ISSN0869-0510

радио любитель

СЕНТЯБРЬ
1995

Ваш компьютер

ВК

Учредитель: НТК "Инфотех"

ЧИТАЙТЕ В НОМЕРЕ:

радио любитель

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
Татьяна БЕЛЬСКАЯ — техническая графика

На 1-й и 4-й странице обложки компьютерная графика
Натальи БЕЛЬСКОЙ

Отдел экспедирования и рассылки журналов — **Наталья ПАСЫНКОВА,** тел. (0172) 78-30-16.
Адрес для писем: 220050, г.Минск-50, а/я 41.
Адрес редакции: 220099, Минск, ул. Авакяна, 30-1-2,
Тел. (0172) 24-86-20.
Факс: (0172) 78 30 16.

Распространение и приобретение очередных номеров журнала — по тел.: (0172) 77-07-87.
Расчетный счет 461496 в Октябрьском РКЦ Ленинского отделения Белбизнесбанка в г.Минске МФО 153001763 код 763, для НТК "Инфотех". Корреспондентский 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 22.10.90г. (рег. удост. N62) и Министерством печати и информации России 17.06.91 (рег. удост. N931).
Подписано к печати 15.08.95. Формат 60 x 84 1/8. Печать офсетная. 4,5 печ. л. Зак. 001.
Отпечатано с оригинал-макета, изготовленного редакцией журнала, в МУ НТК "Инфотех".

ОТ РЕДАКТОРА

Я помню, как в 50-е годы нас, школьников, при наличии в магазинах дешевой колбасы (в столицах), а в городах — дешевой водки в разлив, на политинформациях серьезные дяди и тети убеждали в том, что генетика и кибернетика — лженауки, порождение капитализма. По части генетики мне дал пояснения мой приятель — Володя Парчевский, будущий известный ученый в области генетики. А вот кибернетика так и осталась для меня загадкой, хотя мой однокашник Леня Альтшуллер — будущий известный за пределами СССР ученый-математик — долго и нудно объяснял мне, что такое алгоритм и как, пользуясь двумя действиями— сложением и вычитанием, можно решать трансцендентальные уравнения, которые, оказывается, не решаются вовсе. И только попавшая ко мне чисто случайно переписанная от руки книга Винера "Кибернетика и общество" открыла глаза на все это безобразие. Масла в огонь подлил Артур Кларк со своими фантазиями и предвидениями будущего. И хотя я все равно не понимал, что такое язык Алгол или Фортран, сама мысль о том, что к 2000 году можно будет иметь персональный компьютер, а в начале XXI века компьютер будет на равных играть с человеком — была чистой воды что ни на есть фантастика. А ведь самая-самая БЭСМ-6 занимала к моменту чтения фантазий А.Кларка площадь в тысячи квадратных метров и требовала для своего энергопитания целую электростанцию!

К чему такие предисловия? Меня и сейчас, спустя столько лет, восхищают и захватывают компьютеры, их возможности при использовании в повседневной жизни. И пусть не обидит многих читателей журнала "Радиолюбитель" то, что раздел "Компьютерная техника" с самого начала набрал свою аудиторию, в основном — юную и молодую, интересы которой простирались в неком абстрактном пространстве букв и цифр, программ и подпрограмм, байт и мегабайт, оптических дисководов и карт VGA, пэйджмэйкеров и вентур... И естественной реакцией на это и стал первый журнал — "Радиолюбитель. Ваш компьютер".

Получив первый номер журнала, прошу быть снисходительными к редакции — специфика статей, изобилующих программами, потребовала от редакции утроенного внимания при вычитке корректуры, — мы старались не перекладывать эту работу на плечи читателей, и если вы обнаружите на первых порах неточности или ошибки, не судите нас строго. Особо хочу отметить работу зам. главного редактора Ивана Бельского, его жены Янины и дочерей Татьяны и Наташи. Эта семья — самые главные энтузиасты и создатели этого нового журнала.

Пока нового журнала нет в каталоге, получить его можно через редакцию, перечислив 7000 российских рублей (или по курсу национального банка) на наш р/с 461496 в Октябрьском РКЦ Ленинского отделения Белбизнесбанка в г.Минске МФО 153001763 код 763, для НТК "Инфотех". Корреспондентский счет 700161963 в Главном управлении Национального банка по г.Минску и Минской обл.(адрес банка: 220099, Беларусь, Минск, ул.Казинца, 21, к.3).В эту сумму входят почтовые расходы по состоянию на октябрь месяц 1995 г.

И последнее. Уместен вопрос — ведь в России, на Украине и в Беларуси уже издаются десятки журналов с компьютерной тематикой, — не получится ли так, что мы не наберем достаточно подписчиков, чтобы окупить затраты на производство нового журнала?

И я Вам отвечу так: журналов много, а вот "Радиолюбителя" только три: "Радиолюбитель", "Радиолюбитель. КВ и УКВ" и "Радиолюбитель. Ваш компьютер". Выбирайте любой, а можно и все три! В добрый путь, "Ваш компьютер"!

Главный редактор
В.БЕНЗАРЬ (EU1AA).

А.ЗАКАЛЮКИН,
БГУИР, ФИТиУ, каф. вычислительных методов и программирования,
г.Минск, тел. 39-89-56.

ПРАКТИЧЕСКАЯ ОПТИМИЗАЦИЯ ПРИКЛАДНЫХ ПРОГРАММ

По мере того как вычислительная мощность ЭВМ увеличивается, актуальность проблемы оптимизации вычислений не снижается. И это очевидно, поскольку возможность сокращения вычислительных затрат в несколько раз за счет оптимизации программ или алгоритма позволяет решать такие задачи, от которых приходилось отказываться именно по причине недостаточного быстродействия ЭВМ. Кроме того, в силу предъявляемых требований, узким местом программ часто является их недостаточное быстродействие.

Таким образом, оптимизация программ — это своего рода резерв, часто неиспользуемый, позволяющий повысить производительность ЭВМ для конкретной программы.

Современные компиляторы языков высокого уровня, как правило, проводят, если это возможно, ту или иную оптимизацию. В таких случаях говорят об оптимизирующих компиляторах. Однако степень оптимизации ограничена, так как оптимальность собственно алгоритма вычислений остается на совести программиста. Чтобы разграничить понятия "оптимизация программы" и "оптимальность алгоритма", в данном случае под термином "оптимизация" будем понимать замену медленно работающих операторов на быстрые с частичной модификацией алгоритма в локальных участках программы.

Остановимся на методах и приемах оптимизации программ, написанных на языке PASCAL. Приводимые ниже примеры и рекомендации апробированы и справедливы для компилятора Borland Pascal v.7.0.

1. Арифметические операции и выражения

По степени быстродействия арифметические операции упорядочиваются, начиная с самой медленной, следующим образом:

1) / — деление;
2) * — умножение;
3) + — сложение;
4) - — вычитание.

Последние 3 операции по быстродействию сравнимы и в ряде случаев порядок их следования может быть иным.

Программу следует перестроить таким образом, чтобы медленную операцию заменить на быструю или же сократить по возможности количество обращений к ней. Эта процедура именуется понижением мощности операций.

Например, запись `d:=c*d+c*k+c*m+c*c;` лучше записать как `d:=c*(d+k+m+c);`.

В особенности следует избегать операции "/", так как в этом случае потери можно сократить в несколько раз, например запись `m:=m/2;` лучше заменить на `m:=0.5*m;`.

При записи арифметических выражений не следует разбивать их на промежуточные без особой надобности, экономя тем самым на инициализации "мертвых" переменных. Например, запись

```
x:=x0;
y:=y0;
xy:=x+y;
```

следует заменить на инструкцию `xy:=x0+y0;`.

2. Условные операторы

Приведем фрагменты программ, эквивалентных по логике вычислений, но неравноценных по быстродействию:

а)
```
if i = 1 then begin {...} end;
if i = 2 then begin {...} end;
if i = 3 then begin {...} end;
```
33% (48%)

б)
```
if i = 1 then begin {...} end
else if i = 2 then begin {...} end
else if i = 3 then begin {...} end;
```
17% (35%)

в)
```
case i of
1: then begin {...] end;
2: then begin {...} end;
3: then begin {...} end;
end;
```
50% (17%)

Процентные соотношения указаны относительно суммарного времени просчета фрагментов программ при значениях переключателя i = 1 и в скобках — при i = 3.

Приведенные результаты показывают, что от варианта написания программ (а) следует отказываться в любом случае как от неэффективного. Цепочку операторов `if ... then ... else` следует использовать, если определенные значения переключателя могут появляться с неодинаковой частотой, а именно: наиболее вероятное значение проверки (с результатом `true`) следует располагать в начале цепочки `if`. В любых других случаях предпочтительнее использовать оператор `case`.

3. Циклы

К оптимизации циклов следует относиться с особым вниманием, поскольку именно здесь можно получить основной выигрыш в эффективности программы. Это в особенности относится к вложенным циклам.

Приведем следующие рекомендации в порядке их значимости:

1) Исключение из циклов общих выражений или операций, например

```
s:=0;
for i:=1 to 1000 do s:=s+k*a[i];
```

заменим на

```
s:=0;
for i:=1 to 1000 do s:=s+a[i];
s:=s*k;
```

Из цикла здесь выносится операция умножения, за счет чего быстродействие увеличивается почти в 2 раза.

2) Где возможно, необходимо использовать оператор цикла `for` вместо более медленных операторов `repeat, while`.

3) Исключение из цикла проверок, например:

```
for i:=1 to 100 do
if i<= 50 then a[i]:=sqr(a[i])
else a[i]:=sqrt(a[i]);
```

заменим на

```
for i:=1 to 50 do a[i]:=sqr(a[i]);
for i:=51 to 100 do a[i]:=sqrt(a[i]);
```

4) Объединение циклов, например:

```
for i:=1 to 500 do a[i]:=b[i]+i;
for j:=1 to 500 do c[j]:=d[j]*3;
```

заменим на

```
for i:=1 to 500 do
begin
a[i]:=b[i]+i;
c[i]:=d[i]*3;
end;
```

5) Изменение порядка вложенности циклов, например несколько быстрее выполняется запись

```
for i:=1 to 10 do
for j:=1 to 1000 do begin {...} end;
```

чем

```
for j:=1 to 1000 do
for i:=1 to 10 do begin {...} end;
```

4. Подпрограммы

Включение схожих мест текста в подпрограммы, безусловно, позволяет улучшить структурность и сократить код программы в целом.

Однако с точки зрения оптимизации это не всегда оправдано, если вызов подпрограмм производится в цикле, поскольку расходы, связанные с инициализацией кода подпрограмм и передачей параметров, увеличиваются многократно. Поэтому в особо "критических" циклах вызов подпрограммы можно заменить непосредственно ее операторами.

Это утверждение справедливо и по отношению к библиотечным медленным функциям и процедурам языка. Иногда в этом случае подойдет простая аппроксимация с приемлемой точностью. Если значение функции вычисляется с дискретным шагом, еще лучшие результаты дает замена ее значений массивом с заранее вычисленными значениями. Ниже приводятся три схематичных варианта написания программы в порядке возрастания их эффективности.

а)
```
x:=0.1;
repeat
y:=cos(x);                          {обращение к}
                                    {библиотечной}
                                    {функции}
[...]
x:=x+0.1;
until x>1;                          {74% времени}
                                    {счета}
```

б)
```
x:=0.1;
repeat
y:=1+x*x*(0.03705*x*x-0.4967);      {замена cos(x) на}
                                    {аппроксимирую-}
                                    {щий многочлен}
{...}
x:=x+0.1;
```

```
  until x>1;                          {25% времени}
                                      { счета}
в) x:= 0.1;
   i:=1;
   repeat
   y:=cs[i];                          {использование}
                                      {массива cs со}
                                      {значениями cos(x)}
   {...}
   i:=i+1;
   x:=x+0.1;
   until x>1;                         {менее 3% времени}
                                      { счета.}
```

5. Операции ввода-вывода

Операции ввода-вывода являются потенциально медленными, поскольку быстродействие ограничивается временем доступа к тем или иным периферийным устройствам и их производительностью. Оптимизация здесь сводится, в основном, к сокращению количества обращений к операциям обмена между устройствами. В качестве иллюстрации приведем два схематичных варианта программы считывания заранее подготовленных данных с диска, неравноценных по скорости работы:

```
а) var FR: file of real;
   i: word;
   a:real;
      begin
      assign(FR,'f.dat');
      reset(FR);
      for i:=1 to 100 do
         begin
          read(FR,a);
         {последовательное чтение и обработка}
         {100 значений переменной a ...        }
         end;
      close(FR);
      end.
б) type A_mas=array[1..100] of real;
   var
   a: A_mas;
   FR: file of A_mas;
   i: word;
      begin
         assign(FR,'f.dat');
         reset(FR);
         read(FR,a); {чтение 100 значений массива a}
                     {за один прием                }
         for i:=1 to 100 do begin
                              {обработка 100       }
                              {значений массива a...}
                               end;
         close(FR);
      end.
```

Вариант (а) эффективнее варианта (б) в 20 раз.

При работе с текстовыми файлами обмен с диском происходит порциями посредством буфера ввода/вывода, размер которого по умолчанию равен 128 байтам. Увеличив размер буфера процедурой SetTextBuf, можно существенно ускорить обмен.

Наконец, при обработке особенно больших файлов можно воспользоваться наиболее быстрыми процедурами BlokRead и BlokWrite. Их преимущество в скорости состоит в том, что чтение/запись производятся блоками байт без учета типов переменных и, следовательно, без дополнительных затрат времени на преобразования.

Р.ЖИТОМИРСКИЙ,
г. Минск, СШ N 60 (BarsSoft Group), 9 "Б" класс,
тел. 64-44-63.

РЕЗИДЕНТНАЯ ПОДДЕРЖКА МАКРОСОВ В ЛЮБОЙ СРЕДЕ ПРОГРАММИРОВАНИЯ

Когда я пишу программу на алгоритмическом языке Pascal, я часто вспоминаю его разработчика Никласа Вирта: и где он таких слов набрал — Procedure, Function, Repeat, Until, Implementation, Interface и т.д. Эта проблема стала передо мной особенно остро, когда я начал писать в объектной среде Turbo Vision с ее объектами TInputLine, TStringCollection, TApplication, TScrollBar и др. Ведь ввод их имен с клавиатуры занимает много времени. Вот тогда-то я и решил написать резидентную программу, которая избавила бы меня от многочасового корпения над клавиатурой. Написанная мной программа BarsSoft Professional Key предназначена для создания, редактирования и резидентной поддержки клавиатурных макросов.

Программа реализована в компиляторе языка Паскаль фирмы Borland International Turbo Pascal Version 7.0, хотя после небольшого изменения может быть откомпилирована и в 6-й версии.

Пакет программ состоит из 4-х файлов:

Имя Файла	Размер	Назначение
BSPROKEY.EXE	84 Кб	Головной выполняемый файл
BSPROKEY.TSR	3 Кб	Резидентная порция
BSPROKEY.PKD	6 Кб	Демонстрационный макрос-файл
BSPROKEY.DOC		Файл документации

Головной Модуль

Головной модуль программы имеет имя BSPROKEY.EXE. При запуске без параметров программа выдает краткую информацию о себе.

Автор Родион Житомирский (BarsSoft Group), Белоруссия BarsSoft Group Представляет:BarsSoft Professional Keyboard, Версия 1.0	
Используйте:BSProKey[.exe] [Ключи]	Ключи:
/Q	— Без сообщений
/?	— Помощь по программе
/E	— Редактирование макросов
/L	— Загрузить в память
/R	— Выгрузить из памяти

Назначение ключей:

/Q Не выдавать никаких сообщений.
/? Выдать подробную информацию.
/E Вызов редактора макрос-файлов (см. ниже). (Для вызова редактора вы должны иметь файл BSProKey.PKD.)
/I Загрузиться в память, при этом загружается файл макросов BSProKey.PKD. Резидентная порция занимает в ОЗУ 12 Кб.
/R Выгрузить из памяти. Отключает резидентную поддержку макросов и освобождает занятую память.

Работа программы

Программа перехватывает 16h-е прерывание и просматривает поток данных, идущих от клавиатуры. В случае если в потоке находится "горячая клавиша", поток данных приостанавливается и пользовательской программе передается цепочка клавиш, связанная с "горячей клавишей". Затем поток от клавиатуры восстанавливается.

Перед загрузкой любой программы в память DOS создается блок PSP (Program Segment Prefix), который располагается непосредственно перед программой.

PSP активно используется обыкновенной программой — в нем расположен файловый буфер, командная строка и т.д. Но когда программа становится резидентной, необходимость в PSP для DOS отпадает. Предложенная программа использует его под внутренний циклический клавиатурный буфер размером 100 байт.

Предлагаемая резидентная программа делает проверку своего присутствия в памяти наиболее надежным способом — поиском части кода памяти. В качестве кода используется метка, оставляемая компилятором в EXE-файле (`Portions Copyright ...`), предварительно измененная на `Portions CopyRight ....`

При удалении программы из памяти вектор 16h не восстанавливается, а используется следующий оригинальный алгоритм. При загрузке обычной резидентной программы вектор прерывания устанавливается на процедуру обработки прерывания. Предположим, что поверх загружается другой резидент, который перехватывает тот же вектор. Тогда при выгрузке 1-я программа устанавливает вектор на старый обработчик и, в результате, 2-й резидент оказывается исключенным из цепочки обработчиков прерывания, что может привести к зависанию.

Предлагаемый способ отличается тем, что при загрузке программы в память вектор устанавливается не на процедуру обработки прерывания, а на начало PSP, где ставится команда jmp far <адрес процедуры обработки прерывания>. Таким образом, при выгрузке программы <адрес процедуры обработки прерывания> заменяется на адрес старого обработчика прерывания 16h, а размер блока памяти усекается до размера 16 байтов. При такой выгрузке цепочка не нарушается и машина продолжает нормально работать. Так как в Pascal можно оставить в памяти только всю программу (процедура Keep модуля Dos), а не процедуру обработки прерывания, BSProKey разбита на 2 модуля: один, собственно резидентный, загружается в память без проверки командной строки и сервиса, а второй занимается командной строкой и в случае надобности запускает первый модуль.

Стандартный пакет включает файл BSProKey.PKD, в котором содержатся макросы для языка Pascal. В случае если вы хотите определить новые макросы или отредактировать существующие, вы можете использовать встроенный Редактор Макросов. В рамках данной статьи нет возможности привести его текст. Поэтому в листинге он заменен на простой отладочный редактор, который можно использовать только для полного редактирования макро-библиотеки.

Программа BsProKey может также использоваться для подавления и переопределения клавиш. Выигрыш программы состоит в том, что она эффективно работает с большинством программ, кроме некоторых игр, которые перехватывают 9-е прерывания (от клавиатуры, IRQ2). Ниже приводится листинг программы.

```
{*****************************************}
{   Файл BSProKey.pas                     }
{   BarsSoft Professional Keyboard        }
{   Copyright (c) 1995 by BarsSoft Group  }
{*****************************************}
Uses Dos, Crt  {- Стандартные библиотеки};
{Описание рамки с заставкой}
Const
 HelpInfo:Array [1..13] Of String[80]=
 ('('+Автор Р.Житомирский (BarsSoft Group),     +'
 '¦Белоруссия                                   +',
 '¦     BarsSoft Group Представляет:BarsSoft '  +
 'Professional Keyboard, Версия 1.0             ¦',
 '¦- - - - - - - - - - - - - - - - - - - - - - -¦',
 '¦  Используйте:BSProKey[.exe] [Ключи]'        +
 'Ключи:                                        ¦',
 '¦ - - - - - - - - - - - - - - - - - - - - - - ¦',
 '¦   /Q              -     Без сообщений'      ¦',
 '¦   /?              -     Помощь по программе ',
 '¦   /E              -     Ввод макросов       ¦',
 '¦   /I              -     Загрузить в память  ¦',
 '¦   /R              -     Выгрузить из памяти',
 '¦- - - - - - - - - - - - - - - - - - - - - - -¦',
 '¦                                             ¦',
 '+ - - - - - - - - - - - - - - - - - - - - - +');
Const
   Quit:Boolean=False;
Var
   St:String;
   d:Word;
   ProgSegm:Word;
{Если Quit=False (т.е. если пользователь не
запретил вывод на экран), то выводит строку на
экран, иначе ничего не делает}
Procedure Writel(s:String);
Begin
  If Quit  Then exit;
  Write(s);
End;  {Writel}
{Переносит вектор 16h-го прерывания на PSP
резидентной программы, где в кодах организует
команду ассемблера jmp far <адрес процедуры-
обработчика прерывания> (используется при
загрузке)}
Procedure Direct;
Var poi:Pointer;se,os:Word;sg,om:^Word;
Begin
 GetIntVec($16,Poi);
 Se:=Seg(Poi);os:=ofs(poi);om:=ptr(se,os);sg:=ptr(se,os+2);
 Mem[sg^-16:0]:=$ea;
 MemW[sg^-16:1]:=om^;
 MemW[sg^-16:3]:=sg^;
 setintvec($16,ptr(sg^-16,0));
End;  {Direct}
{Осуществляет поиск резидента в памяти и, если
находит, возвращает True, а в ProgSegm заносит
сегментный адрес найденной метки в памяти
('Portions CopyRight...')}
Function Installed:Boolean;
Var se:Word;Const rt:Boolean=False;
Procedure V5;
Begin
  If Mem[se:4]=Byte('h') Then
  Begin ProgSegm:=se;rt:=True;End;
End;
Procedure V4;
Begin
  If Mem[se:3]=Byte('g') Then V5;
End;
```

```
Procedure V3;
Begin
  If Mem[se:2]=Byte('i') Then V4;
End;
Procedure V2;
Begin
  If Mem[se:1]=Byte('R') Then V3;
End;
Procedure V1;
Begin
  If Mem[se:1]=Byte('y') Then V2;
End;
Begin  {Installed}
  se:=0;
  Repeat
     V1;
     Inc(se);
  Until se=65535;
  Installed:=rt;
End;  {Installed}
      {Выводит таблицу с информацией и, если нужно,
                    строку с информацией об ошибке}
Procedure Error;
Var Answer:String;Iter:Byte;
Begin
 Answer:='';
 Case DosError Of
   2,18 :Answer:='Не найден файл';
   3    :Answer:='Неверный путь';
   5    :Answer:='Не имеете права';
   6    :Answer:='Ошибка файла';
   8    :Answer:='Не хватает памяти';
   10   :Answer:='Неверное окружение';
   11   :Answer:='Неверный формат';
 End;{Case}
 If Quit Then exit;
 TextAttr:=31;GotoXY(1,1);
 For Iter:=1 to 13 do
  Writel(HelpInfo[Iter]);
 GotoXY(3,12);Writel(Answer);
End;  {Error}
          {Выводит информацию об авторе и ключах}
Procedure Help;
Begin
 If Quit Then Exit;
 DosError:=0;Error;
End;  {Help}
            {Процедура-редактор макро-библиотеки}
Procedure Edit;
Type Line      = Array [0..30] Of Word;{Формат
макроса:Нулевой индекс - "горячая" клавиша,
        1-й - длина макроса)}
     MacroLib = Record  {Формат библиотеки
                         макросов:}
      Selector : Word;  {Всего макросов
                         в библиотеке}
      MacroArea: Array [1..100] Of Line;
                        {Массив с макросами}
     End;  {Record}
Var
   MacroVar :MacroLib;
   Macros   :File Of MacroLib;
   TempMacros,Iter: Word;
Function Key:Word;Assembler;
                  {Возвращает полный код клавиши}
Asm
   mov ah,0
   int $16
End; {Key}
Begin
  Help;
  Write('Введите кол-во макросов: ');
  Readln(MacroVar.Selector);
  For TempMacros:=1 To MacroVar.Selector Do
   With MacroVar Do
    Begin
      Iter:=1;
      Write('Введите заголовок макроса:');
      MacroArea[TempMacros,0]:=Key;
      Writeln('Введите Макрос');
      Repeat
        Inc(Iter);
        MacroArea[TempMacros,Iter]:=Key;
        Write(Chr(Lo(MacroArea[TempMacros,Iter])));
      Until MacroArea[TempMacros,Iter]=$1c0d;
      MacroArea[TempMacros,1]:=Iter-2;
      Writeln;
   End;     {With}
   Assign(Macros,'BSProKey.pkd');
   Rewrite(Macros);
   Write(Macros,MacroVar);
   Close(Macros);
End; {Edit}
    {Выводит сообщение о попытке повторной загрузки
                                          программы}
Procedure InstallMessage;
Begin
 If Quit Then exit;DosError:=0;
 Error;
 Writel('Программа уже установлена в память');
End;  {InstallMessage}
 {Выводит сообщение о том, что программа не найдена
                                          в памяти}
Procedure NotInstalled;
Begin
 If Quit Then exit;DosError:=0;
 Error;
 Writel('Программа не найдена в памяти');
End; {NotInstalled}
{Устанавливает программу в памяти]
Procedure Install;
Begin
 If Installed Then Begin InstallMessage;exit;End;
 exec('bsprokey.tsr','');
 If DosError<>0 Then Begin Error;
 Writel(' (bsprokey.tsr)');halt;End;
 direct;
 DosError:=0;Error;
 Writel('Программа установлена в память ');
End;  {Install}
{Выгружает программу из памяти}
Procedure Remove;
Var om,sg:^Word;sz,mc,as:Word;ald:Byte;
Begin If not installed Then
Begin NotInstalled;halt;End;
 For mc:=0 To 6 Do Mem[ProgSegm:mc]:=0;
 as:=MemW[ProgSegm:$2c];
 asm
  mov es,as
  mov ah,$49
  int $21
 End;
 ProgSegm:=ProgSegm-$67;
 om:=ptr(ProgSegm+$b0,$1b0c);sg:=ptr(ProgSegm+$b0,$1b0C+2);
 MemW[ProgSegm:1]:=om^;
 MemW[ProgSegm:3]:=sg^;
 Asm
  mov   ah,$52
  int   $21
  push  ds
  push  es
  pop   ds
  mov   ax,[bx-2]
  mov   as,ax
  pop   ds
 End;
 while Mem[as:0]=Byte('M') Do
 Begin
  If MemW[as:1]=ProgSegm Then mc:=as ;
  as:=as+MemW[as:3]+1;
 End;
 ald:=Mem[mc:0];
 Mem[mc:0]:=Byte('M');
 sz:=MemW[mc:3];
 MemW[mc:3]:=1;
```

```
 Mem[mc+2:0]:=ald;
 MemW[mc+2:3]:=sz-2;
 MemW[mc+2:1]:=0000;
 DosError:=0;Error;
 Writel('Программа выгружена из памяти');
End;  {Remove}
{Приводит строку к верхнему регистру, удаляет все
символы, кроме первых двух. Если первый символ
'-', то заменяет его на '/'}
Function Up(sr:String):String;
Var r:Word;fg:String;
Begin
 fg:='';
 For r:=1 To Length(sr) Do
   fg:=fg+UpCase(sr[r]);
If fg[1]='-' Then fg[1]:='/';
up:=Copy(fg,1,2);
End;  {Up}
Begin   {Program}
 If ParamCount=0 Then Help;
 For d:=1 To paramcount Do If Up(paramstr(d))='/Q'
Then Quit:=True;
 For d:=1 To ParamCount Do
 Begin
   st:=up(paramstr(d));
    If    st='/E'      Then Edit   Else
     If    st='/I'     Then Install Else
      If    st='/R'     Then Remove  Else
       If    st='/E'     Then Edit    Else
        If (st='?') or (st='/?') or (st='/H') Then
Help;
 End;
End. {Program}
```

