Сигналы адреса Z80 (AZ0...AZF) без изменений транслируются в сигналы адреса I8080 (AI0...AIF).

2. Обращение к внешним устройствам (IORQ=0. SYNC=1). Сигналы адреса AZ0...AZ7 передаются на AI8...AIF.

3. Обращение к памяти в режиме “Синклеровский экран” (IORQ=1, SYNC=0). Сигналы адреса транслируются сложным образом, часть из них инвертируется. Необходимость

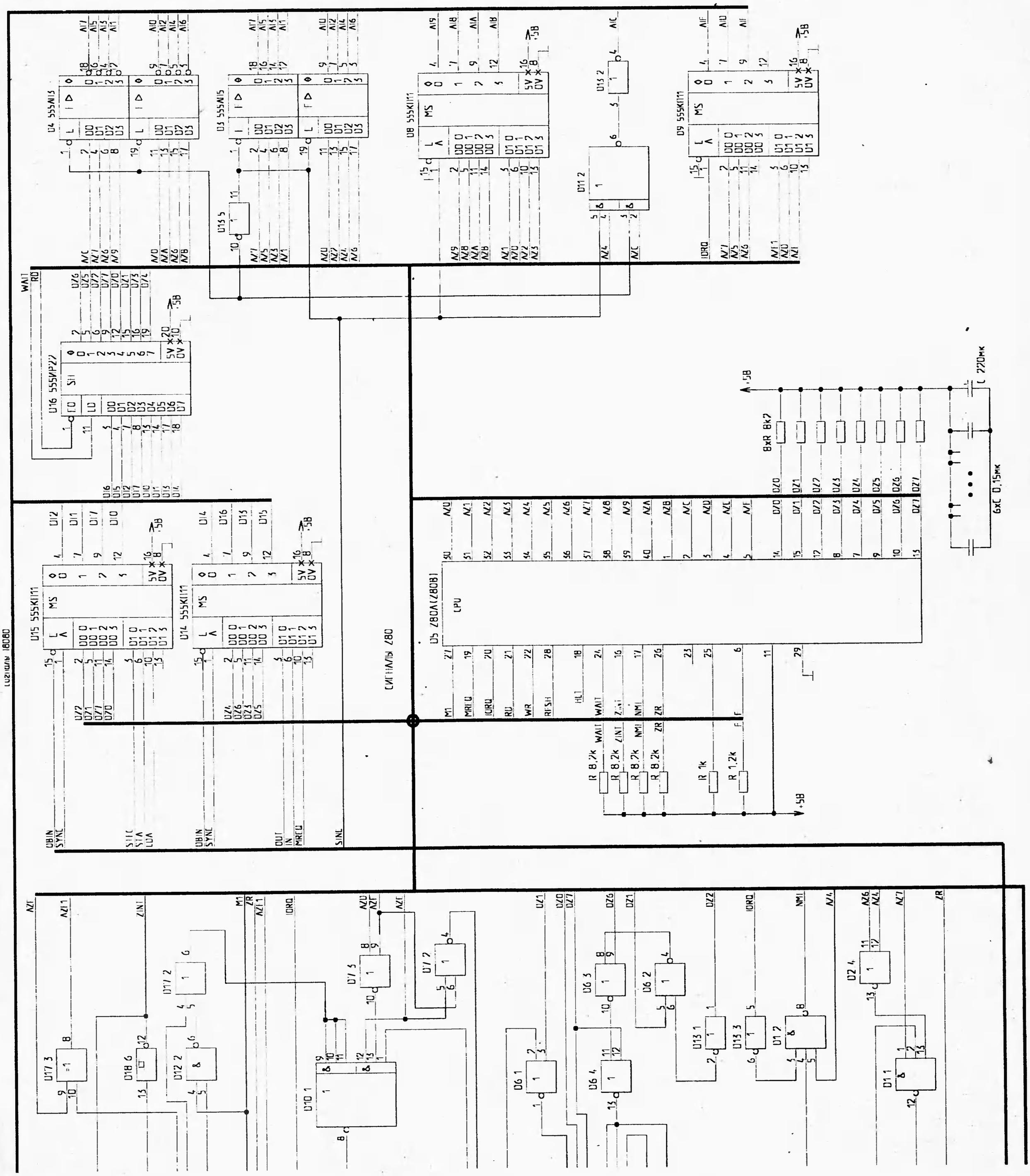


D1 555/ЛА
D2,D6,D7 555/ЛЕ
D10,D11 555/ЛР
D12 1533/ЛА3
D13 555/ЛН1
D17 555/ЛП5
D18 555ТМ2
D21 555ТМ1
D22 555/ЕЦ
D24 1533/ЛР13

такого режима связана с различием строения "синклеровского" и "векторовского" экранов.

В последнем случае адресации, при работе "синклеровских" программ, черно-белая графика заносится на экранную плоскость "Вектора" по координатам, соответствующим положению на "синклеровском" экране. Этот режим решает основную часть проблем, связанных с эмуляцией "синклеровских" программ.

На регистре D20 и элементах логики D1.1 и D2.4 собран



встроенный в адаптер порт вывода с адресом 80h. Порт управляет некоторыми режимами работы компьютера. После включения или аппаратного сброса компьютера все биты порта (сигналы P0...P7) устанавливаются в "0", что соответствует режиму работы "непеределанного" "Вектора". Кратко о назначениях битов порта:

1. Изменение адресного пространства (сигналы P0, P1) — включение "синклеровского" экрана (сигнал SYNC микросхемы D10.2), инверсия старшего бита адреса (посредством D17.3).

2. Формирование сигнала STEC (сигналы P2, P3, P5, P7). Соответствующий сигнал вырабатывается I8080 в операциях обращения к стеку (в мультиплексированном с данными виде) и используется в некоторых случаях работы внешнего ОЗУ для включения последнего. Адаптером этот сигнал формируется существенно более сложным образом, что позволяет использовать всю внешнюю память так же активно, как и внутреннюю.

3. Немаскируемое прерывание (активно при P4=1) позволяет оперативно вмешиваться в обращения к портам ввода-вывода при работе "не-векторовских" программ. Подпрограмма по адресу 66h должна идентифицировать чужеродное устройство, к которому идет обращение, и переадресовать его к "векторовским" устройствам.

4. Замедление выполнения некоторых команд Z80 до времени их выполнения на I8080 происходит при P6=0 благодаря триггерам D21 (для более полной совместимости в "непеределанным" "Вектором"). Сигналы битов порта P1 и P2 задерживаются регистром D19, тактируемым сигналом M1 основного цикла процессора Z80. Обратите внимание на то, что обычно применяемое мгновенное переключение ОЗУ таит в себе немало неприятностей: команда переключения (OUT...) должна находиться вне области, которая переключается. В противном случае продолжение программы, к которой переходит управление после OUT..., должно быть подготовлено в строго определенном месте в новом блоке ОЗУ. Реализованное применение задержки сигналов P1 и P2 на одну команду Z80 (обычно применяется команда PCHL — перехода) позволяет одновременно с изменением в строении ОЗУ сделать и переход на продолжение программы в новом ОЗУ.

Аппаратные возможности адаптера позволили создать программу ZX320 — эмулятор "Spectrum-48". Программа позволяет работать с синклеровскими программами без дополнительных переделок в компьютере: в цвете, со звуком, без замедления быстродействия. Обмен информацией происходит через "векторовский" дисковод или магнитофон в формате ПК "Sinclair".

Статья предоставлена В.Фироновым, 117335, Москва, а/я 101.

С.ГЛЕБОВ, П.САВЫГИН,
г.Минск, ШЮП, секция "Программирование" при кафедре ВМиП, ФИТУ, БГУИР.

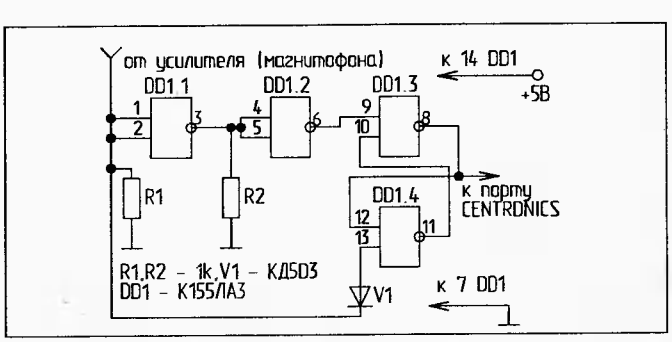
ВВОД ЗВУКОВОЙ ИНФОРМАЦИИ ЧЕРЕЗ ПОРТ CENTRONICS

Предлагаем аппаратно-программный комплекс, с помощью которого возможен ввод звуковой информации в PC через параллельный порт (CENTRONICS). Приведенные схема и программа представляют собой лишь краткий вариант, не оснащенный системами удаления шумов. Приведенную здесь программу можно использовать в качестве своеобразного пробника для проверки схемы и/или параллельного порта. Данный комплекс позволяет вставить в свою программу словесный комментарий, музыку и т.д.

Посредством схемы, представленной на рисунке, осуществляется преобразование звука в прямоугольный двухуровневый сигнал (лог."0" и лог."1"). Преобразованный сигнал поступает на порт CENTRONICS. Сигнал подается на любой из входных контактов (например на контакт 11, предназначенный для сигнализации занятости принтера). С помощью программы, приведенной ниже, осуществляется считывание состояния принтера (байт из порта 379h). Принятый уровень ("0" или "1" соответственно) записывается во 2-й бит порта 61h (управление диффузором динамика). Если из порта читается значение -5041, значит соединительный кабель отключен от принтера и ни на один из его входных контактов не подается "1". Для улучшения качества звука рекомендуется использовать только неинвертированные сигналы (BUSY, PE или SLCT). Если используется инвертированный сигнал (/ERROR или /ACK) необходимо внести соответствующие изменения в программу (инвертировать выводимый сигнал).

Рассмотрим пример демонстрационной программы:

```
/*---Подключение к программе модулей команд--*/
#include <stdio.h>
#include <dos.h>
#include <conio.h>
mair()
{
int i;
/*---Установка невидимого курсора и цветов---*/
textcolor(15);
textbackground(0);
_AH=1;
_CH=32;
geninterrupt(0x10);
clrscr();
puts("Демонстрационная программа по");
puts("качественному воспроизведению");
puts("музыки через PC-Speaker.");
puts("Подключите, пожалуйста, устройство");
puts("к LPT1, затем включите магнитофон");
puts("и по готовности нажмите любую");
puts("клавишу.");
getch();
/*---Воспроизведение-------------------------*/
while(!kbhit()){
i=inport(0x379);
if (i!=-5041) outport(0x61,2);
if (i==-5041) outportr(0x61,0x0);
)
)
```



С.РЮМИК,
250033. г.Чернигов-33. а/я 1772.

ТРИОФОНИЧЕСКИЕ ЭФФЕКТЫ В "ZX-SPECTRUM"

Музыкальный сопроцессор AY3-8912 (AY3-8910) в компьютере "ZX-Spectrum" позволяет создавать панорамное стереофоническое звучание. Слушатель может локализовать направление на кажущийся источник звука по всей ширине панорамы — от левой колонки стереоусилителя до правой.

Ввести объемность звучания помогает третий дополнительный канал. Система с тремя звуковыми каналами называется триофонической. Триофония обладает улучшенным звучанием по сравнению со стереофонией, в то же время более проста и дешева по сравнению с квадрафонией.

В принципе, эффекты, близкие к триофоническим, уже заложены в звучании некоторых игровых программ. Например, в играх DIZZI-4 (Code Masters, 1990), TARGET RENEGADE (Imagine. 1988), ROBOCOP-2 (Ocean, 1990) и т.д. шумовые эффекты, удары, выстрелы, звуки шагов воспроизводятся через моновыход BEEP, а сопровождающая музыка — через стереовыход музыкального сопроцессора.

Если слушатель (С) находится в рамках треугольника, как показано на рис.1, то систему вполне можно назвать псевдотриофонической.

"Псевдо" — так как музыкальный спектр и возможности моноканала BEEP (УНЧ2) гораздо беднее AY-каналов (УНЧ1, УНЧ3). Отсюда резкое различие в звучании сигналов, приходящих из разных направлений.

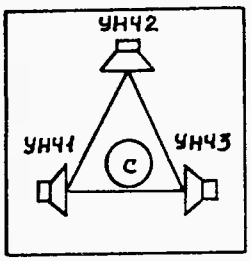


Рис. 1

Кроме того, в компьютерных программах для "ZX-Spectrum" чрезвычайно редко предусматриваются согласованные эффекты BEEP+AY. Приятным исключением из этого правила могут служить игры:

- CHASE H. Q. (Ocean, 1989), где звуки BEEP-ударника дополняют AY-музыку;
- RAW RECRUIT (Mastertronic, 1988), где, наоборот, полифоническая BEEP-имитация сочетается с AY-ударником;
- THE ARC OF YESOD (Odin, 1986) с оригинальной гибридной BEEP+AY музыкой.

Еще один вариант псевдотриофонической системы приведен на рис.2.

Выходные сигналы музыкального сопроцессора A, B, C поступают через коммутатор каналов на блок резистивных делителей и далее, через переходные конденсаторы C1...C4 и переключатель SA4 — на входы трех усилителей.

Переключатели SA1...SA3 позволяют получить шесть неповторяющихся вариантов перестановок каналов A, B, C. Тем самым образуются сигналы физической смены каналов A', B', C'.

Блок резистивных делителей можно использовать из прежней схемы подключения музыкального сопроцессора без изменения номиналов. На рис.2 — это 6 резисторов, другие схемы содержат 7 резисторов. Важно иметь точки выхода для левого (LEFT) и правого (RIGHT) каналов.

Выходные сигналы поступают на стереоусилитель (УНЧ1, УНЧ3) и моноусилитель (УНЧ2). Входное сопротивление УНЧ2 должно быть не менее 25...30 кОм во избежание увеличения переходных помех между стереоканалами.

Смысл третьего дополнительного канала на рис.2 — в воспроизведении разностной составляющей между левым и правым каналами.

Переключатель SA4 меняет полярность сигнала УНЧ2 на противоположную.

Колонки усилителей следует разместить как показано на рис.3.

Расположение громкоговорителей должно быть таким, чтобы слушатель находился в геометрическом центре образовавшегося треугольника.

