Вокруг ZX Spectrum

Ежемесячная газета (1) Декабрь 1991 года Цена 99 коп.

От редакции

Итак, в Ваших руках первый номер газеты "Вокруг ZX Spectrum". Как следует из названия, она предназначена для владельцев домашних компьютеров, совместимых с ZX Spectrum.

Поговорим немного о планах редакции.

Поскольку на "больших" персональных компьютерах наиболее популярными языками программирования являются Си и Паскаль, мы будем уделять особое внимание именно этим языкам. Учитывая существующие темпы компьютеризации нашего общества, многие из наших потенциальных читателей уже в самое ближайшее время получат доступ к компьютерам типа IBM PC. Знакомство с Си и Паскалем на ZX Spectrum позволит Вам чуствовать себя уверенно и на IBM PC.

ZX Spectrum - это домашний компьютер. Поэтому большое внимание будет также уделяться его возможным применениям в "домашнем" хозяйстве. В этом направлении особые надежды мы возлагаем на творчество наших читателей.

Из-за высокой стоимости бумаги мы вынуждены для обеспечения относительно низкой цены довольствоваться об'емом в 1 печатный лист (это то, что Вы держите в руках). На этом количестве бумаги мы будем стараться размещать как можно больше информации. Приносим свои извинения за возникающее вследствие этого ухудшение читаемости.

Читайте в этом номере

Система программирования на языке Си фирмы HiSoft (v1.1)

Сергей Кауль

Система программирования на языке Си фирмы HiSoft (в дальнейшем называемая Си-системой) состоит из двух компонент: текстового редактора и компилятора. Соответственно, имеется два основных режима: режим редактирования и режим компиляции. Рассмотрим особенности работы в этих режимах.

В режиме редактирования производится ввод и исправление текста программы. Управление осуществляется следующими клавишами.

"C" - переход в режим компиляции.

"B" - возврат в Бейсик-систему; повторный вызов Си-системы производится с помощью команды RANDOMIZE USR 25200.

"In1,n2" - переход в подрежим ввода текста. Строки текста автоматически нумеруются начиная с n1, номер увеличивается на n2 при переходе к следующей строке. Выход из этого подрежима осуществляется нажатием "CapsShift"+"1".

"Nn1,n2" - перенумерация строк текста. Первой строке присваивается номер n1, а n2 - приращение номеров в последовательных строках.

"Dn1,n2" - удаление строк с номерами от n1 до n2.

"Mn1,n2" - копирование строки n1 в строку n2.

"Ln1,n2" - вывод на экран текста программы начиная со строки n1 до строки n2. После заполнения части экрана вывод приостанавливается, позволяя просмотреть выведенную порцию текста. Вывод может быть прерван нажатием "CapsShift"+"1", нажатие любых других клавиш приведет к выводу следующей порции текста.

"Kn1" - изменение размера порции текста выводимой командой "L". n1 задает новое количество строк в порции.

"V" - вывод на экран текущих значений основных параметров (первый и второй параметры команд по умолчанию, адрес начала и конца текстового буфера).

"Pn1,n2,filename" - запись на магнитофон (или микродрайв, если он имеется) текста программы от строки n1 до строки n2. Запись производится в файл с именем filename.

"G,,filename" - чтение с магнитофона файла filename. Если в памяти уже находится некоторая программа, то содержимое файла будет размещаться после нее.

"Fn1,n2,text1,text2" - поиск текста text1 (образец поиска), начиная со строки n1 и кончая строкой n2, с возможной заменой text1 на text2. После того, как образец поиска найден, автоматически происходит переход в подрежим редактирования строки (см. ниже), где можно осуществить замену text1 на text2 ("S"), продолжить поиск ("F") или вручную отредактировать найденную строку.

"En1" - переход в подрежим редактирования строки с номером n1. В этом подрежиме допускаются следующие команды:

"CapsShift"+"5" - перемещение курсора на одну позицию влево;

"CapsShift"+"8" - перемещение курсора на одну позицию вправо;

"Enter" - завершение редактирования строки;

"Q" - завершение редактирования строки с отменой внесенных изменений;

"R" - стирание редактируемой строки;

"L" - повторный вывод на экран редактируемой строки;

"K" - удаление символа, находящегося правее курсора;

"Z" - удаление всех символов правее курсора;

"I" - переход в подрежим вставки символов, при этом на фоне мерцающего курсора отображается "*", выход из подрежима вставки происходит после нажатия "Enter";

"X" - перемещение курсора в конец редактируемой строки и переход в подрежим вставки символов;

"C" - переход в подрежим замены символов, при этом на фоне мерцающего курсора отображается "+", выход из подрежима замены происходит после нажатия "Enter";

"F" - поиск следующего образца поиска;

"S" - замена найденного образца поиска на заменяющий текст и поиск следующего образца.

В режиме компиляции осуществляется перевод (компиляция, транслирование) текста программы на языке Си в коды микропроцес-

сора. Переход из режима компиляции в режим редактирования производится нажатием "CapsShift"+"1". Компиляция осуществляется в два этапа. На первом этапе производится предварительная обработка текста; подсистема, которая осуществляет эту работу, называется препроцессором. Все команды препроцессора начинаются с символа "#". Сразу следует оговориться, что возможности описываемого препроцессора существенно хуже, чем фактически сложившийся стандарт, но что поделаешь, - 48 килобайт памяти и внешняя память на магнитной ленте не позволяют размахнуться слишком широко.

Итак, перейдем к командам препроцессора.

#define ИМЯ ЗНАЧЕНИЕ - заменяет каждое вхождение текста ИМЯ на текст ЗНАЧЕНИЕ. Например,

#define ZERO 0

#include "ИМЯфайла" - приводит к включению содержимого файла ИМЯфайла в текст программы. Поскольку текстовый буфер при этом не увеличивается то эту возможность обычно используют для разбиения больших программ, не помещающихся в текстовый буфер, на относительно небольшие фрагменты, хранящиеся в файлах. Кроме того, с помощью этой команды производится подключение библиотек стандартных подпрограмм. Например,

#include "screen" - приводит к считыванию с магнитофона файла screen и включению его в текст программы.

#include "A:screen" - то-же самое, но только чтение с микродрайва или дисковода (в зависимости от варианта Си-системы).