Резидентный модуль:

```
{*******************************************}
{   Файл BSProTSR.pas                       }
{После компиляции BSProTSR.exe переименовать  }
{в 'BSProKey.TSR' и изменить в нем метку      }
{'Portions Copyright ...' на                  }
{'Portions CopyRight ...'                     }
{   BarsSoft Professional Keyboard            }
{   Copyright (c) 1995 by BarsSoft Group      }
{*******************************************}

{$A+,B-,D-,E-,F-,G-,I-,L-,N-,O-,P-,Q-,R-,S-,T-,V-,X-}
{$M 1024,0,0}
Uses dos;
Type aa=Array [1..100,0..30] of word;
     fu=record se:word;mr:aa;end;
Const bufhead:byte=2;buftail:byte=1;
Var war         :fu;
    ft:file of fu;
function getw:word;
var te:word;
begin
 te:=buftail;
 if te=100 then te:=1 else inc(te);
 getw:=Memw[PrefixSeg:te+100];
end;
function get:word;
begin
 if buftail=100 then buftail:=1 else inc(buftail);
 get:=Memw[PrefixSeg:buftail+100];
end;
function is:boolean;
begin
 if bufhead=100 then is:=not buftail=1 else
is:=not((buftail+1)=bufhead);
end;
procedure put(putvar:word);
begin
 if not ((buftail=100) and (bufhead=1) or
((bufhead+1)=buftail)) then begin
 Memw[PrefixSeg:bufhead+100]:=putvar;
 if bufhead=100 then bufhead:=1 else inc(bufhead);
 end;
end;
procedure
int16(Flags,CS,IP,AX,BX,CX,DX,SI,DI,DS,ES,BP:Word);
Interrupt;
Var rax,fg:word;
label ex;
Begin
 If Hi(Ax) in[01,$11] then {если функция 1 - то :}
   Begin
    asm
     mov ah,01
     pushf
     call[SaveInt00]  {Вызвать "Старика"}
     mov rax,ax   {занести результат в rax}
     pushf
     pop ax
     mov flags,ax {в flags - занести флаги}
    end;
    ax:=rax;
    if (flags and (255-$40))=0 then goto ex;{если
                                          ZF=0 TO:}
    if not Is then goto ex;
    {ЕСЛИ есть что-нибудь В БУФЕРЕ:}
    ax:=getw;
    flags:=flags-$40;
    goto ex;
   end;
 if hi(ax) in[0,$10] then
   begin
    if is then
     begin
      ax:=get;
      goto ex;
     end
    else
     begin
      asm
       mov ax,0
       pushf
       call[SaveInt00]
       mov rax,ax
      end;
      ax:=rax;
      for rax:=1 to war.se do
       if war.mr[rax,0]=ax then
        begin
         for fg:=1 to war.mr[rax,1] do
          begin
           put(war.mr[rax,fg+1]);
          end;
         ax:=get;
         break;
        end;
      end;
   goto ex;
   end;
if hi(ax) in [2,$12] then begin
   ax:=mem[0:$417];
   goto ex;
   end;
if hi(ax)=05 then begin
  put(cx);
  goto ex;
 end;
 ex:
end;
begin
 assign(ft,'bspro   .pkd');
 reset(ft);
 read(ft,war);
 close(ft);
 getintvec($16,SaveInt00);
 setintvec($16,addr(int16));
 keep(0);
end.
```

Выражаю благодарность за помощь в подготовке статьи преподавателю кафедры ВМиП БГУИР Корбиту А.Г.

М.ШУСТОВ,
634024, г.Томск,
ул.5-й армии, 9 - 208.

ПЕРЕКЛЮЧЕНИЕ ОБЪЕМА ВИРТУАЛЬНОГО ДИСКА В MS-DOS 6.0

В [1] была описана программа, позволяющая автоматизировать процесс реконфигурации ПЭВМ (переключать объем и количество виртуальных дисков из MS-DOS, NC — по расширению файла и из меню) с последующей перезагрузкой компьютера. С появлением новой версии дисковой операционной системы MS-DOS 6.0 (PS-DOS 6.0) в марте 1993г. у пользователей IBM-совместимых ПЭВМ впервые появилась возможность задавать необходимую для выполнения конкретной задачи конфигурацию непосредственно в процессе загрузки операционной системы.

При соответствующим образом заданных файлах config.sys и autoexec.bat при загрузке ПЭВМ пользователю предъявляется меню, из которого он должен выбрать устраивающий его на данный сеанс работы вариант конфигурации компьютера. Если в течение заданного в config.sys интервала времени не будет выбран ни один из вариантов, процесс загрузки будет продолжен по сценарию, заложенному в файле config.sys "по умолчанию".

Ниже приведено содержание файлов config.sys и autoexec.bat, при помощи которых можно переключать объем и количество виртуальных дисков из меню, реализовывать другие возможности, заложенные в ПЭВМ. Файлы написаны в текстовом редакторе, для последующих пояснений строчки файлов пронумерованы, номера строк в содержание файлов не входят.

Файл config.sys:

```
1 break=on
2 files=25
3 buffers=15
4 country=049,437,c:\dos\country.sys
5  DEVICE=c:\dos\setver.exe
6 [menu]
7 menucolor=2.0
8 menuitem=VD0
9 menuitem=VD60
10 menuitem=VD160
11  menudefault=VD60,3
12 [VD0]
13  [VD60]
14 DEVICE=c:\dos\ramdrive.sys 53 128 16
15 [VD160]
16 DEVICE=c:\dos\ramdrive.sys 160 128 64
```

Строки 1...3 задают возможность прерывания выполнения ряда программ одновременным нажатием клавиш Ctrl-C, максимальное количество одновременно открываемых файлов, устанавливают число буферов для операций ввода-вывода. Строки 4,5 не обязательны (как и предыдущие), они дают возможность вывода на экран даты и времени в формате, привычном для пользователей ПЭВМ СНГ (строка 4), а также задают возможность выполнения ряда программ, которые могут работать только в ранних версиях DOS (строка 5), например DOS 2.0, DOS 3.30, DOS 4.01 и т.д. Предполагается, что соответствующие файлы country.sys и setver.exe находятся в поддиректории DOS на диске C:.

Следующая, шестая строка задает вывод на экран меню. Строка 7 задает зеленый цвет букв меню, последующие строки (8...11) выводят построчно сообщения меню на экран. Пользователь курсором может выбрать интересующий его вариант и нажать клавишу ENTER (ВВОД). Строка 8 определяет загрузку ПЭВМ без использования виртуального диска, строка 9 задает его объем в 60 Кб, строка 10 — в 160 Кб. Строка 11 сообщает, что если в течение 3 секунд не будет избран другой вариант выполнения config.sys, по умолчанию будет загружен виртуальный диск объемом 60 Кб. Строка 12 при условии выбора пользователем варианта VD0 (виртуальный диск отключен) передает процесс последующей загрузки на файл autoexec.bat с прохожденнем последовательности определяемых им процедур через метку VD0. Строки 13 и 14 активизируются при выборе виртуального диска объемом 60 Кб: в конфигурацию ПЭВМ включается виртуальный диск объемом 60 Кб (в примере — 53 Кб для размещения на виртуальном диске файла command.com MS-DOS 6.0 и исключения возможности его вирусного повреждения, т.к. длина файла равна 53 Кб и соответствует объему виртуального диска). Цифра 128 определяет объем сектора создаваемого диска, цифра 16 — максимальное количество файлов и/или каталогов, которые могут быть размещены на виртуальном диске. Для размещения виртуального диска в расширенной памяти (сверх 640 Кб), если таковая имеется, после цифр может быть задан ключ /E. Строки 15 и 16 активизируются при заданнн объема виртуального диска в 160 Кб.

В версии MS-DOS 6.0 драйвер виртуального диска носит название ramdrive.sys. В приводимом примере он находится в поддиректории C:\DOS.

Файл autoexec.bat:

```
1 @echo off
2 prompt $p$g
3 path c:\;c:\nc;c:\dos;c:\nu;c:\arch;..\..;..;
4 set NU=c:\nu
5 set ARJ SW=-jm -jh65535
6 set TEMP= c:\
7  goto  %config%
8 :VD0
9 goto end
10 :VD60
11  :VD160
12 copy command.com d:*.*
13 set comspec=d:command.com
14 :end
15 ver
16 date
17 time
18 nc
```

Строка 1 отключает вывод на экран сообщений "служебного" характера, строка 2 задает вид приглашения DOS к

диалогу (например, C:\DOS>), строка 3 задает пути поиска в подкаталогах исполняемых программ, строки 4...6 не обязательны и задают "среду окружения" для конкретного пользователя.

Следующие строки (7...14) характерны только для MS-DOS 6.0 (PS-DOS 6.0). Строка 7 обеспечивает взаимодействие файлов autoexec.bat и config.sys и выполнение того или иного фрагмента файла autoexec.bat с переходом на соответствующую метку, определяемую в файле config.sys. В частности, при VD0 (виртуальный диск отключен) управление передается на метку :end (строка 14) с последующим выполнением команд из строк 15...18; при выборе VD60 (или VD160) управление передается на строки 10 или 11 с выполнением в том и другом случае команд из строк 12, 13, 15...18. Строки 12 и 13 обеспечивают копирование файла command.com на виртуальный диск D: с передачей управления на этот файл (строка 13). Строки 15...18 выводят на экран номер версии MS-DOS, дату и время, загружают Norton Commander.

Если в ПЭВМ нет накопителя с именем C:, наименование системного диска A: необходимо указать в соответствующих местах файла config.sys, а вместо диска D: в файле autoexec.bat указать диск C:.

При необходимости (если в файлах config.sys или autoexec.bat имеются грубые ошибки) нажатием функциональных клавиш F5 или F8 при загрузке компьютера можно "пропустить" эти файлы или подтвердить необходимость выполнения каждой их строки.

Литература

1. Радиолюбитель.— N 1.— 1993.— с.11-12.

В.НОВИКОВ, А.КОРБИТ,
БГУИР, ФИТиУ, каф. вычислительных методов и программирования,
г.Минск, тел. 39-89-56.

ДРАЙВЕР ANSI.SYS

Не все знают, что коды терминала ANSI позволяют легко создавать красивые эффекты при выводе на экран текстов, заставок и пр. Примерами тому служат многочисленные красочные и динамичные рекламные заставки, распространяемые многими BBS. Для создания таких текстов даже разработаны специальные ANSI-редакторы.

Драйвер ANSI.SYS, входящий в состав MS DOS, предназначен для реализации двух важных функций:

— управления дисплеем;

— переопределения действий клавиш клавиатуры.

Драйвер ANSI.SYS требует не более 2 Кбайт оперативной памяти. Как и любой другой драйвер, он подключается из файла CONFIG.SYS командой:

```
DEVICE=ANSI.SYS
```

или командой DEVICEHIGH.

Драйвер ANSI.SYS перехватывает всю информацию, посылаемую на стандартное устройство вывода (по умолчанию — на экран дисплея), и отыскивает в ней специальные управляющие команды, называемые Esc-последовательностями. Признаком начала Esc-последовательности является пара символов: символ с ASCII кодом 27 (символ Esc) и символ [(открывающаяся квадратная скобка). Эта комбинация выбрана так, чтобы Esc-последовательности невозможно было спутать с другими цепочками символов. В соответствии с информационной частью Esc-последовательности драйвер ANSI.SYS выполняет те или иные операции.

Esc-последовательности имеют следующий общий вид:

<заголовок><параметры><код команды>

где: <заголовок> Esc-последовательности составляют два указанных выше специальных символа;

<код команды> — (буква латинского алфавита) является признаком конца Esc-последовательности и одновременно указывает выполняемую команду;

<параметры> — параметры указанной команды.

Параметром в аргументе <параметры> может быть или десятичный код символа в стандарте ASCII, или строка символов, заключенная в двойные кавычки. Параметры, если их несколько, отделяются друг от друга символом ";".

В Esc-последовательности не должно быть пробелов между символами.

Табл. 1

Последовательность	Выполняемая функция
	1.Перемещение курсора
Esc[<строка>;<колонка>H	устанавливает курсор в <строку> и <колонку>, по умолчанию — 1;1 (верхний левый угол)
Esc[<строка>;<колонка>f	(то же, что и выше)
Esc[<строк>A	перемещает курсор вверх на <строк> строк, по умолчанию на 1 строку.
Esc[<строк>B	перемещает курсор вниз на <строк> строк, по умолчанию на 1 строку.
Esc[<колонок>C	перемещает курсор вправо на <колонок> колонок, по умолчанию на 1 колонку.
Esc[<колонок>D	перемещает курсор влево на <колонок> колонок, по умолчанию на 1 колонку.
	2.Операции стирания
Esc[2J	стирает экран и перемещает курсор в начало экрана
Esc[K	стирает символы от курсора до конца текущей строки
Esc[?7l	отбрасывание конца строки, выходящего за пределы экрана
Esc[?7h	перенос конца длинной строки на следующую строку
	3. Смешанные функции
Esc[6n	выводит текущие строку и колонку в форме: Esc[<строка>;<колонка>R
Esc[s	сохраняет текущую позицию курсора
Esc[u	перемещает курсор в позицию, сохраненную последним Esc[s

Последовательность	Выполняемая функция
	4. Управление экраном
Esc[Xm или Esc[X;Ym или Esc[X;Y;Zm	Устанавливает атрибуты выводимых на экран символов. Вместо Z указывается атрибут выделения символов: 0 нормальный (серый на черном) 1 жирный (символы повышенной яркости) 4 подчеркивание (только монохромный монитор IBM) 5 мерцание (символ мерцает) 7 реверс (черный на сером) 8 нет вывода (цвет символа совпадает с фоном) Вместо Y указывается цвет фона: 40 ЧЕРНЫЙ 41 КРАСНЫЙ 42 ЗЕЛЕНЫЙ 43 ЖЕЛТЫЙ 44 СИНИЙ 45 РОЗОВЫЙ 46 ГОЛУБОЙ 47 БЕЛЫЙ Вместо X указывается цвет символов: 30 ЧЕРНЫЙ 31 КРАСНЫЙ 32 ЗЕЛЕНЫЙ 33 ЖЕЛТЫЙ 34 СИНИЙ 35 РОЗОВЫЙ 36 ГОЛУБОЙ 37 БЕЛЫЙ
Параметры X,Y,Z могут располагаться в любой последовательности. Пример: Esc[1;41;31m — голубые символы повышенной яркости на красном фоне.	
Esc[=<режим>h	устанавливает видеорежим. Значения <режим> следующие: 0 40x25 текстовый режим черно-белый 1 40x25 текстовый режим цветной 2 80x25 текстовый режим черно-белый 3 80x25 текстовый режим цветной 4 320x200 графический режим цветной 5 320x200 графический режим черно-белый 6 640x200 графический режим черно-белый 7 заставляет курсор переходить на новую строку по концу строки ("заворачиваться")
Esc[=7l	запрещает "заворачивание" курсора по концу строки

Пример: `Esc[3B`

Здесь B — код команды, 3 — параметр команды. Символ с кодом 27 здесь и далее обозначается через Esc.

Существует несколько способов формирования Esc-последовательностей и передачи их для обработки драйверу ANSI.SYS.

1. Включение Esc-последовательности в состав аргумента команды PROMPT в вашем .bat файле (в частности, в файле autoexec.bat). В этом случае вместо заголовка нужно записывать пару символов `$e` без пробелов между ними. Например, последовательность `$e3B` приводит к перемещению курсора дисплея на три строки вниз.

2. Включение Esc-последовательности в командном .bat файле в качестве аргумента команды echo. Например:

```
echo Esc[3B (Курсор переместился на три строки вниз).
```

3. Формирование любым способом текстового файла с Esc-последовательностями и направление его на дисплей, например командами DOS type или copy.

4. Формирование Esc-последовательностей в прикладных программах, допустим, написанных на алгоритмических языках Си или Паскаль, и вывод этих последовательностей на стандартное устройство вывода.

В табл.1 приведены Esc-последовательности драйвера ANSI.SYS для управления экраном.

Вторая функция драйвера ANSI.SYS связана с обработкой кодов клавиатуры и позволяет назначать конкретным клавишам или их комбинациям цепочки символов. При нажатии этих клавиш в командную строку DOS посылаются соответствующие им цепочки символов, которые могут быть, например, часто используемыми командами MS DOS или любыми другими текстовыми фрагментами.

Программирование клавиш осуществляется путем посылки в драйвер ANSI.SYS Esc-последовательности следующего вида:

```
Esc[<расширенный ASCII-код клавиши>;<определение клавиши>p
```

Прописная латинская буква p — код команды. Сама команда располагается между заголовком и кодом и состоит из двух частей, которые разделены знаком ";".

Первая часть команды содержит простой или расширенный ASCII-код клавиши, а вторая — цепочку символов, которая должна генерироваться при нажатии данной клавиши. В цепочке символов могут быть разделенные символом ";" скан-коды клавиши или строки символов в двойных кавычках. Например строка `Esc[0;60;"dir c:"p` приписывает клавише F2 цепочку символов dir c: без выполнения dir c: как команды, а строка `Esc[4;"dir c:";13p` при нажатии комбинации клавиш Ctrl-D (код этой комбинации 4) приводит к выполнению команды dir c: (за строкой "dir c:" указан код клавиши Enter — 13).

Для восстановления исходного значения клавиши необходимо в <определение клавиши> указать ее расширенный код ASCII. Например, `Esc[0;60;0;60p` восстанавливает значение клавиши F2.

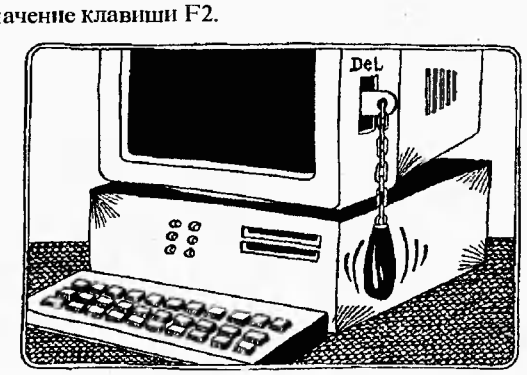


Рисунок О.Попова.

Д.ЯМАЙКИН,
СШ N19, член Школы "Программирование и вычислительная математика" при кафедре ВМиП БГУИР, 220005, г.Минск, пр.Скорины, 39 - 67, тел. 33-97-16.

ПРОГРАММА STINTERV — ПОМОЩЬ ПРОГРАММАМ-РЕЗИДЕНТАМ

Существует ряд программ, в основном игровых, которые для ввода команд не используют стандартную систему ввода/вывода, а изменяют вектор прерывания INT 9H и переводят обработку прерывания на себя, не вызывая старый обработчик (например WINGS OF FURY, DOUBLE DRAGON и др.). Это создает большие неудобства, т.к. при этом не обрабатываются все программы-резиденты, использующие данное прерывание для своей активизации.

Программа STINTERV предназначена для того, чтобы контролировать изменения вектора прерывания INT 9H. Она обрабатывает все резидентные процедуры, использующие данное прерывание, если программа-захватчик загружена.

Алгоритм программы STINTERV основан на использовании таймера (INT 8H) для циклической проверки вектора INT 9H и соответствующей реакции по результатам этой проверки (процедура INT8). Программа способна обнаружить себя в памяти, если пытаться загрузить ее повторно. Программа написана на Ассемблере для компиляции в COM-файл, размер исходного текста (без комментариев) — 2 Кб.

Размер COM-файла — 512 байт, из них в памяти остается 400 байт. При выполнении он не портит содержимого не принадлежащих резиденту ячеек памяти и не занимает свободные вектора прерываний. COM-файл будет работать на любом IBM-совместимом компьютере.

```
CD      SEGMENT PARA            'Code'
        ASSUME  CS:CD,DS:CD,ES:CD,SS:CD
        ORG     100H
BEG:    JMP     MAIN            ;Переход на
                                ;инициализацию
AINT    DW      00              ;Адрес прерывания
AUNT    DW      00              ;INT 9H
FD      DB      00              ;Есть ли в памяти
                                ;программа,
                                ;захватившая INT 9H
                                ; (00=нет)
FN      DB      00              ;Проверка на захват
                                ; (обнуляется
                                ;процедурой     INT9)
INT8    PROC                    ;Обработчик INT 8H
                                ; (таймер)
        PUSHF
CJ1:    NOP                     ;5 байт для вызова
                                ;старого обработчика
        NOP
        NOP
        NOP
        NOP
        PUSH    ES
        PUSH    DS
        PUSH    SI
        PUSH    DI
        PUSH    DX
        PUSH    CX
        PUSH    BX
        PUSH    AX
        MOV     AX,0B902H       ;Запись 2-х байт для
        MOV     ES,AX           ;определения наличия
        SUB     SI,SI           ;резидента в памяти
        MOV     ES:[SI],1234H
        SUB     AX,AX
        MOV     ES,AX
        MOV     SI,24H
        MOV     AX,ES:[SI]      ;Проверка на
                                ;изменение
        MOV     BX,ES:[SI+2]    ;вектора INT 9H
        CMP     AX,CS:AINT      ;и соответствующие
        JNE     A1              ;результату проверки
        CMP     BX,CS:AUNT      ;действия
        JNE     A1
        CMP     CS:FD,00
        JE      J1
        MOV     CS:FD,00
        PUSH    CS
        POP     ES
        LEA     DI,FINDAD       ;Изменения в
                                ;процедуре INT9
        MOV     CS:BYTE PTR [DI],0CFH
J1:     JMP     OK
A1:     MOV     CS:FN,0FFH
        INT     09H
        MOV     AH,07           ;Очистка
                                ;клавиатурного буфера
        INT     21H
        CMP     CS:FN,00
        JE      A2
        MOV     CS:FD,0FFH
        PUSH    CS
        POP     ES
        LEA     DI,FINDAD       ;Изменения в
                                ;процедуре INT9
        MOV     CS:[DI],9A9CH
        SUB     AX,AX
        MOV     ES,AX
        MOV     SI,24H
        MOV     AX,ES:[SI]
        MOV     BX,ES:[SI+2]
        MOV     CS:[DI+2],AX
        MOV     CS:[DI+4],BX
        MOV     CS:BYTE PTR [DI+6],0CFH
        SUB     AX,AX
        MOV     DS,AX
        MOV     AX,CS:AINT
        MOV     BX,CS:AUNT
        DEC     AX
        ADD     BX,10H
        MOV     SI,24H
        CLI
        MOV     [SI],AX
        MOV     [SI+2],BX
        STI
J2:     JMP     OK
A2:     CMP     FD,00
        JNE     J2
        SUB     AX,AX
        MOV     DS,AX
        MOV     SI,24H
        MOV     AX,[SI]
        MOV     BX,[SI+2]
        MOV     CS:AINT,AX
        MOV     CS:AUNT,BX
OK:     POP     AX
        POP     BX
        POP     CX
        POP     DX
```

```
        POP     DI
        POP     SI
        POP     DS
        POP     ES
        IRET                    ;Конец процедуры INT8
INT8    ENDP

INT9    PROC                    ;Фиктивный
                                ;обработчик INT 9H
                                ;(клавиатура)
        PUSHF
CJ2:    NOP                     ;5 байт для вызова
                                ;старого обработчика
        NOP
        NOP
        NOP
        NOP
        PUSH    DS
        PUSH    AX
        PUSH    SI
        CMP     CS:FD,00        ;Если INT 9H
                                ;"захвачено"
        JE      NEX
        SUB     AX,AX
        MOV     DS,AX
        MOV     SI,41AH         ;то очистить буфер
        MOV     AX,[SI]         ;(во избежание
                                ;переполнения)
        MOV     [SI+2],AX
NEX:    MOV     CS:FN,00
        POP     SI
        POP     AX
        POP     DS
FINDAD: IRET
        DD      00              ;6 байт для вызова
                                ;обработчика
        DW      00              ;в программе -
                                ;"захватчике"
INT9    ENDP                    ;Конец процедуры INT9

PR      DB      'Help to resident!',0DH,0AH
        DB      'Programmed by Yamaykin '
        DB      'Dmitry.',0DH,0AH,'$'
AL1     DB      'All rights reserved.',0DH,0AH,'$'
ER1     DB      'Program already loaded.Abnormal '
        DB      'termination.',0DH,0AH,'$'

MAIN    PROC                    ;Процедура
                                ;инициализации
        MOV     AH,09
        LEA     DX,PR
        INT     21H
        MOV     AX,0B902H       ;Проверка на наличие
        MOV     ES,AX           ;программы в памяти
        SUB     SI,SI
        CMP     ES:[SI],1234H
        JNE     GU1
        MOV     AH,09
        LEA     DX,ER1
        INT     21H
        INT     20H
GU1:    MOV     AH,09           ;Инициализация:
        LEA     DX,AL1          ;замена адресов
        INT     21H             ;INT 8H и INT 9H
        MOV     AUNT,CS         ;на свои
        MOV     AX,OFFSET INT9  ;и обеспечение
                                ;вызываемости
        MOV     AINT,AX         ;старых обработчиков
        SUB     AX,AX
        MOV     DS,AX
        MOV     SI,20H
        MOV     AX,[SI]
        MOV     BX,[SI+2]
        MOV     SI,OFFSET CJ1
        MOV     CS:BYTE PTR [SI],9AH
        MOV     CS:[SI+1],AX
        MOV     CS:[SI+3],BX
        MOV     SI,24H
        MOV     AX,[SI]
        MOV     BX,[SI+2]
        MOV     SI,OFFSET CJ2
        MOV     CS:BYTE PTR [SI],9AH
        MOV     CS:[SI+1],AX
        MOV     CS:[SI+3],BX
        MOV     SI,20H
        MOV     AX,OFFSET INT8
        CLI
        MOV     [SI],AX
        MOV     [SI+2],CS
        MOV     SI,24H
        MOV     AX,OFFSET INT9
        MOV     [SI],AX
        MOV     [SI+2],CS
        STI
        MOV     DX,OFFSET PR
        INT     27H             ;Завершить и оставить
MAIN    ENDP                    ;резидентом
CD      ENDS
        END     BEG
```

Д.ШПИЛЕВСКИЙ,
Школа-лицей N165 при БГУИР, 10 "Р" класс,
член Школы "Программирование и вычислительная математика" при кафедре ВМиП БГУИР,
220055, г. Минск, просп.газ."Известия", 14 — 44,
тел. 71-44-02.