Первоначально регулятор громкости тылового усилителя должен быть установлен в положение минимальной громкости. Затем необходимо отрегулировать стереобаланс, громкость и тембр основного стереоусилителя. Постепенно увеличи-

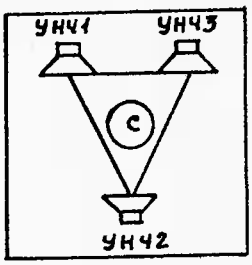
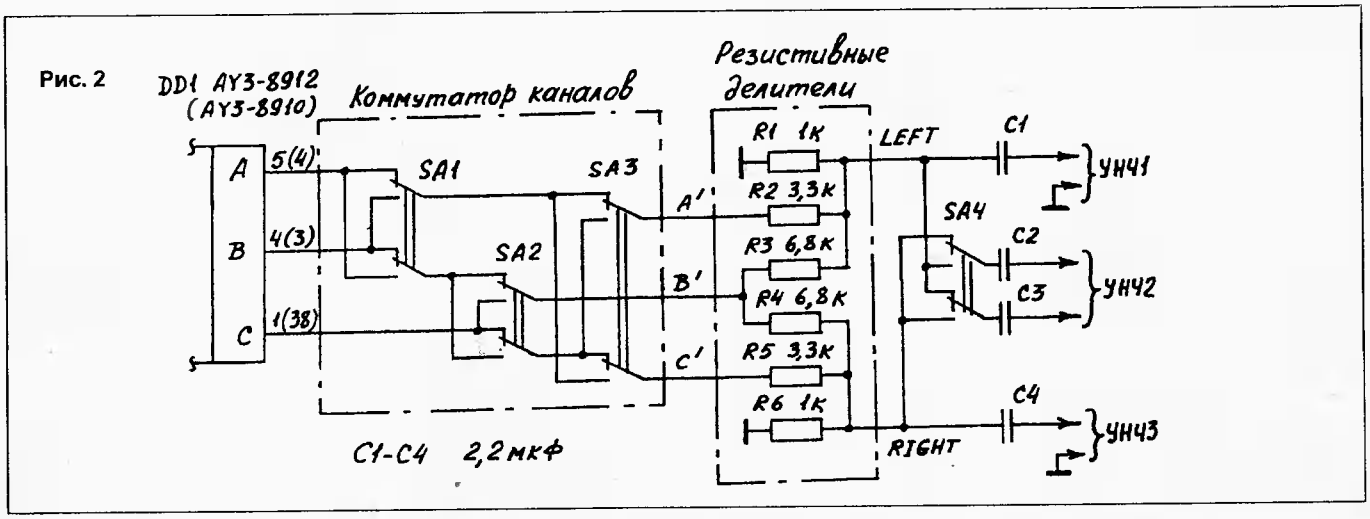
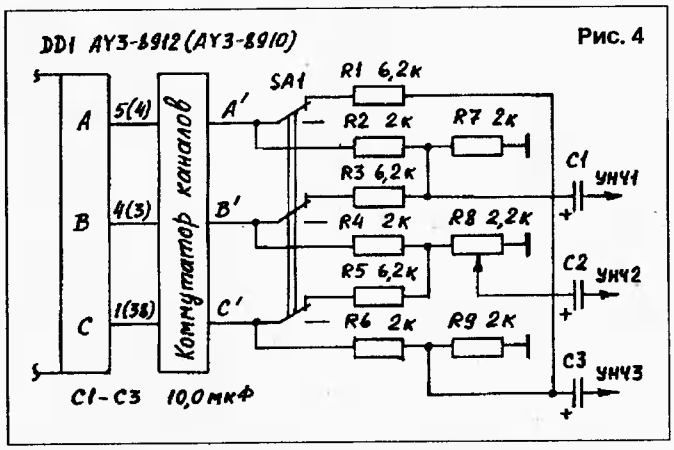


Рис. 3

Рис. 2





вая громкость тылового усилителя, можно заметить, что звуковой образ, который казался ранее "плоским" по фронту, начинает постепенно приближаться к слушателю.

Если подобрать определенные соотношения уровней громкости и поэкспериментировать с положениями переключателей SA1...SA4, звуки компьютерной музыки становятся как бы объемными и заполняют всю комнату. Появляется специфическая полнота и насыщенность звучания.

Конечно, тыловой канал не является в данной схеме чисто триофоническим, т.к. он полностью зависим от двух основных и лишь дополняет звуковую картину подобием эха.

И все же настоящая триофония без приставки "псевдо" в компьютере "ZX-Spectrum" вполне реальна, стоит только повнимательнее присмотреться к структуре музыкального сопроцессора.

Музыкальный сопроцессор AY3-8912 (AY3-8910) имеет 3 независимых канала A, B, C и как бы специально создан для триофонии. Необходимо изменить схему включения аналоговых выходов, как показано на рис.4, и подать сигналы УНЧ1, УНЧ2, УНЧ3 на акустические системы согласно рис.3.

Схема коммутатора каналов ничем не отличается от приведенной на рис.2. Переменный резистор R8 регулирует громкость тылового УНЧ2 и позволяет установить желаемый "триобаланс".

Переключатель SA1 служит для введения "естественности" триозвучания. Этот момент требует пояснения.

В нижнем по схеме (рис.4) положении переключателя SA1 сигналы УНЧ1...УНЧ3 воспроизводятся независимо друг от друга.

При прослушивании реальных AY-мелодий через полностью независимую трехканальную систему отмечается некоторая резкость звучания. Длительное ощущение дискретности источников звука не вписывается в реальную слуховую панораму, окружающую человека.

С подобной проблемой сталкиваются обладатели головных стереотелефонов при прослушивании стереопрограмм [1].

Как выход из положения используется введение перекрестных связей между каналами. В простейшем случае это резисторы R1, R3, R5 (рис.4), включаемые в схему в верхнем положении переключателя SA1.

Экспериментальные исследования показывают, что глубина перекрестных связей должна быть в пределах 6...10 дБ, т.е. должно выполняться соотношение R1/R6 = R3/R2 = R5/R4 = 3...5.

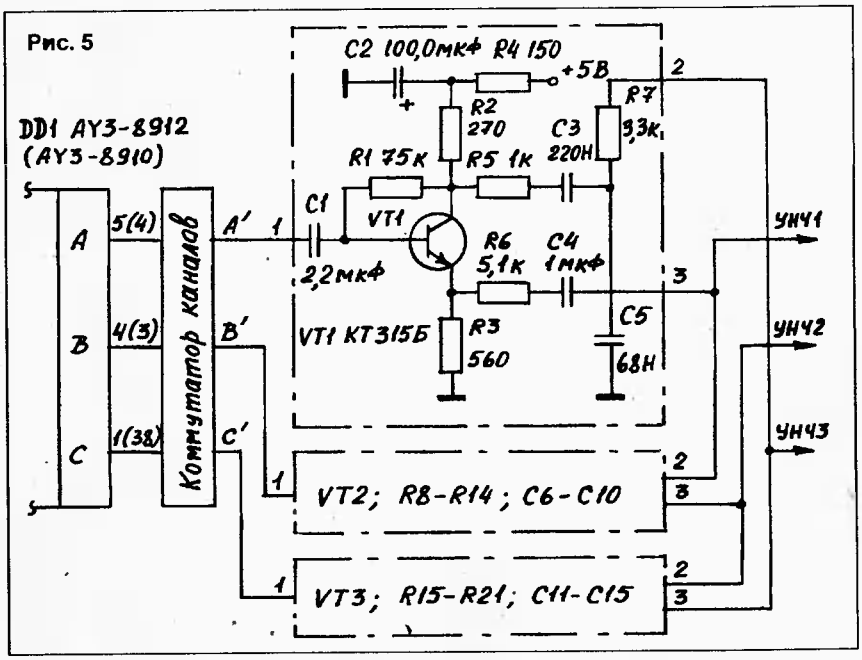
На слух становится заметным приближение звука к слушателю, появление своеобразной сглаженности.

Схемы рис.2 и 4 просты в изготовлении и могут быть опробованы в течение одного вечера. Если эффекты в принципе понравятся, на очереди переход к более сложной схеме, которая помогает ощутить новые нюансы в триозвучании.

Схема, приведенная на рис.5, вполне может быть названа "расширителем триобазы". Действительно, частотозависимые перекрестные связи между каналами создают эффект удаления источников звука.

Схема содержит три одинаковых каскада фазорасщепителей на транзисторах VT1...VT3. Сигналы A', B', C' поступают от коммутатора каналов, имеющего в своем составе 3 переключателя.

Теперь триозвучание будет ощущаться в большем по размерам пространстве. Особенно яркие краски возникают при идентичности и хорошем качестве усилителей УНЧ1, УНЧ3.



На рис.6 приведена универсальная схема, сочетающая в себе разные "триорежимы". Переключатель SA1 включает эффекты схемы рис.2, а переключатель SA2 — эффекты схем рис.4 и рис.5.

Программное обеспечение для триофонии изобретать не следует. Многочисленные стереофонические программы для музыкального сопроцессора чудесно будут звучать в триофоническом исполнении, не требуя какой-либо переделки.

Любой из существующих музыкальных AY-редакторов подходит для написания триофонических программ. Богатые возможности открывает редактор THE MARK TIME MUSIC BOX (Mark Time, 1986), обычно именуемый WHAM-128. Некоторые версии этой программы виртуозно используют трехканальный сопроцессор не только в компьютере "ZX-Spectrum-128", но и в "ZX-Spectrum-48" [2].

Объемные триофонические эффекты в музыкальных программах полностью совместимы со стереофонией.

Рис. 6

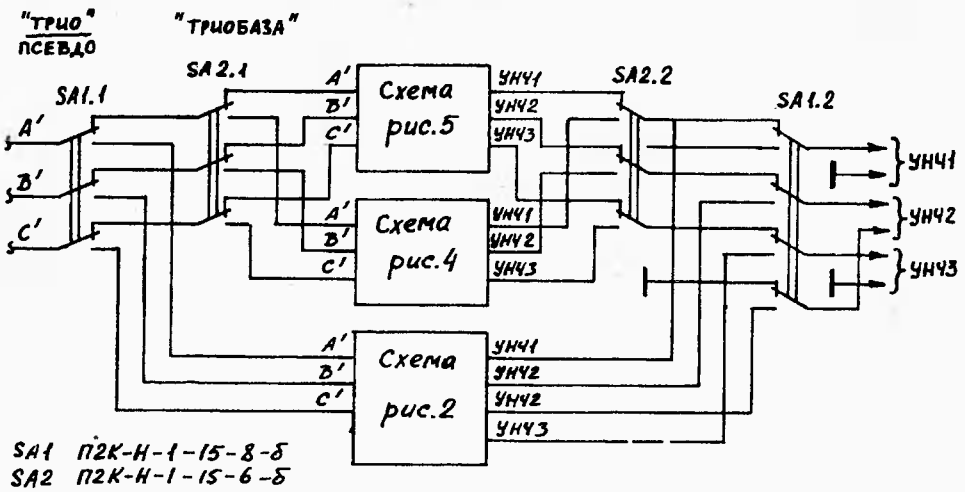


Табл. 1

Фирма	Музыкальные особенности
Code Masters	AY-музыка + BEEP-шаги, прыжки, эффекты; в одном из каналов — одновременно НЧ+ВЧ.
Firebird	Двухканальный звук из B, C; в A — звука нет.
Gremlin Graphics	ВЧ и НЧ — из каналов A и C или C и A.
Odin	Двухканальный звук, в канале B — эффекты; AY-музыка + BEEP-речь, эффекты.
Ocean	AY-музыка + BEEP-выстрелы, эффекты; согласованные эффекты BEEP+AY.
Quicksilva	НЧ — из канала B.
U.S. Gold	Унисонные эффекты по разным каналам.

Как пример можно привести Бейсик-программу (листинг 1) эффекта вращающегося звука — ROTOR SOUND.

Листинг 1.

```
10 LET a$="M56UX2100W5A&&)"
20 LET b$="U&A&)": LET c$="U&&A)"
30 PLAY a$, b$, c$
```

Триофонические эффекты удобно вводить в игры-имитаторы. Представьте, насколько выиграла бы программа типа CONTINENTAL CIRCUS (Taito/Virgin, 1989), если бы AY-звуки обгоняющих машин сначала доносились бы из тылового динамика и постепенно, по мере обгона, переходили бы в левый или правый канал!

Несколько замечаний о необходимости физической смены каналов через коммутаторы на схемах рис.2, рис.4...6.

Анализ фирменного музыкального AY-сопровождения показал отсутствие стандартов в закреплении участков спектра за определенными каналами. Низкие (НЧ), средние (СЧ) и высокие (ВЧ) частоты воспроизводятся достаточно случайно даже в играх одной и той же фирмы. Более того, встречается прием смены каналов звучания прямо по ходу мелодии, например в играх SWITCHBLADE (Gremlin Graphics, 1991), DEATH STALKER (Code Masters, 1988) и т.д.

В табл.1 приведены особенности звучания, замеченные в программах ведущих "Спектрумовских" фирм (разумеется, это не стопроцентные правила).

Физическая смена каналов значительно обогащает слуховую гамму восприятия мелодий и может рассматриваться как необходимая составная часть всех многоканальных систем.

Схемы рис.2, рис.4...6 могут быть выполнены в виде отдельной приставки или, если позволяет место, переключатели можно установить прямо в корпусе компьютера.

Качественное триозвучание можно услышать на качественной звукоусилительной аппаратуре. В идеальном варианте все три усилителя (или, по крайней мере, два фронтальных) должны иметь одинаковые параметры, соответствующие высшей или первой группе сложности. На практике в качестве тыловой применяют одну из разновидностей малогабаритных акустических систем.

Нельзя забывать о необходимости тщательной экранировки проводов, идущих к усилителям низкой частоты. В особенности это касается УНЧ2 в схемах рис.2 и рис.6.

Мощность резисторов в схемах — 0,125 Вт. Переключатели — кнопочные, типа П2К. Конденсаторы — типов КМ-6, К50-35. Разделительные входные и выходные конденсаторы могут иметь емкость от 1 до 10 мкФ.

К транзисторам VT1...VT3 особых требований не предъявляется. В отдельных случаях, возможно, придется подобрать номинал резистора между базой и коллектором, чтобы ток эмиттера составил 1,5...2 мА.

Любителям оригинальных решений можно порекомендовать установить тыловой динамик... прямо над головой. Подобный прием применяется в пентафонии — пятиканальной системе звучания.

Триофония является первым шагом, позволяющим ощутить объемные эффекты. Подкупает простота схемотехнических решений, возможность использования уже наработанного программного обеспечения.

Следующий этап улучшения качества звучания — переход к более сложной четырехканальной квадрафонической системе.

Литература

1. Васильев В.А. Зарубежные радиолюбительские конструкции. — М.: Радио и связь, 1982. — С.26-29.

2. Алексеев А.Г."WHAM" The Music Box//ZX-РЕВЮ. — 1994. — N 1. — С.19-47.

Ю.СТЕФАНОВИЧ,
213860, Беларусь,
Могилевская обл., г/п Глуск, ул.Интернациональная, 14.

РАБОТА С ЭКРАНОМ "ZX-SPECTRUM"

Хочу предложить на суд читателей несколько процедур на Ассемблере, из которых, как из кубиков, можно при желании сложить довольно-таки неплохую программу.

Процедуры рассчитаны на тех, кто уже хотя бы немного знает Ассемблер, ну а остальным придется учиться. На Бейсике далеко не уедешь.