#translate "ИМЯфайла" - скомпилировать программу и записать ее в файл ИМЯфайла. В последующем загрузка и запуск этой программы осуществляются из Бейсик-системы с помощью команд LOAD "ИМЯфайла"CODE: RANDOMIZE USR 25200. При компиляции программы без команды #translate в памяти находятся сама Си-система, текст программы, таблицы имен, область глобальных переменных, стек, генерируемый компилятором код программы, и область свободной памяти. Все они размещены в памяти начиная с адреса 25200 и выше. Если компиляция осуществлена с командой #translate, то после завершения компиляции сгенерированный код программы перемещается в область Си-системы, уничтожая ее, а область глобальных переменных, стек и область свободной памяти расширяются, захватывая память, которая ранее была распределена под текст программы и таблицы имен. Этот механизм нужно понимать и не волноваться, если у вас не хватает памяти во время отладки. Уменьшайте размеры массивов и отлаживайте задачу, а перед тем, как компилировать ее с командой #translate установите нужные размеры и, скорее всего памяти окажется достаточно.

#include - заставляет препроцессор обработать текстовый буфер. Все остальные команды препроцессора обычно включают в текстовый буфер или файлы, подключаемые командой #include "ИМЯфайла".

ЗАМЕЧАНИЯ.

Команда #translate должна находиться в самом начале текстового буфера, или должна вводиться с клавиатуры перед командой #include.

Команда #include "ИМЯфайла" не может быть вложенной, т.е. она может встречаться только в текстовом буфере.

Переход ко второму этапу компиляции производится совместным нажатием клавиш "SymbolShift" и "I" (латинское "И"). Если второй этап завершился нормально, то на экране появится сообщение "type y for run" (нажмите "y" для запуска программы). Нажатие клавиши "Y" приведет к выполнению программы, нажатие любой другой клавиши вызовет возврат в режим компиляции.

Завершим первое знакомство с Си-системой примером простейшего сеанса работы. Сначала загрузим Си-систему с ленты. После загрузки на экране появится сообщение "Save to Microdrive 1 (y/n)?". Если к вашему компьютеру подключен микродрайв, отвечайте "Y" (при этом видимо произойдет запись на микродрайв), все остальные должны отвечать "N", после чего на экране появляется текст "HISOFT C Compiler ...", свидетельствующий о том, что мы оказались в режиме компиляции.

Перейдем в режим редактирования. Для этого нажмем "CapsShift"+"1" и "Enter". Теперь на экране возникнет надпись "edit:". Перейдем в подрежим ввода текста, набирая на клавиатуре "10,10" и "Enter". После появления номера строки 10 наберем текст main() и "Enter". На экране появляется следующий номер - 20. Теперь набираем {printf("hello"); rawin();} и "Enter". Программа введена в текстовый буфер, поэтому надо закончить ввод. Нажмем "CapsShift"+"1" и "Enter".

Скомпилируем программу. Для этого нажав "C" и "Enter" перейдем в режим компиляции. Запустим компилятор командой #include. Если никаких сообщений типа "ERROR ..." нет, а на экране только листинг программы, значит вы правильно ввели текст. В противном случае нужно вернуться в режим редактирования, внимательно просмотреть текст и исправить ошибки. Переходим ко второму этапу компиляции нажав "SymbolShift"+"I". После появления надписи "type y for run" нужно нажать "Y". Нажмем. Программа начала выполняться. Оператор printf("hello") выводит на экран текст hello. Оператор rawin() ожидает ввода с клавиатуры. Нажав любую клавишу мы завершим этот оператор, вслед за чем закончится выполнение и всей программы. На экране вновь появляется надпись "type y for run", позволяя повторно запустить программу. Мы не будем этого больше делать и нажмем "N".

Программа отлажена. Теперь оттранслируем ее и запишем сгенерированный код на ленту. Для этого наберем #translate "hel" и "Enter", затем #include и "Enter". Теперь нужно включить магнитофон на запись и нажать "SymbolShift"+"I". Записанную программу можно загрузить командой LOAD "hel"CODE или просто LOAD ""CODE, а затем запустить на выполнение командой RANDOMIZE USR 25200.

(продолжение следует)

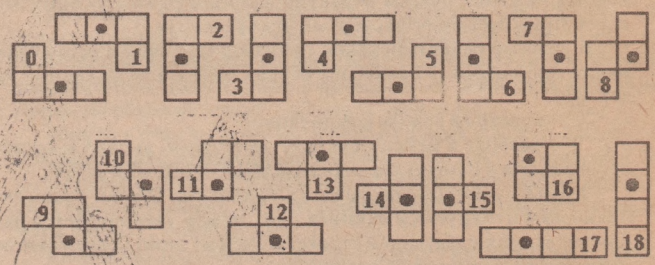
Программирование игр: реализация Тетриса на Си

Сергей Кауль

Суть игры "Тетрис" состоит в следующем. В плоский стакан сверху падают фигуры, построенные из четырех квадратов. Играющий может перемещать их по горизонтали и вращать в плоскости "картинки". В каждый момент времени падает ровно одна фигура. Как только дальнейшее падение становится невозможным, фигура останавливается, и сверху начинает падать следующая. Если некоторый горизонтальный ряд между стенками "стакана" оказывается заполненным, он "схлопывается", в результате чего находящиеся выше ряды опускаются вниз. Цель игры - собрать в "стакан" как можно больше фигур.

Рассмотрим структуры данных, необходимые для эффективной реализации игры. Начнем со "стакана". В качестве его модели самособой напрашивается двумерный массив. Обозначим его stakan[stakw,stakd], где stakw - щирина, а stakd - глубина "стакана". Мы считаем, что "стакан" состоит из stakw*stakd квадратов такогоже размера, как и квадраты, из которых состоят фигуры. Значение stakan[x,y]=0 означает, что квадрат с координатами x,y незаполнен. В противном случае stakan[x,y] интерпретируется как цвет фигуры, покрывающей квадрат с координатами x,y.

Мы будем использовать 19 фигур, показанных ниже.



Внутри каждой фигуры точкой помечен квадрат, который будет называться точкой привязки. Для того, чтобы задать фигуру достаточно описать местоположение каждого из трех оставшихся квадратов по отношению к точке привязки. Обозначим xx[19,3], yy[19,3] - массивы, содержащие координаты квадратов по отношению к точке привязки, т.е. xx[f,i] - это смещение по оси X квадрата i фигуры f относительно точки привязки этой фигуры (yy[f,i] определяется аналогично). Например, для фигуры 0 xx[0,0]=-1, yy[0,0]=-1, xx[0,1]=-1, yy[0,1]=0, xx[0,2]=1, yy[0,2]=0 (ось X расположена как обычно, слева направо, а ось Y - сверху вниз). Некоторые фигуры при поворотах трансформируются друг в друга. Например, поворачивая по часовой стрелке фигуру 0 мы получим последовательно фигуры 2, 1, 3, а затем снова фигуру 0. Это означает, что нам нет необходимости заниматься разработкой какого-то специального алгоритма поворота фигур. Поворот можно реализовать в виде замены одной фигуры на другую. Для хранения информации о порядке смены фигур при повороте будем использовать массив nextfig[19]. Значение nextfig[i] интерпретируется как номер фигуры, которой заменяется фигура i после ее поворота по часовой стрелке на 90 градусов.