ТЕХНОЛОГИЯ РАЗРАБОТКИ РЕЗИДЕНТНЫХ ПРОГРАММ НА ЯЗЫКЕ АССЕМБЛЕРА

На сегодняшний день есть много литературы, излагающей способы построения резидентных программ. Однако некоторые программы, приводимые в качестве примера в различных руководствах, "подвешивают" компьютер, так как содержат множество неточностей. Статья рассчитана на читателя, имеющего по крайней мере поверхностное знакомство с языком ассемблера.

Программа, приводимая здесь, является только примером резидентной программы и не выполняет никаких полезных функций (подаёт звуковой сигнал при нажатии клавиши Scroll-Lock), так как служит только для объяснения принципа работы резидентных программ. Впрочем, для того чтобы изменить выполнение программы, достаточно лишь немного изменить процедуру MY.

Для компиляции приведенной в конце статьи программы нужно использовать компилятор TASM.EXE и компоновщик TLINK.EXE следующим образом:

A:\>TASM.EXE RES.ASM RES.OBJ

A:\>TLINK.EXE RES.OBJ /T

В результате получается COM-вариант программы.

С ЧЕГО НАЧАТЬ ?

Для оптимального использования памяти, занимаемой программой-резидентом, необходимо писать её в COM-формате. Для этого нужно поместить в начале ассемблерного текста программы заголовок следующего вида:

```
CSEG    SEGMENT                  ;Установка
        ASSUME CS:CSEG,DS:CSEG   ;сегментных
        ASSUME ES:CSEG,SS:CSEG   ;регистров

        ORG 100H                 ;Отступ от начала сегмента

BEGIN:  JMP MAIN                 ;Переход на основную
                                       процедуру

        - - - - -                ;
        - - - - -                ;Раздел описаний
        - - - - -                ;

MAIN    PROC NEAR                ;Главная процедура загрузки
```

Сначала описывается сегмент для размещения кода и данных программы и на него устанавливаются все сегментные регистры, для того чтобы и данные, и код располагались в одном сегменте (необходимое условие для COM-программы). После этого резервируется место для PSP (Program Segment Prefix — префикс программного сегмента) с помощью псевдооператора ORG и происходит переход на начало кода загрузки (JMP MAIN).

СЛЕДУЮЩИЙ ШАГ

Резидентной называется программа, сохраняющаяся в памяти компьютера после её завершения. Для того чтобы иметь возможность активизироваться, такая программа должна "захватывать" на себя вектор какого-либо прерывания, чтобы иметь возможность выполниться при вызове этого прерывания.

Для того чтобы перехватить вектор прерывания (в данном случае прерывания от клавиатуры — Int9), необходимо записать в таблицу векторов прерываний вместо старого вектора адрес процедуры, являющейся новым обработчиком этого прерывания. Расчёт смещения в таблице векторов прерываний чрезвычайно прост: нужно умножить номер "перехватываемого" прерывания на четыре и, начиная с полученного смещения, записать адрес нового обработчика в следующем порядке: сначала младший байт смещения, затем старший, а потом адрес сегмента в таком же порядке. Например для прерывания 9 вектор располагается по адресам 36, 37, 38, 39 в десятичной системе (24h, 25h, 26h, 27h в шестнадцатиричной), и если записать в эти байты значения 12h, 13h, 14h и 15h, то при активизации девятого прерывания управление будет передано по адресу 1514h:1312h. Для того чтобы компьютер не "повис", необходимо не просто изменять адрес обработчика, уничтожая старый, а сохранить предыдущий вектор, "повесив" его на какой-либо не используемый операционной системой вектор. Благодаря этому можно после выполнения собственно процедуры обработки обратиться к старому обработчику, что позволяет избежать такой неприятной вещи как блокирование клавиатуры. В данном случае в качестве "резервного" вектора используется вектор прерывания 64 (40h) — на него переназначается вектор девятого прерывания (прерывание от клавиатуры):

```
        MOV DX,0         ;Установка базы
        MOV DS,DX        ;адресации
        MOV SI,36        ;Получение смещения для старого
        MOV AX,[SI]      ;обработчика
        MOV SI,38        ;Получение сегмента
        MOV BX,[SI]
        MOV SI,256       ;
        MOV [SI],AX      ;Установка вектора 9 на
        MOV SI,258       ;вектор 40h
        MOV [SI],BX      ;
```

Для адресации байтов памяти младших адресов ОЗУ необходимо занести в регистр DS ноль, что означает адресацию смещения относительно сегмента 0000. Чтобы прочитать слово (2 байта) по определённому смещению, нужно занести смещение в регистр SI и обращаться к соответствующему ему слову как [SI], что эквивалентно адресу DS:SI. Таким образом переписывается значение девятого вектора на вектор 40h.

После сохранения адреса старого обработчика нужно переназначить вектор девятого прерывания на собственную процедуру обработки:

```
        MOV SI,36          ;
        MOV AX,OFFSET MY   ;Установка смещения
        MOV [SI],AX        ;
        MOV SI,38          ;
        MOV AX,CS          ;Установка сегмента
        MOV [SI],AX        ;
```

ПИШЕМ ОБРАБОТЧИК ПРЕРЫВАНИЯ

Теперь мы подошли к написанию нового обработчика прерывания от клавиатуры. От старого он отличается лишь тем, что при нажатии клавиши Scroll-Lock (ФСД-СТОП) производится подача звукового сигнала посредством вывода символа с ASCII-кодом 7 (Bell). Следовательно, можно написать только процедуру обработки нажатия клавиши Scroll-Lock, а к старому обработчик обращается с помощью вызова прерывания 40h:

```
MY      PROC FAR
        INT 40H          ;Вызов старого обработчика
        IRET             ;Возврат из прерывания
MY      ENDP
```

Эту процедуру нужно поместить в "раздел описаний" (между меткой BEGIN и процедурой MAIN).

Однако такой обработчик ничем не отличается от старого, и чтобы при нажатии клавиши Scroll-Lock выполнялись предусмотренные действия, необходимо перед вызовом старого обработчика (INT 40h) поместить участок, обрабатывающий нажатие клавиши. Теперь процедура MY должна иметь следующий вид:

```
MY      PROC FAR
        PUSH AX      ;Сохранение AX
        IN AL,60H    ;Определение кода
        CMP AL,70    ;нажатой клавиши
        JNE EXIT
                     ;Процедура обработки нажатия
                     ;Scroll-Lock.
        PUSH DX      ;Сохранение DX
        MOV AH,2     ;
        MOV DL,7     ;Вывод седьмого символа (Bell)
        INT 21H      ;
        POP DX       ;Восстановление регистров,
        POP AX       ;вызов старого обработчика
        INT 40H      ;и возврат из прерывания.
        IRET         ;
EXIT:   POP AX       ;
        INT 40H      ;Вызов старого обработчика и выход
        IRET         ;
MY      ENDP.
```

Вначале сохраняется в стеке регистр AX, так как его значение будет меняться в дальнейшем, а обработчику нужно передать его первоначальное значение. Затем из порта 60h

считывается код последней нажатой клавиши, и если он равен 70 (Scroll-Lock), производится подача звукового сигнала, вызывается старый обработчик и возврат из прерывания. В противном случае подача звука не производится (совершается переход на метку EXIT).

Теперь нужно подумать о том, как избежать повторной загрузки нашего резидента. Сделать это можно следующим образом: перед изменением векторов прерываний нужно поместить в какой-либо регистр, например в CX, ключевое значение, скажем 1234h, а в процедуре MY поставить проверку на наличие этого ключевого значения, и если оно будет обнаружено, в регистр AX занести 4321h и выйти из прерывания. Теперь, если резидент уже загружен, после вызова прерывания в AX должно быть значение 4321h:

```
        MY PROC FAR
        CMP CX,1234H   ;Проверка ключа
        JE EXIT1       ;Переход при наличии
        PUSH AX
        IN AL,60H
        CMP AL,70
        JNE EXIT
        PUSH DX
        MOV AH,2
        MOV DL,7
        INT 21H
        POP DX
        POP AX
        INT 40H
        IRET
EXIT:   POP AX
        INT 40H
        IRET
EXIT1:  MOV AX,4321H   ;Занесение значения в AX
        IRET           ;Возврат из прерывания
MY      ENDP
MAIN    PROC NEAR
        MOV CX,1234H   ;Занесение ключа
        INT 9          ;Вызов прерывания
        CMP AX,4321H   ;Проверка на наличие резидента
        JE QUIT        ;
         - - - - - -   ;Изменение
         - - - - - -   ;векторов
         - - - - - -   ;прерываний
QUIT:   RET            ;Выход
MAIN    ENDP
CSEG    ENDS
        END BEGIN
```

ХРАНЕНИЕ ПРОГРАММЫ РЕЗИДЕНТОМ

Для того чтобы сохранить нашу программу в памяти, воспользуемся прерыванием 27h. Для этого нужно занести в регистр DX адрес, начиная с которого программа не должна сохраняться (другими словами, адрес конца обработчика):

```
MAIN    PROC NEAR
         - - - - - -
         - - - - - -
         - - - - - -
        MOV DX,OFFSET MAIN   ;Сохранение резидентом
        INT 27H              ;
QUIT:   RET
MAIN    ENDP
CSEG    ENDS
        END BEGIN
```

В заключение привожу полный листинг программы RES.ASM, дополненный некоторыми сообщениями:

```
CSEG    SEGMENT
        ASSUME CS:CSEG,DS:CSEG,ES:CSEG,SS:CSEG
        ORG 100H
BEGIN:  JMP MAIN
MY      PROC FAR
        CMP CX,1234H
        JE EXIT1
        PUSH AX
        IN AL,60H
        CMP AL,70
        JNE EXIT
        PUSH DX
        MOV AH,2
        MOV DL,7
        INT 21H
        POP DX
        POP AX
        INT 40H
        IRET
EXIT:   POP AX
        INT 40H
        IRET
EXIT1:  MOV AX,4321H
        IRET
MY      ENDP
MES1    DB 'Resident already installed.',
        DB 0AH,0DH,'$'
MES2    DB 'Resident Demo.Copiright(C)'
        DB 'Claws Software,'
        DB '1995.Use Scroll-Lock for activate.',
        DB 0AH,0DH,'$'
MAIN    PROC NEAR
        MOV CX,1234H
        INT 9
        CMP AX,4321H
        JE QUIT
        MOV AH,9
        MOV DX,OFFSET MES2
        INT 21H
        MOV DX,0
        MOV DS,DX
        MOV SI,36
        MOV AX,[SI]
        MOV SI,38
        MOV BX,[SI]
        MOV SI,256
        MOV [SI],AX
        MOV SI,258
        MOV [SI],BX
        MOV SI,36
        MOV AX,OFFSET MY
        MOV [SI],AX
        MOV SI,38
        MOV AX,CS
        MOV [SI],AX
        MOV DX,OFFSET MAIN
        INT 27H
QUIT:   MOV AH,9
        MOV DX,OFFSET MES1
        INT 21H
        RET
MAIN    ENDP
CSEG    ENDS
        END BEGIN
```

От редакции. *На время перестановки векторов прерываний рекомендуется запрещать прерывания командой CLI.*

А.ИВАНЧИКОВ,
БГУИР, ФИТиУ, каф.информационных технологий и автоматизированных систем,
г.Минск, тел. 39-88-23.

"СИМБИОЗ" FoxPro и Assembler

Чем больше знаешь, тем больше можешь.
Э. Абу

Любая система управления базами данных (СУБД), в том числе и FoxPro, — это, прежде всего, наличие мощного высокоуровневого аппарата, направленного на обработку данных, и как следствие — недостаточное внимание к вопросам системного программирования.

При написании подавляющего большинства простых программ на входном языке СУБД FoxPro вполне достаточно оперировать лишь широким многообразием имеющихся в распоряжении программиста встроенных команд и функций языка. Разрабатывая современный пакет программ под управлением СУБД FoxPro, если необходимо иметь полный набор средств интерактивного диалога с пользователем и контроля всевозможных ошибок, крайне трудно, а порой и невозможно обойтись без небольших сервисных программ на Ассемблере. СУБД FoxPro обладает простым интерфейсом с языком Ассемблера, позволяющим загружать в память СУБД двоичный код программы и при необходимости выполнять его как дальнюю процедуру.

Соглашения

Весь интерфейс FoxPro и Ассемблера поддерживают в FoxPro всего три команды:

а) LOAD <имя_файла> — считывает двоичный файл с диска и загружает его в память компьютера;

б) CALL <имя_файла> WITH <идентификатор_переменной> — вызывает двоичную программу, передавая ей аргументом адрес переменной-параметра;

в) RELEASE MODULE <имя_файла> — освобождает память компьютера от двоичного кода.

В тоже время, программы на Ассемблере должны удовлетворять следующим условиям:

а) первая выполняемая инструкция должна быть загружена со смещением 0;

б) адрес параметра передается функции на Ассемблере в регистре BX относительно сегментного регистра DS;

в) бинарная программа не должна изменять содержимое памяти за пределами переменных памяти, передаваемых ей в виде аргументов;

г) размер используемой или размещаемой памяти не может превышать фактический размер программы, так как размер программы используется для определения полного объема памяти, который нужно выделить;

д) регистры SS и CS должны быть восстановлены до того как управление будет передано FoxPro;

е) для передачи управления FoxPro должен быть использован дальний возврат.

Подготовка двоичной программы

Создание бинарного модуля начинается с набора соответствующей программы на Ассемблере в любом текстовом редакторе (предположим, программный файл называется SIMPLE.ASM). Сегментные директивы (упрощенные или стандартные) должны быть указаны как для создания .COM файла.

После вызова компилятора имеется объектный файл данной программы — SIMPLE.OBJ (если в исходном тексте не было допущено ошибок). Для получения объектного файла можно воспользоваться командой tasm simple для вызова Turbo Assembler.

На следующем шаге необходимо отработать компоновщику. Команда tlink simple создает файл SIMPLE.EXE (запуск этого файла из системной среды разрушит ее).

И наконец, после выполнения команды exe2bin simple получается файл SIMPLE.BIN, являющийся конечным продуктом, адаптированным для работы в среде СУБД FoxPro.

Определение времени создания файла

Надежность хранения и использования информации является одним из обязательных элементов каждой "законченной" программы. Надежность информации подразумевает: периодическое сохранение информационных файлов на внешних носителях (гибких дисках, ленте и т.д.); сохранение во время сеанса работы текущих установок на твердом магнитном диске; автоматическое корректное закрытие всех открытых файлов при завершении работы; восстановление структуры потерянных файлов базы данных (ФБД); своевременное переиндексирование ФБД (в случае необходимости).

Пакеты программ, использующие большое количество ФБД и связанных с ними индексных файлов, становятся уязвимыми при изменении ФБД без открытых индексов (это может произойти при автономной работе с ФБД без загрузки управляющей программы).

Для определения данной ситуации в начале каждого сеанса работы следует сравнивать времена последнего изменения ФБД и соответствующего ему индекса.

Это может быть, например, такая программа:

```
  load datetime
* создание списка используемых файлов
dime FileName(4)
FileName(1) =  "path.dbf "
FileName(2) =  "path.cdx "
FileName(3) =  "main.dbf "
FileName(4) =  "main.cdx "
* проверка наличия файлов и соответствующее действие
 for i=1 to 4 step 2
 fn1 = FileName(i)
 fn2 = FileName(i+1)
 cmp = compare(fn1,fn2)
 do case
  case cmp = 0   * каких-либо файлов не существует
   wait  "Rebuilding! " wind nowa
   do build      * восстановление
  case cmp = 1   * устаревший индекс
   wait  "Reindex file  " + FileName(i) +  "! ";
  wind nowa
   use &FileName(i)
   rein          * переиндексация
  case cmp = 2   * все в норме
   use &FileName(i)
 endc
endf
rele modu datetime
```

```
* Дальнейшие действия ...
* Сравнение времен создания двух файлов
*
* параметры - имена файлов, заканчивающиеся
* символом с кодом 0                .
* возвращаемые значения: 0 - файлов не существует
*                        1 - первый файл создан*
                         раньше второго
*                        2 - второй файл создан*
                         раньше первого
func compare
para _file1, _file2
* длина передаваемой строки должна быть не менее
* 13 символов
 _file1 = _file1 + chr(0) + spac(12 - len(_file1))
 _file2 = _file2 + chr(0) + spac(12 - len(_file2))
  call datetime with _file1
  call datetime with _file2
* если строка не изменилась - ошибка
  if isal(_file1) .or. isal(_file2)
   retu 0
  endi
* если двоичная процедура отработала нормально,
* имя файла содержит строку в виде 'гггммддччммсс'
retu iif(val(_file1) > val(_file2),1,2)
```

Ассемблерный файл DATETIME.ASM имеет вид:

```
  CSEG SEGMENT
  ASSUME cs:CSEG
Start_pr:
              ;- открыть файл
  mov dx,bx
  mov ax,3d00h
  int 21h
  jc short ErrorOpen
              ;- получить дату и время его создания
  push ax
  push bx
  mov bx,ax
  mov ax,5700h
  int 21h
  jc short ErrorRead
  pop bx
  push bx
  push cx
  inc bx
              ;- вывести дату
  push dx
  and dx,0fe00h
  mov cl,9
  shr dx,cl
  add dx,50h
  mov ax,dx
  call Dessist
  add bx,4
  pop dx
  push dx
  and dx,01e0h
  mov cl,5
  shr dx,cl
  mov ax,dx
  call Dessist
  add bx,4
  pop dx
  mov ax,001fh
  and dx,ax
  mov ax,dx
  call Dessist
  add bx,4
              ;- вывести время
  pop cx
  push cx
  and cx,0f800h
  mov ax,cx
  mov cl,0bh
  shr ax,cl
  call Dessist
  add bx,4
  pop cx
  push cx
  and cx,07e0h
  mov ax,cx
  mov cl,5
  shr ax,cl
  call Dessist
  add bx,4
  pop cx
  mov ax,1fh
  and cx,ax
  mov ax,cx
  call Dessist
              ;- закрыть файл .
ErrorRead:
  pop cx
  pop bx
  mov ah,3eh
  int 21h
  mov bx,cx
ErrorOpen:
  retf
 ; функция перевода числа в строку символов
Dessist PROC NEAR
  mov si,10
  mov cx,2
Next_integ:
  sub dx,dx
  div si
  add dx,30h
  mov [bx],dl
  dec bx
  loop Next_integ
  ret
ENDP Dessist
  CSEG ENDS
  END Start_pr
```

В FoxPro, начиная с версии 2.0, имеется функция ADIR(), заполняющая массив информацией из оглавления каталога. Программисты на FoxPro 2.0 и выше могут сопоставить эту разницу во времени работы программ, реализованных с помощью этих двух методов — результат будет отнюдь не в пользу "чистого" FoxPro.

Получение списка каталогов на диске

Для быстрого и, главное, удобного выбора пользователем поддиректории логического устройства для операций ввода/вывода необходимо представлять список имеющихся каталогов в виде меню.

В поздних версиях FoxPro в распоряжении программиста в этом случае имеется упомянутая выше функция ADIR(). Однако, как и в предыдущем случае, более "быстрой" заменой ей является программа на Ассемблере, использующая функции DOS для поиска первого/следующего файла по маске (называется SHOWCAT.ASM):

```
  CSEG SEGMENT
  ASSUME cs:CSEG
Start_pr:
  jmp short Main_pr
DtaAdr DW 0   ;смещение начала области
              ;умалчиваемой DTA
DtaSeg DW 0   ;сегмент области умалчиваемой DTA
BxZn   DW 0   ;адрес параметра
Main_pr:
  push bx
  mov cs:[BxZn],bx
  xor si,si
  mov al,30h
  cmp [bx],al
  jne NextDirFind
              ;- получить адрес умалчиваемой DTA
  push bx
```

```
  push es
  mov ah,2fh
  int 21h
  mov cs:[DtaAdr],bx
  mov ax,es
  mov cs:[DtaSeg],ax
  pop es
  pop bx
              ;- установить адрес новой DTA
  mov ah,1ah
  mov dx,bx
  int 21h
              ;- поиск первого каталога
  mov dx,bx
  inc dx
  mov ah,4eh
  mov cx,10h
  int 21h
  jc Errors
              ;- имя каталога в строку
Next_Dir:
  inc si
  add bx,15h  ;смещение к атрибуту
  mov al,10h
  cmp [bx],al ;если это не каталог
  jne Errors  ;ничего не делать
  add bx,9    ;смещение к началу имени
  mov al,2eh
  cmp [bx],al
  jne This_no_dot
  cmp [bx+1],al
  jne short Errors
This_no_dot:
  inc si
  mov cx,0ch  ;длина имени каталога - 12 символов
  sub ah,ah
Next_plus:
  cmp ah,0
  jne Del_define
  mov dl,0
  cmp [bx],dl
  jne Character
  mov ah,20h
Del_define:
  add bx,0eh
  mov [bx],ah
  sub bx,0eh
  jmp short No_no
Character:    ;копировать имя каталога
  mov dl,[bx] ;в строку, за исключением
  add bx,0eh
  mov [bx],dl ;разделяющей точки
  sub bx,0eh
No_no:
  inc di
  inc bx
  loop Next_plus
  jmp short Errors
NextDirFind:
              ;- поиск следующего каталога
  mov bx,cs:[BxZn]
  xor si,si
  mov dx,bx
  mov ah,4fh
  int 21h
  jnc Next_Dir
  xor si,si
Errors:
  cmp si,0
  je Free_p
  cmp si,1
  jne short Con_end
  jmp short NextDirFind
Free_p:
  add bx,2ch
  mov cx,0ch
  mov ah,20h
Next_space:
  mov [bx],ah
  inc bx
  loop Next_space
              ;- установить адрес умалчиваемой DTA
  push ds
  mov dx,cs:[DtaAdr]
  mov ax,cs:[DtaSeg]
  mov ds,ax
  mov ah,1ah
  int 21h
  pop ds
Con_end:
  pop bx
  retf
  CSEG ENDS
  END Start_pr
```

В приведенной программе интересен, прежде всего, метод выделения памяти для области DTA, с которой работают функции поиска файлов 4Eh, 4Fh. Область DTA распределяется в переданной параметром строке при первом вызове (в ней первоначально задан путь поиска директорий) и используется затем при последующих вызовах. Поэтому размер передаваемой параметром строки устанавливается равным 60 символам. Имена найденных каталогов помещаются в этой строке со смещением 45 и заканчиваются пробелом. Если каталогов больше нет, по смещению 45 находится 12 пробелов.