При написании программ использовался двухпроходной Ассемблер GENS4.1.

ПРОЦЕДУРА СТИЛИЗАЦИИ ШРИФТА

```
10       ORG 40000
20       ENT $
30       LD HL,15616-256
40       LD DE,38000
50       LD (23606),DE
60       LD C,128
70 N1    LD B,4
80 N2    LD A,(HL)
90       LD (DE),A
100      INC HL
110      INC DE
120      DZNZ N2
130      LD B,4
140 N3   LD A,(HL)
150      RRA
160      OR (HL)
170      LD (DE),A
180      INC HL
190      INC DE
200      DJNZ N3
210      DEC C
220      JR NZ, N1
230      RET
```

Вызов: RANDOMIZE USR адрес.

Процедура переместимая и может располагаться в любом месте памяти.

В строках 30, 40 задается адрес расположения оригинального шрифта минус 256 и адрес расположения нового шрифта.

50: В системную переменную заносится новое значение.

60: Число символов.

70...120: Верхние 4 строки символа переносятся без изменений.

130...200: нижние 4 строки символа сдвигаются на 1 пиксел вправо и накладываются на старое значение.

210...220: уменьшение счетчика символов и если не конец, то возврат для следующего символа.

230: выход из процедуры.

Таким образом, у нового шрифта все 4 нижние линии символа оказываются утолщенными относительно верхних.

ПРОЦЕДУРА УТОЛЩЕНИЯ ФОНТА

```
10       ORG 40000
20       ENT $
30       LD HL,15616
40       LD DE,39000
50       LD BC,768
60 P1    LD A,(HL)
70       RRA
80       OR (HL)
100      LD (DE),A
110      INC HL
120      INC DE
130      DEC BC
140      LD A,B
150      OR C
160      JR NZ,P1
170      LD DE,39000
180      DEC D
190      LD (23606),DE
200      RET
```

Вызов: RANDOMIZE USR адрес.

Процедура переместима.

Описание:

В строках 30...50 задаются: адрес расположения оригинального фонта, адрес, по которому будет размещаться новый фонт, и число байт в фонте.

60-160: сдвиг каждой линии символа вправо на 1 пиксел и наложение на старое значение. Увеличение адресов на единицу и уменьшение счетчика байт фонта на единицу.

Затем проверка на достижение счетчиком нуля и если нет, то возврат для следующего байта.

170...190: в системную переменную заносится адрес нового фонта.

200: возврат из процедуры.

Если оригинальный фонт уже утолщен, эта процедура не рекомендуется, хотя в некоторых случаях приемлема.

ПРОЦЕДУРА ОЧИСТКИ ЭКРАНА С ЭФФЕКТОМ

```
10       ORG 40000
20       ENT $
30       LD DE,0
40       LD B,15
50 L1    PUSH BC
60       LD HL,16384
70 L2    LD A,(DE)
80       AND (HL)
90       LD(HL),A
100      INC HL
110      INC DE
120      LD A,H
130      CP 88
140      JR NZ,L2
150      EX DE,HL
160      LD BC,6000
170      AND A
180      SBC HL,BC
190      EX DE,HL
200      POP BC
210      DJNX L1
220      RET
```

Вызов: RANDOMIZE USR адрес.

Программа переместима в памяти.

Описание:

30: в регистровую пару помещается адрес начала "таблицы" случайных значений, в данном случае использовано ПЗУ.

40: задается число проходов для полного стирания экрана. Рекомендуется значение в диапазоне чисел от 10 до 20.

50: запомним на стеке этот счетчик.

60: адрес начала экранной памяти.

70...110: берем байт из таблицы и совмещаем его с байтом экрана по команде AND. В результате сбрасываются заданные биты и в строке 90 посылаются обратно на экран.

Далее увеличиваем счетчики адресов на единицу.

120...140: проверка конца экрана по старшему байту адреса и возврат для следующего байта, если не конец.

150...190: смещение адреса из "таблицы" для более случайного стирания.

200...210: восстановление счетчика проходов и если не 0, то возврат для повтора.

220: выход из процедуры.

С.БИРЮКОВ,
659322. Россия. Алтайский край.
г.Бийск. ул.Радищева. 8 — 61.

ДОПОЛНИТЕЛЬНЫЕ КЛАВИШИ В "ZX-SPECTRUM"

Часто на компьютерах типа "Spectrum" рядом с основной клавиатурой устанавливают дополнительные кнопки управления курсором. В зависимости от подключения это может быть "Sinclair" или "Cursor" джойстик.

Собрав на клавиатуре предлагаемую схему (рис.1), можно, в зависимости от положения переключателя S, иметь тот или иной тип джойстика. Преимущество схемы особенно заметно при работе с различными дисковыми загрузчиками и сервисными программами.

Владельцам компьютера "Ленинград-1" предлагаю установить тумблер "WAIT" (рис.2). В этом случае разрывается цепь, идущая к выводу 1 триггера D9.1 (К555ТМ2), и вводятся два элемента — резистор и выключатель. При замыкании тумблера S компьютер выполняет циклы ожидания, а вы при этом можете заниматься чем угодно (думать, рассматривать картинку на экране и т.д.).

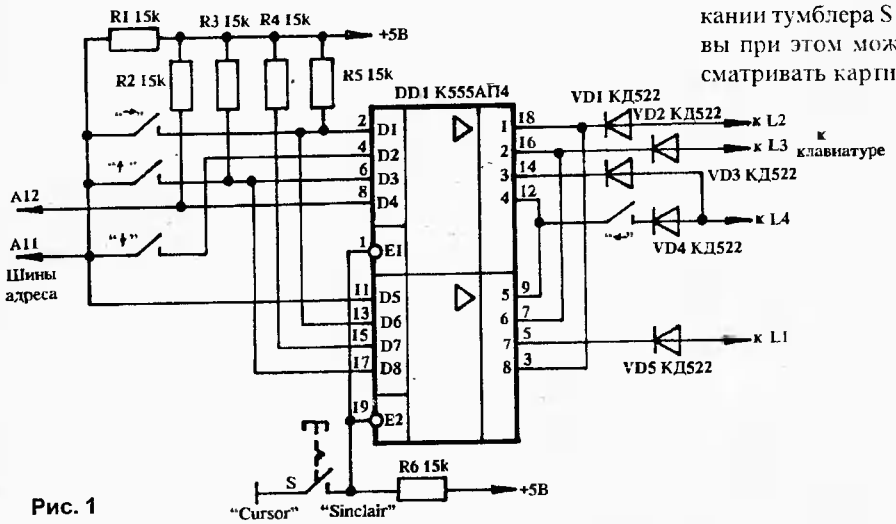


Рис. 1

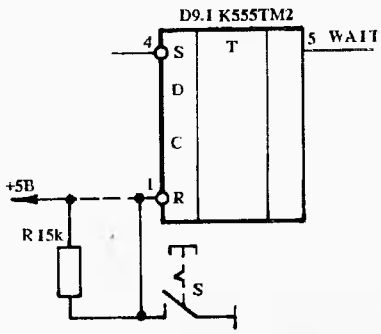


Рис. 2

А.ЛЕВОНЮК,
225850. Брестская обл.,
Дрогичинский р-н,
г.п.Антополь, до востребования.

КАК ОСТАНОВИТЬ ПРОГРАММУ?

Во многих программах, написанных для "ZX Spectrum", особенно в фирменных версиях, возникают трудности при остановке основного загрузчика, написанного на Бейсике, т.к. он записан с помощью команды SAVE "..." LINE... и после загрузки стартует со строки, указанной в LINE. Это было бы просто, если бы не блокировка клавиши <BREAK> и зашита от MERGE " ". Так что при выполнении команды MERGE " " и загрузке нужного вам блока происходит зависание системы, либо выполняется RESET.

Преодолеть эту трудность можно разными способами. Можно, например, изменить номер строки старта программы непосредственно в заголовке файла. Можно также использовать директиву "R", имеющуюся в копировщике "COPY 86/M". Но наиболее удобный способ — использование программы "LOAD/MERGE" в машинных кодах. Вот текст этой программы на Бейсике:

```
10 FOR N=60000 TO 60025: READ A: POKE N,A: NEXT N
20 RANDOMIZE USR 60000
30 DATA 1, 34, 0, 247, 213, 221, 225, 253, 54, 58,
1, 221, 54, 1, 255, 205, 29, 7, 42, 66, 92, 34,
69, 92, 207, 255.
```

Текст машинных кодов "LOAD/MERGE" на Ассемблере для желающих разобраться в ее работе:

```
20 ; LOAD/MERGE
30 ORG 60000
40 LD EC, 34
50 RST 48
60 PUSH DE
70 POP IX
80 LD (IY+58),1
90 LD(IX+1),255
100 CALL 1821
110 LD HL,(23618)
120 LD (23621),HL
130 RST 8
```

После набора программы на Бейсике или загрузки ее с кассеты нужно выполнить RUN <ENTER>. После этого — загрузить нужную вам программу (на Бейсике). Если не было ошибки, после окончания загрузки появляется сообщение OK, свидетельствующее о том, что загрузка прошла нормально. При просмотре листинга вы не обнаруживаете программы "LOAD/MERGE". На ее месте находиться листинг загруженной вами программы. Теперь вы можете внести в нее любые изменения. Если вы закончили работу с этой программой и хотите перейти к другой, вам не обязательно снова набирать весь листинг "LOAD/MERGE". Нужно лишь выполнить команду RANDOMIZE USR 60000 <ENTER>.

С.ВЕРЕМЕЕНКО,
620040, г.Екатеринбург, ул.Ст.Большевиков, 24 — 41,
тел.3432-340-124.

DENDY ПОД МИКРОСКОПОМ

(Окончание. Начало в NN2-3/96)

ВИДЕОПРОЦЕССОР

С видеопроцессором дела обстоят несколько хуже. Мне удалось определить, что он управляется по 8 адресам. В DENDY для него задействованы адреса 2000h...2007h. Функции некоторых регистров удалось выяснить, анализируя игры. Практически все игрушки начинаются процедурой инициализации, которая выглядит приблизительно так:

```
CLD             ; сброс десятичного режима
SEI             ; запрет прерываний
LDA   #00h      ; очистка аккумулятора
STA   2000h
STA   2001h
LOOP  LDA 2002h ; ожидание установки
BPL   LOOP      ; разряда D7 в 0
```

Кроме того, известно, что координаты позиции вывода на экран определяются регистром 2005h.

CPU через видеопроцессор может записывать и считывать информацию из памяти видеопроцессора. Адрес устанавливается путем записи двух байтов (вначале старший, потом младший) по адресу 2006h. Чтение или запись осуществляется путем передачи массива по адресу 2007h. Причем, при чтении первый считанный байт недостоверен. Только второй и последующие считываются из заданного адреса и могут быть использованы.

Видеопроцессор формирует полный видеосигнал полностью самостоятельно, не нуждаясь в контроле CPU. Обращения CPU происходят только при необходимости сменить картинку. Но взаимодействие с видеопамятью идет непрерывно. Видеопроцессор может формировать немаскируемые прерывания для CPU. В некоторых случаях их генерация прекращается, но это никак не сказывается на картинке. Это пока все, что я могу сообщить о видеопроцессоре.

Цоколевка видеопроцессора приведена на рис.3.

В большинстве DENDY разъем картриджа предоставляет единственную возможность подключиться к ней, т.к. они однокристальные и добраться до внутренних узлов невозможно. Поэтому сигналам, выведенным на разъем, необходимо уделить самое пристальное внимание. Цоколевка разъема приведена на рис.4.

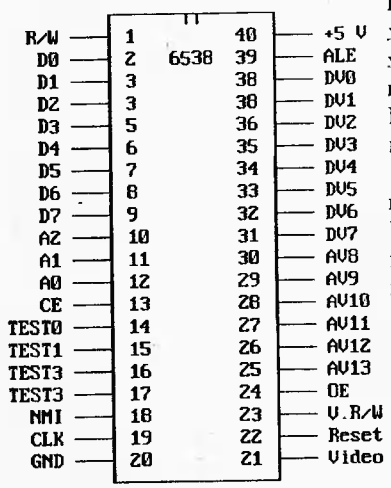


Рис. 3

Разъем картриджа четко делится на две части. Контакты с 1 по 15 идут к CPU, а с 17 по 30 — к видеопроцессору.

Рассмотрим вначале группу контактов, через которую CPU связан с картриджем. Мы видим обычную системную шину микропроцессора. Контакты B6...B13 образуют шину данных, A2...A13 и B3...B5 — шину адреса. Адрес A15 на разъем не выведен, но сигнал выбора C8 (B14) появляется только при A15=1.

Сигнал WR (A14) устанавливается в "0" в цикле записи и равен "1" в цикле считывания. На контакт A15 выведен вход маскируемого прерывания INT, а на контакт B2 — синхросигнал микропроцессора CLK. Контакты A1 и A16 — земля. B1 — питание +5 В.

Сигналы CS, WR, INT инверсные, активен нулевой уровень, как и в большинстве микропроцессоров, а у сигнала CLK активен единичный уровень.

Все шины небуферизованы, поэтому допускается нагрузка не более 1 входа TTL. Сигнал WR шире сигнала CS (рис.5). Это очень удобно для подключения памяти, но вызывает затруднения, если вы хотите подключить порт 580ВВ55. К сожалению, на разъем не выведен сигнал сброса.

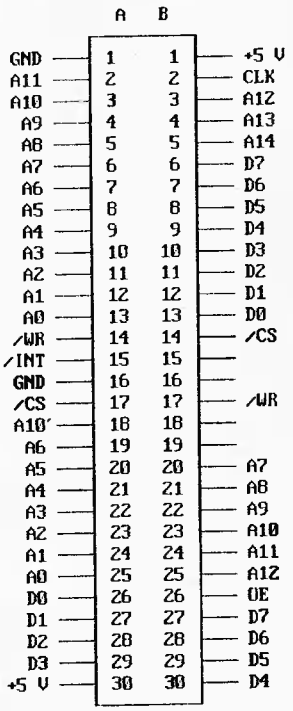


Рис. 4

При нажатии кнопки "Сброс" все шины переходят в третье (оборванное) состояние.

Если вы хотите подключить к DENDY какое-то свое устройство, нужно учитывать, что хотя у большинства DENDY сетевые адаптеры на 650 мА, более чем на 50 мА нагружать ее источник +5 В нельзя.

Это связано с тем, что ее внутренний стабилизатор слаб и еле тянет те 180...220 мА, которые потребляет сама DENDY. Однако, вполне возможно подключить к адаптеру дополнительный стабилизатор, например типа 142ЕН5А, с которого можно снять 200...300 мА без вреда для DENDY.

Видеопроцессор имеет шину, очень похожую на шину CPU. Также есть шина данных (A26...A29, B27...B30), шина адреса (A19...A25 и B20...B25), CS (B26) и WR(B17). Кроме того, есть линия A10', назначение которой рассмотрим чуть позже, когда будем разглядывать картридж.

