Scan, OCR & spellcheck by NUK, km,
Василий Кормилицын, Lexys (Cybrex)

МКП "ИНФОРКОМ" 121019, Москва, Г-19, а/я 16

Уважаемые читатели!

Мы хотели бы обратиться к Вам со следующей информацией. Просим ею руководствоваться в дальнейшей деятельности.

1. Мы определились в наших планах на 1993 год и продолжим выпуск "ZX-РЕВЮ" в том же виде, т.е. это останется все такое же внутреннее "фирменное" издание для своих клиентов. Правда, по понятным экономическим причинам тираж его будет еще более сокращен.

2. Мы не сможем принимать от Вас подписку по почте так, как это делалось в прошлые годы и организуем в Москве корреспондентский пункт. Подписаться смогут только те, кто лично придут по известному им адресу или пришлют своих друзей, находящихся в Москве проездом. Подробности о том, как связаться с корпунктом - в прилагаемом к данному выпуску информационном листке.

3. В порядке исключения для жителей особо удаленных районов, не имеющим никакой возможности прибыть на корпункт мы рассмотрим проведение подписки по почте, хотя гарантируем это не всем. Будет отдано безусловное предпочтение нашим постоянным клиентам и авторам.

4. На корпункте кроме подписки на 1993 год можно будет приобрести выпуски 1992 года, полный комплект 1991 года, выполненный в виде отдельной книги улучшенного качества, и прочие сопутствующие материала.

5. Стоимость подписки и прочих материалов, при приобретении их через корпункт будет ниже, чем при аналогичном обслуживании по почте.

6. Мелкооптовые операции будут обеспечены дополнительными льготами.

7. Мы заключили соглашение о сотрудничестве с редакцией журнала "МОНИТОР", согласно которому часть материалов этого многотиражного журнала, посвященная особо сложным и интересным игровым программам для IBM-совместимых машин будет готовиться нами. Начиная с сентября месяца этого года (N5) и далее ежемесячно наши материалы будут печататься в этом журнале. Рекомендуем это издание Вашему вниманию как для подписки, так и для размещения рекламы.

Журнал "Монитор":
Тираж 30 000 экз.
Периодичность в 1992 году - 9 номеров в год.
Издатель: НТО "Софт-Москва".
Основная форма распространения розничная продажа, но подписка по почте возможна.
Адрес для писем: 117419, Москва, а/я 765. Тел. 247-36-25; 237-21-36.

* * *

Сообщаем вам график выхода очередник выпусков ZX-РЕВЮ этого года (по началу рассылки): N7-8 - 20 сентября N9-10 - 30 октября N11-12 - 15 декабря

До свидания, до встречи в следующем выпуске.

Ваш ИНФОРКОМ.

СПЕКТРУМ В ШКОЛЕ

В прошлом выпуске "ZX-РЕВЮ" мы напечатали тестирующую программу, которая может быть полезной на уроке истории. Сегодня мы приводим рекомендации по тому, как можно использовать компьютер на уроке географии.

Перед учащимся появляется на несколько секунд карта какой-либо страны (области, края) с указанным расположением городов. Затем города исчезают и остается только контур страны. Учащийся должен курсором указать где находится тот или иной город.

Вы можете сами задать ту или иную карту (в нашем примере рассмотрена Австралия). Это может быть, например Красноярский край.

Не надо думать, что эта программа применима только для определения месторасположения городов. С тем же самый успехом можно проверять знание учащимся основных районов добычи тех или иных полезных ископаемых, знание расположения горных хребтов, названий рек, расположения гидрокомплексов и т.п. Вы сами легко разберетесь, как Вам развить и применить эту программу.