Массивы xx, yy, nextfig должны быть проинициализированы в начале программы, поскольку в них содержатся исходные данные.

```
#define xscr 3  /* координаты левого верхнего */
#define yscr 0  /* угла "стакана"             */
#define stakw 10   /* ширина и                */
#define stakd 20   /* глубина "стакана"       */
#define wait 30
#define waitkey 60
#define rotate 'z'  /* клавиша "поворот"      */
#define left 'o'   /* клавиша "влево"         */
#define right 'p'  /* клавиша "вправо"        */
typedef char * charptr;
/*-----------------------------------------------*/
int nextfig[19]=
 {3,2,0,1,6,7,5,4,9,8,11,10,14,15,13,12,16,18,17};
int xx[19][3]= {
     {-1,-1,1},{-1,1,1},{0,0,1},{-1,0,0},{-1,-1,1},
     {-1,1,1},{0,0,1},{-1,0,0},{-1,-1,0},{-1,0,1},
    {-1,-1,0}, {-1,0,1},{-1,0,1},{-1,0,1},{0,-1,0},
     {0,1,0},{1,0,1}, {-1,1,2},{0,0,0}          };
int yy[19][3]= {
    {-1,0,0},{0,0,1},{1,-1,-1},{1,1,-1},{1,0,0},
     {0,0,-1},{-1,1,1},{-1,-1,1},{1,0,-1},{-1,-1,0},
    {-1,0,1},{0,-1,-1},{0,-1,0},{0,1,0},{-1,0,1},
    {-1,0,1},{0,1,1},{0,0,0},{-1,1,2}      };
int figgen[7]={2,7,9,11,14,16,18}; /* номера
      "начальных" фигур */
int score,maxscore;
int stakan[stakw][stakd];
int xfig,yfig,fig,colorfig;
int new;
/*-----------------------------------------------*/
main()
{ int k; maxscore=0;
 inline(0xCD,0xD6B); /* чистим экран */
 while(1)
  { score=0;
   setpos(13,18);
   printf("          ");
   game();       /* начинаем игру */
   if(score>maxscore)maxscore=score;
   setpos(13,3);
   printf(" max %6d",maxscore);
   setpos(13,18);
   printf("Play game?(Y/N)");
   while((k=rawin())!='y')if(k=='n')return;
  }
}
/*-----------------------------------------------*/
setpos(x,y) int x,y;  /* установка местоположения
                         выводимого текста */
{ printf("%c%c%c",22,y,x);
}
/*-----------------------------------------------*/
game()
{ int time,k;
 initgame();  /* инициализация */
 colorfig=7;
 while( setfig(5,1,newfig()) )  /* цикл-пока для новой
                                   фигуры есть место */
 {
  while(1)
   {time=wait;
   while(time--)if(k=key())
    {if(k==left)setfig(xfig-1,yfig,fig);
     if(k==right)setfig(xfig+1,yfig,fig);
     if(k==rotate)setfig(xfig,yfig,nextfig[fig]);
    }
   if(setfig(xfig,yfig+1,fig)==0)  /* если дошли до низа */
    { figinst();
     compress();
     break;
    }
   }
  }
}
/*-----------------------------------------------*/
newfig()  /* возвращает номер новой фигуры и
             вычисляет счет */
{ charptr p;
 new=1; score++;
 setpos(13,1);
 printf(" score %4d",score);
 p=cast(charptr)23672;  /* указатель на
псевдослучайное число */
 colorfig=((*p)+score)%7+1;
 return figgen[(*p)%7];
}
/*-----------------------------------------------*/
figinst()  /* запись в stakan */
{ int i;
 stakan[xfig][yfig]=colorfig;
 for(i=0;i<3;i++)stakan[xfig+xx[fig][i]][yfig+yy[fig][i]]=
                 colorfig;
}
/*-----------------------------------------------*/
compress() /* сжать заполненные ряды */
{ int x,y,z,str,ax;
 for(y=stakd-1;y;y--)
  { str=1;
   for(x=0;x<stakw;x++)if(stakan[x][y]==0)str=0;
   if(str)
    { for(x=0;x<stakw;x++)for(z=y;z;z--)
      { ax=x*stakd+z;
       stakan[0][ax]=stakan[0][ax-1];
       setscr(x,z,stakan[0][ax]);
      }
     y++;
    }
  }
}
/*-----------------------------------------------*/
initgame()  /* установка начальных параметров */
{ int i,j;
 score=0; xfig=5; yfig=1;
 for(i=0;i<stakw;i++)for(j=0;j<stakd;j++)
  {stakan[i][j]=0; setscr(i,j,0);
```

```
  }
}
/*-------------------------------------------------*/
setscr(x,y,c) int x,y,c;  /* закрасить знакоместо (x,y)
                             цветом c */
{ charptr p;
  p=cast(charptr) (0x5800+x+xscr+((y+yscr)<<5));
        *p=9*c;
}
/*-------------------------------------------------*/
setfig(x,y,f) int x,y,f;  /* передвинуть фигуру f в новое
                             место с координатами (x,y) */
{
 if(isfigset(x,y,f)==0)return 0;
 if(new==0)drawfig(0);
 new=0;
 xfig=x; yfig=y; fig=f;
 drawfig(colorfig);
 return 1;
}
/*-------------------------------------------------*/
isfigset(x,y,f) int x,y,f;  /* проверка возможности
установки f в (x,y) */
{ int i,a,b;
 for(i=0;i<3;i++)
  { if((a=x+xx[f][i])<0)return 0;
    if(a>=stakw)return 0;
    if((b=y+yy[f][i])<0)return 0;
    if(b>=stakd)return 0;
    if(stakan[a][b])return 0;
  }
 if(stakan[x][y])return 0;
 return 1;
}
/*-------------------------------------------------*/
drawfig(c) int c; /* нарисовать фигуру fig цветом c */
{ int i;
 setscr(xfig,yfig,c);
 for(i=0;i<3;i++)setscr(xfig+xx[fig][i],yfig+yy[fig][i],c);
}
/*-------------------------------------------------*/
```

Обработка прерываний

Сергей Кауль

Прерывание - это сигнал, с помощью которого процессор оповещается о внешнем событии. При его поступлении процессор приостанавливает выполнение текущей программы и передает управление специальной подпрограмме обработки прерывания, которую часто называют обработчиком прерывания. Прерывания подразделяются на маскируемые и немаскируемые. Первое может быть разрешено или запрещено (в этом случае процессор игнорирует сигнал прерывания) командами EI (Enable Interrupt) или DI (Disable Interrupt). Второе-же безусловно выполняется при поступлении сигнала прерывания.