Контакты B15, B16 и B18, B19 на большинстве картриджей замкнуты между собой. Через B15, B16 проходит сигнал звукового сопровождения. Блокировка звука, возможно, предусмотрена чтобы избежать "рычания" телевизора, если кому-то взбредет в голову включить DENDY без картриджа.

Функция B18, B19 сложнее. Если для работы программы достаточно внутренней видеопамяти, то изменив эту перемычку, можно подключить встроенное ОЗУ на адреса видеопроцессора 0000h...07FFh. В этом случае видеоПЗУ в картридже может отсутствовать. Правда, мне таких картриджей пока не попадалось.

УСТРОЙСТВО КАРТРИДЖА

Видимо, вам уже приходилось открывать картриджи для DENDY.

В них установлена небольшая печатная плата с торцевым печатным разъемом. На плате установлено от 2 до 5-6 микросхем, часто бескорпусных, залитых компундом. С корпусными микросхемами более ли менее ясно, но как определить тип и функции бескорпусных?

Большинство бескорпусных микросхем представляют собой ПЗУ емкостью от 8 до 128 Кб. Иногда возле такой микросхе-

мы есть надписи типа 64k, 256k, 1 M. Необходимо помнить, что емкость приведена в битах. Чтобы определить емкость в килобайтах, это число нужно разделить на 8. У микросхем ПЗУ как правило 28 выводов. Иногда один из выводов кристалла не выводится наружу и из-под компаунда выходят 27 ножек. Дополнительным признаком ПЗУ можно считать то, что на него поданы все разряды шины данных и все, или большая часть, адресных линий. Цоколевка ПЗУ приведена на рис.6. Вначале прозвонкой находим выводы питания и земли, затем убеждаемся в соответствии линий шины данных и адреса.

Кроме ПЗУ в состав картриджа может входить статическое КМОП ОЗУ емкостью 2, 8 или 32 Кб. Такие микросхемы можно отличить по тому, что на них заводится сигнал WR, кроме того, они, как правило, корпусные.

Третья разновидность микросхем, применяемых в картриджах — заказные ПЛМ. Они всегда бескорпусные с количеством выводов от 20 до 40. Отличительной особенностью ПЛМ является нерегулярное ее подключение ко всем линиям микропроцессора. Часто на ПЛМ бывают заведены сигналы CLK и INT, которые никогда не подключаются к ПЗУ и ОЗУ. Микросхемы ПЛМ наиболее трудны для анализа. Самое печальное, что невозможно дать какую-то общую рекомендацию. Слишком разнообразны как сами ПЛМ, так и схемы их подключения. Правда, в большей части картриджей их, к счастью, нет.

РАЗНОВИДНОСТИ КАРТРИДЖЕЙ

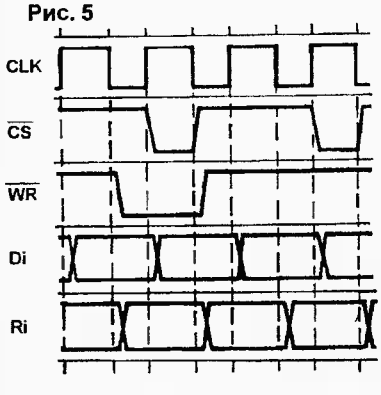
В простейшем случае в картридже установлены две микросхемы — ПЗУ программ, подключенное к шинам CPU, и ПЗУ картинок, подключенное к видеопроцессору.

В этом случае емкость ПЗУ программ не превышает 32 Кб, а емкость видеоПЗУ — 8 Кб.

Это самый приятный с точки зрения взлома вариант. Достаточно прочитать такой картридж — и все проблемы решены.

Гораздо чаще попадаются картриджи, в которых кроме ПЗУ установлен регистр банков, как правило, на микросхеме SN74ALS161 (аналог нашей 1533ИЕ10).

Рис. 5



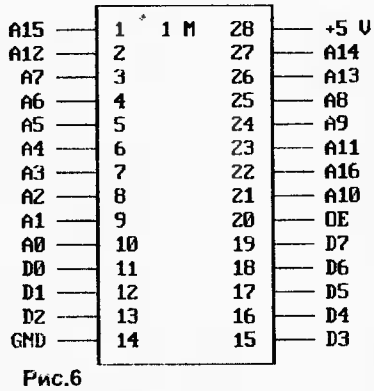


Рис.6

Хотя вообще-то это синхронный двоичный счетчик, в картриджах DENDY он используется как регистр с параллельной загрузкой. Его выходы подключаются к старшим адресам ПЗУ программ и часто — к видеоПЗУ. Входы регистра могут быть подключены как к шине данных, так и к адресной шине CPU. С точки зрения анализа и взлома особой разницы нет, но при считывании появляются некоторые нюансы. Запись в регистр производится по любому адресу в диапазоне 8000h...0FFFFh.

Кроме регистра, могут присутствовать микросхемы мелкой логики, функции которых просты и легко определяются после разрисовки схемы картриджа. Банки ПЗУ программ обычно имеют емкость 32 Кб или 16 Кб. Это зависит от того, заведена ли на ПЗУ адресная шина A14. Банки видеоПЗУ обычно имеют емкость 8 Кб, а в одном случае я обнаружил ОЗУ в 32 Кб.

К CPU ОЗУ, в принципе, подключить можно и вроде бы даже целесообразно, но таких картриджей я не видел. Если они и есть, то очень редки.

И, наконец, последняя и самая сложная разновидность.

Я имею в виду картриджи, содержащие ПЛМ (это не совсем точное название, т.к. функции заказных микросхем в картриджах DENDY посложнее простых логических матриц).

Эти картриджи определяются с одного взгляда, и каждый раз когда я с ними сталкиваюсь, у меня портится настроение. Практически невозможно считать такой картридж с первого раза. Приходится после считывания одного банка путем анализа программы, записанной в нем, определять условия для переключения на следующий. Поэтому, чтобы достоверно считать такой картридж, приходится делать до 8, а иногда и более попыток. Кроме того, иногда для переключения банков формируется прерывание, а сама процедура переключения находится в подпрограмме обработки прерывания. Если две предыдущих разновидности функциональны и их структура логически оправдана, в третьей отчетливо просматривается стремление "засекретить" картридж.

Есть еще одна особенность, которой картриджи отличаются друг от друга. Речь идет о линии A10' (контакт разъема A18).

В большинстве картриджей эта линия соединена с A10 (контакт B23). При этом встроенное видеоОЗУ занимает адреса видеопроцессора 2000h...27FFh и отображается, вследствие неполной адресации, на адреса 2800h...2FFFh, 3000h...37FFh и 3800h...3FFFh.

Линия A10' представляет собой адресный вход A10 встроенного видеоОЗУ и иногда подключается к шинам A11 (B24) или A12 (B25) видеопроцессора. При этом меняется включение встроенного видеоОЗУ в адресное пространство видеопроцессора. Подробно рассматривать эти изменения нет особого смысла. Однако в некоторых случаях вариант подключения линии A10' нужно иметь в виду.

СЧИТЫВАНИЕ КАРТРИДЖЕЙ

Для того чтобы прочитать картридж, нужно в первую очередь определить его тип и структуру. Кроме того, нужно иметь компьютер с дополнительным устройством типа программатора РПЗУ. Тип компьютера особого значения не имеет. Очень хорош "ZX-Spectrum", но годится даже какая-нибудь "Микроша" или IBM PC/AT-486. Очень желательно иметь дисковод.

Картридж первого типа, не имеющий регистра банков, считывается очень просто. Достаточно через переходник, изготовленный из разъема "дохлой" DENDY или слота IBM-XT, подключить его к устройству считывания и считать.

Считывание картриджей второго типа осложняется необходимостью переключать банки ПЗУ. Если банки переключаются по шине данных, перед считыванием нужно записать по одному из адресов 8000h...0FFFFh нужное слово. Если же переключение происходит по шине адреса (обычно задействованы младшие адреса), то слово может быть любым, но задействованные разряды адреса следует установить соответственно банку.

Есть одна тонкость. Дело в том, что во время записи в регистр банков ПЗУ картриджа включается на чтение. Если байт, который вы подаете на шину данных картриджа, не совпадает с байтом, который записан по этому адресу в ПЗУ, происходит так называемый "конфликт на шине данных".

Если ваш программатор имеет достаточно мощные выходы, он конечно "передавит" довольно хилое ПЗУ картриджа. Но в этом случае трудно поручиться за сохранность картриджа. А если программатор достаточно "деликатный", регистр банков переключается в состояние, определяемое байтом из ПЗУ.

Единственный выход из этого положения заключается в том, что записываемый байт должен быть равен байту из ПЗУ. В случае переключения банков по шине адреса нужно вначале считать байт по требуемому адресу и записывать именно его. Если же переключение идет по шине данных, то вначале нужно найти в ПЗУ адрес, по которому содержится требуемый байт, и записать этот байт именно по этому адресу. Такой байт непременно найдется, потому что эта проблема стояла в свое время и перед программистом, готовившим картридж. Причем имейте в виду, что переключение банков видеоПЗУ также происходит при управлении по шинам CPU.

Если картридж содержит ОЗУ, то его считывать, разумеется, нет смысла.

Наибольшие трудности появляются при попытках считать картридж с ПЛМ. Если ваша квалификация достаточно высока, приведенной выше информации вам достаточно. Дать какую-то общую рекомендацию невозможно. Каждый картридж такого типа требует индивидуального подхода.

АНАЛИЗ ПРОГРАММ

После получения копии картриджа на дискете или кассете можно приступить к ее анализу. Для этого желательно иметь кроссмонитор, работающий в кодах DENDY. Такой кроссмонитор написан мною для "ZX-Spectrum" с системой TR-DOS. Можно использовать также компьютер, система команд которого совпадает с DENDY'евской, например "COMMODORE" или "ПРАВЕЦ-2".

Методика анализа зависит от типа картриджа, количества игр на нем и поставленной цели. Картриджи первого типа, имеющие всего один банк, обычно содержат одну небольшую игру. При запуске сразу появляется заставка этой игры. Это наиболее простой и приятный случай. С такой игрой можно делать все что угодно: сделать ее бессмертной, русифицировать, практически без всякой адаптации перевести на кассету или дискету и т.д. Поскольку объем ПЗУ такого картриджа невелик, можно даже запрограммировать РПЗУ и сделать самодельный картридж.

Сложнее работать с картриджами второго типа. Первая задача — определить ведущий банк, в котором расположено начало программы. Поскольку при включении регистр банков может произвольно установиться в любое состояние, программист должен позаботиться о том, чтобы программа правильно началась из любого банка. Поэтому из любого ведомого банка есть переход в ведущий. Но здесь есть тонкость. Дело в том, что программа, переключив банк, продолжит свое выполнение уже в другом. Решается эта задача двумя различными способами. Иногда во всех банках кроме ведущего по одним и тем же адресам располагается фрагмент типа

```
LOOP LDA  #03h     ; номер ведущего банка 03h
     STA  LOOP+1   ; запись по адресу, в котором
                   ; уже содержится байт 03h
                   ; сопровождается установкой
                   ; банка 03h
```

И после этого фрагмента идет либо совершенно другая программа, либо просто "мусор", которого, кстати, в китайских картриджах более чем достаточно. Иногда даже обрывки ассемблерного текста попадаются.

А в ведущем банке есть продолжение этой программы. Дальше идет инициализация, установка стека и переменных и выход в меню.

Этот способ прост, но имеет существенный недостаток. Для его использования нужно, чтобы во всех банках было свободное место на одних и тех же адресах. А поскольку авторы программ не подозревали, что фирма "СТИБРЕР" запихнет их программы в картриджи, они не озаботились созданием удобств для этой процедуры. Поэтому чаще используется другой вариант.

Процедура включения ведущего банка переносится из ПЗУ картриджа в ОЗУ, например по адресу 0400h, и управление передается в ОЗУ. В разных банках он может располагаться в разных адресах, но переносится в одно и то же место ОЗУ. Из ОЗУ включается ведущий банк и после этого дается безусловный переход с индексной адресацией на вектор START.

Как правило, в картридже второго типа бывает от одной сложной до 5...6 простых игр. Иногда говорят о 20, 30, даже 100 и более игр, но это неверно. Все эти сумасшедшие цифры получаются нехитрым мошенничеством. Берется одна и та же игра, меняются начальные установки, чтобы сразу выйти на другой уровень или наделить героя новыми свойствами, и все готово. "Синклеристы" знают, что такое POKE. Так вот, механизм "создания" десятков, а то и сотен "новых" игр именно таков. Если в картридже не одна игра, то их желательно разделить. Как правило, простая игра размещается в одном банке и во время своей работы банки не переключает. Это касается и банков видеопамяти. Таким образом, для разделения игр нужно определить, в каком банке какая игра и найти адреса их запуска.

Бывают тонкости, но в основном это так. Проще всего это сделать, если найти подпрограмму, формирующую меню. Для ее поиска можно использовать текст меню, который никогда не кодируется и легко обнаруживается любым монитором. Этого нельзя сказать о тексте игр. Часто он бывает закодирован. Правда, алгоритм кодировки несложен и с ним без труда справляется даже одна из моих первых программ для раскодирования — "СЕРВИС-2", которая получила широкое распространение.

Работа по анализу программы DENDY проста ("синклеровские" программы гораздо более "закручены"), но очень нудная, из-за того что пока нет системных программ такого качества, как для "ZX-Spectrum".

ЗАКЛЮЧЕНИЕ

Как вы, видимо, убедились, легенда о полной закрытости и "секретности" DENDY не имеет ничего общего с реальностью. Конечно, совершенствовать DENDY, как "ZX-Spectrum" (от "Балтика" и "Ленинграда" к "ZX-Scorpion" и "ПРОФИ") невозможно. Но абсолютно ничего не мешает делать для нее дополнительные внешние устройства. Подключение даже кассетного магнитофона уже открывает путь для творчества. "Спектрумисты" дружно презирают DENDY. А собственно говоря, почему? Графика у DENDY лучше, а звук не хуже, чем у муз-процессора "ZX-Spectrum". Не потому ли, что "зуб неймет"?

Теперь, когда открывается возможность самим писать программы для DENDY, не пора ли повернуться к ней лицом и рассмотреть как следует? Ведь достаточно написать для DENDY буквально десяток системных программ, которые можно загружать с внешнего носителя или даже разместить в ПЗУ, и "ZX-Spectrum" лишится своего решающего преимущества — открытости. Помню, когда я впервые столкнулся с "ZX-Spectrum", то воспринял его как неприличный анекдот на компьютерную тему. Какая-то 8-битная игрушка с мизерной памятью и несуразной консолью. Оценил его я позже, и сейчас считаю, что это одна из самых удачных машин. DENDY я тоже занялся с некоторым предубеждением, а когда познакомился с ее процессором, вообще чуть не бросил. Но сейчас мое мнение уже начало меняться. Не так уж она и плоха. Во всяком случае, попробовать стоит. Кто со мной?