Фрагмент программы на входном языке FoxPro, использующей вызов приведенной процедуры, может выглядеть следующим образом:

```
load showcat
colcat = 0
* входная строка
cat =  "0d:\tc\include\*.* " + chr(0) + spac(50)
call showcat with cat
* выделение имени каталога
strTmp = subs(cat,45,at(chr(32),subs(cat,45,13))-1)
* повторять пока еще есть каталоги
do whil len(_strTmp) != 0
 colcat = colcat + 1
 dime subdir(colcat)
 subdir(colcat) = _strTmp
 call showcat with cat
_strTmp = subs(cat,45,at(chr(32),subs(cat,45,13))-1)
endd
rele modu showcat
* В массиве subdir перечислено colcat каталогов
* Дальнейшие действия ...
```

Определение количества установленных на компьютере дисководов

Для того чтобы иметь возможность выбора активного дискового накопителя из меню (по аналогии с Norton Commander), следует знать количество накопителей на гибких магнитных дисках (НГМД) и количество логических устройств на жестком магнитном диске.

Простейшая программа на Ассемблере позволяет получить эту информацию. Ниже приводится исходный текст программы с необходимыми комментариями (имя — MAXDRIVE.ASM):

```
;принимает строку из трех элементов и заполняет ее
;по следующему формату:
;         1-й символ  - количество НГМД,
;         2-3 символы - общее количество
;               логических дисков
  CSEG SEGMENT
  ASSUME cs:CSEG
Start_pr:
              ;- получить количество гибких дисков
```

```
  int 11h
  push ax
  and ax,1
  pop ax
  jz NoFloppy
  and ax,0c0h
  mov cl,6
  shr ax,cl
                ;— вывести количество гибких дисков
  add al,31h
  mov [bx],al
  jmp short YesFloppy
NoFloppy:
  mov [bx],byte ptr 0
YesFloppy:
  inc bx
  inc bx
                ;— получить текущий диск
  mov ah,19h
  int 21h
                ;— получить количество устройств
  mov ah,0eh
  mov dl,al
  int 21h
  xor ah,ah
  call DesSist
  retf
  CSEG ENDS
  END Start_pr
```

Вызываемая в этой программе функция DesSist полностью идентична одноименной функции, приведенной в одном из предыдущих разделов.

Фрагмент программы на входном языке СУБД FoxPro, вызывающий рассматриваемую программу на Ассемблере, выглядит так:

```
load maxdrive
NumberDrive = spac(3)
call maxdrive with NumberDrive
* количество накопителей на гибких дисках и общее
* количество дисководов
Gmd = int(val(subs(NumberDrive,1,1)))
All = int(val(subs(NumberDrive,2,2)))
rele modu maxdrive
* Дальнейшие действия ...
```

Определение готовности НГМД

Перед выполнением операций ввода/вывода с/на НГМД необходимо определять их готовность и в случае любой из ошибок (нет дискеты в дисководе, дисковод открыт, ошибка чтения дискеты) предупреждать о них пользователя.

Показательна в этом плане программа, имеющаяся в виде двоичного модуля на дистрибутивных дискетах FoxPro — ISDISKIN.BIN. Ниже приводится ее исходный текст:

```
;получает параметром символ — имя дисковода
;в случае ошибки возвращает вместо этого символа
;символ '0'
  CSEG SEGMENT
  ASSUME cs:CSEG
Main:
  int 11h               ;получить список
                        ;оборудования
  mov dx,ax
  and dl,1
  jz Endss              ;если нет дисководов,
                        ;закончить
  mov cl,6
  shr ax,cl             ;получить количество
                        ;дисководов
  and ax,3
  mov dl,[bx]
  and dl,0dfh
  sub dl,41h            ;получить номер дисковода
                        ;из символа
  cmp al,dl
  jl Endss              ;если такого дисковода
                        ;нет, закончить
  mov cx,2
Two_it:
  push cx
  mov ax,401h
  mov dh,0
  mov cx,1
  int 13h               ;проверить готовность
                        ;дисковода
  pop cx
  loop Two_it           ;читать два раза
  jnc Endss
  mov [bx],byte ptr 30h
                        ;если не готов -
                        ;возвратить '0'
Endss:
  retf
  CSEG ENDS
  END Main
```

Вызов данной процедуры может осуществляться из программы на FoxPro примерно следующим образом:

```
load isdiskin
Drv = 'a'
* только для гибких дисков
if inli(uppe(Drv),'A','B')
 =isdisk(Drv)
endi
rele modu isdiskin
* Дальнейшие действия ...
func isdisk
para Drive
 do while .t.
    Dd=Drive
    call isdiskin with Dd
    if Dd='0'
       wait 'Вставьте дискету в дисковод '+Drive+;
            ': и нажмите любую клавишу...' wind
    else
       return
    endi
 endd
```

Операции со строками

СУБД FoxPro имеет большой набор функций для работы со строками. Но, во-первых, обработка алфавитных символов ориентирована только на принадлежащие таблице ASCII и не распространяется на символы из расширенной таблицы, где, в частности, и расположены символы кириллицы; во-вторых, все многообразие операций над строками охватить просто невозможно; в-третьих, скорость работы FoxPro с возрастанием длины обрабатываемой строки падает по экспоненте.

Все это позволяет говорить о том, что и в области обработки символьных выражений (где преимущество Ассемблера перед другими языками не вызывает сомнения) поле для деятельности программиста на Ассемблере остается обширным.

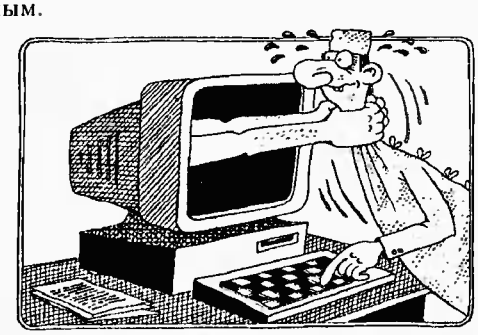


Рисунок О.Попова.

А.ВОРОТНИКОВ.

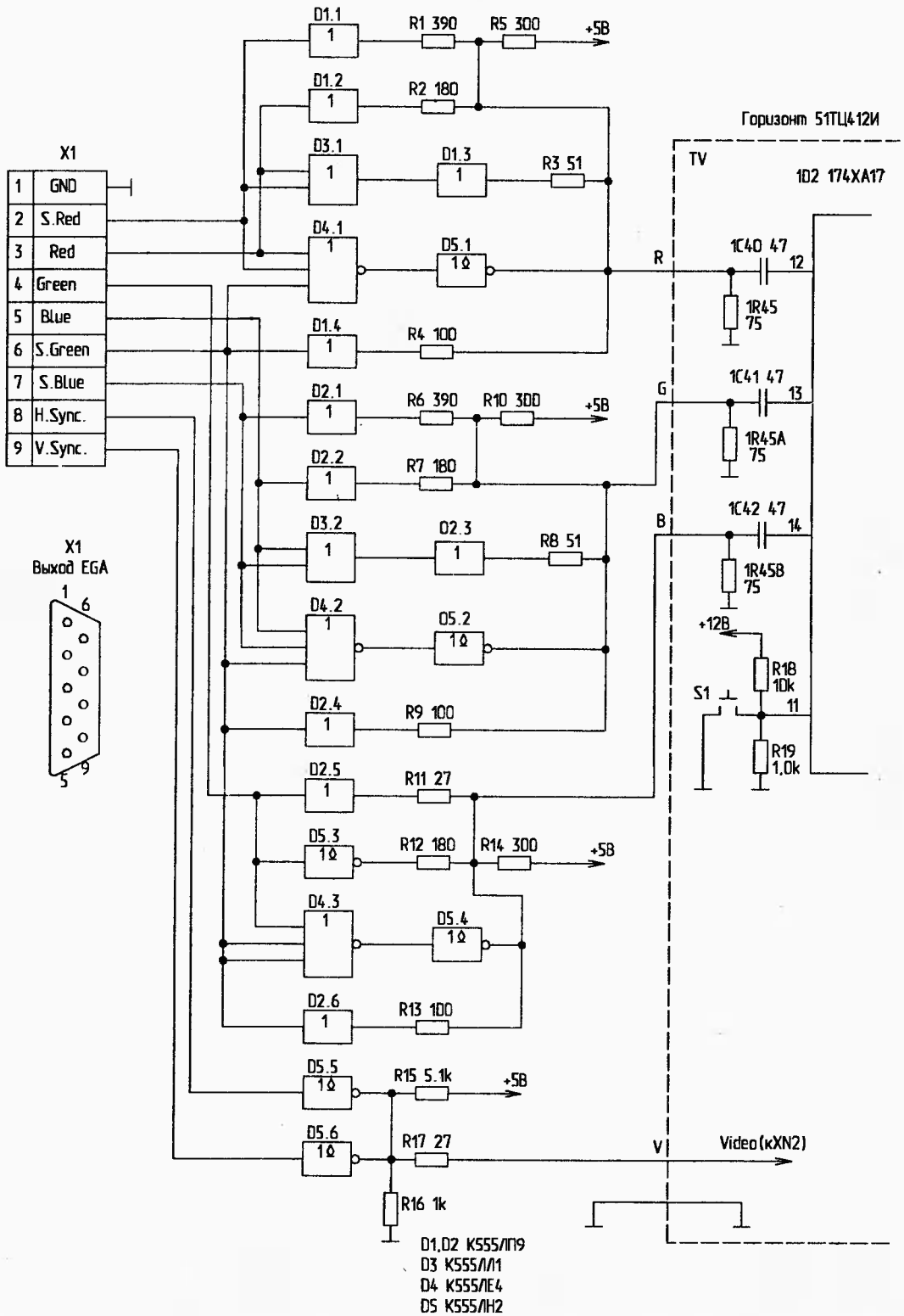
ПОДКЛЮЧЕНИЕ EGA-АДАПТЕРА К ЦВЕТНОМУ ТЕЛЕВИЗОРУ

Представляю схему, не требующую никакой настройки, где правильно передаются все 64 цвета.

Номиналы резисторов подобраны так, чтобы максимальная амплитуда сигналов (когда все буферные элементы закрыты) была чуть больше 1 В с учетом того, что в телевизоре линии RGB соединены с корпусом через резисторы 75 Ом. D4 и D5 нужны для того, чтобы получить наименьший уровень сигнала (уровень черного). D1...D3 создают пониженные уровни сигналов (градации серого). Напряжение питания +5 В взято из компьютера.

Адаптер EGA настраивается в режим 640x200 точек и "enhanced RGB" (положение переключателей — 11101). Соответственно надо использовать знакогенератор с матрицей символов 8x8.

Для получения черно-белого сигнала можно позаимствовать схему от "Спектрума": соединить линии RGB в одну точку через резисторы 1 кОм, 480 Ом, 2 кОм соответственно, добавить синхросигналы и усилить транзистором.



М.РОМАНОВ, А.ПЕТРОВ,
198261, г.С.-Петербург, а/я 213.

ДРАЙВЕР "МЫШИ" ДЛЯ "СПЕКТРУМА"

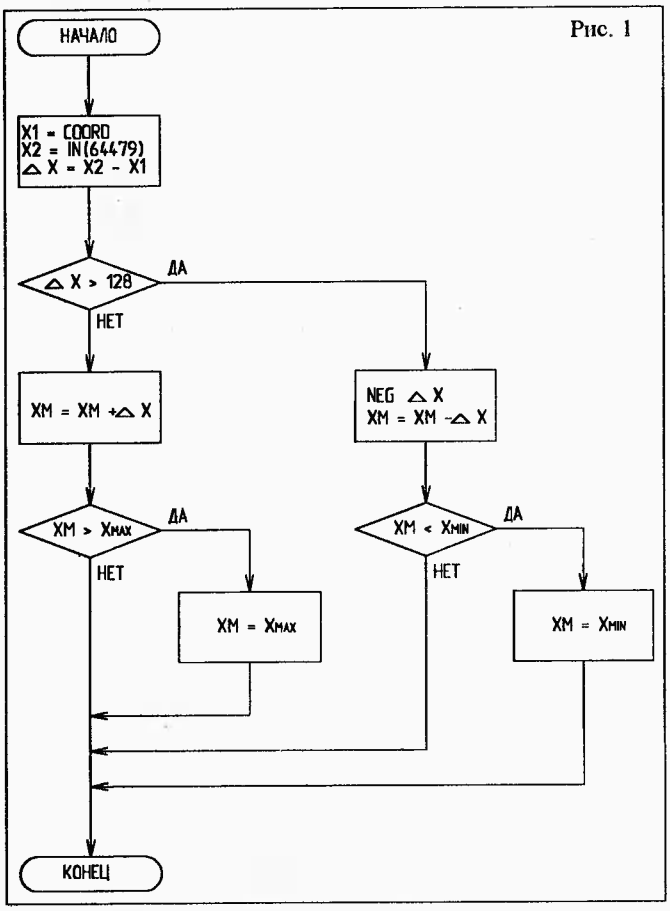
Многие хотели бы более быстро и удобно работать с программным обеспечением "Спектрума". Эту задачу отчасти решает манипулятор типа "мышь". Бытовые компьютеры Amiga, PC, Apple имеют возможность использовать этот интерфейс. "Спектрум" также имеет стандартные интерфейсы "мыши", хотя они и не распространены у нас в стране.

Известны два типа интерфейсов, работающих на "Спектруме": AMX MOUSE и KEMPSTON MOUSE. Первый — более "загадочный" с точки зрения анализа программ, написанных для него. По нашим сведениям, в нем используется контроллер прерываний.

Более простым и доступным является KEMPSTON MOUSE (KM). При подключении контроллера "мыши" в компьютере появляются три порта. Порт #FBDF (64479) — координата X, порт #FFDF (65503) — координата Y. Диапазон значений, считываемых из этих портов, лежит в пределах от 0 до 255. Из порта #FADF (64223) считывается состояние кнопок.

Координаты курсора при управлении "мышью" вычисляются по следующему алгоритму. Считывание информации из портов производится через промежутки времени t1, t2...tn. Желательно, чтобы интервалы между циклами считывания информации из портов были одинаковыми и не очень большими. Почему — будет сказано ниже. Для примера разберем работу одного канала, например X (второй канал — Y — почти идентичен).

Рис. 1



При считывании информации из порта в момент времени t1 канал X имеет значение X1 в интервале 0...255, а в момент времени t2 — X2 в том же интервале. Если в этот промежуток времени манипулятор не двигался, X1 и X2 равны.

Для определения величины перемещения и его направления воспользуемся следующим алгоритмом (рис.1):

1. Определим разность ΔX=X2-X1 (X принимает значения от 0 до 255);

2. Сравним ΔX и 128;

3. Если ΔX<128, происходит увеличение текущей координаты на ΔX; если ΔX>128, уменьшение текущей координаты на ΔX.

В данной блок-схеме реализован режим ограничения координат. Задавая переменные Xmax и Xmin, можно ограничить "окно", в котором двигается курсор. Переменные, введенные в данный алгоритм, имеют следующие значения:

COORD — "старое" значение порта X;

XM — координаты курсора.

В большинстве программ отсчет координат происходит сверху вниз. Если в алгоритме в канал X поставить значения канала Y, то отсчет координат будет происходить снизу вверх, что не совсем удобно. Поэтому в алгоритм канала Y необходимо ввести изменения. С учетом всего вышесказанного приводим текст драйвера на языке Ассемблер для каналов X и Y. Входные данные — координаты XM и YM, они же являются и выходными.

Подпрограмма MOUSST в тексте драйвера осуществляет ускоренное перемещение курсора при увеличении скорости движения манипулятора. Входная величина — приращение координат; если приращение больше определенной величины, оно удваивается. Можете поэкспериментировать с этим режимом, меняя значение, с которым сравнивается шаг. При ненадобности этот режим можно отключить, поставив вместо NOP команду RET или повысив величину, на которой происходит ускорение, до числа больше 128. При минимизации размера драйвера возможно исключение обращений к подпрограмме MOUSST и исключение самой подпрограммы.

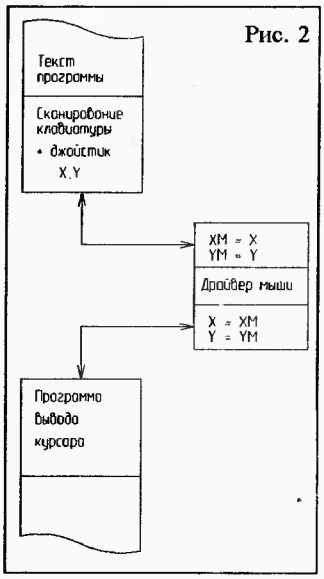
Данный интерфейс требует минимум времени процессора, но накладывает ограничения на время между обращениями к портам координат. Если за время между обращениями манипулятор перемещается более чем на 128 точек (на практике 1 точка соответствует 0,5...0,6 мм, 128 точек — 60...80 мм), возникает ошибка в определении шага и направления. Легко посчитать, что при обращении 50 раз в секунду максимальная скорость перемещения мыши равна 60 мм x 50 сек$^{-1}$ = 3000 мм/сек=3 м/с. Такая скорость практически не используется, поэтому это ограничение несущественно.

Положение кнопок можно определить считав информацию из порта #FADF: 1 — кнопка не нажата, 0 — нажата. Расположение по битам такое: бит 0 — левая кнопка, бит 1 — правая кнопка, бит 2 — средняя кнопка.

Если вы подключаете "мышь" с одной кнопкой, то логично подключить ее к биту 0 — это "выбор"; если кнопок две, вторую (правую) кнопку надо подключить на бит 1 — "отмена" или вторая функция (часто функция ESCAPE из меню); при трех кнопках третья (средняя) подключается к биту 2.

Для совмещения программного обеспечения по клавишам рекомендуется использовать бит 0 как "огонь". Наберите текст драйвера в редакторе Ассемблер, задайте адрес ассемблирования, ассемблируйте текст в объектный код и определите адреса переменных XM, YM. В MONS4

Рис. 2



для этого надо установить ключ 1 в команды ассемблирования. Небольшая программа на Бейсике наглядно показывает, как свʼязаны переменные XM, YM с перемещением манипулятора по столу:

```
10 RANDOMIZE USR
ADDR: PLOT PEEK (XM), 176-
(YM):GO TO 10
```

Напоминаем, что драйвер считает начало координат в левом верхнем углу, а так как в Бейсике нули координат — левый нижний угол, необходимо из 176 вычитать значение YM. При запуске программы на экране отображаются точки в координатах XM, YM. Дополнив программу проверкой нажатой клавиши, можно написать маленький графический редактор.

При переделке уже готовых программ или написании своих под "мышь" возможно использование следующего способа: в тексте программы найти подпрограмму обработки клавиатуры или джойстика (выходными данными этих программ являются координаты курсора). Эти данные могут являться входными для драйвера "мыши". Выходные данные "мыши" записываются вместо выходных данных с программы обработки клавиатуры. Схема на рис.2 показывает пример подключения драйвера к программе.

Обработку нажатой клавиши на "мыши" удобно использовать там же, где проверяется "огонь" джойстика либо "Space" или "Enter" клавиатуры. Авторами уже адаптировано под описываемый интерфейс около 2,5 Мб программного обеспечения и работа продолжается.

Использование "мыши" в ваших программах сделает их более удобными в обращении. Желаем успехов.

```
 ORG 65000
MOUSE LD HL,(COORD)
 LD BC,#FBDF
 IN A,(C)
 LD (COORD+1),A
 SUB H
 CP #80
 JP NC,MOUSE1
 CALL MOUSST
 LD H,A
 LD A,(XM)
 ADD A,H
 JP C,MOUSE2
XMAX CP 255
 JP C,MOUSE3
MOUSE2 LD A,(XMAX-1)
 JP MOUSE3
MOUSE1 NEG
 CALL MOUSST
 LD H,A
 LD A,(XM)
 SUB H
 JP C,MOUSE4
XMIN CP 1
 JP NC,MOUSE3
MOUSE4 LD A,(XMIN+1)
MOUSE3 LD (XM),A
 LD BC,#FFDF
 IN A,(C)
 LD (COORD),A
 SUB L
 CP #80
 JP NC,MOUSE5
 CALL MOUSST
 LD L,A
 LD A,(YM)
 SUB L
 JP C,MOUSE6
YMAX CP 3
 JP NC,MOUSE7
MOUSE6 LD A,(YMAX+1)
 JP MOUSE7
MOUSE5 NEG
 CALL MOUSST
 LD L,A
 LD A,(YM)
 ADD A,L
 JP C,MOUSE8
YMIN CP 173
 JP C,MOUSE7
MOUSE8 LD A,(YMIN+1)
MOUSE7 LD (YM),A
 RET
MOUSST NOP
 CP 10
 RET C
 SLA A
 RET
COORD DEFB 128,128
XM NOP
YM NOP
```

В.ЕРМОЛЕНКО (EW1OM),
220012, Минск, ул. Ф.Скорины, 7 - 23,
тел. (0172) 39-59-51.

МОРЗЕ - ДАМП

Среди читателей журнала "Радиолюбитель" немало коротковолновиков и связистов, знакомых с телеграфной азбукой Морзе. Их непременно заинтересует программа, позволяющая самыми простыми средствами превратить свой компьютер "Радио-86РК" в исполнительного собеседника. Программа Морзе-Дамп предназначена для ввода в компьютер шестнадцатиричных дампов программ, которые публикуются на страницах журналов в виде распечаток, с помощью телеграфного ключа, а также для прослушивания содержимого памяти компьютера. Дело в том, что набор подобного дампа на клавиатуре и последующий поиск сделанных ошибок — занятие весьма трудоемкое. Звучащий компьютер позволяет оператору сосредоточить взгляд на распечатке, а также внести в утомительную работу элемент игры или тренировки. Предлагаемый подход может оказаться особенно полезным для операторов - инвалидов по зрению.

Телеграфный ключ, используемый для работы данной программы, представляет собой манипулятор с двумя замыкающими контактами, аналогичный манипуляторам для полуавтоматических телеграфных ключей. При нажатии манипулятора в одну сторону (например, большим пальцем руки) программно вырабатывается серия точек, в другую (указательным пальцем) — серия тире. Контакты подключаются к клавиатуре компьютера параллельно служебным клавишам "УС" и "НР", замыкая при нажатии соответствующие линии порта на общий провод. Эти же клавиши могут использоваться для проверки работоспособности программы при отсутствии манипулятора. Служебная клавиша "РУС" задействована в программе для возврата в режим управления с клавиатуры.

Машинные коды программы приведены в конце статьи. Программа размещается в верхней части ОЗУ компьютера начиная с адреса 6400H, чтобы освободить место для набираемого текста. Контрольная сумма программы — 61F7H. Особенностью программы для "Радио-86РК" является способ извлечения звука путем переключения состояния линии INTA микропроцессора (командами EI и DI), а также анализ состояния манипулятора и служебных клавиш путем опроса порта клавиатуры по адресу 8002H. В остальном программа использует только стандартные обращения к Монитору "Радио-86РК".

Работа с программой напоминает работу с соответствующими директивами Монитора. По директиве "M" программа запрашивает адрес первой ячейки для изменения, а затем предлагает изменить ее содержимое с использованием телеграфного ключа. Таблица кода Морзе составлена так, что цифру "0" можно вводить и как "T", а цифру "9" — и как "N". Если по ошибке была введена комбинация, отсутствующая в таблице кода Морзе, оператора предупреждает звуковой сигнал низкого тона и на экране появляется знак вопроса. В таком случае следует повторно ввести новое значение байта (2 символа). Символ "дробь" (-..-.) вызывает возврат к предыдущей ячейке, символ "знак раздела" (-...-) — переход к следующей ячейке без изменения. Символ "перебой" (...-.) служит для исправления неверно введенного символа. Для этого же достаточно ввести последовательность из более чем 6 точек или тире.

По директиве "D" программа также запрашивает адрес начала выводимой области, а затем начинает передавать дамп кодом Морзе с одновременным выводом его на экран. Прекратить звучание можно как и в предыдущем случае — нажав клавишу "РУС".

Директива "S" предусмотрена для установки тона звучания и скорости передачи азбуки Морзе. В связи с особенностями организации вывода звука на "Радио-86РК" и с целью упрощения, в данной программе выполняется установка двухбайтной константы, в которой младший (второй при вводе) байт определяет тон звука (полупериод), а старший (первый) — длительность данного тона (количество циклов повторения периодов). Радиолюбитель может самостоятельно установить на своем компьютере наиболее приемлемый тон и прокалибровать скорость звучания по секундомеру. Чтобы не выполнять эту установку каждый раз, новое значение константы можно вписать в ячейки по адресам 6758H, 6759H.

Для завершения работы программы предусмотрена директива "." (точка).