В компьютере ZX Spectrum установлен микропроцессор Z80, имеющий две входные линии, на которые поступают сигналы прерываний: линия NMI используется для регистрации немаскируемых прерываний, а линия INT - для маскируемых. В компьютере с минимальной конфигурацией линия NMI не задействована (обычно она выводится на раз'ем для подключения внешних устройств), а линия INT подключена к блоку, вырабатывающему синхронизирующие сигналы для телевизора и активизируется с частотой кадровой развертки (50 раз в секунду). Стандартный обработчик этого прерывания, код которого находится в постоянном запоминающем устройстве (ПЗУ), выполняет две функции: во-первых увеличивает на 1 значение трехбайтовой переменной, находящейся по адресу 23672 (#5C78), которая может использоваться прикладными программами для измерения промежутков времени с дискретностью 1/50 сек., и во-вторых, опрашивает состояние клавиатуры, преобразует коды нажатых клавиш во внутреннее представление и помещает их в буфер клавиатуры из которого они будут считываться интерпретатором Бейсика или прикладными программами.

Поскольку в ZX Spectrum стандартный обработчик прерывания записан в ПЗУ, его код не может быть модифицирован программой. Единственный способ изменения процедуры обработки прерывания состоит в "перехвате" преывания другим обработчиком. Рассмотрим как это можно сделать.

Микропроцессор Z80 имеет три режима прерываний - 0,1 и 2. Выбор нужного режима осуществляется командами IM 0, IM 1 или IM 2 соответственно. В режиме 0 прерывание отрабатывается также, как и на микропроцессоре i8080 (K580BM80), т.е. после поступления прерывания с шины данных считывается и исполняется код команды, формируемый обычно специальной микросхемой контроллера прерываний. В ZX Spectrum микросхема контроллера прерываний отсутствует, а используется схемотехническая особенность, состоящая в том, что при отключении от шины данных всех устройств ее линии переводятся в состояние "1". Поэтому, когда при поступлении сигнала прерывания от шины данных отключаются все устройтва, это эквивалентно приходу от контроллера прерываний байта #FF, что соответствует команде RST #38. В режиме 1 при поступлении сигнала прерывания микропроцессор автоматически выполняет команду RST #38 ничего не считывая с шины данных. Т.е. режимы 0 и 1 на ZX Spectrum полностью идентичны. Наконец, режим 2. Это самый "хитрый" режим преываний в Z80. Специально для этого режима в микропроцессоре имеется регистр вектора прерываний I. Значение регистра I может быть прочитано или записано командами LD A,I или LD I,A. При поступлении сигнала прерывания процессор считывает с шины данных байт и формирует адрес, старший байт которого равен I, а младший равен считанному байту (напомним, что на ZX Spectrum он всегда равен #FF). Двухбайтовая величина, расположенная по этому адресу, интерпретируется далее процессором как адрес обработчика прерывания.

Таким образом "перехват" прерывания осуществляется следующим образом. По адресу #xxFF записывается адрес точки входа в новый обработчик прерывания. Затем в регистр I заносится значение xx и выполняется команда IM 2.

Проиллюстрируем эту технику на примере драйвера джойстика.

```
ORG   #FEFF
VECTOR DEFW  ENTRY
TABL
    DEFW  #2713  ; значение DE для кнопки "вправо"
    DEFW  #2704  ; значение DE для кнопки "влево"
    DEFW  #2703  ; значение DE для кнопки "вниз"
    DEFW  #270B  ; значение DE для кнопки "вверх"
    DEFW  #2723  ; значение DE для кнопки "огонь"
ENTRY
    PUSH  AF
    PUSH  HL    ; увеличиваем на 1 системный
    LD    HL,(#5C78)    ; счетчик времени
    INC   HL
    LD    (#5C78),HL
    LD    A,H
    OR    L
    JR    NZ,L0048
    INC   (IY+64)
L0048 PUSH  BC
    PUSH  DE
    CALL  GETKEY
    POP   DE
    POP   BC
    POP   HL
```

```
    POP   AF
    EI
    RET
GETKEY
    IN    A,(#DF)  ; опрос джойстика
    LD    HL,#FFFF
NOBIT  RRCA
    INC   HL     ; поиск справа-налево единичного
    JR    NC,NOBIT ; бита
    LD    A,L
    CP    5      ; если не нажата ни одна из кнопок
    JP    Z,#02BF  ; джойстика, то опрос клавиатуры
    ADD   HL,HL
    LD    DE,TABL  ; в противном случае опрос
                   : клавиатуры
    ADD   HL,DE    ; имитируется занесением в DE
    LD    E,(HL)   ; значения из таблицы TABL
    INC   HL
    LD    D,(HL)
    JP    #02C3
INIT
    LD    A,#FE
    LD    I,A
    IM    2
    RET
```

Процедура ENTRY идентична соответствующей процедуре обработчика прерываний, "прошитого" в ПЗУ, - значение 3-байтового системного счётчика времени увеличивается на 1 и производится вызов процедуры GETKEY, осуществляющей опрос джойстика и клавиатуры. Различия появляются в процедуре GETKEY. В ПЗУ находится следующий код:

```
02BF:  CALL   #028E
02C2:  RET    NZ
02C3:  LD     HL,#5C00
  ...
```

Процедура #028E осуществляет опрос клавиатуры и возвращает в регистровой паре DE коды нажатых клавиш. Если ни одна клавиша не нажата, то DE=#FFFF, если нажата ровно одна клавиша, то D=#FF, а в E находится код клавиши. Если нажато две клавиши, причем одна из них CapsShift или SymbolShift, то код CapsShift или SymbolShift находится в D, а в E находится код второй клавиши. Во всех этих случаях флаг Z установлен, свидетельствуя о разрешенной комбинации нажатых клавиш, для остальных комбинаций он сбрасывается. На рисунке схематично показана клавиатура и коды клавиш, возвращаемые в регистровой паре DE.