В.ЕРМОЛЕНКО (EW1OM),
г.Минск.

МОДЕМ БЕЗ ПРОБЛЕМ

(Продолжение. Начало в N3/96)

Работа в режиме MNP

Протокол MNP (Microcom Networking Protocol) служит для полнодуплексной безошибочной связи по телефонным линиям. Протокол детектирует и исправляет ошибки, возникающие из-за шумов и искажений сигнала. MNP работает при связи на 2400 bps и выше. Модемы 14400 bps поддерживают MNP класса 2...5:

- MNP2 использует асинхронный байт-ориентированный формат со стартовым и стоповым битами и полный дуплекс (одновременная передача в обе стороны);
- MNP3 использует синхронный бит-ориентированный формат, причем биты обрамления удалены. Это увеличивает пропускную способность примерно на 20%;
- MNP4 использует более эффективный метод обрамления и допускает передачу больших блоков данных. Размер блока данных подстраивается в зависимости от качества телефонных линий (меньше для худших линий), чтобы уменьшить количество перезапросов данных;
- MNP5 включает все свойства MNP4, а также сжатие данных, что значительно увеличивает пропускную способность. Если MNP5 запрещен, модем по умолчанию работает по MNP4. Сигналы MNP5 и MNP4 несовместимы.

После соединения в режиме MNP модем выполняет те же функции, что и при нормальном соединении, за исключением нескольких перечисленных ниже отличий.

Восстановление синхронизации (retrain). При высокой скорости передачи модем может потерять синхронизацию со входным сигналом. "Retrain" — это процедура подстройки, которая выполняется, когда один из модемов обнаруживает, что целостность данных находится под угрозой. На слух это напоминает процесс вхождения в связь (какое-то время передается немодулированная несущая, затем происходит обмен на 2400 bps и переход на более высокую скорость).

Прием данных при переходе в командный режим. Если модем при MNP-связи переходит в командный режим по команде "+++", он продолжает принимать и подтверждать данные. Данные накапливаются в модеме, пока он не будет переведен в режим "on-line", после чего они пересылаются в терминал (компьютер). В обычном режиме связи все данные теряются как только модем переходит в командный режим.

Сжатие данных (MNP5). Если в режиме MNP разрешено сжатие данных, модем сжимает данные в "токены" (кодовые комбинации) перед передачей удаленному модему и восстанавливает их при приеме.

Программируемый таймер бездействия. Если данные не принимаются и не передаются, модем может ожидать в течение заданного времени перед тем как рассоединиться. В режиме MNP таймер ожидания устанавливается вновь, когда возобновляется прием или передача данных.

Блочный режим передачи данных. MNP может функционировать в блочном или потоковом режиме. В потоковом режиме MNP пересылает данные кадрами переменной длины в зависимости от времени между принимаемыми символами. В блочном режиме MNP посылает кадры фиксированной длины (64, 128, 192 или 256 символов). Чем больше размер кадра, тем меньше затраты времени на передачу служебной информации и выше пропускная способность канала. Однако при плохом качестве связи (когда перезапрос кадров для восстановления ошибок происходит часто) выгоднее иметь минимальный размер кадра. Подстройка размера кадра происходит автоматически, однако вы можете задать максимальный размер кадра для режима MNP. Ваша коммуникационная программа должна поддерживать использование блочного режима передачи.

Если установить связь в режиме MNP невозможно (например отвечающий модем не поддерживает этот протокол), модем переходит в один из двух стандартных асинхронных режимов: нормальное или прямое соединение. При *нормальном* соединении (со скоростным буферированием) данные могут пересылаться в терминал на скорости, отличающейся от физической скорости связи между модемами. В этом режиме допускается управление потоком с помощью специальных символов (XON и XOFF), которые соответственно разрешают и запрещают передачу данных в терминал. При *прямом* соединении, наоборот, данные передаются в терминал с той же скоростью, как и принимаются из линии, и символы управления потоком не распознаются. Прямой режим обычно вводится лишь для совместимости с очень старыми модемами.

Два слова о скоростном буферировании в режиме MNP. Терминал или компьютер, соединенный с модемом через последовательный порт (внутренние модемы содержат такой порт на плате модема), использует для передачи одного байта до 11 бит (с битом четности, стартовым и стоповым битом). Если скорость терминала установить равной физической скорости связи (Бод), то терминал не успевает принимать данные и, в итоге, уже при MNP класса 3 тормозит связь. MNP5 теоретически позволяет достичь 4-кратного выигрыша в объеме данных по сравнению с физической скоростью. Это означает, что всегда выгодно устанавливать скорость терминала выше физической скорости в линии и использовать скоростное буферирование.

Однако как многие модемы, так и многие коммуникационные программы автоматически (по умолчанию) поддерживают скорость порта равной скорости соединения. Правильное использование команд модема позволяет избежать этого. Иногда бывает достаточно заменить "скорость DCE" (связного оборудования) в сообщении CONNECT XXXX на "скорость DTE" (терминала), например командой ATW. Тогда модем сообщает коммуникационной программе только ту скорость, на которую терминал был настроен вначале, и программа не пытается перенастроить порт. Есть и команды, запрещающие модему подстраивать скорость через порт при изменении скорости связи (lock — "замкнуть" порт). При отсутствии подобных команд для этой же цели могут использоваться S-регистры.

Рассмотрим команды управления режимом MNP (некоторые из рассматриваемых здесь команд управляют общими режимами работы модема).

\An — максимальный размер блока MNP:
- n=0 64 символа;
- n=1 128 символов (по умолчанию);
- n=2 192 символа;
- n=3 256 символов.

\Bn — передать сигнал разрыва длительностью n x 100 мс.

Ниже покажем, как можно раздельно запрограммировать реакции модема на сигнал разрыва, поступающий от терминала и удаленного модема при связи в режиме MNP, нормальном и прямом соединении.

\Gn — программное управление потоком данных (XON/XOFF):
- n=0 выключить;
- n=1 включить.

\Jn — подстройка скорости последовательного порта в соответствии со скоростью связи между модемами:

n=0 запретить (lock);
n=1 разрешить.

\Kn — реакция на сигнал разрыва.

Возможные комбинации приведены в табл.3. Варианты действий модема в этой таблице обозначены цифрами:

1 — модем высылает сигнал разрыва удаленному модему;
2 — модем выдает сигнал разрыва в терминал;
3 — сигнал разрыва выдается последовательно с передаваемыми данными;
4 — модем очищает буферы принятых и передаваемых данных;
5 — модем переходит в командный режим.

\Ln — блочный режим MNP:

Табл. 3

Команда	Сигнал разрыва получен от:		
	терминала		удаленного модема
	(MNP или норм. режим)	(прямой режим)	(норм. режим)
\K0	5	1,5	2,4
\K1	1,4	1,5	2,4
\K2	5	1	2
\K3	1	1,5	2
\K4	5	1	2,3
\K5	1,3	1	2,3

n=0 выключен (используется потоковый режим);
n=1 включен.

\Nn — использование протокола коррекции ошибок.

Указывает, какой режим будет использоваться после установления соединения:

n=0 нормальный режим;
n=1 прямой режим;
n=2 RELIABLE — режим с коррекцией ошибок (V.42bis или MNP5), при невозможности — рассоединение;
n=3 AUTO-RELIABLE (при невозможности установить связь с коррекцией ошибок — нормальный режим);
n=4 коррекция ошибок только V.42bis (LAPM);
n=5 коррекция ошибок MNP5.

\Tn — таймер бездействия:

n=0...90 минут.

\Wn — работа с раздельными скоростями (V.23).

Позволяет использовать скорость передачи 75 bps при скорости приема 1200 bps.

n=0 запрещено;
n=1 разрешено.

***Hn** — скорость переговоров об MNP режиме линии:

n=0 на наивысшей доступной скорости (по умолчанию);
n=1 на 1200 bps.

Работа в режиме V.42/V.42bis

V.42 — это стандарт CCITT (МККТТ — Международный Комитет по Телеграфии и Телефонии), который поддерживает протоколы коррекции ошибок LAPM и MNP4 и поэтому совместим с модемами, использующими MNP4.

V.42bis — это стандарт сжатия данных, который требует протокола коррекции ошибок LAPM и при этом теоретически обеспечивает вдвое большую пропускную способность, чем стандарт MNP5. Как уже говорилось, более медленный MNP5 требует протокола сжатия данных MNP4 и поэтому он несовместим с V.42bis.

Эффективность сжатия данных на практике представляет гораздо более сложный вопрос. Она зависит как от характера данных, так и от используемого алгоритма, в чем убеждается каждый пользователь программ-архиваторов типа ARJ или PKZIP. В любом случае сжатие архиватором, анализирующим весь объем файла данных, происходит более эффективно, а передача архивированных (сжатых) данных под протоколом сжатия может происходить даже несколько медленнее, чем без использования такого протокола.

Вхождение в связь по V.42 предусматривает две фазы: детектирование (обмен символами, подтверждающими поддержку протоколов коррекции ошибок в обоих модемах) и переговоры о протоколе.

Если модемам не удалось договориться о протоколе V.42bis, модем может выполнить один из следующих вариантов:

- перейти к переговорам о режиме MNP;
- перейти к стандартному асинхронному режиму;
- повесить трубку.

Порядок действий определяется регистрами S36 и S48. Поскольку MNP не поддерживает фазу детектирования, для использования только режима MNP это должно быть задано принудительно.

Рассмотрим команды управления режимами V.42/V.42bis.

%An — устанавливает ASCII символ для запроса процедуры "retrain" (n=0...127).

%Cn — управление сжатием данных:

n=0 без сжатия;
n=1 разрешить использование MNP5;
n=2 разрешить использование V.42bis;
n=3 разрешить оба протокола сжатия.

%En — автоматический вызов процедуры "retrain":

n=0 запретить контроль качества линии и вызов "retrain";
n=1 разрешить контроль качества линии и вызов "retrain";
n=2 разрешить автоматический вызов "retrain" и автоматическое изменение скорости связи.

%Fn — выбор направления работы с раздельными скоростями, V.23 (если выбрано \W1):

n=0 передача 75 bps, прием 1200 bps;
n=1 передача 1200 bps, прием 75 bps.

%L — сообщить уровень принимаемого сигнала.

Диапазон измерения — от 0 до -43 дБм (децибел относительно мощности 1 мВт). Результат выводится тремя цифрами, например 009 — это -9 дБм, 010 — -10 дБм и т.д. Чтобы вычислить отношение сигнал/шум, следует из уровня сигнала вычесть уровень шума.

%Q — сообщить оценку качества линии.

Выводит старший байт расчетной оценки качества (значение от 000 до 255). Для нормального качества связи это значение находится в диапазоне 0...2 и с ухудшением связи увеличивается. При значении оценки 8 и более выполняется "retrain" (если разрешено командой %E1).

Указанные сервисные функции (измерение и т.д.) специфичны для конкретных моделей модемов. В вашем случае их может не быть вообще, либо они могут вызываться совершенно другой командой.

S-регистры модема

Как уже говорилось, все команды влияют на состояние S-регистров модема, определяющих его работу. Команду непосредственной модификации регистров ATS=n можно использовать, когда нужно изменить действие какой-либо команды или когда подходящая команда отсутствует. В стандартный набор Hayes входят регистры с 0 по 18, функции остальных регистров могут значительно отличаться для разных моделей. Лучше всего, изучив инструкцию к своему модему, поэкспериментировать с разными регистрами, записав их начальные установки. Имейте в виду, что терминальные программы могут при работе выдавать модему свои команды, так что содержимое регистров будет отличаться от того, которое вы установили заранее.

Как правило, содержимое регистров задается в десятичном виде в пределах байта (диапазон 0...255), если это не оговорено специально. Это может быть счетчик, значение кода ASCII или реальная величина в определенных единицах (доли секун-

ды и пр.) Величина параметра, приведенная ниже, рекомендована изготовителем модема (см. описание конкретной модели). Некоторые регистры не используются.

S0 — количество сигналов звонка перед ответом модема.
Величина: 000.

При S0=0 автоответ выключен, при S0>0 — включен (на внешнем модеме горит индикатор AA).

S1 — счетчик сигналов звонка (информационный параметр — только считывание).

Содержимое регистра увеличивается каждый раз, когда модем получает сигнал звонка из телефонной линии, и по истечении 8 с после ответа сбрасывается.

S2 — символ выхода.
Величина: 043 (ASCII "+").

Устанавливает символ, который используется для перехода в командный режим без разрыва соединения с удаленным модемом (см. команду "+++").

S3 — символ возврата каретки.
Диапазон: 0...127.
Величина: 013 (ASCII "CR" или Ctrl-M).

S4 — символ перевода строки.
Диапазон: 0...127.
Величина: 010 (ASCII "LF" или Ctrl-J).

S5 — символ "забой" (backspace).
Диапазон: 0...32, 127.
Величина: 008 (ASCII "BS" или Ctrl-H).

S6 — время ожидания ответа станции при наборе "вслепую".
Диапазон: 2...255.
Единицы: с.
Величина: 002.

Указанная пауза выдерживается после поднятия трубки перед набором первой цифры, вместо того чтобы анализировать ответ станции (DIALTONE — гудок).

S7 — ожидание сигнала несущей после набора.
Диапазон: 1...255.
Единицы: с.
Величина: 030 (иногда 050 и др.).

Определяет сразу два интервала: время ожидания ответа модема после набора номера и время ожидания ответа станции, если в строке набора имеется символ W. Иногда включает в себя время набора номера, которое при импульсном методе набора довольно значительно. Если ваш модем внезапно вешает трубку, не дождавшись ответа с другой стороны — не стесняйтесь увеличить это значение до 90 с и более. Нужное время ожидания (меньшее, чем S7) можно установить прямо в терминальной программе.

S8 — время паузы на символ "," в строке набора.
Единицы: с.
Величина: 002.

Если вашей "сонной" АТС не хватает совсем чуть-чуть, чтобы распознать набираемые цифры, ожидать 2 секунды каждый раз довольно утомительно. Установите S8=1 и вводите цифры номера через запятую. Если понадобится пауза 2 с, вы можете ввести 2 запятые подряд и т.д.

S9 — время реакции сигнала DCD на обнаружение несущей.
Диапазон: 1...255.
Единицы: 0,1 с.
Величина: 006 (0,6 с).

S10 — задержка разъединения после потери несущей.
Диапазон: 1...255.
Единицы: 0,1 с.
Величина: 014 (1,4 с).

Значения S9 и S10 важны для успешного поддержания связи при частых потерях несущей удаленного модема. Их значения взаимосвязаны. Так, если установить S9>S10, модем вообще не может войти в связь (вешает трубку до того как появляется сигнал обнаружения несущей). При S10=255 модем действует так, как если бы несущая всегда присутствовала. На практике рекомендуется устанавливать S10 в 2...4 раза больше, чем S9, однако конкретные значения лучше подобрать на месте. Например на очень плохих линиях при тихом сигнале может помочь вариант S9=30, S10=140.