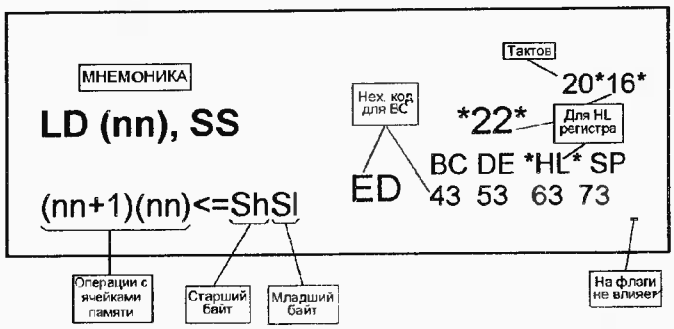
```
6400 21 BD 66 CD 18 F8 CD 03 F8 4F CD 09 F8 79 E6 5F
6410 FE 44 CA 5F 64 FE 4D CA 19 65 FE 53 CA 35 64 FE
6420 0E CA 6C F8 FE 0D C2 2F 64 21 DC 66 CD 18 F8 CD
6430 27 66 C3 00 64 21 3F 67 CD 18 F8 2A 58 67 CD 5A
6440 66 CD 53 64 3A 5A 67 32 58 67 3A 5B 67 32 59 67
6450 C3 00 64 21 53 64 22 89 65 21 4F 67 C3 E3 64 CD
6460 DA 64 2A 5A 67 CD 57 66 C3 80 64 7D E6 7F CC 4C
6470 66 E6 0F CA 62 64 E6 03 C2 80 64 0E 20 CD 09 F8
6480 7E CD 15 F8 0E 20 CD 09 F8 4E 41 78 E6 0F 47 79
6490 E6 F0 0F 0F 0F 0F 4F CD B6 64 CD 35 66 48 CD B6
64A0 64 CD 35 66 CD 35 66 23 22 5A 67 CD FB 65 E6 80
64B0 CA 9F 65 C3 6B 64 E5 21 AD 66 79 85 6F 5E E1 16
64C0 09 15 C8 7B 07 5F D2 C1 64 15 C8 7B 07 5F DC 0E
64D0 66 CD 11 66 CD 38 66 C3 C9 64 21 DA 64 22 89 65
64E0 21 36 67 CD 18 F8 16 02 21 5C 67 CD 55 65 07 07
64F0 07 07 77 23 CD 55 65 77 23 15 C2 EB 64 CD 4C 66
6500 16 02 32 5A 67 2B 7E 2B 86 15 C2 02 65 32 5B 67
6510 21 22 65 22 89 65 C3 2E 66 21 09 67 CD 18 F8 CD
6520 DA 64 2A 5A 67 CD 57 66 7E CD 15 F8 0E 20 CD 09
6530 F8 CD A8 65 CD 5C 65 07 07 07 07 32 5C 67 CD A8
6540 65 CD 5C 65 E5 21 5C 67 86 E1 77 23 22 5A 67 CD
6550 4C 66 C3 22 65 CD 03 F8 4F CD 09 F8 79 D6 30 DA
6560 6F 65 FE 0A D8 D6 07 FE 0A DA 6F 65 FE 10 D8 F1
6570 79 FE FF CA 9F 65 FE 2F CA 8B 65 FE 2D CA 98 65
6580 CD 27 66 0E 3F CD 09 F8 C3 22 65 CD 4C 66 2B 22
6590 5A 67 CD 2E 66 C3 83 65 CD 4C 66 23 C3 8F 65 CD
65A0 2E 66 CD 4C 66 C3 00 64 0E 01 16 FF CD FB 65 E6
65B0 80 CA F5 65 7B E6 40 CA D5 65 7B E6 20 CA E7 65
65C0 79 FE 01 CA AA 65 15 C2 AC 65 E5 21 6D 66 85 6F
65D0 4E E1 C3 09 F8 79 07 FE 40 D2 F8 65 3C 4F CD 0B
65E0 66 CD 38 66 C3 AA 65 79 07 FE 40 D2 F8 65 4F CD
65F0 11 66 C3 E1 65 0E FF C9 0E 00 C9 3A 02 80 E6 E0
6600 5F 3A 02 80 E6 E0 BB C2 FB 65 C9 CD 11 66 CD 11
6610 66 E5 2A 58 67 7D FB 3D C2 16 66 7D F3 3D C2 1C
6620 66 25 C2 15 66 E1 C9 E5 21 1A 50 C3 15 66 E5 21
6630 13 50 C3 15 66 CD 38 66 E5 2A 58 67 7D 3D C2 3D
6640 66 7D 3D C2 42 66 25 C2 3C 66 E1 C9 E5 F5 21 06
6650 67 CD 18 F8 F1 E1 C9 CD 4C 66 C5 7C CD 15 F8 7D
6660 CD 15 F8 0E 3A CD 09 F8 0E 20 CD 09 F8 C1 C9 45
6670 30 49 41 39 4D 53 55 52 57 44 4B 47 4F 48 56 46
6680 60 4C 71 50 4A 42 58 43 59 5A 51 7E 7B 35 34 2A
6690 33 7C 2A 2A 32 3B 2A 40 2A 2A 2A 2A 31 36 2D 2F
66A0 2A 2A 2A 2A 2A 37 2A 2A 2A 38 2A 39 30 3F 2F 27
66B0 23 21 20 30 38 3C 3E 05 18 1A 0C 02 12 0D 0A 4D
66C0 4F 52 53 45 2D 44 55 4D 50 3E 20 64 69 72 65 6B
66D0 74 69 77 61 28 53 4D 44 2E 29 3D 00 22 77 69 6C
66E0 69 71 73 6F 66 74 22 20 66 65 77 2E 31 39 39 30
66F0 20 55 43 32 41 41 53 20 6D 69 6E 73 6B 20 33 39
6700 2D 35 39 2D 35 31 0D 0A 00 0D 0A 64 72 6F 62 7E
6710 2D 6E 61 7A 61 64 2C 20 7A 6E 61 6B 20 72 61 7A
6720 64 65 6C 61 2D 77 70 65 72 65 64 2C 20 72 75 73
6730 2D 73 74 6F 70 00 0D 0A 61 64 72 65 73 3D 00 0D
6740 0A 73 6B 6F 72 6F 73 74 78 2A 74 6F 6E 3D 00 77
6750 77 65 64 69 74 65 3D 00 08 C0 00 00 00 00 00 00
К.С. - 61F7H.
```

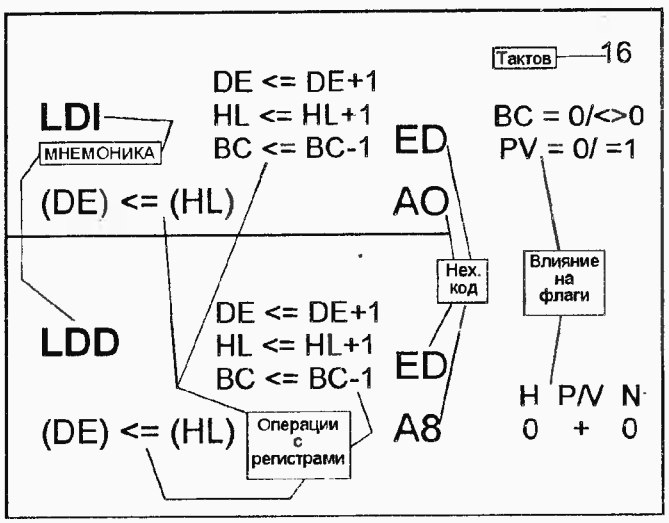
А.СУЛИМА.
184372, Мурманская обл.,
пос.Видяево, ул.Заречная, 20-21.

СИСТЕМА КОМАНД МИКРОПРОЦЕССОРА Z-80

Эти справочные таблицы, на мой взгляд, удобны при программировании в Ассемблере, отладке программ в машинных кодах в системе команд микропроцессора Z-80, особенно на "SINCLAIR"-совместимых компьютерах. Сразу хочу оговориться, что нового ничего не придумал, но думаю, что данные таблицы выигрывают своей упорядоченностью, информатированностью, наглядностью, что удобно как для обучения, так и в практической работе.

Во всяком случае, те из программистов и пользователей, кому были подарены данные таблицы, пользуются ими интенсивно и с хорошими отзывами. Это настольный вариант, который должен быть всегда под рукой у того, кто занимается "SPECCY" и вообще программированием.





В таблице приведены мнемоника команды, основное воздействие на регистры и флаги, время выполнения операции в машинных тактах, машинные коды для указанных регистров и их пар (если регистры не указаны для однобайтных операций, машинные коды расположены в общепринятом порядке для 8-разрядных регистров и ячеек памяти — B, C, D, E, H, L, (HL), A. (IX+d), (IY+d).

ОДНОБАЙТНАЯ ЗАГРУЗКА, ПЕРЕСЫЛКА

LD r,r — 4[7]

r<=r` — –

	B`	C`	D`	E`	H`	L`	(HL)`	A`
B	40	41	42	43	44	45	46	47
C	48	49	4A	4B	4C	4D	4E	4F
D	50	51	52	53	54	55	56	57
E	58	59	5A	5B	5C	5D	5E	5F
H	60	61	62	63	64	65	66	67
L	68	69	6A	6B	6C	6D	6E	6F
(HL)	70	71	72	73	74	75	—	77
A	78	79	7A	7B	7C	7D	7E	7F

LD r, n — 7[10]

r<=n — 06, 0E, 16, 1E, 26, 2E (36) 3E — –

LD (IX+d), n — DD 19

(IX+d)<=n — 36 –

LD (IY+d), n — FD 19

(IY+d)<=n — 36 –

LD r, (IX+d) — DD 19

r<=(IX+d) — 46, 4E, 56, 5E, 66, 6E (–) 7E –

LD (IX+d), r — DD 19

(IX+d)<=r — 70, 71, 72, 73, 74, 75 (–) 77 –

LD r, (IY+d) — FD 19

r<=(IY+d) — 46, 4E, 56, 5E, 66, 6E (–) 7E –

LD (IY+d), r — FD 19

(IY+d)<=r — 70, 71, 72, 73, 74, 75 (–) 77 –

LD A, (BC) — 0A 7

A<=(BC) –

LD (BC), A — 02 7

(BC)<=A –

LD A, (DE) — 1A 7

A<=(DE) –

LD (DE), A — 12 7

(DE)<=A –

LD A, (nn) — 3A 13

A<=(nn) –

LD (nn), A — 32 13

(nn)<=A –

LD A, I — ED 9

A<=I, p/v<=IFF2 — 57 SZP/V

LD I, A — ED 9

I<=A — 47 –

LD A, R — ED 9

A<=R, p/v<=IFF2 — 5F SZP/V

LD R, A — ED 9

I<=R — 4F –

ОПЕРАЦИИ С ПОРТАМИ

ВВОД ИЗ ПОРТА

IN A, (N) — D8 11

A<=(pAN) –

IN r, (C) — ED 12

r<=(pBC) — 40, 48, 50, 58, 60, 68 (xx), 78 SZHPN

INI — ED A2 16

(HL)<=(pBC) — B<=B-1, HL<=HL+1, B=0/<>0, Z=1/=0

INIR — ED B2 BC=0 21, BC<>0 16, ZN=1

(HL) <=(pBC) — B<=B-1, HL<=HL+1

IND — ED AA 16, B=0/<>0, Z=1/=0

(HL)<=(pBC) — B=<=B-1, HL<=HL-1

INDR — ED BA BC=0 21, BC<>0 16, ZN=1

(HL)<=(pBC) — B=<=B-1, HL<=HL-1

ВЫВОД В ПОРТ

OUT (N), A — D3 11

(pAN)<=A –

OUT (C), r — ED 12

(pBC)<=r — 41, 49, 51, 59, 61, 69 (—), 79 –

OUTI — ED A3 16

(pBC)<=(HL) — B<=B-1, HL<=HL+1, B=0/<>0, Z=1/=0

OUTIR — ED B3 BC=0 21, BC<>0 16, ZN=1

(pBC)<=(HL) — B<=B-1, HL<=HL+1

OUTD — ED AB 16

(pBC)<=(HL) — B<=B-1, HL<=HL-1, B=0/<>0, Z=1/=0

OUTDR — ED BB BC=0 21, BC<>0 16, ZN=1

(pBC)<=(HL) — B<=B-1, HL<=HL-1

ДВУХБАЙТНАЯ ЗАГРУЗКА, ПЕРЕСЫЛКА

LD SS, (nn) — *2A* ED 20*16*

ShSl<=(nn+1)(nn) — BC DE *HL* SP, 48 58 68 78 –

LD IX, (nn) — DD 20

IX<=(nn+1)(nn) — 2A –

LD IY, (nn) — FD 20

IY<=(nn+1)(nn) — 2A –

LD SS, nn — BC DE HL SP, 01 11 21 31 10

SS<=nn –

LD IX, nn — DD 10

IX<=nn — 21 –

LD IY, nn — FD 14

IY<=nn — 21 –

LD (nn), SS — *22* ED 20*16*

(nn+1)(nn)<=ShSl — BC DE *HL* SP, 43 53 63 73 –

LD (nn), IX — DD 20

(nn+1)(nn)<=IX — 22 –

LD (nn), IY — FD 20

(nn+1)(nn)<=IY — 22 –

EX AF,AF` — 8 4

AF<=>AF` +

EXX — D9 4

BC<=>BC`, DE<=>DE`, HL<=>HL` –

EX DE,HL — EB 4

DE<=>HL –

ОПЕРАЦИИ СО СТЕКОМ

ИЗ СТЕКА

POP BC — C1 10

BC<=(SP) –

POP DE — D1 10

DE<=(SP) –

POP HL — E1 10

HL<=(SP) –

POP AF — F1

AF<=(SP) +

POP IX — DD E1 14

IX<=(SP) –

POP IY — FD E1 14

IY<=(SP) –

В СТЕК

PUSH BC — 11

(SP)<=BC C5 –

PUSH DE — 11

(SP)<=DE D5 –

PUSH HL — 11

(SP)<=HL E5 –

PUSH AF — 11

(SP)<=AF F5 –

PUSH IX — DD E5 15

(SP)<=IX –

PUSH IY — FD E5 15

(SP)<=IY –

ЗАГРУЗКА, ПЕРЕСЫЛКА

EX (SP),HL — E3 19

H<=>(SP+1) L<=>(SP) –

EX (SP), IX — DD 23

IX h1 <=> (SP) E3 –

EX(SP), IY — FD 23

IY h1<=>(SP+1)(SP) — E3 –

LD SP, HL — F9 6

SP<=HL –

LD SP, IX — DD 6

SP<=IX — F9 –

LD SP, IY — FD 10

SP<=IY — F9 –

БЛОЧНЫЕ ПЕРЕСЫЛКИ, СРАВНЕНИЯ

ОДНОГО БАЙТА

LDI — DE<=DE+1, HL<=HL+1, BC<=BC-1 — ED 16

(DE)<=(HL) — A0 — BC=0/<>0, PV=0/=1

LDD — DE<=DE-1, HL<=HL-1, BC<=BC-1 — ED — HP/VN 0+0

(DE)<=(HL) — A8

CPI — HL<=HL+1, BC<=BC-1 — ED 16

A<=A-(HL) — A1 — BC=0/<>0, PV=0/=1

CPD — HL<=HL-1, BC<=BC-1 — A=HL/<>HL, Z=1/=0, SZHP/VN

A<=A-(HL) — ED A9

БЛОКА БАЙТОВ

LDIR — DE<=DE+1, HL<=HL+1, BC<=BC-1 — ED — BC=0, BC<>0, HP/VN 0+0 — 21

(DE)<=(HL) — B0 — 16

LDDR — DE<=DE-1, HL<=HL-1, BC<=BC-1 — ED

(DE)<=(HL) — B8

CPIR — HL<=HL+1, BC<=BC-1 — ED — BC=0, BC<>0 — 16, 21

A<=A-(HL) — B1

CPDR — HL<=HL-1, BC<=BC-1 — ED — SZHP/VN +++ + 1

A<=A-(HL) — B9

ОДНОБАЙТНЫЕ АРИФМЕТИЧЕСКИЕ ОПЕРАЦИИ

ADD A, r	80, 81, 82, 83 84, 85 (66) 87	4[7]	**ADC A, r**	88, 89, 8A, 8B 8C, 8D (8E) 8F	4[7]
A<=A+r		SZHVNC ++++0+	A<=A+r+c		SZHVNC ++++0+
ADD A, (IX+d)	DD	19	**ADC A, (IX+d)**	DD	19
A<=A+(IX+d)	86	SZHVNC ++++0+	A<=A+(IX+d)+c	8E	SZHVNC ++++0+
ADD A, (IY+d)	FD	19	**ADC A, (IY+d)**	FD	19
A<=A+(IY+d)	86	SZHVNC ++++0+	A<=A+(IY+d)+c	8E	SZHVNC ++++0+
ADD A,n	C6	4	**ADC A,n**	CE	7
A<=A+n		SZHVNC ++++0+	A<=A+n+c		SZHVNC ++++0+
SUB r	90, 91, 92, 93 94, 95 (96) 97	4[7]	**SBC A,r**	98, 99, 9A, 9B 9C, 9D (9E) 9F	4[7]
A<=A-r		SZHVNC ++++1+	A<=A-r-c		SZHVNC ++++1+
SUB (IX+d)	DD	19	**SBC A, (IX+d)**	DD	19
A<=A-(IX+d)	96	SZHVNC ++++1+	A<=A-(IX+d)-C	9E	SZHVNC ++++1+
SUB (IY+d)	FD	19	**SBC A, (IY+d)**	FD	19
A<=A-(IY+d)	96	SZHVNC ++++1+	A<=A-(IY+D)-C	9E	SZHVNC ++++1+
SUB n	D6	7	**SBC A, n**	DE	7
A<=A-n		SZHVNC ++++1+	A<=A-n-c		SZHVNC ++++1+
CP r	B8, B9, BA, BB BC,BD,BE(BF)	4[7]	**INC**	04, 0C, 14, 1C 24, 2C (34) 3C	4[11]
A-r		SZHVNC ++++1+	r<=r+1		SZHVN ++++0
CP (IX+d)	DD	19	**INC (IX+d)**	DD 34	23
A-(IX+d)	BE	SZHVNC ++++1+	(IX+d)<=(IX+d)+1		SZHVN ++++0
CP (IY+d)	FD	19	**INC (IY+d)**	FD 34	23
A-(IY+d)	BE	SZHVNC ++++1+	(IY+d)<=(IY+d)+1		SZHVN ++++0
CP n	FE	7	**DEC r**	05, 0D, 15, 1D 25, 2D(35) 3D	4[11]
A-n		SZHVNC ++++1+	r<=r-1		SZHVN ++++1
			DEC (IX+d)	DD 35	23
			(IX+d)<=(IX+d)-1		SZHVN ++++1
			DEC (IY+d)	FD	23
			(IY+d)<=(IY+d)-1	35	SZHVN ++++1

ДВУХБАЙТ. АРИФ. ОПЕРАЦИИ

ADD HL, SS		11
HL<=HL+SS	BC DE HL SP 09 19 29 39	HNC x0+
ADD IX, SS	DD	15
IX<=IX+SS	BC DE IX SP 09 19 29 39	HNC x0+
ADD IY, SS	FD	15
IY<=IY+SS	BC DE IY SP 09 19 29 39	HNC x0+
ADC HL,SS	ED	15
HL<=HL+SS+c	BC DE HL SP 4A 5A 6A 7A	SZHVNC ++x+0+
SBC HL, SS	ED	11
HL<=HL-SS-c	BC DE HL SP 42 52 52 72	SZHVNC ++x+0+
INC SS		6
SS<=SS+1	BC DE HL SP 03 13 23 33	
INC IX	DD	10
IX<=IX+1	23	—
INC IY	FD	10
IY<=IY+1	23	—
DEC SS		6
SS<=SS-1	BC DE HL SP 0B 1B 2B 3B	—
DEC IX	DD	10
IX<=IX-1	2B	—
DEC IY	FD	10
IY<=IY-1	2B	—

ЛОГИЧЕСКИЕ ОПЕРАЦИИ

AND r	B C D E A0 A1 A2 A3	4[7]
A<=A&r	H L (HL) A A4 A5 A6 A7	SZHPNC ++1+00
AND (IX+d)	DD	19
A<=A&(IX+d)	A6	SZHPNC ++1+00
AND (IY+d)	FD	19
A<=A&(IY+d)	A6	SZHPNC ++1+00
AND n	E6	7
A<=A&n		SZHPNC ++1+00
OR r	B C D E B0 B1 B2 B3	4[7]
A<=AVr	H L (HL) A B4 B5 B6 B7	SZHPNC ++0+00
OR (IX+d)	DD	19
A<=AV(IX+d)	B6	SZHPNC ++0+00
OR (IY+d)	FD	19
A<=AV(IY+d)	B6	SZHPNC ++0+00
OR n	F6	7
A<=AVn		SZHPNC ++0+00
XOR r	B C D E A8 A9 AA AB	4[7]
A<=A()r	H L (HL) A AC AD AE AF	SZHPNC ++0+00
XOR (IX+d)	DD	19
A<=A()(IX+d)	AE	SZHPNC ++0+00
XOR (IY+d)	FD	19
A<=A()(IY+d)	AE	SZHPNC ++0+00
XOR n	EE	7
A<=A()n		SZHPNC ++0+00

ПЕРЕХОДЫ

АБСОЛЮТНЫЕ

JP nn	C3	10
PC<=nn		—
JP C, nn	DA	10
C =1/ =0 PC<=nn/PC<=PC+3		—
JP NC, nn	D2	10
C =0/ =1 PC<=nn/PC<=PC+3		—
JP Z, nn	CA	10
Z =1/ =0 PC<= nn/ PC<= PC+3		—
JP NZ, nn	C2	10
Z =0/ =1 PC<=nn/ PC<= PC+3		—
JP PE, nn	EA	10
P =1/ =0 PC<=nn/ PC<= PC+3		—
JP PO, nn	E2	10
P =0/ =1 PC<= nn/ PC<= PC+3		—
JP M, nn	FA	10
S =1/ =0 PC<= NN/ PC<= PC+3		—
JP P, nn	F2	10
S =0/ =1 PC<= mm/ PC<= PC+3		—
JP (SS)		4 [9]
PC<=(SS)	(HL) [(IX)] [(IY)] E9 DDE9 FDE9	—

ОТНОСИТЕЛЬНЫЕ

JR d	18	12/7
PC<=PC+d		—
JR C, d	38	12/7
C =1 / =0 PC<=PC+d/ PC<=PC+2		—
JR NC, d	30	12/7
C =0 / =1 PC<=PC+d/PC<=PC+2		—
JR Z, d	28	12/7
Z =1 / =0 PC<=PC+d/PC<=PC+2		—
JR NZ, d	20	12/7
Z =0 / =1 PC<=PC+d/PC<=PC+2		—
DJNZ d	10	13/8
B= 0/ <>0 B<=B-1:PC<=PC+2/PC+d		—

ПОДПРОГРАММЫ

ВЫЗОВ

CALL nn	CD	17
PC<=nn (SP)<=PC+3 SP<=SP-2		—
CALL C, nn	DC	17/10
C= 1/ =0 PC<= nn/ PC+3		—
CALL NC, nn	D4	17/10
C =0/ =1 PC<= nn/ PC+3		—
CALL Z, nn	CC	17/10
Z =1/ =0 PC<= nn/ PC+3		—
CALL NZ, nn	C4	17/10
Z =0/ =1 PC<=nn/ PC+3		==
CALL PE, nn	EC	17/10
P =1/ =0 PC<=nn/ PC+3		—
CALL PO, nn	E4	17/10
P =0/ =1 PC<=nn/ PC+3		—
CALL M, nn	FC	17/10
S =1/ =0 PC<=nn/ PC+3		—
CALL P, nn	F4	17/10
S =0/ =1 PC<=nn/ PC+3		—
RST n	00 08 10 18 C7 CF D7 DF	11
PC<=00n H (SP)<=PC+1	20 28 30 38 E7 EF F7 FF	—

ВОЗВРАТ

RET	C9	10
PCl<=(SP) PCh<=(SP+1) SP<=SP+2		—
RET C	D8	11/5
C =1/ =0 PC<=(SP)/PC+1		—
RET NC	D0	11/5
C =0/ =1 PC<=(SP)/ PC+1		—
RET Z	C8	11/5
Z =1/ =0 PC<=(SP)/ PC+1		—
RET NZ	C0	11/5
Z =0/ =1 PC<=(SP)/ PC+1		—
RET PE	E8	11/5
P =1 / =0 PC<=(SP)/ PC+1		—
RET PO	E0	11/5
P =0/ =1 PC<=(SP)/ PC+1		—
RET M	F8	11/5
S =1 / =0 PC<=(SP)/ PC+1		—
RET P	F0	11/5
S =0 / =1 PC<=(SP)/ PC+1		—