36	28	20	12	4	3	11	19	27	35
37	29	21	13	5	2	10	18	26	34
38	30	22	14	6	1	9	17	25	33
39	31	23	15	7	0	8	16	24	32

В процедуре GETKEY опрашивается состояние порта джойстика. Положения кнопок индицируют биты 0-4: бит 0 - кнопка "вправо" (1 - нажата, 0 - ненажата); 1 - "влево", 2 - "вниз", 3 - "вверх", 4 - "огонь". Биты 5-7 всегда равны "1". Если нажатие кнопки джойстика не зафиксировано, то управление передается по адресу #02BF (стандартная обработка прерывания). В противном случае в регистровую пару DE помещается нужное значение из таблицы TABL (имитация вызова процедуры опроса клавиатуры #028E) и управление также передается стандартному обработчику, но уже минуя опрос клавиатуры.

Активизация драйвера джойстика производится вызовом процедуры INIT.

Ниже приводится текст драйвера джойстика на Бейсике.

```
 1 DATA 65279,3
 2 DATA "0BFF1327042703270B272327F5E52A785C
2322785C7CB52003FD3440C5D5CD2658"
 3 DATA "FFD1C1E1F1FBC9DBDF21FFFF0F2330F
C7DFE05CABF02291101FF195E2356C3C319"
 4 DATA "023EFEED47ED5EBD"
100 CLEAR 65278: GO SUB 9993
110 RANDOMIZE USR 65344: STOP
9993 READ a,S: FOR f=1 TO S
9994 READ l$: LET l=LEN l$: LE f s=0: LET k=2
9995 LET a$=l$(k-1): LET b$=l$(k)
9996 LET c=(CODE a$-48-(7*(a$>"@")))*16+CODE
 b$-48-(7*(b$>"@"))
9997 IF k<l THEN POKE a,c: LET s=s+c: LET k=k+2:
 LET a=a+1: GO TO 9995
9998 IF s-256*INT (s/256)<>c THEN PRINT
 "error in string";f+1: STOP
```

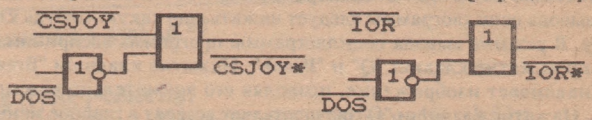
(продолжение следует)

Советы *Дмитрия Козлова*

...Если подключив контроллер дисковода к компьютеру Вы получаете сообщение "No disk" при вставленной в исправный дисковод дискете, не отчаивайтесь. Причиной этого может быть конфликт между имеющимся в Вашем компьютере интерфейсом джойстика и контроллером дисковода.

Есть несколько путей решения этой проблемы. Самый простой из них - на время работы с дисководом отключать "Chip select" джойстика. Но это не оптимальное решение проблемы, поскольку Вам достаточно быстро надоест щелкать кнопкой и, кроме того, такой способ ограничивает круг используемых программ. Более правильным является использование выходных сигналов контроллера, позволяющих автоматически отключать конфликтующую периферию. В зависимости от используемого контроллера для этой цели можно использовать либо сигнал IORQOUT, либо DOS.

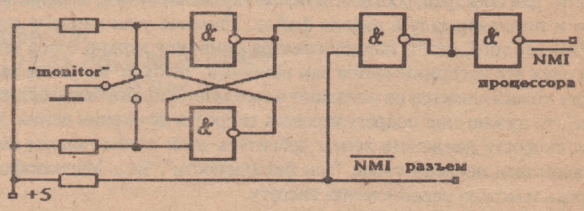
IORQOUT - это сигнал, аналогичный IORQ процессора, но



предназначенный для всей периферии, кроме контроллера дисковода. Если Ваш контроллер имеет такой сигнал, то нужно разорвать цепь IORQ между процессором и периферийными устройствами, и подать на них сигнал IORQOUT с контроллера.

Если-же Ваш контроллер имеет выходной сигнал DOS, то логический "0" на этом выходе соответствует состоянию, когда активны периферийные устройства контроллера дисковода. В этот момент конфликтующие устройства компьютера должны быть отключены. Возможные варианты модификации:

...Если Вы причисляете себя к числу хакеров, иначе говоря, "взломщиков" программ, то Вам пригодится следующее приспособление



Как известно, версии ПЗУ в ZX Spectrum начиная с 1986 года имеют встроенный монитор памяти. Для обращения к нему обычно используется команда Бейсика "*". Аналогичный результат дает команда RANDOMIZE USR 102 или CALL #0066. При использовании данного устройства появляется возможность нажав кнопку "monitor" в любой момент прервать выполнение программы и перейти в режим монитора памяти. В этом режиме можно просмотреть (модифицировать) содержимое памяти, а в случае необходимости также и сохранить дамп памяти на ленте. Для этого требуется в регистровую пару AF занести значение #FF00, в DE - количество записываемых байтов, в IX - стартовый адрес, в PC - #04C2, и нажать SymbolShift и "0".

Копирование программ

Сергей Борисов

Программа настройки магнитофона diag

Копирование программ - один из наиболее широко распространенных видов деятельности владельцев ZX Spectrum. Если осуществляется переписывание с кассеты, записанной на другом магнитофоне и чтение идет с ошибками, не отчаивайтесь, - наиболее вероятная причина этого - несовпадение угла наклона головки или скорости перемещения ленты. Для того, чтобы прочитать такую кассету нужно перенастроить магнитофон.

Вообще говоря, копирование программ лучше всего осуществлять с помощью двух магнитофонов. Магнитофон, с которого производится чтение, нужно немного модифицировать. Прежде всего следует просверлить отверстие на передней панели (если его нет) таким образом, чтобы отверткой небольшого диаметра можно было регулировать угол наклона головки. Кроме того, желательно иметь возможность изменять в небольших пределах скорость перемещения ленты. В магнитофонах с батарейным питанием это делается довольно-таки просто. На плате управления двигателя нужно найти переменный резистор, которым производится установка частоты вращения двигателя, и перенести его в более доступное место. На дорогих стационарных магнитофонах это лучше не делать. Второй магнитофон - эталонный - используется только для записи и чтения "своих" кассет.