S11 — длительность посылки при частотном наборе.
Единицы: 0,01 с.

S12 — время паузы до и после подачи последовательности выхода ("+++").
Единицы: 0,02 с.
Величина: 050 (1 мс).

S14 — биты управления функциями (здесь и далее указано состояние при значении бита 1 или два состояния — 1/0):
бит 0 — игнорируется;
бит 1 — эхо (ATE1);
бит 2 — вывод результата (ATQ1);
бит 3 — словесный результат (ATV1);
бит 4 — резерв;
бит 5 — метод набора (ATP/ATT);
бит 6 — резерв;
бит 7 — вызывной/ответный режим (ATD/ATA, R).

S16 — биты управления функциями тестирования.

S18 — таймер тестирования.
Единицы: с.
Величина: 000.

Устанавливается при выборе функций тестирования (команда &T).

S21 — биты управления функциями:
бит 0 — AT&J1;
бит 1 — резерв;
бит 2 — AT&R1;
биты 3, 4 — AT&D;
бит 5 — AT&C1;
бит 6 — AT&S1;
бит 7 — ATY1.

S22 — биты управления функциями:
биты 0, 1 — ATL;
биты 2, 3 — ATM;
биты 4, 5, 6 — ATX;
бит 7 — резерв.

S23 — биты управления функциями:
бит 0 — AT&T4/AT&T5;
биты 1, 2, 3 — скорость связи (0 — 300 bps, 1 — 600, 2 — 1200, 3 — 2400, 4 — 4800, 5 — 9600, 6 — 19200 bps);
биты 4, 5 — контроль по четности (0 — четность, 2 — нечетность, 3 — не используется);
биты 6, 7 — AT&G.

S24 — таймер бездействия для перехода в режим с пониженным энергопотреблением.
Единицы: с.
Величина: 000.

S25 — задержка реакции на изменение сигнала DTR.
Единицы: 0,01 с.
Величина: 005 (50 мс).

S26 — задержка изменения сигнала CTS после изменения сигнала RTS.
Единицы: 0,01 с.
Величина: 001 (10 мс).

S27 — биты управления функциями:
бит 2 — AT&L1;
биты 4, 5 — AT&X;
бит 6 — ATB1;

бит 7 — резерв.
S28 — биты управления функциями:
бит 0 — AT\W1;
бит 1 — AT%F1;
бит 2 — V.23 полудуплекс (AT%F3);
биты 3, 4 — AT&P;
биты 5, 6, 7 — резерв.
S29 — длительность опускания трубки по символу "!" в строке набора (flash dial).
Единицы: 0,01 с.
Величина: 000.
S30 — таймер бездействия до рассоединения.
Единицы: 0,1 с.
Величина: 000.
S31 — биты управления функциями:
бит 0 — резерв;
бит 1 — ATN1;
биты 2, 3 — ATW;
биты 4...7 — резерв.
S32 — символ XON.
Величина: 017 (ASCII).
S33 — символ XOFF.
Величина: 019 (ASCII).
S36 — управление переговорами V.42bis (если соглашение не достигнуто):
0 — рассоединиться;
1 — прямое соединение;
3 — нормальное соединение;
4 — пробовать MNP, если неудачно — рассоединиться;
5 — пробовать MNP, если неудачно — прямое соединение;
7 — пробовать MNP, если неудачно — нормальное соединение (по умолчанию).
S37 — желаемая скорость соединения:
0 — автодетектирование (ATF0);
1...3 — 300 bps (ATF1);
5 — 1200 bps (ATF4);
6 — 2400 bps (ATF5);
7 — V.23 (ATF3);
8 — 4800 bps (ATF6);
9 — 9600 bps (ATF8);
10 — 12000 bps (ATF9);
11 — 14400 bps (ATF10);
12 — 7200 bps (ATF7).
S38 — задержка перед принудительным рассоединением после получения команды ATH0.
Единицы: с.
Величина: 020.
Если S38=255, модем продолжает передавать данные до тех пор, пока весь буфер данных не будет передан или связь потеряна.
S39 — управление потоком данных:
биты 0...2 — AT&K.
S40 — биты управления функциями (MNP):
бит 0 — расширенный сервис MNP (AT-K1);
бит 1 — подстройка уровня при сотовом протоколе (AT)M1);
бит 2 — AT*H;
биты 3...5 — действия по сигналу разрыва (AT\K);
биты 6, 7 — размер блока MNP (AT\A).
S41 — биты управления функциями (MNP):
биты 0, 1 — AT%C;
бит 2 — AT%E1;
бит 3 — AT\G1;
бит 4 — AT\L1;
биты 5...7 — резерв.
S46 — управление сжатием данных:
136 — коррекция ошибок без сжатия;
138 — коррекция ошибок со сжатием.
S48 — управление переговорами V.42:
0 — запретить переговоры и продолжать по LAPM;
7 — разрешить переговоры (по умолчанию);
128 — разрешить переговоры и продолжать в соответствии с S36.
S82 — действие сигнала разрыва:
3 — "отложенное" — модем пересылает сигнал разрыва немедленно, целостность данных сохраняется;
4 — "деструктивное" — сигнал разрыва передается немедленно, обрабатываемые в этот момент данные разрушаются;
128 — "последовательное" — модем пересылает сигнал разрыва последовательно с данными, целостность данных сохраняется.
S86 — причина рассоединения или неудачи установления связи:
0 — нормальное рассоединение без ошибок (OK);
4 — потеря несущей (NO CARRIER);
5 — переговоры V.42 не обнаружили протокола коррекции ошибок в удаленном модеме;
9 — модемы не нашли общего протокола;
12 — нормальное рассоединение по инициативе удаленного модема;
13 — удаленный модем не отвечает после 10 повторений одного и того же сообщения;
14 — нарушение протокола.
S91 — уровень передачи данных.
Диапазон: 0...15.
Единицы: -дБм.
Величина: 010 (-10 дБм).
В некоторых странах мощность сигнала передачи по телефонной линии строго регламентирована (США — 0 дБм, Япония — -15 дБм). Повышение уровня передачи, во-первых, затрудняет работу системы эхо-компенсации, выделяющей принимаемый сигнал на фоне собственной передачи, а во-вторых, может вызвать перегрузку и искажения в тракте АТС. Поэтому увеличивать сигнал до предела (S91=0) может оказаться невыгодно. Если удаленный модем принимает вас недостаточно громко и при потере несущей первым вешает трубку, попробуйте постепенно уменьшать значение регистра.
S92 — уровень передачи в режиме факса.
Диапазон: 0...15.
Единицы: - дБм.
Величина: 010 (-10 дБм).
Поскольку спектр факс-сигнала отличается, а АЧХ линии неравномерна, оптимальное значение может отличаться от S91 даже при связи с тем же удаленным факс-модемом.
S95 — вывод расширенных сообщений:
бит 0 — CONNECT XXXX указывает скорость DCE вместо скорости DTE;
бит 1 — добавить /ARQ (автоматический перезапрос) к CONNECT, если используется протокол коррекции ошибок;
бит 2 — сообщение о скорости в линии CARRIER XXXX;
бит 3 — сообщение о протоколе коррекции ошибок PROTOCOL XXXX;
бит 5 — сообщение о методе сжатия COMPRESSION XXXX;
биты 4, 6, 7 — резерв.
В заключение подчеркнем еще раз, что приведенный набор команд и регистров может сильно отличаться от варианта, имеющегося в вашей модели. Поэтому в первую очередь тщательно изучите инструкцию к своему устройству, а уж затем оценивайте, насколько применимы в вашем случае те или иные рекомендации.
Чистых линий и надежных CONNECT'ов!

В.СТЕПАНОВ, 9-й класс,
443063, Россия, г.Самара,
ул.Средне-Садовая, 34"А" — 1.

РЕШЕНИЕ КВАДРАТНЫХ И БИКВАДРАТНЫХ УРАВНЕНИЙ, ИССЛЕДОВАНИЕ КВАДРАТИЧНОЙ ФУНКЦИИ НА "ZX-SPECTRUM"

В школе часто приходится решать квадратные и биквадратные уравнения и исследовать квадратичную функцию. Чтобы облегчить себе работу дома, я и написал эту программу. Поскольку в "Spectrum'е" нет символов "Принадлежность", и "Бесконечность", пришлось заменить их:

& — "Принадлежность" ($\in$);

$ — "Бесконечность" (∞).

Надеюсь, что моя программа будет полезна не только ученикам школ.

```
 10 CLS
 20 REM STEPANOV VIACHESLAV. SAMARA. 1993
 30 BODER 2: PAPER 3: INK 0: CLS
 40 PRINT AT 0,0; "ВАМ НУЖНЫ ИНСТРУКЦИИ?(1-Д/2-
Н":INPUT C$
 50 IF C$="2" THEN GO TO 180: IF C$<>"1" OR
C$<> "2" THEN GO TO 40
 60 CLS
 70 PRINT AT 0,0;"НА ЗАПРОС КОМПЬЮТЕРА ВВЕДИТЕ"
 80 PRINT AT 1,0;"НУЖНУЮ ВАМ ЦИФРУ:"
 90 PRINT AT 2,0;"-1 КВАДРАТНЫЕ УРАВНЕНИЯ"
 100 PRINT AT 3,0;"-2 БИКВАДРАТНЫЕ УРАВНЕНИЯ"
 110 PRINT AT 4,0;"-3 ИССЛЕДОВАНИЕ КВАДРАТИЧНОЙ
ФУНКЦИИ"
 130 PRINT AT 7,0;"ВЫ ГОТОВЫ?(1-Д)":PRINT AT
8,0;"БУДЕТЕ СЧИТАТЬ?(2-Н)"
 140 PAUSE 60
 150 IF INKEY$="2" THEN NEW
 160 PAUSE 60
 170 IF INKEY$="" THEN GO TO 140
 180 CLS: PRINT 0,0;"ВВЕДИТЕ ЦИФРУ"
 190 INPUT A$
 200 IF A$="1" THEN GO TO 250
 210 IF A$="2" THEN GO TO 420
 220 IF A$="3" THEN GO TO 680
 240 IF A$<>"1" OR A$<>"2" OR A$<>"3" OR A$=" "
THEN GO TO 190
 250 CLS
 260 GO SUB 1100
 270 LET X1=0: LET X2=0
 280 IF (B*B-4*A*C)<0 THEN GO TO 370
 290 LET X1=((-B)-SQR (B*B-4*A*C))/(2*A)
 300 LET X2=((-B)+SQR (B*B-4*A*C))/(2*A)
 310 PAUSE 100
 320 CLS
 330 PRINT AT 0,0;"X1=";X1
 340 PRINT AT 1,0;"X2=";X2
 350 PRINT AT 2,0;A;"(X-";X1;")(X-";X2;")"
 360 GO TO 390
 370 CLS
 380 PRINT AT 0,0;"РЕШЕНИЙ НЕТ"
 390 PRINT AT 3,0;"БУДЕТЕ ЕЩЕ СЧИТАТЬ?(1-Д/2-
Н)"; INPUT M
 400 IF M=1 THEN GO TO 250: IF M=2 THEN GO TO 60
 410 GO TO 60
 420 CLS
 430 GO SUB 1100
 440 IF (B*B-4*A*C)<0 THEN GO TO 640
 450 LET Y1=((-B)-SQR (B*B-4*A*C))/(2*A)
 460 LET Y2=((-B)+SQR (B*B-4*A*C))/(2*A)
 470 IF Y1<0 THEN GO TO 550
 480 IF Y2<0 THEN GO TO 500
 490 IF Y1<=0 AND Y2<=0 THEN GO TO 640: GO TO 530
 500 LET X1=-SQR Y1
 510 LET X2=SQR Y1
 520 GO TO 570
 530 LET X1=-SQR Y1
 540 LET X2=SQR Y1
 550 LET X3=-SQR Y2
 560 LET X4=SQR Y2
 570 PAUSE 100
 580 CLS
 590 PRINT AT 0,0;"X1=";X1
 600 PRINT AT 1,0;"X2=";X2
 610 PRINT AT 2,0;"X3=";X3
 620 PRINT AT 3,0;"X4=";X4
 630 GO TO 650
 640 CLS: PRINT AT 0,0;"РЕШЕНИЙ НЕТ"
 650 PRINT AT 5,0;"БУДЕТЕ ЕЩЕ СЧИТАТЬ?(1-Д/2-Н)"
 660 INPUT M: IF M=1 THEN GO TO 420
 670 GO TO 60
 680 CLS
 690 GO SUB 1100
 710 LET N=0
 720 LET M=0
 730 LET M=(-B)/(2*A)
 740 LET N=(4*A*C)/(4*A)
 760 IF (B*B-4*A*C)<0 THEN GO TO 790
 770 LET X1=((-B)-SQR (B*B-4*A*C))/(2*A)
 780 LET X2=((-B)+SQR (B*B-4*A*C))/(2*A)
 790 CLS
 800 PAUSE 70
 810 IF A<0 THEN GO TO 940
 820 PRINT AT 0,0;"D(F)=(-$;+$)"
 830 PRINT AT 1,0;"E(F)=[";M;";+$)"
 840 PRINT AT 2,0;"Y ВОЗРАСТАЕТ X&[";M";+$)"
 850 PRINT AT 3,0;"Y УБЫВАЕТ X&(-$;";M;"]"
 860 IF (B*B-4*A*C)<0 THEN GO TO 920
 870 PRINT AT 4,0;"Y<0 X&(";X1;";";X2;")"
 880 PRINT AT 5,0;
"Y>0  X&(-$;";X1;")U(";X2;";+$)"
 890 PRINT AT 6,0;"Y=0 X=";X1;",";X2
 900 PRINT AT 7,0;"ВЕРШИНА:(";M;";";N;")"
 910 GO TO 1060
 920 PRINT AT 9,0;"С ОСЬЮ X": GO TO 1060
 940 CLS: PRINT AT 0,0;"D(F)=(-$;+$)"
```

```
 950 PRINT AT 1,0;"E(F)=(-$;";M;"]"
 960 PRINT AT 2,0;"Y ВОЗРАСТАЕТ X&(-$;";M;"]"
 970 PRINT AT 3,0;"Y УБЫВАЕТ X&[";M;";+$)
 980 IF (B*B-4*A*C)<0 THEN GO TO 1040
 990 PRINT AT 4,0; "Y<0 X&(-$;";X1;")U(";X2;
";+$)"
 1000 PRINT AT 5,0;"Y>0 X&(";X1;";";X2;")"
 1010 PRINT AT 6,0;"Y=0 X=";X1;",";X2
 1020 PRINT AT 7,0;"ВЕРШИНА:(";M;";";N;")"
 1030 GO TO 1060
 1040 PRINT AT 8,0;"ПАРАБОЛА НЕ ИМЕЕТ ОБЩИХ
ТОЧЕК"
 1050 PRINT AT 9,0;"С ОСЬЮ X"
 1060 PRINT AT 12,0;"БУДЕТЕ ЕЩЕ ИССЛЕДОВАТЬ
(1-Д/2-Н)?": INPUT L
 1070 IF L=2 THEN GO TO 60
 1080 GO TO 680
 1100 CLS: PRINT AT 0,0; "ВВЕДИТЕ ПЕРВЫЙ
КОЭФФИЦИЕНТ": INPUT A
 1110 PRINT AT 1,0;"A=";A
 1120 PRINT AT 2,0;"ВВЕДИТЕ ВТОРОЙ
КОЭФФИЦИЕНТ": INPUT B
 1130 PRINT AT 3,0;"B=";B
 1140 PRINT AT 4,0;"ВВЕДИТЕ СВОБОДНЫЙ ЧЛЕН":
INPUT C
 1150 PRINT AT 5,0;"C=";C
 1160 RETURN
```

ДОМАШНЕЕ ЗАДАНИЕ

В 1980 году Учебно-производственный центр вычислительной техники Октябрьского района г.Москвы провел первую олимпиаду по программированию среди школьников. Участникам было предложено одиннадцать задач, разбитых на три группы в порядке убывания трудности, причем внутри одной группы задачи считались равноценными. Из каждой группы в зачет шло не более одной задачи.