ДОПОЛНИТЕЛЬНЫЕ КОМАНДЫ

SCF	37	4	**CCF**	3F	4	—	PC<=0066H		**RETN**	ED 45	14
Cf<=1	УСТАНОВКА ФЛАГА ПЕРЕНОСА	HNC 001	Cf<=$\overline{Cf}$	ИНВЕРТИРОВАНИЕ ЗНАЧЕН. ФЛАГА ПЕРЕНОСА	HNC x0+	IFF2<=IFF1 IFF1<=0	АППАРАТНОЕ НЕМАСКИРУЕМОЕ ПРЕРЫВАНИЕ ПО СИГНАЛУ NMI		PC<=(SP) IFF1<=IFF2	ВОЗВРАТ ПО ОКОНЧАНИИ ПЛТ НЕМАСКИРУЕМОГО ПРЕРЫВ.	—
DAA	27	4	**CPL**	2F	4	**IM0**	ED 46	8	**RETI**	ED 4D	14
	ДЕСЯТИЧНАЯ КОРРЕКЦИЯ СОДЕРЖ. АККУМУЛЯТОРА	SZHPC +-++++	A<=$\overline{A}$	ИНВЕРТИРОВАНИЕ СОДЕРЖ АККУМУЛЯТОРА	HN 11	PC<= *0030H* IFF1, 2<=0	УСТАНОВКА РЕЖИМА "0" МАСКИРУЕМ. ПРЕРЫВАНИЯ	—	PC<=(SP) IFF1<=IFF2	ВОЗВРАТ ПО ОКОНЧАНИИ ПЛТ МАСКИРУЕМОГО ПРЕРЫВАНИЯ	—
NEG	ED 44	8				**IM1**	ED 56	8	**DI**	F3	4
A<=0-A	ПРЕОБРАЗОВАН. ЧИСЛА В ДОПОЛНИТЕЛЬНЫЙ КОД	SZHVN ++++1+				PC<=*0030H* IFF1,2<=0	УСТАНОВКА РЕЖИМА "1" МАСКИРУЕМ. ПРЕРЫВАНИЯ	—	IFF1<=0 IFF2<=0	ЗАПРЕТ МАСКИРУЕМЫХ ПРЕРЫВАНИЙ	—
NOP	00	4	**HALT**	76	4	**IM2**	ED 5E	8	**EI**	FB	4
PC<=PC+1 R<=R+1	НЕТ ОПЕРАЦИЙ			ОСТАНОВ ДО ПОЯВЛЕНИЯ ПРЕРЫВАНИЯ	—	PC<=(I,D7-D0) IFF1, 2<=0	УСТАНОВКА РЕЖИМА "2" МАСКИРУЕМ. ПРЕРЫВАНИЯ	—	IFF1<= IFF2<=	РАЗРЕШЕНИЕ МАСКИРУЕМЫХ ПРЕРЫВАНИЙ	—

ПРОВЕРКА БИТОВ

BIT b,r b =0/ =1 z =1/ =0 — 8(12)[(20)]

bit	B	C	D	E	H	L	(HL)	A	[(IX+d)]	[(IY+d)]
	CB	CB	CB	CB	CB	CB	CB	CB	DDCB	FDCB
0	40	41	42	43	44	45	46	47	d 46	d 46
1	48	49	4A	4B	4C	4D	4E	4F	d4E	d4E
2	50	51	52	53	54	55	56	57	d 56	d 56
3	58	59	5A	5B	5C	5D	5E	5F	d5E	d5E
4	60	61	62	63	64	65	66	67	d 66	d 66
5	68	69	6A	6B	6C	6D	6E	6F	d6E	d6E
6	70	71	72	73	74	75	76	77	d 76	d 76
7	78	79	7A	7B	7C	7D	7E	7F	d7E	d7E

СБРОС БИТОВ

RES b,r b<=0 — 8(12)[(20)]

bit	B	C	D	E	H	L	(HL)	A	[(IX+d)]	[(IY+d)]
	CB	CB	CB	CB	CB	CB	CB	CB	DDCB	FDCB
0	80	81	82	83	84	85	86	87	d 86	d 86
1	88	89	8A	8B	8C	8D	8E	8F	d 8E	d 8E
2	90	91	92	93	94	95	96	97	d 96	d 96
3	98	99	9A	9B	9C	9D	9E	9F	d 9E	d 9E
4	A0	A1	A2	A3	A4	A5	A6	A7	d A6	d A6
5	A8	A9	AA	AB	AC	AD	AE	AF	dAE	dAE
6	B0	B1	B2	B3	B4	B5	B6	B7	dB6	dB6
7	B8	B9	BA	BB	BC	BD	BE	BF	dBE	dBE

УСТАНОВКА БИТОВ

RES b,r b<=1 — 8(12)[(20)]

bit	B	C	D	E	H	L	(HL)	A	[(IX+d)]	[(IY+d)]
	CB	CB	CB	CB	CB	CB	CB	CB	DDCB	FDCB
0	C0	C1	C2	C3	C4	C5	C6	C7	d C6	d C6
1	C8	C9	CA	CB	CC	CD	CE	CF	d CE	d CE
2	D0	D1	D2	D3	D4	D5	D6	D7	d D6	d D6
3	D8	D9	DA	DB	DC	DD	DE	DF	d DE	d DE
4	E0	E1	E2	E3	E4	E5	E6	E7	dE6	dE6
5	E8	E9	EA	EB	EC	ED	EE	EF	dEE	dEE
6	F0	F1	F2	F3	F4	F5	F6	F7	dF6	dF6
7	F8	F9	FA	FB	FC	FD	FE	FF	dFE	dFE

ОБОЗНАЧЕНИЯ

- r - 8-разрядные регистры МП Z80 B, C, D, E, H, L, A или ячейки памяти, адресуемые по (HL), (IX+d), (IY-d)
- SS - 16-разрядные регистры или регистровые пары BC, DE,HL,IX, IY, SP
- n - 1-байтная величина (8-разрядное число)
- nn - 2-байтная величина (16-разрядное число)
- d - 1-байтное (8-разрядное) относительное смещение (-127<= d<=128)
- b - номер бита в операциях с битами
- ** - предпочтительнее, преимущественно, для данного значения, операции...
- [] - значения, операции...

Содержание "F"-регистра (флаги)

bit	7	6	5	4	3	2	1	0
flag	S	Z	-	H	-	P/V	N	C

- S - флаг ЗНАКА (состояния 7 бита аккумулятора): S=1 число отрицательное
- Z - флаг НУЛЯ (признак 0-го результата операции): Z=1 результат операции =0
- C - флаг ПЕРЕНОСА - признак переноса 7 бита (заема в 7 бит) при операции: C=1 был перенос (заем) в C (из C)
- H - флаг ПОЛУПЕРЕНОСА в операциях с двоично - десятичными числами - признак переноса 3 бита (заема в 3 бит) при операции: H=1 был перенос (заем) в 4 бит (из 4 бита)
- V - флаг ПЕРЕПОЛНЕНИЯ в арифметических операциях: V=1 результат арифметической операции не укладывается в 8 битов
- P - флаг ЧЕТНОСТИ в логических операциях: P=1 - признак, что в результате операции число установленных (=1) битов аккумулятора четное
- N - флаг ВЫЧИТАНИЯ: N=1 последней операцией было вычитание

Условия истинности выполнения операции

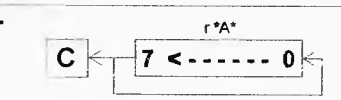
NZ - не ноль (флаг Z=0)	Z - ноль (флаг Z=1)
NC - нет переноса (фл. C=0)	C - перенос (флаг C=1)
PO - нечетность (фл.P/V=0)	PM - четность (фл.P/V=1)
P - положительное (фл.S=0)	M - отрицательное (S=1)

Влияние операций на флаги

- *SZHPVNC* , "+" - изменяет указанный флаг
- "—"- не влияет на флаги
- 1 - устанавливает указанный флаг
- 0 - сбрасывает указанный флаг
- x - состояние флага не определено

КОМАНДЫ СДВИГА

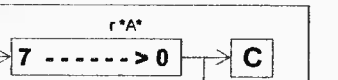
RLC r — 8(15)[23] SZPC, HN=0



B	C	D	E	H	L	(HL)	A	[(IX+d)]	[(IY+d)]
CB	CB	CB	CB	CB	CB	CB	CB	DDCB	FDCB
00	01	02	03	04	05	06	07	d 06	d 06

RLCA *07* 4 C

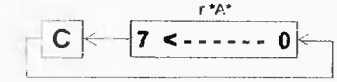
RRC r — 8(15)[23] SZPC, HN=0



B	C	D	E	H	L	(HL)	A	[(IX+d)]	[(IY+d)]
CB	CB	CB	CB	CB	CB	CB	CB	DDCB	FDCB
08	09	0A	0B	0C	0D	0E	0F	d 0E	d 0E

RRCA *0F* 4 C

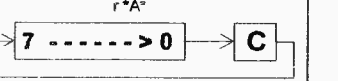
RL r — 8(15)[23] SZPC, HN=0



B	C	D	E	H	L	(HL)	A	[(IX+d)]	[(IY+d)]
CB	CB	CB	CB	CB	CB	CB	CB	CB	CB
10	11	12	13	14	15	16	17	d 16	d 16

RLA *17* 4 C

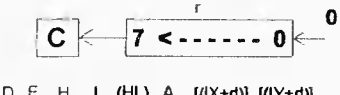
RR r — 8(15)[23] SZPC, HN=0



B	C	D	E	H	L	(HL)	A	[(IX+d)]	[(IY+d)]
CB	CB	CB	CB	CB	CB	CB	CB	CB	CB
18	19	1A	1B	1C	1D	1E	1F	d 16	d 16

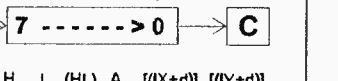
RRA *1F* 4 C

SLA r — 8(15)[23] SZPC, HN=0



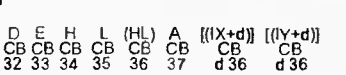
B	C	D	E	H	L	(HL)	A	[(IX+d)]	[(IY+d)]
CB	CB	CB	CB	CB	CB	CB	CB	CB	CB
20	21	22	23	24	25	26	27	d 26	d 26

SRL r — 8(15)[23] SZPC, HN=0



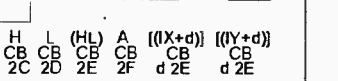
B	C	D	E	H	L	(HL)	A	[(IX+d)]	[(IY+d)]
CB	CB	CB	CB	CB	CB	CB	CB	DDCB	FDCB
38	39	3A	3B	3C	3D	3E	3F	d 3E	d 3E

SLI r — 8(15)[23] SZPC, HN=0



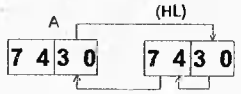
B	C	D	E	H	L	(HL)	A	[(IX+d)]	[(IY+d)]
CB	CB	CB	CB	CB	CB	CB	CB	CB	CB
30	31	32	33	34	35	36	37	d 36	d 36

SRA r — 8(15)[23] SZPC, HN=0

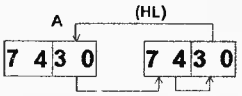


B	C	D	E	H	L	(HL)	A	[(IX+d)]	[(IY+d)]
CB	CB	CB	CB	CB	CB	CB	CB	CB	CB
28	29	2A	2B	2C	2D	2E	2F	d 2E	d 2E

RLD ED 18 SZP, HN=0 6F



RRD ED 18 SZP, HN=0 67



А.ПЕРМИНОВ,
625006, г.Тюмень, ул.Елизарова, 30 - 201,
тел.24-71-25.

"ZX SPECTRUM" УПРАВЛЯЕТ СВЕТОМ

Предлагаемое устройство служит приставкой к "Спектрум"-совместимым компьютерам и эмулирует работу автомата световых эффектов. Работать может как в домашних условиях, так и на дискотеке. В основу его работы положен способ передачи данных с компьютера через порт на периферию [1].

Все тонкости работы автомата реализованы программно, в то время как аппаратной частью служит приставка, описанная ниже (и, разумеется, компьютер). Возможности данного устройства весьма многообразны при сравнительно низкой себестоимости.

Основные технические характеристики устройства

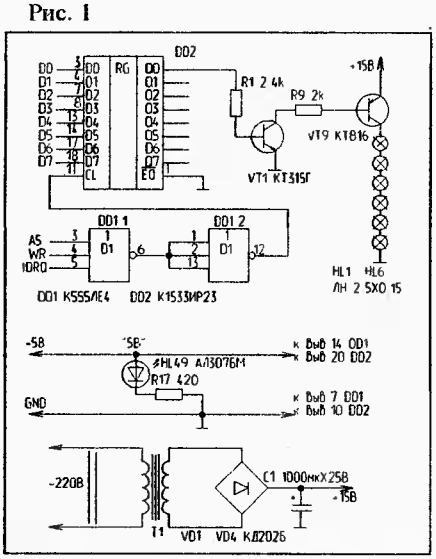
Напряжение питания, В	
- логика	5±10%
- силовая часть	15±10%
Число каналов	8
Кол-во световых эффектов	16
Кол-во тактов в эффекте	16

Рассмотрим порт вывода, преобразующий 8-битовые слова компьютера в потенциальные сигналы 0 или 5 В на выходных линиях Q0...Q7 регистра DD2 (рис.1). Микросхема DD2 порта вывода принимает сигналы с шины данных "Спектрума" после поступления сигнала на ее вход CL. Такое состояние микросхемы сохраняется до поступления следующего импульса на вход CL. Говорят, что микросхема "защелкивается", т.е. "защелкивает" данные.

Схема выборки выходного порта выполнена на двух элементах микросхемы DD1. Адрес формируется из отдельных сигналов — на адресной линии A5 (т.к. в нашем случае использован порт 31) вместе с сигналом WR (линия записи) и IORQ (линия запроса ввода/вывода). Первый элемент выделяет сигнал выборки, а второй инвертирует его для получения необходимого для записи сигнала на входе CL микросхемы DD2.

При выводе на блок ключей некоторого числа, например, 255 (двоичный код — 11111111, т.е. все лампы загораются) выполняется следующая последовательность действий: выполняется оператор OUT 31,255; двоичный код 11111111 появляется на шине данных D0...D7 и попадает на входы микросхемы DD2; одновременно формируется сигнал выборки, полученный из сигналов A5, WR, IORQ и дает команду микросхеме DD2 "защелкнуть" все 8 триггеров регистра и передать 8 единиц на транзисторные ключи; следующая команда программы сбрасывает сигналы выборки и продолжает выполнение программы, однако на выходе 8 единиц сохранятся до тех пор, пока не будет выполнена команда OUT 31.

Рис. 1



ТТЛ выводы Q0...Q7 микросхемы DD2 управляют транзисторными ключами, которые, в свою очередь, коммутируют ток на лампы накаливания. Принцип работы ключей известен.

Работа программы основана на образовании в ОЗУ компьютера блока данных и упорядоченной выдаче их в порт 31 (возможно использование другого порта) [2].

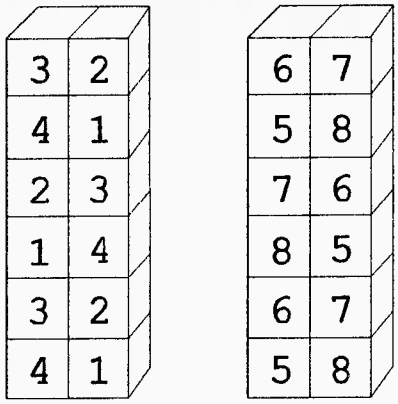


Рис.2

Сведения по управлению эффектами индицируются на мониторе компьютера. На нем также показан номер используемого эффекта, скорость и состояние автоматического режима. Программная поддержка написана на языке BASIC и предельно проста в эксплуатации, в связи с чем считаю разумным подробно на ней не останавливаться.

Мною выполнено выносное оптическое устройство в виде двух панно (рис.2). В каждой ячейке световой колонки размещаются 2 лампочки накаливания типа ЛН 2,5х0,15, соединенные последовательно в каждом канале. Лампы предварительно окрашены в четыре цвета: 1 и 5 — красные, 2 и 6 — зеленые и т.п. Снаружи наложено матовое оргстекло.

Описанное устройство питается от двух источников: 5 В — от компьютера и 15 В — от собственного источника питания. В нем использованы микросхемы серий 1533 или 555. Можно использовать м/с серии 155, но они работают хуже. Резисторы — МЛТ-0,25. Транзисторы: VT1...VT8 — КТ315 с любым буквенным индексом; VT9...VT16 — КТ816, КТ814А...Г, КТ816А...Г или аналогичные с током коллектора ≥150 мА [3]. Конденсатор — К50-6, К50-16, К50-35. Диоды — на ток ≥1,2 А. Светодиод — АЛ307БМ или другой с соответствующей заменой резистора R17.

Устройство подключается к шине компьютера или непосредственно к самому микропроцессору Z80. Оно просто в изготовлении и при правильном монтаже не требует наладки.

ПРОГРАММА

```
  1 LET SP=100
  2 LET NO=1
  4 RESTORE
  10 INK 9:PAPER 0:FLASH 0:BRIGHT 0:BORDER 0
  15 CLS
  20 DIM A(16,16)
  25 PRINT "WAIT !!!":GO SUB 1000:CLS
  30 PRINT AT 0,0;BRIGHT 1;PAPER 1;INK 7;
"PERMINOV A. TYUMEN TEL.24-71-25"
  40 PRINT BRIGHT 1;FLASH 1;PAPER 2;
INK 7;"▄▄▄▄▄▄▄▄ LIGHT SHOW -2▄▄▄▄▄▄▄▄"
  45 INK 4
  50 PRINT:PRINT:PRINT "CURSOR JOYSTIK IN USE"
  60 PRINT: "LEFT & RIGHT - SPEED"
  70 PRINT "UP & DOWN - NUMBER OF EFFECT"
  80 PRINT "FIRE - AUTO (ON/OFF)"
  85 LET AUTO=0
  90 INK 9
  100 PRINT AT 12,0;"AUTO:"
  110 PRINT AT 14,0;"SPEED:"
  120 PRINT AT 16,0;"N EF.:"
  130 PRINT AT 20,0;INK 3;"GOOD LUCK!"
  200 FOR I=1 TO 16
  210 FOR J=1 TO 16
  220 OUT 31,A(I,J)
  230 IF INKEY$="0" THEN LET AUTO=NOT AUTO
  240 IF AUTO=1 THEN PRINT AT 12,10;"ON"
  250 IF AUTO=0 THEN PRINT AT 12,10;"OFF"
  260 IF INKEY$="8" THEN LET SP=SP+5
  270 IF INKEY$="5" THEN LET SP=SP-5
  280 IF SP<0 THEN LET SP=0
  290 PRINT AT 14,10;SP;"  "
  300 IF INKEY$="7" THEN LET NO=NO+1
  310 IF INKEY$="6" THEN LET NO=NO-1
  320 IF NO>16 THEN LET NO=16
```

```
330 IF NO<1 THEN LET NO=1
335 PRINT AT 16,10;NO;"  "
340 FOR F=0 TO SP:NEXT F
350 NEXT J
360 IF AUTO=1 THEN NEXT I
365 IF I>16 THEN GOTO 200
370 LET I=NO:GOTO 210
1000 FOR I=1 TO 16
1010 FOR J=1 TO 16
1020 READ M
1030 LET A(I,J)=M
1040 NEXT J
1050 NEXT I
1060 RETURN
2010 DATA 1,2,4,8,16,32,64,128
2011 DATA 1,2,4,8,16,32,64,128
2020 DATA 1,0,16,0,2,0,32,0
2021 DATA 4,0,64,0,8,0,128,0
2030 DATA 15,0,240,0,15,0,240,0
2031 DATA 15,0,240,0,15,0,240,0
2040 DATA 1,3,7,15,14,12,8,0
2041 DATA 16,48,112,240,224,192,128,0
2050 DATA 9,6,96,144,9,6,96,144
2051 DATA 9,6,96,144,9,6,96,144
2060 DATA 12,0,3,0,48,0,192,0
2061 DATA 12,0,3,0,48,0,192,0
2070 DATA 8,12,14,15,31,63,127,255
2071 DATA 247,243,241,240,224,192,128,0
2080 DATA 132,150,183,255,123,105,72,0
2081  DATA  132,150,183,255,123,105,72,0
2090 DATA 5,0,10,0,5,0,10,0
2091 DATA 160,0,80,0,160,0,80,0
2100 DATA 34,68,34,68,34,68,34,68
2101 DATA 17,136,17,136,17,136,17,136
2110 DATA 255,0,0,0,255,0,0,0
2111 DATA 0,16,1,16,1,16,1,0
2120 DATA 153,102,153,102,153,102,153,102
2121 DATA 153,0,102,0,153,0,102,0
2130 DATA 40,20,65,130,40,20,65,130
2131 DATA 40,20,65,130,40,20,65,130
2140 DATA 17,34,68,136,17,34,68,136
2141 DATA 17,34,68,136,17,34,68,136
2150 DATA 1,3,7,15,16,48,112,240
2151 DATA 14,12,8,0,224,192,128,0
2160 DATA 9,6,9,6,144,96,144,96
2161 DATA 3,12,3,12,48,192,48,192
9999 SAVE "L.SHOW-2" LINE 1:VERIFY ""
```

Литература

1. Электронные устройства (Периферия "ZX Spectrum"). "V A PRINT". Москва, 1993. (Автор неизвестен).
2. А.Ларченко, Н.Раднонов. "ZX Spectrum" для пользователей и программистов. ИПК "Импакс". С.-Петербург, 1991.
3. Транзисторы (справочник). "Радио и связь". Москва, 1990.
4. В.Л.Шило. Популярные цифровые микросхемы. "Металлургия". Челябинск, 1988.
5. Е.П.Воробьев, К.В.Сенин. Интегральные мнкросхемы производства СССР и их зарубежные аналоги. "Радно н связь". Москва, 1990.

В.ИГНАТОВИЧ,
г.Минск, ул. Романовская слобода, 5,
DATAStream Communication Company,
тел. (0172) 203842, факс (0172) 265447,
e-mail:vitaly@ dast.minsk.by fido: 2:450/22, 2:450/23.

КОМПЬЮТЕРЫ ВСЕХ СТРАН— СОЕДИНЯЙТЕСЬ!

Распространение персональных компьютеров, оснащенных модемами, уже повлекло за собой множество изменений в устоявшихся формах общения между людьми. Привычного для всех почтальона в наше время не без успеха заменяет компьютер, позволяющий доставлять корреспонденцию во все уголки нашей планеты. Подключив однажды модем к своему персональному компьютеру, вы попадаете в бездонный мнр телекоммуникаций.

Как же происходит обмен информацией в сетях? Какое программное обеспечение для этого используется? Что нужно знать начинающему пользователю? Каждый вопрос требует подробного ответа.

Прежде всего следует отметить, что в отличие от простой почты, электронная имеет ряд разновидностей. Для начала остановимся на некоммерческой сети FIDONET, получившей заметное развитие в странах СНГ уже более 5 лет назад.

FIDONET— это крупнейшая в мире любительская сеть персональных компьютеров, нечто вроде сообщества радиолюбителей-коротковолновиков, но на другом уровне. FIDO — общественная неправительственная организация, служащая целям свободного обмена информацией между своими членами через станции BBS (электронная доска объявлений). Сеть FIDO имеет жесткую централизованную структуру, построенную на основе признака территориального деления. Основной структурной единицей сети является NODE (узел), имеющий свой сетевой адрес и устаиовленные маршруты соединения (routing map) с другими узлами в сети. Узлы в одном географическом регионе объединяются в сети, сети объединяются в регионы, регноны объединяются в зоны. На каждом уровне перархии сети имеется свой координатор, который выполняет функции администратора, а в случае возникновения спорных вопросов — функции судьи. Каждый узел в сети обязан поддерживать обмен информацией с узлами согласно установленным маршрутам, для чего выделяется особое время — NetMail Hour (час обмена почтовыми сообщениями). За работой станции BBS следит системный оператор (SysOp).

В сети FIDONET выделяют следующие типы информации: NetMail — личная корреспонденция между членами сети, EchoMail — сообщения, распространяемые через эхоконференции и доступные для всех членов сети. Также по сети возможна передача файлов — программ, документаций, графических изображений и пр.

Информация в сети FIDO носит некоммерческий характер и касается всевозможных сторон человеческой деятельности — науки, увлечений, музыки, кухонных рецептов и т.д. Естественно, значительная часть разделов посвящена компьютерам, электронной аппаратуре и программному обеспечению. Наименования эхоконференций могут включать уточнение тематики или регнона распространения, например SU — страны СНГ, BEL — Беларусь, RU — Россия, MO — Москва, PVT (private) — частная переписка и т.д. Вот для примера некоторые из эхоконференций, доступ к которым имеется на нашем узле 2:450/22 в Минске.

BEL.FILES	"Belarus Talks About Files" — переписка о файлах
BEL.HARDWARE	"Belarus Talks About Hardware" — переписка об аппаратном обеспечении
BEL.HOBBY	"Belarus Hobbies" — различные увлечения
BEL.SOFTWARE	"Belarus Talks About Software" — переписка о программном обеспечении
CDROM	"Int CD-ROMs Echo" — международная переписка о CD-ROM
NET_DEV	"Int Network Development" — о развитии сетей
NOVELL	"Int Talks About Novell" — о сетях Новелл
ZYXEL	"Int Talks About ZyXEL" — о модемах фирмы ZyXEL
PVT.CRACK	"Pvt Craking..." — проблемы взлома защит

PVT.EXCH. AUDIOVIDEO	"Pvt Exchange: Audio & Video" — обмен — аудио и видео
PVT.EXCH. COMPUTER	"Pvt Exchange: Computers" — обмен компьютерной техникой
RU.ANEKDOT	"Ru Funny Stories" — российские анекдоты
RU.BBSNEWS	"Ru BBS News" — новости российских BBS
RU.BORLAND.AO	"Ru Borland-Russia Centre Echo" — переписка центра Борланд-Россия
RU.GAME.RPG	"Ru Talks About RP Games" — для любителей компьютерных игр
RU.HACKER	"Ru Hackers' Echo" — переписка хаккеров
RU.MODEM	"Ru Talks About Modems" — о модемах
RU.STRACK	"Ru PC Music" — о музыке на персональных компьютерах
RU.VIDEO	"Ru Video Lovers Echo" — клуб любителей видео
SU.CHAINIK	"Su Echo For Beginners" — ответы начинающим
SU.HARDW. CHAINIK	"Su For Beginners in Hardware" — ответы начинающим по аппаратному обеспечению
SU.HARDW.PC. VIDEO	"Su PC Video" — переписка о видеосистемах PC
SU.OS2.FAQ.D	"Su OS/2 FAQ Discussions" — часто задаваемые вопросы (Frequently Asked Questions) в дискуссиях по OS/2
SU.POL	"Su Talks About Politics" — переписка о политике
SU.TOLKIEN	"Su JRRT Echo" — клуб любителей сюжетов Дж. Толкина
ADINF.SUPPORT	"Su AdInf Support Echo" — поддержка пользователей антивирусного сторожа AdInf
SU.VIRUS	"Su Talks About Viruses" — переписка о компьютерных вирусах
MO.SOFTEXCHANGE	"Moscow Software Exchange" — обмен программами в Москве

Пользователь, не являющийся членом сети, может познакомиться с имеющейся информацией, позвонив на любую из BBS, принадлежащих сети FIDO.