Итак, приступаем к настройке магнитофона. Сначала нужно запустить программу diag. После ее запуска экран заполнится текстом - это главное меню программы. Мерцанием выделены буквы, показывающие возможные варианты выбора режима диагностирования. Для выбора нужного режима следует нажать клавишу с соответствующим буквенным обозначением. Рассмотрим возможные режимы.

Help - описание работы программы diag. Занимает несколько экранов, нажатие любой клавиши приводит к выводу следующего экрана.

Oscillogram - осциллограмма поступающего с магнитофона сигнала. После нажатия "O" на экране появится поясняющий текст; для вывода осциллограммы следует нажать еще какую-нибудь клавишу. В режиме вывода осциллограммы программа воспринимает только нажатие клавиш "Q" и "Break". Нажатие клавиши "Break" останавливает изображение, позволяя его внимательно рассмотреть. Нажатие клавиши "Q" осуществляет возврат в главное меню.

Time - частотный спектр поступающего с магнитофона сигнала. Управление осуществляется также, как и в режиме осциллограммы.

Record - на магнитофон записывается файл. Эквивалентно выполнению команды SAVE в Бейсике.

Verify - проверка файла, записанного предыдущей командой. Эквивалентно выполнению команды VERIFY в Бейсике.

Exit - завершение работы программы diag.

С точки зрения настройки магнитофона на "чужую" кассету наиболее информативным является спектр сигнала. Запустив программу diag переведите ее в режим вывода спектра и включите магнитофон с "чужой" кассетой. В нижней части экрана будет нарисована полоса красного цвета, на которой другими цветами выделены "зоны безопасности" для спектров, соответствующих нулевым битам, единичным битам и пилот-сигналу в начале файла. Изменяя угол наклона головки добейтесь того, чтобы полосы спектра стали как можно более узкими. На слух это воспринимается как наиболее "звонкое" звучание записи. Если полосы спектра не попадают в соответствующие "зоны безопасности", то нужно еще подрегулировать скорость движения ленты. Изменяя скорость движения ленты добейтесь того, чтобы полосы спектра находились посреди своих "зон безопасности". Все. Магнитофон настроен и можно переписывать кассету.

Несколько замечаний.

Не пугайтесь, если при отсутствии сигнала у вас будет "грязь" на спектрограмме. Это происходит из-за того, что на многих компьютерах узел чтения с магнитофона при отсутствии сигнала начинает самовозбуждаться.

Иногда между спектрами "0" и "1" может появиться еще одна полоса. Это явление обусловлено тем, что в программе diag при анализе спектра промежутки времени измеряются не между последовательными перепадами входного сигнала, а между положительными, т.е. через один. Поэтому если входной сигнал оказывается инвертированным, то измеряется не время передачи бита, а время передачи половины предыдущего и половины последующего битов. Следовательно когда за нулевым битом следует единичный или наоборот, будет зафиксирован промежуток времени равный среднему арифметическому от времени передачи нулевого и времени передачи единичного битов, что и приводит к появлению лишней средней полосы в спектрограмме.

Копировщик TURBO

Существует большое количество различных копировщиков. Больше всего мне нравится TURBO. Поясню почему. Во-первых, он осуществляет сжатие данных, поэтому у меня не было ни разу проблем с "большими" файлами. Во-вторых, он работает как с файлами, записанными с нормальной плотностью, так и с файлами, записанными с двойной плотностью. Работать с TURBO очень просто. После запуска программы на экране появляется картинка, в верхней части которой написано 44288 (размер свободной памяти), а немного ниже середины экрана строка меню "Load Save Delete Turbo Reset", в которой перечислены команды копировщика. Выбор нужной команды осуществляется нажатием клавиши, на которой обозначена первая буква наименования команды.

Load - чтение с магнитофона. Каждый прочитанный блок нумеруется и информация о нем отображается на экране. Обычно (но не всегда) файл записывается в виде двух блоков - заголовка и данных. Имя файла, его тип, длина, начальный адрес записываются в заголовке. Во многих игровых программах заголовки файлов не записаны, поскольку программа сама "знает" тип и где его нужно размещать. Поэтому не удивляйтесь если при копировании вам встретятся файлы без параметров. Чтение может завершиться неправильно по двум причинам: 1) из-за ошибки чтения, в этом случае в строке, где расположено меню, появится сообщение "Accept PARITY AT ...."; 2) из-за нехватки места в памяти, тогда сообщение будет таким " MEMORY AT ....". Чтение в любой момент может быть прервано нажатием клавиши "BREAK".

Save - запись на магнитофон. После нажатия клавиши "S" появится сообщение "Save from" и копировщик ожидает ввода двузначного номера начального блока для записи (однозначные номера вводятся с первым нулем, например, 1 - 01). Затем после его ввода появится текст "to" и копировщик ожидает ввода двузначного номера последнего записываемого блока. Запись в любой момент может быть прервана нажатием клавиши "BREAK".

Delete - удаление группы блоков для освобождения места в памяти. Номера первого и последнего блоков запрашиваются также, как и в команде Save.

Turbo - переключение плотности записи для операций чтения/записи. После нажатия клавиши "T" выводится текст "Speed L-H-T" и копировщик ожидает нажатия одной из клавиш "L","H" или "T". "L" соответствует одинарной плотности, "T" - двойной, а "H" - случаю, когда заголовки записываются с одинарной плотностью, а данные с двойной (кстати, записей в таком формате я еще ни разу не встречал). Установленная плотность отображается в верхней половине экрана правее размера свободной памяти.

Reset - завершение работы.

Программа "Мультипликатор"

Максим Злотников

Программа предназначена для создания, просмотра и записи на магнитофон мультфильмов.

Управляющие клавиши:

Q - движение курсора вверх.

A - движение курсора вниз.

O - движение курсора влево.

P - движение курсора вправо.

Курсор - это мигающая точка, в начале работы находится в левом нижнем углу экрана.

CapsShift+U - установка курсора по координатам;

CapsShift+K - запись кадра в память. По этой команде запоминаются все изменения произошедшие на экране со времени предыдущего выполнения этой команды.

CapsShift+P - режим перемещения курсора без изменения экрана;

CapsShift+R - режим рисования;

T - режим ускоренного перемещения курсора без изменения экрана;

CapsShift+T - выход из режима ускоренного перемещения;

CapsShift+Z - запись на магнитофон составленного мультфильма или его части (запись предлагается также при переполнении буфера).

CapsShift+I - изменение цвета (цвет вводится номером 0 - 7).

CapsShift+L - просмотр мультфильма.
CapsShift+C - очистка экрана.