Считаем, что предложенные тогда задачи до сих пор не утратили актуальность и предлагаем вашему вниманию условия задач последней группы, носивший явно "утешительный" характер. Проверьте свои силы.

1. Составить программу вывода всех трехзначных десятичных чисел, сумма цифр которых равна данному целому числу.

2. Целое неотрицательное число M задано массивом своих двоичных цифр a_0, a_1,..., a_{n-1}:

$M=a_{n-1}\cdot 2^{n-1}+a_{n-2}\cdot 2^{n-2}+...+a_1\cdot 2+a_0$.

где $a_i=0$ или $a_i=1$ (i=0,...,n-1). Напечатать массив двоичных цифр числа M+1.

3. В массиве X (M, N) все числа различны. В каждой строке выбирается минимальный элемент, затем среди этих чисел выбирается максимальное. Напечатать номер строки массива X, в которой расположено выбранное число.

4. В массиве X(N) каждый элемент равен 0, 1 или 2. Переставить элементы массива так, чтобы сначала располагались все нули, затем все единицы и, наконец, все двойки (дополнительного массива не заводить).

ИГРАЙТЕ С НАМИ

Здравствуйте, дорогие друзья! Вот еще две игры из нашей коллекции.

Rise of the Triad: Dark War

Автор: FormGen / Apogee, 1995.

Требования к компьютеру: AT-386/33МГц, 4...8 Mb RAM, CD-ROM Drive, VGA, звуковая карта, совместимая с SoundBlaster.

Объем: 1 CD.

Стиль: Arcade.

Это еще одна Doom-подобная стрелялка, но здесь много нового и интересного: она сложнее и "кровавее", чем Doom, предоставляет больше типов оружия, позволяет выбрать 5 различных игроков, каждый из которых обладает своими спецнавыками, имеет более крутые уровни с множеством препятствий и ловушек, позволяет превратиться в бога, собаку, полетать и т.п. Но давайте все по порядку и более подробно.

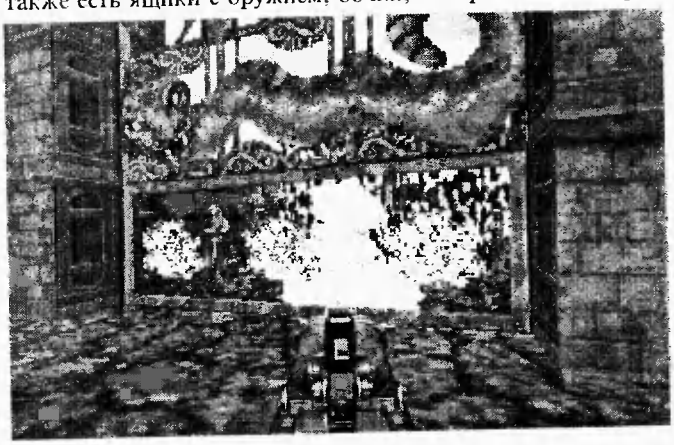
Начнем со сложности. На самом первом уровне, буквально с первых шагов, вы сталкиваетесь с несколькими типами врагов. Особенно опасны те, которые сначала просят пощады, а потом стреляют в спину. Здесь же вы можете найти второй пистолет — это для того чтобы стрелять с двух рук.

Насчет крови следует сделать отдельное замечание. В игре есть несколько уровней "кровавости". Для людей с повышенной впечатлительностью я рекомендую уровень Violence Level None, ну а для того, чтобы "оторваться" на полную катушку — уровень Excessive. Здесь при точном попадании из тяжелого оружия в вас летят окровавленные остатки ваших врагов.

Теперь об оружии. Его здесь 11 типов: всегда есть пистолет с неограниченным числом патронов (кулаков и бензопилы, в отличии от Doom, нет), имеется автомат с бездонным магазином, базука, огнемет и различные виды тяжелого оружия.

Несколько слов о врагах. Есть солдаты (Low Guard и High Guard), патрульные в черной форме (Overpatrol), босс — генерал Дарнан, во втором эпизоде — это Lightning Guard, Strike Team и Triad Enforces, еще один босс — Себастьян Крайст.

В третьем все еще круче: появляются механические враги — Patrol Robot, Ballistikraft и еще один босс, на этот раз механический робот. В четвертом эпизоде вам придется сражаться с двумя типами монахов — Death Monk и Death-Fire Monk, а также с самым главным боссом — El Oscuro. Помимо этого есть еще газовые и огненные кратеры, стены, стреляющие огнем, горелки, перемещающиеся стены, огненные стены, перемещающиеся лезвия и т.п. Не расстраивайтесь. Здесь также есть ящики с оружием, бочки, которые можно взры-



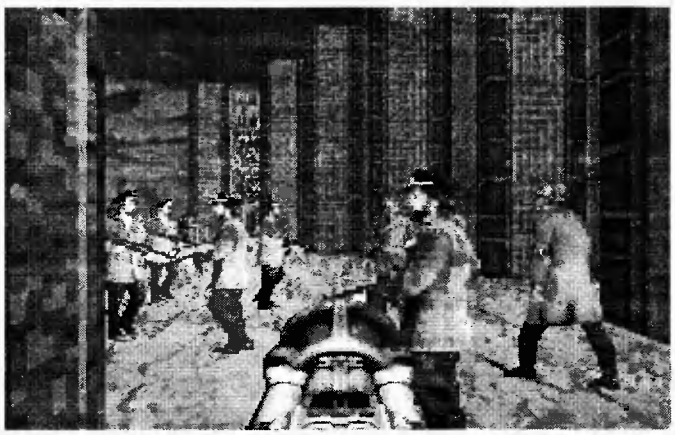
вать, переключатели, поднимающиеся платформы, ключи, лифты и т.п.

Одним словом, если играть по-честному, это — надолго. Уровни, составляющие эпизоды, достаточно сложны для начинающих — препятствия встречаются на каждом шагу и времени на их освоение у вас просто нет.

И наконец — о превращениях. На достаточно короткое время (15-30 секунд) вы можете стать богом и уничтожать врагов наложением божьей кары, собакой, которая может не только загрызть врага, но и открывать двери. Есть возможность полетать и достичь тех мест, куда не запрыгнуть, можно стать эластичным и дотянуться до чего душа пожелает и т.д.

Для тех, кто захочет пройти игру за несколько дней, в игре существуют секретные коды (см.таблицу). Для того чтобы можно было ими воспользоваться, необходимо ввести ключевое слово DIPSTICK.

Код	Описание
WOOF	Превратиться в собаку
GOOBERS	Начать уровень сначала
LONDON	Включить туман
NODNOL	Отключить туман
BOING	Стать эластичным
PANIC	Отменить режим (в режиме собаки, бога и т.п.)
SHOOTME	Бронежилет
GOTO	Перейти на другой уровень
GOARCH	Завершить уровень
DIMON	Включить источники света
DIMOFF	Отключить источники света
BADTRIP	Режим shroom (попробуйте!)
TOOSAD	Превратиться в бога
GOTA386	Отключить текстуры пола и стен
GOTA486	Включить текстуры пола и стен
BURNME	Асбестовая защита
LUNGDUNG	Газовая маска
HUNTPACK	Получить Split missile
86ME	Умереть
REEN	Начать уровень с самого начала
JHONWOO	Получить второй пистолет
VANILLA	Получить базуку
HOTTIMES	Получить Heat seeker
BOOZE	Получить Drunk missile
FIREBOMB	Получить Fire bomb
BONES	Получить Flame Wall
SEEYA	Снова стать богом
SPLIT	Получить Split missile
CUJO	Снова стать собакой
CHOJIN	Полное оружие и бессмертие
WHERE	Отображение координат
RECORD	Записать демо
PLAY	Воспроизвести демо
CARTIER	Получить полную карту
GOGATES	Полностью завершить игру



The 7th Guest

Автор: Virgin Games, 1993.

Требования к компьютеру: AT-386, 2Mb RAM, CD-ROM Drive (скорость передачи данных не менее 150Kb в секунду), SVGA (16-битный, 512Kb), звуковая карта, совместимая с SoundBlaster, 10 Mb на жестком диске, мышь.

Объем: 1 CD.

Стиль: Adventure, Logic.

Перед вами крутейшая головоломка всех времен и народов! Эту игру написали, судя по всему, для того чтобы показать, на что способны CD-ROM. В ней были засняты реальные актеры.

Действие игры происходит в замке Генри Страуфа, злого кукольного мастера, который завлекает шестерых гостей к себе обещаниями дать им власть и деньги (незваный седьмой гость сам прокрался в дом).

Открывается книга, играет тихая музыка, переворачиваются страницы и вдруг... мы видим, как рисунки оживают, разворачивается действие и нашему взору представляется сказка о разбогатевшем нищем, конец которой делаете вы сами. Прибывают гости, находят записку, в которой хозяин советует им самим позаботиться о своем развеселом времяпрепровождении. По ходу сюжета гости погибают один за другим, только седьмой гость остается в живых, и теперь его жалкая судьбинушка полностью зависит от вас.

Лучше начать с головоломки в гостиной. Вам нужно разделить тортик на шестерых гостей поровну — по пять кусочков, причем так, чтобы края кусочков, предназначенных одному гостю, соприкасались. Каждый гость должен получить два кусочка с черепами, два — с надгробиями и один — без обозначения. Если успешно разделите тортик, большинство дверей 2-го этажа откроются и можно будет посмотреть, что же там когда-то происходило.

Вся игра управляется "мышкой". Изящная кисть руки скелета указывает, "куда пойдем?". Если она превращается в челюсти, вам предстоит встреча с чем-то мистическим и, обычно, неожиданным, если в "раскалывающийся" мозг — вас ожидает головоломочка, если в маску с трагическим выражением "лица" — драматический эпизод. По мере разгадывания головоломок разворачиваются события, помогающие вам понять, что же происходило в этом замке. Если вы не знаете как решить головоломку, зайдите в библиотеку (там есть книжка с подсказками). Первый раз вам объяснят суть загадки, во второй — растолкуют, по каким принципам она действует, в третий — загадка решается сама собой, но решения вы не увидите.

В "седьмом госте" есть, на мой взгляд, все типы головоломок, которые существую на белом свете: шахматные (к примеру, переставить 8 ферзей так, чтобы ни один не попал на путь другого), лингвистические (переставить в группе гласных и согласных только согласные так, чтобы получилось правильное английское предложение), графические (переставлять маленьких паучков в определенном порядке), загадки с переключением клапанов и т.п.

Играть будет легче, если нарисовать план дома. В подвале вас ожидает мрачный запутанный лабиринт, проходить ко-

торый нужно "только по правой руке" либо "только по левой". По тайным ходам (вроде камина) можно попасть на второй этаж. Иногда в коридорах появляется женщина в белом — не бойтесь, идите за ней.

"The 7th Guest" захватывает мгновенно и надолго, хотя пройти ее довольно сложно. Я играл в нее достаточно долго и не жалею о потраченном времени. Изящная графика, ненавязчивая музыка, разнообразные звуковые эффекты постоянно держат в напряжении и не дают оторваться от компьютера ни на минуту. Хочу предостеречь впечатлительных — не играйте по ночам и на сон грядущий.

До новых встреч!

Играл А. Ермаченко,
г.Минск, БГУИР, факультет информационных технологий и управления, II курс.

А.АГАЛЬЦОВ,
140414, Московская обл., г.Коломна, ул. Советская, 54 — 51.

"МОРСКОЙ БОЙ"