Для вызова BBS следует воспользоваться любым из распространенных коммуникационных пакетов (Telix, Comit. ProComm, Telemate, Mte и др.) Встроенные средства для работы с модемами имеются в Microsoft Windows, а также в полной поставке Norton Commander. Настройку модема, если для данной BBS она неизвестна, следует сделать такой: скорость максимальная для вашего модема — отсутствие контроля по четности (N) — длина слова 8 бит — 1 стоповый бит. Обязательно необходимо предварительно узнать часы работы BBS, иначе можно причинить значительное беспокойство своими неурочными звонками оператору станции. Помните, все обязанности члены сети принимают на себя добровольно и несут их на некоммерческих основаниях (бесплатно).

Расширить свои возможности можно, получив собственный адрес в сети FIDO — point. Наличие собственного адреса даст вам возможность осуществлять переписку с другими участниками сети, а также получать файлы через файловые конференции. Настроив программное обеспечение соответствующим образом, вы сможете участвовать в работе сети без затрат собственного времени: информация будет приниматься и рассылаться по нужным адресам в выделенное для этого время, скажем, когда вы спите.

Для этого следует связаться с системным оператором одного из узлов сети FIDO, который поможет уточнить технические моменты. Это можно сделать, направив письмо на имя SYSOP, например в Минске для адреса 2:450/22 или 2:450/23 с помощью пакетов Bink, Xenia и др.

Кроме сети FIDO, действует много доступных для публики BBS, не входящих в структуру сетей, и число их постоянно увеличивается. Такие BBS могут быть установлены в организациях, фирмах, или принадлежать частным лицам. Многие из них работают только в ночное время, когда телефонная линия владельца не занята.

Списки BBS, действующих в данном регионе, можно найти в местной печати. Кроме того, такие списки, регулярно обновляющиеся, имеются на других BBS. Следует иметь в виду, что поскольку работа BBS — это личное дело системного оператора, режим работы может быть нерегулярным. Это же относится и к содержанию информации: в вашем городе могут быть коллекционеры ПО для Windows, музыкальных программ, фантастической литературы и т.д. Переписка на таких BBS зачастую не делится по тематике.

Соединяясь с BBS, важно вести себя так, чтобы не доставлять неудобств системному оператору и другим пользователям. Если вы звоните первый раз, держите под рукой телефонную трубку. Может оказаться, что вам ответят голосом. В этом случае обязательно следует извиниться, объяснить причину своего звонка и поинтересоваться графиком работы BBS. Помните, SysOp не получает никаких денег за поддержание станции BBS и вы не имеете морального права отрывать его от работы или отдыха.

Важно следить за объявлениями, которые предъявляются вам при входе в систему, и точно следовать указаниям системного оператора. Чтобы не пропустить такие объявления из-за случайного нажатия клавиши, обязательно включите режим захвата (capture) информации в файл — тогда вы сможете потом разобраться в своих ошибках. При первом соединении с BBS следует зарегистрироваться — сообщить данные о себе (желательно соответствующие действительности). После регистрации ваш уровень доступа будет повышен, хотя возможно и не сразу, а это значит, что вам будет выделено больше времени и килобайт на каждый звонок (или на сутки).

Наверняка первое время вас будет больше всего интересовать получение файлов, имеющихся на BBS. Файлы на BBS хранятся в тематических областях (area). Хотя меню BBS предлагает такую возможность, считается некорректным просматривать списки файлов (иногда по объему достигающие сотни Кбайт) прямо на экране во время соединения (on-line), отнимая возможность дозвониться у других пользователей. Эти списки имеются на каждой BBS в виде отдельного архивированного файла (например, DATA-ALL.ARJ), который вы можете внимательно просмотреть, а затем позвонить на BBS вновь и заказать нужные вам файлы.

То же относится и к просмотру почты. Вместо того чтобы перекачивать тексты сообщений на экран, следует воспользоваться специальным форматом упаковки почты Blue Wave. Упакованный файл затем можно просмотреть программой просмотра (off-line reader) Blue Wave. Таким образом, Вы не будете занимать рабочего времени BBS, что позволит обслужить большее число пользователей.

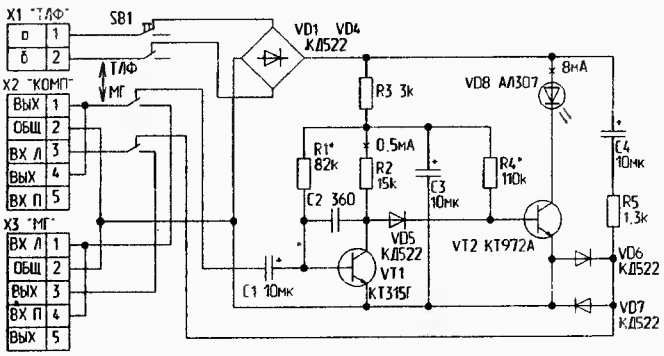
Передача файлов на BBS, вообще говоря, ценится и увеличивает ваши возможности по приему файлов. Однако надо строго следить, нет ли уже таких файлов на данной BBS, и соблюдать пожелания системных операторов. Иначе дело обернется понижением уровня доступа. Лучше всего предварительно оставить записку оператору и выяснить его мнение о необходимости передачи. Очень большие файлы, а также находящиеся на сменных дисках, отмечаются в списках как недоступные (offline) и для их получения также нужно обратиться с просьбой к системному оператору. Если BBS находится в том же городе, файл большой, а скорость вашего модема мала — проще всего обменяться программами лично, скопировав их на дискеты.

Некоторые BBS вводят оплату за свои услуги, чтобы компенсировать расходы владельца на получение информации, оплату телефона и т.п. Практика показывает, однако, что количество пользователей таких BBS резко сокращается, а объем информационных услуг чаще всего не оправдывает вложенные средства.

По всем затронутым здесь вопросам вы можете получить ответы, связавшись с DATAStream BBS в Минске по телефону 20-38-42 или 20-49-70 (время работы 21:00 — 9:00 МСК).

А.КОЖЕВКО, В.ФИЛАТОВ,
220092, г.Минск-92, а/я 86,
тел. 52-60-08.

ПОДКЛЮЧЕНИЕ "СПЕКТРУМА" К ТЕЛЕФОННОЙ ЛИНИИ



К сожалению, промышленными изготовителями упущена интереснейшая область применения бытовых компьютеров — связь по телефонным сетям. В [1] описан модем, способный успешно работать на "ZX Spectrum", но он имеет сравнительно сложную схему и должен подключаться к шине компьютера, что неудобно для большинства пользователей. Модем, разработанный для ATM [2], располагается на общей плате, отчего возникают сложности с его повторением.

Нами разработано устройство для согласования стандартного магнитофонного порта "Спектрума" с телефонной линией. При разработке схемы ставились задачи использования минимального количества доступных деталей, простоты подключения к компьютеру и достижения удовлетворительного качества связи. Компромисса удалось достичь программно-аппаратным способом. При отсутствии специальных программ можно пользоваться обычными копировщиками. К недостаткам конструкции можно отнести отсутствие гальванической развязки от телефонной сети.

Схема устройства состоит из диодного моста VD1...VD4 для подключения к ТЛФ линии с любой полярностью, светодиода VD8, индицирующего подключение к линии; формирователя коммутатора линии на транзисторах VT1 и VT2; входного фильтра и C4, R5 и ограничителя VD6, VD7 и переключателя SB1 "телефонная линия ◄► магнитофон". Настройка правильно собранного устройства сводится к подбору резисторов R4 и R1 по величине тока в коллекторных цепях транзисторов.

Устройство подключается параллельно телефонному аппарату. Для связи необходимо, как обычно, созвониться и договориться о передаче. Затем — переключателем SB1 включить устройство в линию. Как уже говорилось, в простейшем случае можно воспользоваться встроенными функциями интерпретатора BASIC (LOAD, SAVE, LOAD CODE, SAVE CODE), либо программой-копировщиком (желательно такой, которая не делает автоматическое уплотнение данных, например COPY-COPY). В этом случае качество связи сильно зависит от "удачности" соединения. При наличии сбоев можно попытаться перезвонить или покомбинировать с поднятием или опусканием телефонных трубок.

Чтобы меньше зависеть от качества канала и для удобства работы была написана программа CONNET, позволяющая работать как в режиме обычного копировщика с нормальной, двойной или тройной скоростью, либо с автоматической подстройкой скорости, так и в режиме квитирования с возможностью задания длины блока от 256 до 10000 байт. Эта программа имеет возможность работать с расширенной памятью.

Некоторые проблемы возникают при работе устройства с компьютером "БАЙТ" производства Брестского ПО вычислительной техники. Это связано с тем, что при обращении к встроенным функциям одновременно с приемом сигнала происходит вывод в магнитофонный порт, отчего возникает конфликт сигналов в линии. Кроме того, не каждый копировщик хорошо работает с этим компьютером из-за торможения процессора при работе с экранной областью ОЗУ.

Испытания модема и программного обеспечения с моделями "Baltik", "Ленинград" и др. показали вполне удовлетворительные результаты как при работе внутри города, так и при междугородной связи.

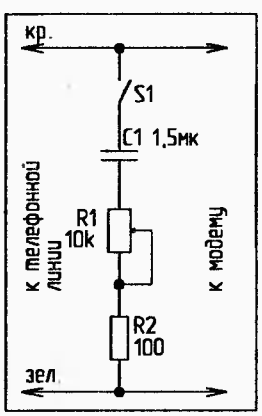
Литература

1. ZX-Ревю. — 1992. — N 2. — С.37.
2.Радиолюбитель.—1993.—N3.—С.8 - 9.

Из недр FIDO Net

T.NOFSINGER.

ПОДАВЛЕНИЕ ШУМОВОЙ ПОМЕХИ ДЛЯ МОДЕМА



Простое устройство оказалось полезным при работе с модемом US Robotics 2400 bps для устранения "мусора" на экране из-за шумовых помех в случае плохого качества телефонной линии.

Конденсатор должен быть неполярным и рассчитанным на напряжение не ниже 160 В. Емкость его может быть 1...2 мкФ.

Для настройки поверните движок потенциометра до упора в любую сторону. Соединитесь с той BBS, с которой у вас всегда есть проблемы из-за шумов. После того как соединение достигнуто и начал появляться "мусор", подключите устройство к линии (параллельно модему). Если потенциометр был выведен не в ту сторону, это обнаружится тотчас же: "мусора" станет больше или связь вообще оборвется. В таком случае поверните движок до упора в противоположную сторону. Если же никаких изменений после подключения устройства к линии не заметно, напечатайте какие-нибудь команды, следя за символами "мусора". Медленно вводите движок потенциометра — в какой-то точке связь должна резко ухудшиться или оборваться. Оптимальное положение регулятора, дающее эффект улучшения связи, находится опытным путем и на разных линиях может различаться, при этом сдвиг в пределах ±1 кОм практически не сказывается на работе схемы.

В.УБИЙКОНЬ,
БГУИР, ФИТиУ, каф. вычислительных методов и программирования",
тел. 39-89-56.

БЫСТРОДЕЙСТВУЮЩИЙ АЛГОРИТМ СОРТИРОВКИ ДАННЫХ

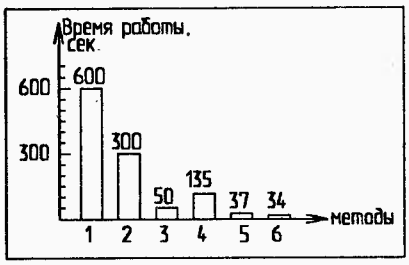
В учебные программы по информатике включены вопросы, посвященные алгоритмам сортировки различных данных. Под сортировкой понимается упорядочение набора данных по какому-нибудь признаку.

Существует много способов сортировки данных. К наиболее популярным относятся:

1) — сортировка по индексам;
2) — метод "пузырька";
3) — сортировка Шелла;
4) — сортировка Бэтчера;
5) — пирамидальная сортировка;
6) — сортировка Хоара и т.д. [1].

В рамках секции Школы юного программиста при кафедре ВМиП БГУИР и отдела Интеллектуального творчества "Одаренные дети" при МДДиМ была составлена программа на алгоритмическом языке Паскаль версии 5.5 для анализа скорости сортировки для ПЭВМ типа ЕС-1849 (80286/12 МГц). В программе генерируется массив 5000 случайных чисел, который сортируется различными способами. Время сортировки этого массива:

- сортировка по индексам — 10 мин;
- метод "пузырька" — 5 мин;
- метод Шелла — 50 с;
- метод Бэтчера — 2 мин 15 с;
- пирамидальная сортировка — 37 с;
- быстрая сортировка Хоара — 34 с.



Таким образом, наиболее быстродействующим оказывается алгоритм сортировки Хоара [2]. Для объективного сравнения методов в качестве исходных данных были сгенерированы 2000 чисел по следующему правилу: первое число равно 2000, второе число — 1999, третье — 1998, четвертое — 1997 и так далее, и наконец, последнее сгенерированное число равно 1. Числа упорядочивались по возрастанию.

Время работы алгоритмов сортировки с таким набором данных таково:

- сортировка по индексам — 24 мин 45 с. Потребовалось 1999000 сравнений чисел и столько же пересылок;
- метод Шелла — 9 с, 6764 сравнения и столько же пересылок;
- метод Бэтчера — 40 с, 113555 сравнений и 8128 пересылок;
- пирамидальная сортировка — 13 с, 10986 сравнений и 1999 пересылок;
- быстрая сортировка Хоара — 7 с, 2022 сравнений и столько же пересылок.

Суть алгоритма Хоара в следующем [2]:

1. Имеется два указателя i и j, причем вначале i=1, j=N (N — количество сортируемых чисел).

2. Сравнить a[i] и a[j], и если нужно (по условиям сортировки) — поменять эти элементы местами. Уменьшить j на 1 и повторить сравнение.

После первого обмена увеличить i на 1, сравнить a[i] с a[j] для очередного обмена. Потом опять уменьшить j на 1 и т.д. Процесс продолжается, пока не выполнится условие i=j.

3. В итоге выполнения пункта 2 исходный массив A делится на две части согласно условию сортировки.

4. Аналогично описанному алгоритму сортируются элементы подмассива a[1], a[2], ..., a[i-1] и элементы подмассива a[i+1], a[i+2], ..., a[n].

Рассмотрим числовой пример. Пусть исходный массив чисел таков:

{6, 2, 11, 8, 1, 13, 12, 7}.

1. Установим значение i=1 и значение j=8;

2. Сравним a[1]=6 и a[8]=7. Так как 6<7, обмен делать не надо. Уменьшаем j на 1 (j=7) и сравниваем a[1]=6 с a[7]=12. Обмен делать не надо. Опять уменьшаем j и делаем сравнение. Процесс продолжается до тех пор, пока не обменяются a[1]=6 и a[5]=1. Устанавливаем j на место i (j=1), а i устанавливаем на место j (i=5). Продолжаем процесс проверки (теперь j увеличиваем на 1 и сравниваем числа a[j] с a[i].

3. При достижении условия i=j=4 массив чисел разделен на две части.

4. Аналогично описанному алгоритму сортируются подмассивы {1, 2, 6} и {11, 13, 7}.

Листинг программы быстрой сортировки Хоара для 5000 псевдослучайных чисел приведен далее. Программа занимает 1598 байт, и в ней применяется рекурсия.

```
program HOAR;(* 05.06.95 *)
(* быстрая сортировка Хоара *)
uses crt;
label 99;
const nn=5000;
type aa=array[1..nn] of real;
var a:aa;n,i,move,compare:integer;
(* * * * * * * * * * * * * * * * * * * * * * * * * * * *)
PROCEDURE QUICK;
(* ..........................................*)
procedure sort(l,r:integer);
   var i,j:integer;w,x:real;
   begin i:=l;j:=r;
   x:=a[(l+r) div 2];
   REPEAT
      while a[i]<x do begin compare:= compare+1;
      i:=i+1;end;
      while x<a[j] do begin  compare:=compare+1;
      j:=j-1;end;
      if i<=j then begin
      w:=a[i];a[i]:=a[j];a[j]:=w;i:=i+1;j:=j-1;
      move:=move+1;end
   UNTIL i>j;
   if l<j then SORT(L,J);
   if i<r then SORT(I,R);
   end;
(*.........................................*)
begin
compare:=0;move:=0;
SORT(1,n);
end;
(*QUICK*)
(* * * * * * * * * * * * * * * * * * * * * * * * * * * *)
BEGIN
   clrscr;
   writeln(' введите N НЕ БОЛЕЕ 5000! ');readln(n);
   if n>=5001 then begin
   writeln('     ВВЕДЕНО N=',n:6,
   ' РАБОТА ОКОНЧЕНА ');goto 99;end;
   for i:=1 to n do a[i]:=random;
   QUICK;
   writeln('... отсортированный массив ... ');writeln;
   for i:=1 to n do write('a[',i:2,']=',
   a[i]:6:3,' '); writeln;
   writeln(' пересылок=',move:6,'     сравнений=',
   compare:6);
   repeat until keypressed;
   99: repeat until keypressed;
END.
```

Литература

1. Д.Кнут. Искусство программирования для ЭВМ.Т.3. — М.: Мир., 1978. — 843 с.

2. Д.Алкок. Язык Паскаль в иллюстрациях. — М.: Мир, 1991, — 192 с.

ДОМАШНЕЕ ЗАДАНИЕ

ЗАДАЧИ МЕЖДУНАРОДНОЙ ОЛИМПИАДЫ ПО ПРОГРАММИРОВАНИЮ

В мае 1994 года в белорусском городе Могилеве состоялась Международная олимпиада школьников под эгидой "INTERNATIONAL COMPETITION IN INFORMATICS". Приводим для любознательных условия задач первого и второго туров. Оцените свои силы и возможности.

Задача 1

Матрица размером N строк и M≤255 столбцов представляет собой компьютериый холст. На нем изображена картина, нарисованная с помощью взаимно непересекающихся контуров фигур. Каждая фигура представляет собой одну из трех геометрических фигур: треугольник, прямоугольник, окружность.

Каждая фигура нарисована своим изображаемым символом. В качестве символа для заполнения фона выбран символ "."(точка). Пример изображения одной из фигур:

```
.....
....T.....
....TT....
....T.T...
....T..T..
....T...T.
....T....T
....T.....T
....T.....T
....TTTTTTTT
.....
```

Задание:

Определить, каким символом нарисована какая фигура и количество фигур.

Входные данные:

Вводится имя файла данных (длина имени файла <13). Структура файла имеет следующий вид:

1-я строка (N)
2-я строка (M)
3-я строка (1-я строка холста)
4-я строка (2-я строка холста)
.....
N+2-я строка (N-я строка холста)

Результат должен быть представлен в следующем виде:

Символом <изображение символа> нарисован(а) <название фигуры>.
.....
Символом <изображение символа> нарисовани(а) <название фигуры>.

Количество фигур <количество>.

Примечание: Входные данные корректны.

Задача 2

Пусть задана система односторонних дорог между городами. Это значит, что определены пары городов, где каждая пара определяет одностороннюю дорогу из города, первого в паре, в город, второй в паре. При этом имена городов есть некоторые натуральные числа. Количество городов — не более 1000. Длина каждой дороги равна 1. Количество дорог — не более 4096.

Необходимо в заданной системе дорог найти один из минимальных по длине замкнутых путей. Путь называется замкнутым, если он начинается и оканчивается в одном и том же городе и имеет ненулевую длину. Длина пути есть суммарная длина дорог, входящих в путь.

Ввод данных осуществляется из файла с именем FILE.IN.

Вывод данных осуществляется в файл с именем FILE.OUT.

Входные данные:

- количество тестов;

Для каждого теста вводится следующая информация:

- количество городов;
- количество пар городов;
- пары городов.

Выходные данные:

Для каждого теста выводится следующая информация:

Тест номер: <номер>

Длина минимального замкнутого пути= <длина>

Последовательность городов пути: <последовательность городов>.

Пример ввода:

```
2   — количество тестов
2   — количество городов в 1-ом тесте
1   — количество пар городов в 1-ом тесте
1 2 — пара
3   — количество городов во 2-ом тесте
3   — количество пар городов во 2-ом тесте
1 2 — пара 1
2 3 — пара 2
3 1 — пара 3
```

Примечание: Входные данные корректны.

Задача 3

Начиная с 80-х годов в военной области большое распространение получили автоматические роботы-шпионы. Работая только на приём, они накапливают информацию и при получении определённого сигнала быстро выдают её в эфир, что делает их трудно обнаруживаемыми.

Схема, моделирующая систему "база-шпион", состоит из двух интерфейсов Международного союза ОРТ и двух компьютеров. Каждый интерфейс подключён к отдельному компьютеру, один из которых является "базой", а другой — "шпионом". Интерфейсы соединены друг с другом следующим образом: все восемь выходов одного из интерфейсов соединены с восемью входами другого интерфейса. Для того чтобы послать сигнал (байт) на компьютер-шпион, необходимо записать байт в порт с адресом 642. Для получения сигнала (байта) от компьютера-шпиона требуется прочитать байт из порта с адресом 642. Биты в байте нумеруются по мере возрастания разрядов.

Задание:

В компьютере-шпионе находится текст неизвестной длины, заканчивающийся кодом 0. При поступлении от компьютера-базы сигнала с кодом 255 компьютер-шпион начинает передачу текста по следующим правилам:

1. Каждый символ из строки последовательно, с задержкой в 3 миллисекунды, посылается на 642-ой порт компьютера.
2. При достижении конца текста компьютер-шпион начинает передачу сначала.
3. Сигнал с кодом 0 от компьютера-базы приводит к прекращению передачи.
4. При передаче в силу работы систем подавления 1 или 2 бита в передаче каждого байта искажаются.
5. Передача завершающего текст кода 0 также проходит с искажениями.

Написать программу для компьютера-базы, которая выводит на монитор текст, находящийся в компьютере-шпионе.

Примечание:

Выдача байта информации D в порт с адресом 642 и считывание байта D из порта с адресом 642 производится следующим образом.

Язык	Запись в порт	Чтение из порта
C	OUTPORTB(642,	D=INPORTB(642)
PASCAL	PORT[642]:=D	D:=PORT[642]
BASIC	OUT 642,D	D=INP(642)

А.КАЛИНОВСКИЙ,
Школа-лицей N 165 при БГУИР,
11 "Р" класс,
220117, г. Минск,
просп.газ. "Известия", 28 — 194,
тел. 71-50-75.

"THE COLUMNS"

Даиная игра является одиой из разиовидностей популярнейшей игры "TETRIS". Цель игры — набрать как можно больше очков, ставя колоики таким образом, чтобы одинаковые элемеиты (цвета, символы, цифры, буквы) образовывали линию, состоящую из трёх и более одинаковых элементов по вертикали, по горизоитали или по любой из диагоналей. Такие линии исчезают, и вам начисляется одно очко. После набора каждых десяти очков вы переходите на следующий уровень. На каждом последующем уровие скорость игры увеличивается и вам остаётся всё меньше времени для раздумий.

Игра очень динамична и довольио-таки трудна. Попробуйте набрать хотя бы сто очков!

Представлеиная здесь версия игры THE COLUMNS является самой короткой и самой трудной из всех существующих.