Пример составления простого мультфильма.

В мультфильме нужно изобразить человека взмахивающего рукой. Сначала рисуем человека в исходном состоянии и записываем этот кадр в память (CapsShift+K). Далее стираем у него руку и рисуем ее в новом положении. Записываем в память этот кадр. И таким же образом еще несколько кадров.

При просмотре (CapsShift+L) вы увидите человека у которого изменяется положение руки.

Примечание.

В программе не предусмотрены проверки при вводе цвета и координат, поэтому их неправильный ввод может вызвать ошибку.

```
 1 DATA 65000,4
 2 DATA "64010000798479842178691100400100 1B1
ABE280F77E52AEEFD73237223772313"
 3 DATA "22EEFDE123130B78B120E6C921004011
786901001BEDB0C90179842AEEFDEB211B"
 4 DATA "00007AB820047BB928060303032318
F2ED4BE8FD1807ED4BE8FD2AEAFDEB2179DD"
 5 DATA "84C5C1C50B78B120FB7AB3280
A4E2346237E23021B18EBC1C900A2"
 6 PRINT "wait": GO SUB 9990
 19 CLEAR 26999: LET rf=1: LET w=20: LET c=0: CLS :
LET n=199: LET x=0: LET y=0: LET i=0
 20 RANDOMIZE USR 65044
 60 LET a$=INKEY$
 65 IF a$="t" THEN LET rf=10: LET zxc=n: LET n=990
 66 IF a$="T" THEN LET rf=1: LET n=zxc
 70 IF a$="U" THEN GO SUB 350: GO TO 60
 75 IF a$="K" THEN GO SUB 4000: GO TO 60
 77 IF a$="L" THEN RANDOMIZE USR 65056:
GO TO 60
 80 IF a$="C" THEN CLS : GO TO 60
 85 IF a$="Z" THEN GO SUB 5000: GO TO 60
 90 IF a$="R" THEN LET n=199: GO TO 60
100 IF a$="S" THEN LET n=230: GO TO 60
110 IF a$="I" THEN GO SUB 360: GO TO 60
115 IF a$="P" THEN LET n=990: GO TO 60
120 IF a$="H" THEN FOR x=1 TO 13: PRINT #0;
AT 0,0;q$(x): PAUSE 0: NEXT x
130 IF a$="q" THEN LET y=y+rf
140 IF a$="o" THEN LET x=x-rf
150 IF a$="a" THEN LET y=y-rf
160 IF a$="p" THEN LET x=x+rf
180 LET asw=22528+(x/8)+32+((192-y)/8):
 LET attr=PEEK asw
181 IF y<0 THEN LET y=174
182 IF y>175 THEN LET y=0
183 IF x>255 THEN LET x=0
184 IF x<0 THEN LET x=255
190 LET asw=22528+INT (x/8)+32*INT ((175-y)/8):
LET attr=PEEK asw
196 GO TO n
197 GO SUB 990
199 PLOT INK i;x,y
205 PRINT #0;AT 0,0;"  ","  "
210 PRINT #0;AT 0,0;x,y
225 GO TO 60
230 GO SUB 990
232 PLOT INVERSE 1;x,y
245 PRINT #0;AT 0,0;"  ","  "
250 PRINT #0;AT 0,0;x,y
255 GO TO 60
350 PRINT #0;AT 0,0;"WWEDITE KOORDINATY ";:
PAUSE 0: INPUT x,y: RETURN
360 PRINT #0;AT 0,0;"WWEDITE CWET ";: PAUSE 0:
INPUT i: RETURN
990 LET fg=POINT (x,y): LET w=20: LET c=0
992 LET a$=INKEY$
993 IF w>0 THEN LET w=w-1: GO TO 1000
995 LET w=20: LET c=1-c
1000  PLOT INVERSE c;x,y
1005 IF a$<>"" THEN GO TO 1020
1010 GO TO 992
1030 PLOT INVERSE 1-fg;x,y: IF N=990 THEN
 GO TO 60
1040 RETURN
4000 IF PEEK 65006+256*PEEK 65007>63000 THEN
 GO SUB 5000
4010 RANDOMIZE USR 65008
4030 RETURN
5000 LET adr=PEEK 65004+256*PEEK 65005:
 LET adr1=PEEK 65006+256*PEEK 65007:
 SAVE "cod"CODE adr,adr1-adr
5020 POKE 65006,PEEK 65004: POKE 65007,PEEK
 65005: RETURN
9990 RESTORE 1
9993 READ a,S: FOR f=1 TO S
9994 READ l$: LET l=LEN l$: LET s=0: LET k=2
9995 LET a$=l$(k-1): LET b$=l$(k)
9996 LET c=(CODE a$-48-(7*(a$>"@")))*16+CODE
 b$-48-(7*(b$>"@"))
9997 IF k<l THEN POKE a,c: LET s=s+c: LET k=k+2:
 LET a=a+1: GO TO 9995
9998 IF s-256*INT (s/256)<>c THEN PRINT
 "error in string";f+1: STOP
9999 NEXT f: RETURN
```

Текст программы просмотра мультфильма, записанного на магнитофон

```
 1 DATA 60000,6
 2 DATA "C30BEB1103003EFFB737DD2131EB08F33E
0FD3FE213F05E5DBFE1FE620CD6F3CEB"
 3 DATA "BFC0CDE70530FA21690010FE2B7CB520F9
CDE30530EB069CCDE30530E43EC6B866"
 4 DATA "30E02420F106C9CDE70530D578FED430F4C
DE705D079EE034F260006B0181F089D"
 5 DATA "2007300FDD7500180FCB11ADC0791F4F
131807DD7E00ADC0DD231B0806B22E0113"
 6 DATA "CDE305D03ECBB8CB1506B0D2E0EA7CAD6
77AB320CAE53A33EB2A31EB77DD213148"
 7 DATA "EB110300E108DD750018CD111200DD2117
EBAF37C356050046"
 20 GO SUB 9993
 40 RANDOMIZE USR 60000
 50 RANDOMIZE USR 60003
 60 GO TO 40
9993 READ a,S: FOR f=1 TO S
9994 READ l$: LET l=LEN l$: LET s=0: LET k=2
9995 LET a$=l$(k-1): LET b$=l$(k)
```

```
9996 LET c=(CODE a$-48-(7*(a$>"@")))*16+CODE
 b$-48-(7*(b$>"@"))
9997 IF k<l THEN POKE a,c: LET s=s+c: LET k=k+2:
 LET a=a+1: GO TO 9995
9998 IF s-256*INT (s/256)<>c THEN PRINT
 "error in string";f+1: STOP
9999 NEXT f: RETURN
```