Эта программа почти полностью соответствует всем известной игре. Написана она на Бейсике для компьютера УКНЦ (МС-0511), но легко может быть адаптирована практически на любой компьютер. Напомним, что знак "?" заменяет оператор PRINT, а знак "'" — оператор REM.

```
10 'МОРСКОЙ БОЙ
20 CLS
30 SCREEN 2
40 LINE (80,10)-(560,250),8,BF
50 LINE (110,25)-(530,235),7,BF
60 LINE (140,40)-(500,220),6,BF
70 LINE (170,55)-(470,205),5,BF
80 LINE (200,70)-(440,190),4,BF
90 LINE (230,85)-(410,175),3,BF
100 LINE (260,100)-(380,160),2,BF
110 LINE (290,115)-(350,145),1,BF
120 FOR I=0 TO 5000
130 NEXT
140 LINE (0,264)-(640,0),3,BF
150 SCREEN 1
160 DIM X(10),Y(10),X3(150),Y3(150),
X1(10),Y1(10)
170 COLOR 8,3,3
180 CLS
190 GOSUB 620
200 WIDTH 75
210 SCREEN 2
220 T=0
230 P=0
240 L=0
250 J=0
260 X=80
270 G=0
280 Y=24
290 Z%=0
300 FOR I=1 TO 11'ФОРМИРОВАНИЕ ИГРОВОГО ПОЛЯ
310 Y=Y+11
320 X=X+16
330 LINE (96,Y)-(256,Y),8
340 LINE (X,36)-(X,144),8
350 NEXT
360 M=X+80
370 K=24
380 FOR I=1 TO 11
390 K=K+11
400 M=M+16
410 LINE (352,K)-(512,K),8
420 LINE (M,36)-(M,144),8
430 NEXT
440 ? AT(0,0)"г.КОЛОМНА т.(261) 3-95-94
450 ? AT(0,1)"~~~~~~~~~~~~~~~~~~~~~~~~~
460 ? AT(60,0)"МОРСКОЙ БОЙ
470 ? AT(60,1)"~~~~~~~~~~~
480 ? AT(9,9)"0 1 2 3 4 5 6 7 8 9 (13
пробелов) 0 1 2 3 4 5 6 7 8 9
490 ? AT(30,11)"А        А
500 ? AT(30,12)"Б        Б
510 ? AT(30,13)"В        В
520 ? AT(30,14)"Г        Г
530 ? AT(30,15)"Д        Д
540 ? AT(30,16)"Е        Е
550 ? AT(30,17)"Ж        Ж
560 ? AT(30,18)"З        З
570 ? AT(30,19)"И        И
580 ? AT(30,20)"К        К
590 LINE (150,180)-(450,230),8,B
600 ? AT(15,22)"КОМПЬЮТЕР (13 пробелов) "I$
610 GO TO 660
620 ? AT(25,12)"КАК ВАС ЗОВУТ ?
630 ? CHR$(14)
640 INPUT I$
650 RETURN
660 ? AT(17,4)I$",ВВЕДИ КООРДИНАТЫ 10 КОРАБЛЕЙ
670 ? AT(17,5)"КОРАБЛЬ СОСТОИТ ИЗ ОДНОЙ КЛЕТКИ
680 ? AT(20,6)"И РАСПОЛОЖЕН В ЛЮБОМ МЕСТЕ
690 FOR I=1000 TO 0 STEP -1
700 ? AT(32,22)I
710 NEXT
720 BEEP
730 T=0
740 ? AT(20,6)"  (33 пробела)  "
750 ? AT(32,22)"(4 пробела) "
760 ? AT(17,5)"  (33 пробела)  "
770 ? AT(18,4)"  (47 пробелов)  "
780 LINE (150,180)-(450,230),8,B
790 T=T+1 'ВВОД КООРДИНАТ ИГРОКА
800 ? AT(17,4)I$",ВВОДИ КООРДИНАТЫ "T"КОРАБЛЯ
810 ? AT(17,5)"ЦИФРА-"
820 ? AT(17,6)"БУКВА-"
830 J=J+2
840 Z$=INKEY$
850 IF Z$="" THEN 840
860 IF Z$="0" THEN X(T)=360
```

```
  870 IF Z$="1" THEN X(T)=376
  880 IF Z$="2" THEN X(T)=392
  890 IF Z$="3" THEN X(T)=408
  900 IF Z$="4" THEN X(T)=424
  910 IF Z$="5" THEN X(T)=440
  920 IF Z$="6" THEN X(T)=456
  930 IF Z$="7" THEN X(T)=472
  940 IF Z$="8" THEN X(T)=488
  950 IF Z$="9" THEN X(T)=504
  960 IF Z$="0" OR Z$="1" OR Z$="2" OR Z$="3" OR
Z$="4" OR Z$="5" THEN 980
  970 IF Z$="6" OR Z$="7" OR Z$="8" OR Z$="9"
THEN 980 ELSE 840
  980 ? AT(25+J,5):Z$
  990 M$=INKEY$
 1000 IF M$="" THEN 850
 1010 IF M$="А" THEN Y(T)=140
 1020 IF M$="Б" THEN Y(T)=129
 1030 IF M$="В" THEN Y(T)=118
 1040 IF M$="Г" THEN Y(T)=107
 1050 IF M$="Д" THEN Y(T)=96
 1060 IF M$="Е" THEN Y(T)=85
 1070 IF M$="Ж" THEN Y(T)=74
 1080 IF M$="З" THEN Y(T)=63
 1090 IF M$="И" THEN Y(T)=52
 1100 IF M$="К" THEN Y(T)=41
 1110 IF M$="А" OR M$="Б" OR M$="В" OR M$="Г"
OR M$="Д" OR M$="Е" THEN 1130
 1120 IF M$="Ж" OR M$="З" OR M$="И" OR M$="К"
THEN 1130 ELSE 990
 1130 ? AT(25+J,6);M$
 1140 PAINT (X(T),Y(T)),8'ЗАКРАШИВАНИЕ
КОРАБЛЕЙ ИГРОКА
 1150 IF T<10 THEN 790
 1160 FOR I=1 TO 10
 1170 FOR F=1 TO 10
 1180 IF F=I THEN 1200
 1190 IF Y(F)=Y(I) AND X(F)=X(I) THEN 1260
 'АНАЛИЗ ОБМАНА
 1200 NEXT F
 1210 NEXT I
 1220 ? AT(17,4)"  (34 пробела)  "
 1230 ? AT(17,5)"  (34 пробела)  "
 1240 ? AT(17,6)"  (34 пробела)  "
 1250 GOTO 1360
 1260 ? AT(17,4)"  (34 пробела)  "
 1270 ? AT(17,5)"  (34 пробела)  "
 1280 ? AT(17,6)"  (34 пробела)  "
 1290 ? AT(17,4)I$ ", ВЫ НАРУШИЛИ ПРАВИЛА
ИГРЫ"
 1300 ? AT(17,5)"ЕСЛИ ХОТИТЕ ИГРАТЬ ЧЕСТНО, ТО"
 1310 ? AT(17,6)"НАЖМИТЕ КЛАВИШУ <Д>
 1320 V$=INKEY$
 1330 IF V$="" THEN 1320
 1340 IF V$="Д" THEN 210
 1350 END
 1360 FOR I=1 TO 10 'ВВОД КООРДИНАТ КОРАБЛЕЙ
КОМПЬЮТЕРА
 1370 Q%=INT(RND(1)*100)'ПЕРЕМЕШИВАНИЕ СЛ.ЧИСЕЛ
 1380 FOR A%=0 TO Q%
 1390 U%=RND(1)
 1400 NEXT A
 1410 X1(I)=INT(RND(1)*10)
 1420 Y1(I)=INT(RND(1)*10)
 1430 NEXT I
 1440 FOR I=1 TO 10
 1450 FOR F=1 TO 10
 1460 IF I=F THEN 1480
 1470 IF Y1(F)=Y1(I) AND X1(F)=X1(I) THEN 1360
 1480 NEXT F
 1490 NEXT I
 1500 FOR I=1 TO 10
 1510 IF Y1(I)=0 THEN Y1(I)=140
 1520 IF Y1(I)=1 THEN Y1(I)=129
 1530 IF Y1(I)=2 THEN Y1(I)=118
 1540 IF Y1(I)=3 THEN Y1(I)=107
 1550 IF Y1(I)=4 THEN Y1(I)=96
 1560 IF Y1(I)=5 THEN Y1(I)=85
 1570 IF Y1(I)=6 THEN Y1(I)=74
 1580 IF Y1(I)=7 THEN Y1(I)=63
 1590 IF Y1(I)=8 THEN Y1(I)=52
 1600 IF Y1(I)=9 THEN Y1(I)=41
 1610 NEXT
 1620 FOR I=1 TO 10
 1630 IF X1(I)=0 THEN X1(I)=104
 1640 IF X1(I)=1 THEN X1(I)=120
 1650 IF X1(I)=2 THEN X1(I)=136
 1660 IF X1(I)=3 THEN X1(I)=152
 1670 IF X1(I)=4 THEN X1(I)=168
 1680 IF X1(I)=5 THEN X1(I)=184
 1690 IF X1(I)=6 THEN X1(I)=200
 1700 IF X1(I)=7 THEN X1(I)=216
 1710 IF X1(I)=8 THEN X1(I)=232
 1720 IF X1(I)=9 THEN X1(I)=248
 1730 NEXT
 1740 FOR I=1 TO 10
 1750 PAINT (X1(I),Y1(I)),3,8
 1760 NEXT
 1770 ? AT(17,4)I$",КТО НАЧНЕТ ПЕРВЫМ ?"
 1780 ? AT(17,5)"ЕСЛИ НАЧНЕШЬ САМ НАЖМИ КЛАВИШУ"
 1790 ? AT(17,6)"<Я>, ЕСЛИ КОМПЬЮТЕР - НАЖМИ <Т>"
 1800 Q$=INKEY$
 1810 IF Q$="" THEN 1800
 1820 IF Q$="Я" THEN 1840
 1830 IF Q$="Т" THEN 2380
 1840 ? AT(17,4)"  (40 пробелов)  "
 1850 ? AT(17,5)"  (33 пробела)  "
 1860 ? AT(17,6)"  (31 пробел)  "
 1870 LINE (150,180)-(450,230),8,B
 1880 L=0
 1890 L=L+1
 1900 IF P=10 THEN 2880
 1910 ? AT(17,4)I$",НАБЕРИ КООРДИНАТЫ"L
 "ВЫСТРЕЛА"
 1920 ? AT(17,5)"ЦИФРА-"
 1930 ? AT(17,6)"БУКВА-"
 1940 R$=INKEY$
 1950 IF R$="" THEN 1940
 1960 IF R$="0" OR  R$="1" OR R$="2" OR
R$="3" OR R$="4" OR R$="5" OR THEN 1980
```

```
 1970 IF R$="6" OR R$="7" OR R$="8" OR R$="9"
THEN 1980 ELSE 1940
 1980 ? AT(25,5)R$
 1990 IF R$="0" THEN X2=104
 2000 IF R$="1" THEN X2=120
 2010 IF R$="2" THEN X2=136
 2020 IF R$="3" THEN X2=152
 2030 IF R$="4" THEN X2=168
 2040 IF R$="5" THEN X2=184
 2050 IF R$="6" THEN X2=200
 2060 IF R$="7" THEN X2=216
 2070 IF R$="8" THEN X2=232
 2080 IF R$="9" THEN X2=248
 2090 R$=INKEY$
 2100 IF R$="" THEN 2090
 2110 IF R$="А" OR R$="Б" OR R$="В" OR R$="Г"
OR R$="Д" THEN 2130
 2120 IF R$="Ж" OR R$="З" OR R$="И" OR R$="К"
OR R$="Е" THEN 2130 ELSE 2090
 2130 ? AT(25,6)R$
 2140 J=J+2
 2150 IF R$="А" THEN Y2=140
 2160 IF R$="Б" THEN Y2=129
 2170 IF R$="В" THEN Y2=118
 2180 IF R$="Г" THEN Y2=107
 2190 IF R$="Д" THEN Y2=96
 2200 IF R$="Е" THEN Y2=85
 2210 IF R$="Ж" THEN Y2=74
 2220 IF R$="З" THEN Y2=63
 2230 IF R$="И" THEN Y2=52
 2240 IF R$="К" THEN Y2=41
 2250 FOR I=1 TO 10
 2260 FOR I=1 TO 10
 2270 IF X2=X1(I) AND Y2=Y1(I) THEN 2280 ELSE
2360
 2280 IF POINT (X2,Y2)=8 THEN 2360
 2290 PAINT (X2,Y2),8
 2300 BEEP
 2310 ? AT(23,5)"  (13 пробелов)  "
 2320 ? AT(23,6)"  (13 пробелов)  "
 2330 P=P+1
 2340 I=10
 2350 GOTO 1890
 2360 NEXT
 2370 CIRCLE (X2,Y2),3,8,,,0.7
 2380 ? AT(23,5)"  (30 пробелов)  "
 2390 ? AT(23,6)"  (30 пробелов)  "
 2400 G=G+1
 2410 X3(G)=INT(RND(1)*10)
 2420 Y3(G)=INT(RND(1)*10)
 2430 FOR I=1 TO G
 2440 IF I=G THEN 2460
 2450 IF X3(G)=X3(I) AND Y3(G)=Y3(I) THEN 2410
 2460 NEXT
 2470 IF X3(G)=0 THEN X0=360
 2480 IF X3(G)=1 THEN X0=376
 2490 IF X3(G)=2 THEN X0=392
 2500 IF X3(G)=3 THEN X0=408
 2510 IF X3(G)=4 THEN X0=424
 2520 IF X3(G)=5 THEN X0=440
 2530 IF X3(G)=6 THEN X0=456
 2540 IF X3(G)=7 THEN X0=472
 2550 IF X3(G)=8 THEN X0=488
 2560 IF X3(G)=9 THEN X0=504
 2570 IF Y3(G)=0 THEN Y0=140
 2580 IF Y3(G)=1 THEN Y0=129
 2590 IF Y3(G)=2 THEN Y0=118
 2600 IF Y3(G)=3 THEN Y0=107
 2610 IF Y3(G)=4 THEN Y0=96
 2620 IF Y3(G)=5 THEN Y0=85
 2630 IF Y3(G)=6 THEN Y0=74
 2640 IF Y3(G)=7 THEN Y0=63
 2650 IF Y3(G)=8 THEN Y0=52
 2660 IF Y3(G)=9 THEN Y0=41
 2670 IF POINT(X0,Y0)=8 THEN 2680 ELSE 2740
 2680 FOR D=1 TO 5
 2690 CIRCLE (X0,Y0),D,3,,,0.7
 2700 NEXT
 2710 BEEP
 2720 Z%=Z%+1%
 2730 IF Z%<10% THEN 2400 ELSE 2760
 2740 CIRCLE (X0,Y0),3,8,,,0.7
 2750 GOTO 1890
 2760 ? AT(17,4)"  (37 пробелов)  "
 2770 ? AT(17,5)"  (31 пробел)  "
 2780 ? AT(17,6)"  (31 пробел)  "
 2790 ? AT(19,4)I$",ВЫ ПРОИГРАЛИ"
 2800 ? AT(17,5)"ЕСЛИ БУДЕТЕ ИГРАТЬ ЕЩЕ"
 2810 ? AT(17,6)"НАЖМИТЕ КЛАВИШУ <Д>"
 2820 F$=INKEY$
 2830 IF F$="" THEN 2820
 2840 IF F$="Д" THEN 210
 2850 ? CHR$(15)
 2860 COLOR 8,2,2
 2870 END
 2880 ? AT(17,4)"  (32 пробела)  "
 2890 ? AT(17,5)"  (32 пробела)  "
 2900 ? AT(17,6)"  (32 пробела)  "
 2910 ? AT(19,4)I$", ТЫ ВЫИГРАЛ"
 2920 ? AT(17,5)"ЕСЛИ ХОЧЕШЬ ЕЩЕ ИГРАТЬ"
 2930 ? AT(17,6)"НАЖМИ КЛАВИШУ <Д>"
 2940 U$=INKEY$
 2950 IF U$="" THEN 2940
 2960 IF U$="Д" THEN 210
 2970 IF U$="Н" THEN 2980 ELSE 2940
 2980 ? CHR$(15)
 2990 COLOR 8,2,2
 3000 END
```

Дорогие друзья!

Страницы журнала "Радиолюбитель. Ваш компьютер" открыты для всех, кто хочет и умеет сделать общение с компьютером лучше, интереснее и полезнее. Поделитесь с коллегами своими идеями, схемными решениями, программами и советами.

Присылайте нам и свои вопросы — возможно, кто-то уже знает ответ. Многочисленному форуму наших читателей под силу найти решения самых сложных проблем. В общем, ждем ваших писем.

Редакция.

220050, Минск, а/я 41, BELARUS, Minsk, Box 41

ISSN 0869-0510. Радиолюбитель. Ваш компьютер, 1996.

Цена договорная.

"ВАШ КОМПЬЮТЕР" –

ЭТО ЖУРНАЛ ДЛЯ ВАС!