При работе используются только стаидартные TPU-модули Turbo Pascal 5.5 — Dos и Crt.

```
PROGRAM Columns;
USES Dos,Crt;
CONST
  B:array[1..3] of byte=(0,48,65);
  kb5=53;kbB=98;kbShiftB=66;kb4=52;kb6=54;
  kbQ=113;kbShiftQ=81;kbSpace=32;kbEnter=13;
VAR
  M,N:array[1..20,1..6] of byte;
  F:array[1..3] of byte;
  P:array[1..6] of byte;
  Flag,I,J,X,Y,A,Q:byte;
  L,Speed1,Speed,Sn,C,SC,SC1:word;
  Flag1,Flag2:boolean;
  kbHeadPtr:word absolute 0:$41a;
  kbTailPtr:word absolute 0:$41c;
  R:Registers;

FUNCTION GetKey:word;
BEGIN
  IF kbTailPtr=kbHeadPtr THEN
  BEGIN GetKey:=0;Exit END;
  R.AH:=1;Intr($16,R);
  IF R.AL=0 THEN GetKey:=R.AH+1000
  ELSE GetKey:=R.AL
END;

PROCEDURE AnyKey;
BEGIN
 kbTailPtr:=kbHeadPtr;REPEAT UNTIL GetKey<>0
END;

PROCEDURE Game;
BEGIN
  A:=TextAttr;
  TextColor(Red);GotoXY(14,2);
  Writeln('THE  COLUMNS');
  TextColor(Yellow);Writeln;Writeln;
  Writeln('  Данная игра является одной из ');
  Writeln('разновидностей популярной игры TETRIS.');
  TextColor(White);Writeln;
  Writeln('            Задача всё та же: ');
  Writeln('     набирать очки, ставя колонки ');
  Writeln('   таким образом, чтобы одинаковые ');
  Writeln('    цвета (символы, цифры, буквы)');
  Writeln('    образовали линию по дигонали,');
  Writeln('     вертикали или горизонтали.');
  Writeln;TextColor(Red);
  Writeln('   Данная версия этой известной игры');
  Writeln('   является самой короткой из всех');
  Writeln('   существующих. При этом игра ');
  Writeln('   не утратила своей динамичности.');
  Writeln('     Игра довольно-таки трудна и');
  Writeln('развивает реакцию и быстроту мышления.');
  TextAttr:=A;AnyKey
END;

PROCEDURE Music;
BEGIN
  Sound(1397);Delay(60);
  Sound(1568);Delay(60);
  Sound(1760);Delay(60);
  Sound(2200);Delay(150);
  Sound(1760);Delay(60);
  Sound(2200);Delay(200);NoSound;
end;

PROCEDURE HideCursor;
BEGIN
  R.AH:=1;R.CH:=32;R.CL:=1;Intr(16,R)
END;

PROCEDURE GetFigure;
BEGIN
  FOR I:=1 TO 3 DO F[I]:=Round(1+Random(6))
END;
PROCEDURE Fon;
BEGIN
  A:=TextAttr;GoToXY(18,21);
  TextBackGround(3);Write('         ');
  FOR I:=1 TO 20 DO BEGIN
    GoToXY(18,I);Write(' ');GoToXY(25,I);Write(' ')
  END;
  GotoXY(1,5);HighVideo;TextBackGround(0);TextColor(15);
  Writeln('  4  : LEFT');Writeln('  6  : RIGHT');
  Writeln('  5  : SWAP');Writeln('Space: DOWN');
  Writeln('Enter: CHANGE');
  Writeln('  B  : NEW GAME');
  Writeln('  Q  : QUIT');NormVideo;
END;

PROCEDURE ShowFigure;
BEGIN
  A:=TextAttr;FOR I:=0 TO 2 DO IF Flag=1 THEN
  BEGIN
  TextBackGround(F[I+1]);GoToXY(X,Y-I);
  Write(' ') END else
  BEGIN GotoXY(X,Y-I);Write(chr(F[I+1]+B[Flag-1]))
  END;
  TextAttr:=A
END;

PROCEDURE ChangeFigure;
BEGIN
  Q:=F[1];F[1]:=F[2];F[2]:=F[3];F[3]:=Q
END;

PROCEDURE ShowAll;
BEGIN
  FOR I:=1 TO 20 DO FOR J:=1 TO 6 DO
  IF Flag=1 THEN BEGIN
    TextBackGround(M[I,J]);GoToXY(18+J,21-I);
    Write(' ')
  END ELSE BEGIN
    IF M[I,J]=0 THEN BEGIN
      GotoXY(18+J,21-I);Write(' ')
    END ELSE BEGIN
      GoToXY(18+J,21-I);
      Write(chr(B[Flag-1]+M[I,J]))
    END
  END
END;

PROCEDURE Check(II,JJ:shortint);
```

```
BEGIN
  IF (M[I+II,J-JJ]=M[I,j]) AND (M[I,j]=M[I-II,J+JJ])
  THEN BEGIN Inc(SC);N[I,J]:=1;N[I+II,J-JJ]:=1;
  N[I-II,J+JJ]:=1 END;
END;

PROCEDURE Main1;
BEGIN
  FOR I:=1 TO 6 DO FOR J:=1 TO 20 DO N[J,I]:=0;
  FOR J:=2 TO 5 DO FOR I:=2 TO 19 DO
  IF M[I,j]<>0 THEN BEGIN
    Check(0,1);Check(-1,1);Check(1,1);Check(1,0);
  END;
  FOR I:=2 TO 19 DO BEGIN
    IF (M[I,1]<>0) THEN BEGIN J:=1;Check(1,0) END;
    IF (M[I,6]<>0) THEN BEGIN J:=6;Check(1,0) END;
  END;
  I:=1;FOR J:=2 TO 5 DO IF (M[I,J]<>0)
  THEN Check(0,1);
  FOR I:=1 TO 6 DO P[I]:=1;Flag1:=False;
  FOR I:=1 TO 20 DO FOR J:=1 TO 6 DO
  IF N[I,J]=1 THEN BEGIN
    Flag1:=True;TextBackGround(0);GotoXY(18+J,21-I);
    Write(' ')
  END;
  IF Flag1 THEN BEGIN
    SN:=500;
    REPEAT Sound(SN);Delay(50);
    Inc(SN,20) UNTIL SN>700;
    NoSound;
  END;
  FOR I:=1 TO 20 DO FOR J:=1 TO 6 DO
  IF N[I,J]=0 THEN BEGIN M[P[J],J]:=M[I,J];
  Inc(P[J]) END
END;

PROCEDURE DelFigure;
BEGIN
  FOR I:=0 TO 2 DO
  BEGIN GoToXY(X,Y-I);Write(' ') END;
END;

PROCEDURE MAIN;
BEGIN
  ShowAll;
  REPEAT
    GetFigure;
    X:=Round(19+Random(3));Y:=3;Flag2:=False;
    Speed1:=Speed;
    REPEAT
      GotoXY(1,1);TextColor(15);HighVideo;
      Write('SCORE: ');
      LowVideo;Writeln(SC);HighVideo;Write('LEVEL: ');
      LowVideo;Write(SC DIV 10);kbTailPtr:=kbHeadPtr;
      DelFigure;Inc(Y);ShowFigure;Delay(Speed);
      C:=GetKey;
      CASE C OF
        kbspace:begin speed:=50;flag2:=true; end;
        kbenter:begin if flag<4 then inc(flag)
        else flag:=1;
        showall;showfigure end;
        kbshiftq,kbq:begin
          kbHeadPtr:=kbTailPtr;textmode(co80);HALT(0);
        end;
        kbshiftb,kbb:EXIT;
        kb4:IF (X>19) then if (M[21-Y,X-19]=0) THEN
          BEGIN DELFIGURE;DEC(X) END;
        kb6:IF (X<24) then if (M[21-Y,X-17]=0) THEN
          BEGIN DELFIGURE;INC(X) END;
        kb5:CHANGEFIGURE
      END
  UNTIL (M[20-Y,X-18]<>0) OR (Y=20);
  IF Y=4 THEN EXIT;
  M[21-Y,X-18]:=F[1];M[22-Y,X-18]:=F[2];
  M[23-Y,X-18]:=F[3];
  REPEAT SC1:=SC;MAIN1;SHOWALL;if flag2 and
  (sc<>sc1) then inc(sc,1) UNTIL SC1=SC;
  speed:=speed1;
  if (sc>=1) then begin inc(1,10);dec(speed,30);
  music; end;
  UNTIL FALSE
END;

BEGIN
  REPEAT
    FOR I:=1 to 6 DO P[I]:=1;Randomize;SC:=0;
    TextMode(Co40);SC1:=0;HideCursor;Game;
    Flag:=1;Speed:=400;Speed1:=400;L:=10;
    TextColor(0);TextBackGround(0);ClrScr;Fon;
    FOR I:=1 TO 20 DO FOR J:=1 TO 6 DO M[I,J]:=0;
    Main;ClrScr;GotoXY(17,12);TextColor(15);
    HighVideo;Write('SCORE:');
    LowVideo;Write(SC);kbTailPtr:=kbHeadPtr;AnyKey
  UNTIL False
END.
```

В.ДОНЧЕНКО.

ДЖОЙСТИК ДЛЯ IBM PC

Во 2-ом номере сборника "Компьютер" за 1990 год я прочитал статью "Палка, приносящая радость" Анджея Поплавского о джойстике. Автор подробно описал подключение джойстика к 8-разрядным домашним компьютерам, но только упомянул о возможности использования его на IBM PC. Это и натолкнуло меня на мысль попробовать здесь свои силы.

Джойстик привлек меня тем, что, во-первых, он очень прост и понятен в обращении (особенно для новичков в компьютерном деле: куда надо, туда и гни "палку"), во-вторых, это новое устройство, не требующее доработки машины и не мешающее уже имеющимся. Принцип указания направления наталкивает на мысль о разделении функций с традиционной "мышью" ("мышь" — указывает и выбирает, джойстик — направляет и перемещает).

Остановка за малым — купить джойстик и подключить к машине. Здесь я встал перед удивительным фактом: джойстиков в магазинах хватает, цены на них достаточно приемлемые, а спрос совсем невелик. Но здравый смысл (как, впрочем, и различия в разъемах) подсказывали, что эти джойстики, вероятно, просто так подключаться к IBM PC не захотят.

Информацию о физическом принципе и строении джойстиков и Game-адаптера я нашел в технической документации по IBM PC XT.

Игровой адаптер допускает подключение к системе до четырех ручек управления или двух джойстиков. Плата Game-адаптера устанавливается в одну из панелей расширения системной платы.

Интерфейсный кабель присоединяется в задней части адаптера. Положение ручки или джойстика определяется изменением величины сопротивления, передающегося на адаптер. Адаптер вместе с системным программным обеспечением преобразует имеющуюся величину сопротивления в относительное положение ручки или джойстика. Кроме того, обеспечивается 4 входа для подключения переключателей (кнопок). Этот адаптер может быть использован как схема ввода-вывода общего назначения с четырьмя аналоговыми (резистивными) и четырьмя цифровыми входами.

Каждый из четырех цифровых входов имеет резистор величиной 1 кОм, подключенный к +5В. Если на эти входы ничего не подается, на них присутствует логическая "1". Для того чтобы прочитать логический "0", эти входы должны быть замкнуты на общий провод.

Четыре аналоговых входа подключаются через переменные

резисторы на +5В. С помощью схемы одновибраторов сопротивление преобразуется в длительность импульса в соответствии со следующим соотношением:

T, мкс = 24,2 + 0,011 (R, кОм)

Пользователь должен сначала начать диалог, выдавая команду OUT по адресу 201h. Эта команда запускает все четыре одновибратора, их выходы одновременно переходят в состояние "1" и остаются в этом состоянии в течение времени, соответствующего величине сопротивлений.

Команда IN по адресу 201h позволяет прочитать текущее состояние адаптера.

Бит 7	Бит 6	Бит 5	Бит 4	Бит 3	Бит 2	Бит 1	Бит 0
цифровые входы				резисторные входы			

Типовым входным устройством для игрового адаптера являются джойстик либо игровые ручки. Джойстиков обычно два, каждый из них может иметь одну или две нормально разомкнутые кнопки и два переменных резистора величиной от 0 до 100 кОм. Одно переменное сопротивление указывает координату X, а другое — координату Y. Они должны присоединяться к адаптеру для получения следующих данных:

Бит 7	Бит 6	Бит 5	Бит 4	Бит 3	Бит 2	Бит 1	Бит 0
B-#2	B-#1	A-#2	A-#1	B-Y	B-X	A-Y	A-X
кнопка	кнопка	кнопка	кнопка	координ.	координ.	координ.	координ.

Игровые ручки бывают в наборе из двух (A, B) или четырех (A, B, C, D) ручек. Каждая из них имеет одну кнопку и одно переменное сопротивление величиной 100 кОм. Они должны присоединяться для получения следующих входных данных:

Бит 7	Бит 6	Бит 5	Бит 4	Бит 3	Бит 2	Бит 1	Бит 0
D	C	B	A	D	C	B	A
кнопка	кнопка	кнопка	кнопка	координ.	координ.	координ.	координ.

Точную схему разводки гнезда Game-адаптера (рис.1) я почерпнул из "Baby AT I/O Card GW211 User's manual" (эта карта стояла у меня в машине).

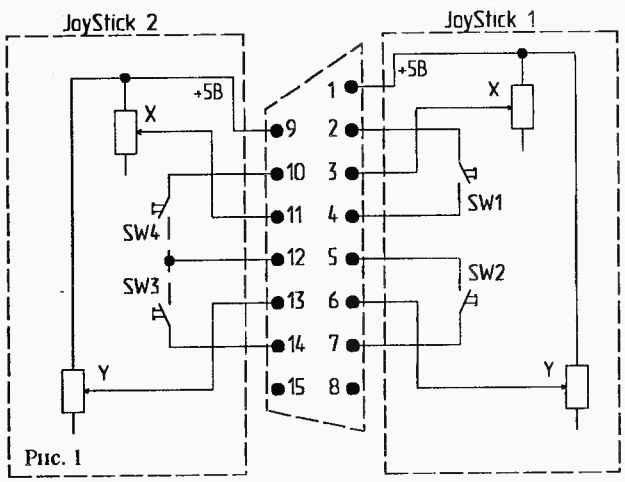


Рис. 1

Итак, я решил переделать стандартный (магазинный) джойстик для использования в IBM PC. Первое, что необходимо было сделать, это обеспечить резистивность сигналов координат в интервале от 0 до 100 кОм. Так как джойстик я собирался использовать для указания направлений, то решил обойтись дискретной резистивностью с тремя порогами: 0, 50 и 100, где 0 соответствует максимальной координате, 50 — среднему положению, а 100 — минимальному.

В тот же день мною был куплен игровой манипулятор "Джойстик" производственного предприятия "Альфа" (г.Ростов-на-Дону). В нем при вскрытии оказалась классическая схема с замыканием по координатам (рис.2).

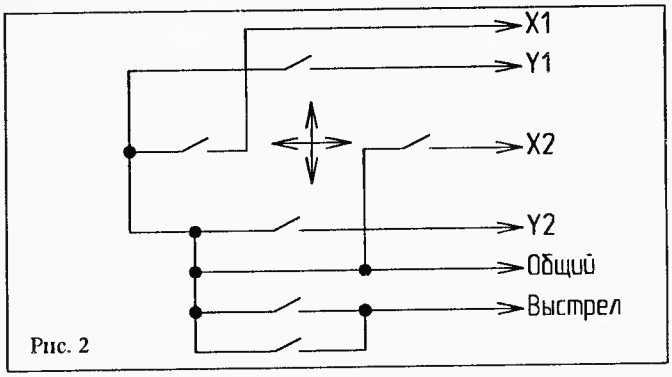


Рис. 2

После установки четырех резисторов по 100 кОм и превращения двух замыкающих контактов на координатах в размыкающие получилась схема, приведенная на рис.3. Необходимо отметить, что кнопки замыкаются на сигнальную "землю", а координаты — на +5 В (в исходной схеме они шли в одну точку). Игнорирование этого момента приводит к сбоям (у меня это проявлялось остановкой винчестера при нажатии на любую кнопку джойстика).

Так как собственный шнур джойстика был коротким, но достаточно качественным и содержал необходимые 6 проводов, его разумно было оставить, а соединение произвести через переходник в два этапа (рис.4). Кабель от джойстика остается со своим разъемом, а переходник от машины оканчивается двумя гнездами, что позволяет, во-первых, подключать сразу два джойстика, а во-вторых, свободно их переключать.

Тестировать джойстик удобно программой CheckIt. При входе в тест джойстика CheckIt сбрасывает Game-адаптер, поэтому первые движения курсора джойстика случайны и неуправляемы, но через пару секунд все приходит в норму.

И вот, наконец, джойстик был собран. Первое впечатление от использования джойстика я получил, запустив игру "PRINCE OF PERSIA" в режиме управления джойстиком (Ctrl+J). Мне очень понравилось.

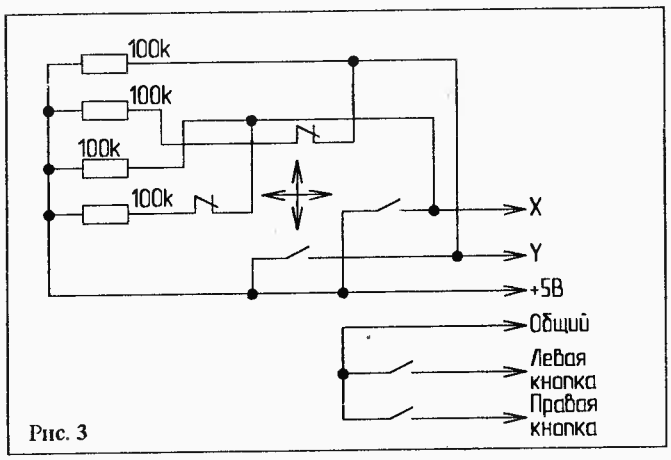


Рис. 3

Наступил момент организации взаимодействия с новым устройством из программ.

Функции получения информации о состоянии кнопок и координатах джойстиков (BIOS AT, Int 15h, Fn 84h) были описаны и тривиальны, но они работали только в том случае, если подключен хотя бы один джойстик и вообще имеется Game-адаптер. Пришлось написать следующую процедуру проверки наличия Game-адаптера и подключенных джойстиков:

```
        . . .
; — Инициализация джойстика —
   ; — Тестируем Game-адаптер
mov   dx,201h   ;Game port
out   dx,al     ;Сбросили данные в Game
```

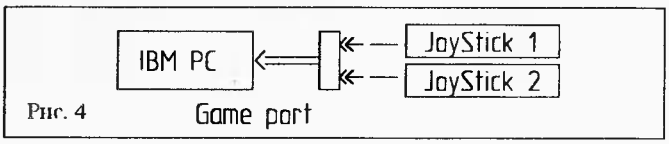


Рис. 4

```
 mov  bx,3      ;3 глобальных цикла
 mov  cx,-1
ask_joistick:
 in   al,dx     ;Читаем данные Game
 mov  bx,bx
 loop ask_joistick
 cmp  al,0FCh  ;Проверка установленных данных
 jz   Yes_Game
 mov  cx,-1
 dec  bx
 jnz  ask_joistick
 jmp  No_Game  ;По всей видимости, Game-адаптер
               ;отсутствует
       ; - Читаем клавиши джойстиков
Yes_Game:
 mov ah,84h ! mov dx,0
 int 15h        ;Запрос состояния клавиш джойстиков
 test al,0F0h
jz No_Game      ;Неправдоподобное состояние
                ;означает отсутствие Game-адаптера
       ; - Читаем координаты джойстика
 mov ah,84h ! mov dx,1 ! int 15h
 cmp ax,40h ! jb No_Joi1
 cmp ax,70h ! ja No_Joi1
 cmp bx,40h ! jb No_Joi1
 cmp bx,70h ! ja No_Joi1
 or  @Joi1,84h   ; джойстик 1 - активен
 mov @Joi1_time,10
No_Joi1:
 cmp cx,40h ! jb No_Joi2
 cmp cx,70h ! ja No_Joi2
 cmp dx,40h ! jb No_Joi2
 cmp dx,70h ! ja No_Joi2
 or  @Joi2,84h   ; джойстик 2 - активен
 mov @Joi2_time,10
No_Joi2:
No_Game:
       ; - Конец инициализации --
```

Таким образом, я получил новое устройство взаимодействия с программами. Надеюсь, что мой опыт пригодится кому-нибудь и внесет некоторое оживление в будущее пользовательских интерфейсов.

В.БАБЫНИН,
353660, г.Ейск, Краснодарского края,
ул.Нижнесадовая, 402.

РЕМОНТ ДЖОЙСТИКА ДЛЯ "ДЕНДИ"

Наиболее частой причиной выхода из строя джойстика — пульта управления игрового компьютера типа "Денди" — являются обрывы проводников соединительного кабеля. Практика ремонта показывает, что такой кабель целесообразно выбросить сразу, не ремонтируя. Новый кабель проще всего изготовить из сплетенных проводов МГШВ-0,2, подключив их к вилке, а в компьютере установить розетки соединителя ОНЦ-ВГ-4-5/16В (бывший разъем СГ-5).

При выходе из строя микросхемы на печатной плате джойстика ее можно заменить двумя микросхемами: ЛН1 и ИР9 серий 555 или 1533. Для этого следует аккуратно перерезать печатные проводники вокруг неисправной микросхемы; на стороне печат-

Вы перевернули последнюю страницу первого номера приложения к нашему журналу — "Радиолюбитель. ВАШ КОМПЬЮТЕР". Чтобы редакция могла сделать это приложение максимально интересным для каждого читателя, нам необходимо знать Ваше мнение о его будущем облике. Поэтому мы и предлагаем эту анкету. Подчеркните нужное и допишите непредусмотренное нами.

АНКЕТА

1. Какие вопросы применения компьютеров Вас интересуют?
- работа с пользовательскими пакетами (редакторы текстов, базы данных, САПР и т.д.)
- программирование и алгоритмы решения задач
- работа с графикой, звуком, видеоинформацией
- игры
- другое (что именно)

2. Какая тематика интересует Вас больше всего?
- языки программирования
- локальные сети и коммуникации
- схемотехника ПК, ремонт и диагностика неисправностей
- периферийные устройства
- другое (что именно)

3. Какие компьютеры Вас интересуют?
- IBM PC
- ZX Spectrum
- Бытовые компьютеры выпуска прошлых лет (БК-0010, Микроша, Вектор-06Ц)
- радиолюбительские компьютеры (Радио-86РК, Орион-128)
- другое (что именно)

4. Что бы Вы хотели видеть в приложении "ВАШ КОМПЬЮТЕР"?
- обзоры состояния и тенденций развития компьютерной техники
- материалы для начинающих
- учебные материалы по информатике
- инструкции для пользователей ПК
- материалы для конструкторов и разработчиков программного обеспечения
- другое (что именно)

5. Назовите 2 — 3 материала из напечатанных в журнале "Радиолюбитель", которые оказались лично для Вас полезными или особенно интересными.

Ниже Вы можете написать, о чем бы Вам хотелось прочитать в ближайших номерах "ВАШ КОМПЬЮТЕР", и добавить сведения о себе, которые Вы бы хотели сообщить редакции.

Анкеты (можно и в произвольной форме) просим высылать по адресу: 220095, г.Минск, а/я 199. Бельскому И.М.

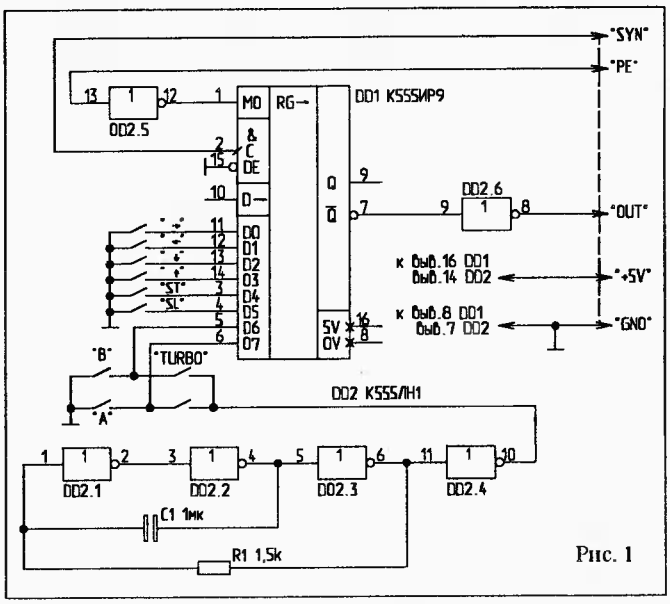


Рис. 1

ной платы, свободной от проводников, приклеить корпуса микросхем ЛН1 и ИР9 так, чтобы они не мешали сборке пульта, и тоикими проводниками произвести соединения согласно принципиальной схеме (рис.1).

Расположение проводников в разъемах соединительного кабеля показано на рис.2.

При использовании микросхемы ИР9 серии 1533 ее выводы 3, 4, 5, 6, 11, 12, 13, 14 необходимо дополнительно подключить через резисторы (на принципиальной схеме не показаны) сопротивлением 51...100 кОм к проводу источника питания 5 В.

Работает джойстик следующим образом. Информация о положении кнопок управления в виде параллельного кода поступает на входы D. При поступлении сигнала "PE" информация с входов D записывается в регистр, и на выходе Q появляется уровень, соответствующий разряду D7 входного кода (рис. 3). Тактовые импульсы "SYN", поступающие на вход C, сдвигают информацию вправо, т.е. вызывают поочередное появление соответствующих разрядов входного кода на выходе Q. Через инвертор DD2.6 сформированная кодовая последовательность импульсов поступает на выход джойстика "OUT".

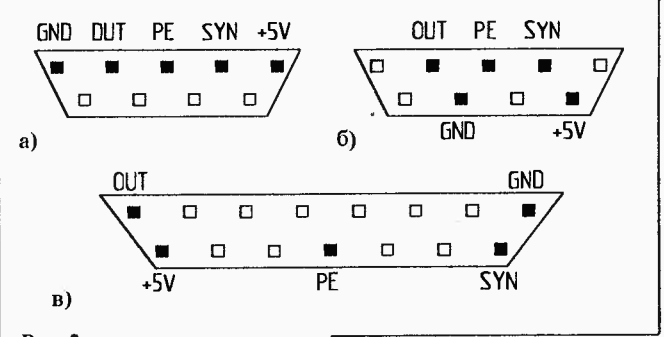


Рис. 2

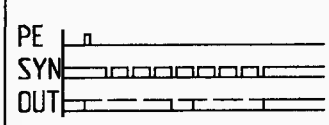


Рис. 3

В режиме "TURBO" при нажатии соответствующих кнопок на входы A и B подаются от генератора на элементах DD2.1...DD2.4 импульсы, имитирующие многократное нажатие кнопок A или B.

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

"ВАШ КОМПЬЮТЕР"-

ЭТО ЖУРНАЛ ДЛЯ ВАС!

ISSN 0869-0510. Радиолюбитель, Ваш компьютер, сентябрь 95, Индекс 74996.

Цена договорная.