Программа для обучения сложению

Геннадий Шахов

```
 20 PRINT "Level: ";
 30 INPUT a: PRINT a: RANDOMIZE 0
 40 LET b=1+INT (a*RND )
 50 LET c=1+INT (a*RND )
 60 PRINT b;"+";c;"=";
 70 INPUT d
 80 IF d=c+b THEN GO TO 1000
 90 BEEP 1,0: BEEP 1,2: BEEP .5,3: BEEP .5,2
300 BEEP 1,0: BEEP 1,2: BEEP .5,3
330 BEEP .5,2: BEEP 1,0: BEEP 1,3: BEEP 1,5
340 BEEP 2,7: BEEP 1,3: BEEP 1,5: BEEP 2,7
430 GO TO 70
1000 BEEP .5,16: BEEP .5,12: BEEP .5,16: BEEP .5,12
3000 BEEP .5,17: BEEP .5,16: BEEP 1,14
3030 BEEP .5,7: BEEP .5,7: BEEP .5,9: BEEP .5,11
3040 BEEP .5,12: BEEP .5,12: BEEP 1,12
3050 PRINT d
3060 GO TO 40
```

Описания игр

THE WAY OF THE EXPLOADING FIST

Если вы хотите освоить восточные единоборства, попробуйте поиграть в эту игру. Играть могут двое или только один человек. В последнем случае в качестве второго играющего выступает компьютер.

Настройка программы:

0 - выбор джойстиков или клавиатуры;

1 - один играющий (игра с компьютером);

2 - двое играющих.

Лучше всего начинать играть, настроив программу на двух игроков. В этом случае второй игрок бездействует, и вы можете освоить клавиатуру или джойстик. Имейте ввиду, что некоторые приемы осуществляются при одновременном нажатии нескольких клавиш, поэтому перепробуйте различные сочетания и посмотрите к каким действиям это приводит. Освоив несколько приемов попробуйте сыграть с компьютером. Сначала вам легко удастся его победить, но после каждого проигрыша класс его игры будет возрастать, и в конце концов вы проиграете. Не унывайте, а продолжайте тренироваться, и вас ждут лавры Брюса Ли.

GUNFRIGHT

Вы - шериф, патрулирующий город.

В начале игры сверху падают мешки с деньгами, и ваша задача - набрать их побольше, потому, что в процессе игры вам придется платить буквально за все: за поездку на лошади, за патроны, штрафы за столкновения с жителями города.

После того, как вы запаслись деньгами, вы попадаете в город. В левой части экрана находится об'явление - "Разыскивается опасный преступник ...", внизу изображен барабан вашего револьвера и ценник, в котором показаны текущие цены на патроны, стоимость поездки на лошади и размер штрафа. Теперь вы можете приступать к розыску преступника.

Для того, чтобы изменить способ изображения города, нажмите клавишу Z или CapsShift.

Разыскивая преступника нужно помнить следующее:

- подпрыгивающие пацаны показывают где скрывается преступник;

- попадающиеся время от времени мешки нужно подбирать - там находятся деньги, которые вам совсем не повредят;

- сев на лошадь вы сможете передвигаться значительно быстрее, но это достаточно дорогое удовольствие;

- суетящиеся женщины опасны: в случае столкновения вы не только платите штраф, но и лишаетесь одной жизни. Вообще старайтесь избегать столкновений с любыми движущимися предметами (и кактусами!).

После того, как преступник обнаружен (а выглядит он также, как и шериф), прицельтесь в него и стреляйте. Если вы попали в него, не радуйтесь преждевременно, - вы его только поцарапали своей пулей, и теперь вам предстоит дуэль с ним. Бандит появляется на экране в полный рост и примерно за одну секунду вы должны навести на него прицел и выстрелить. Если вы успели это сделать, то получив денежный приз, можете продолжить поиск следующего (более опасного и, соответственно, более "дорогого") преступника. В противном случае (если у вас еще остались "жизни"), вам придется производить поиск этого-же преступника заново.

WEST BANK

Эта игра развивает ловкость и быструю реакцию.

Вы охранник "Западного Банка", в котором имеется 12 дверей. Эти двери время от времени открываются и пропускают посетителей банка. Некоторые посетители приносят в банк деньги, а некоторые наоборот приходят с желанием ограбить ваш банк. Вы можете стрелять в посетителей, но, естественно, убивать нужно только грабителей. После нескольких промахов вас просто выгонят.

Грабителя можно распознать по наличию оружия в руке. Как только вы узидите грабителя - стреляйте в него. Торопитесь, ибо грабитель может опередить вас. Не спутайте грабителя с обычным вооруженным посетителем, а внимательно следите за ним, и как только он обнажит "ствол" - обезвреживайте его.

Иногда в банк забегает мальчишка с кучей шляп на голове. Под шляпами у него находится либо яблоко, либо мешок с деньгами. Если вы собьете с него все шляпы и там окажется мешок с деньгами, то он оставит его в банке. Во всех остальных случаях он просто помашет в дверях, а потом уйдет.

Собственно цель игры - пропустить через все 12 дверей посетителей, принесших деньги. Как только через какую-то дверь прошел посетитель с деньгами, в верхнем табло, состоящем из 12 окошек (по одному для каждой двери), в окошке соответствующем двери появляется знак доллара. В каждый момент времени разрешен вход только через 3 двери. Для того, чтобы разрешить вход через другие двери, дождитесь, когда все 3 двери окажутся закрытыми и, нажав клавиши "влево" или "вправо", переместитесь на другую тройку дверей.

После того, как через все двери принесли деньги, на банк производят налет 3 бандита. Стрелять в бандита можно только после того, как он вытащит оружие. На этом этапе оценивается скорость вашей реакции. Если всех троих удается уложить с максимальной скоростью, то появляется надпись "EXTRA", и количество допустимых ошибок, которые вам разрешается сделать, увеличивается на 1.

Если вы успешно пройдете все уровни сложности, то можете смело отправляться на любое предприятие, и наниматься туда охранником.

Главный редактор С.Б.Кауль
Адрес редакции: 630105,
Новосибирск-105, а/я 258

Подписано в печать
Заказ 161
Объем 1 п.л.
Тираж 15000 экз.

Отпечатано в типографии МГП
"Репринт". Адрес типографии:
630098 г.Новосибирск-98,
ул.Приморская, д. 22.