Преподаватели биологии смогут применить аналогичный подход для проверки правильности знания учащимся названий различных частей растений или скелета животного. Возможностей много. Мы надеемся, что Вы сумеете ими воспользоваться.

```
10 REM Урок географии
20 LET another = 150:
       LET nextcity = 300:
       LET tryagain = 380:
       LET move = 400:
       LET wait = 420:
       LET finish = 580:
       LET outline = 1000:
       LET printcity = 2500:
       LET test = 2700:
       LET data = 3000
30 REM Инструкции обучаемому
40 BORDER 1: PAPER 7: INK 9: BRIGHT 1: CLS
50 PRINT PAPER 1; FLASH 1; AT 9,9; "УРОК ГЕОГРАФИИ"
60 PAUSE 150 70 CLS
80 PRINT AT 2,3; "Эта программа проверит Ваши знания по географии Австралии. На десять
       секунд Вам будет показано расположение австралийских городов. Потом они исчезнут и
       останется только контур. Ваша задача установить указатель в том месте, где должен
       находиться заданный Вам город. Вы имеете по три попытки на отыскание каждого города."
90 PRINT AT 19,5; FLASH 1; "Нажмите любую клавишу"
100 PAUSE 0 110 CLS
120 PRINT AT 2,2;
       "Клавиша 8 - указатель вправо
       Клавиша 5 - указатель влево
       Клавиша 7 - указатель вверх
       Клавиша 6 - указатель вниз
       Нажмите 0, когда указатель
       займет правильное положение
       После каждой попытки курсор
       будет возвращен в левый
       нижний угол."
150 PRINT FLASH 1;AT 19,5; "Нажмите любую клавишу"
140 PAUSE 0
150 REM another
160 DIM r(4)
170 LET error = 0
180 GO SUB outline
190 RESTORE data
200 FOR n=0 TO 6
210 READ a, b, Y, X, a$
220 GO SUB printcity
```

```
230 NEXT n
240 RESTORE data
280 PAUSE 500
290 GO SUB outline
300 REM nextcity
310 LET xcity = 0: LET ycity  = 0: LET tries = 1
320 LET s$ = "          ": REM десять пробелов
330 PRINT AT 1, 1; s$
340 PRINT AT 18, 1; "Найдем-"; AT 19, 1; s$
350 READ a, b, Y, X, a$
360 IF a$ = "eof" THEN GO TO finish
370 PRINT AT 19, 1; a$
380 REM tryagain
390 PRINT AT 1, 1; "attempt "; tries
400 REM move
410 PLOT OVER 1; xcity, ycity
420 REM wait
430 LET fall = 0: LET result = 0
440 IF INKEY$ = "0" THEN GO SUB test
450 IF result = 2 THEN LET r(tries) = r(tries)+1: PRINT FLASH 1;AT 20,1; "CORRECT": PAUSE
      25: PAUSE 150: PRINT AT 20,1; s$: GO SUB printcity
460 IF tries = 4 THEN LET r(4)=r(4)+1: LET error = 1: GO SUB printcity
470 IF result = 1 OR tries = 4 THEN GO TO nextcity
480 IF fail = 1 OR tries = 4
490 LET dx=(INKEY$="8")-(INKEY$="5")
500 IF xcity+dx=256 OR xcity+dx=-1 THEN LET dx=0
510 LET dy=(INKEY$="7")-(INKEY$="6")
520 IF ycity+dy=176 OR yclty+dy=-1 THEN LET dy=0
530 IF dz=0 AND dy=0 THEN GO TO wait
540 PLOT OVER 1; xcity, ycity
550 LET xcity=xcity+dx
560 LET ycity=ycity+dy
570 GO TO move
580 REM finish
590 CLS
600 PRINT AT 4,2; "Верно с первой попытки:   ";r(1)
610 PRINT AT 7,2; "Верно со второй попытки:  ";r(2)
620 PRINT AT 10,2;"Верно с третьей попытки: ";r(3)
630 PRINT AT 13,2;"Неверно: ";r(4)
640 INPUT "Попробуем еще раз?",y$
650 IF CODE y$=89 OR CODE y$=121 THEN GO TO another
660 STOP
950 REM*****************
       *                *
       *  Подпрограммы  *
       *                *
       *****************
1000 REM контуры
1010 CLS
1020 PLOT 51,58
1030 DRAW 0,4
1040 DRAW 2,0
1050 DRAW 0,9
1060 DRAW -15,31
1070 DRAW 4,-3
1060 DRAW 1,2
1090 PLOT 51,128
1100 DRAW -8,-27,1.5
1110 PLOT 51,128
1120 DRAW 3,-1
1130 DRAW 6,3
1140 DRAW 12,9,1.5
1150 DRAW 3,1
1160 DRAW -1,4
1170 DRAW 3,5
```

```
1180 DRAW 3,-6
1190 DRAW 2,3
1200 DRAW -2,2
1210 DRAW 1,2
1220 DRAW 4,-2
1230 DRAW 0,7
1240 DRAW 10,7
1250 DRAW 3,-1
1260 DRAW 3,5
1270 DRAW 4,-2
1280 DRAW -2,4
1290 DRAW 7,9
1300 DRAW 8,1,1
1310 DRAW 0,2
1320 DRAW 2,1
1340 DRAW 1,-2
1350 DRAW 19,0,1
1360 DRAW 2,-2
1370 DRAW -7,-13
1380 DRAW 21,-14
1390 DRAW 4,1
1400 DRAW 5,11
1410 DRAW 2,20
1420 DRAW 3,1
1430 DRAW 1,-6
1440 DRAW 2,-4
1450 DRAW 1,-9
1460 DRAW 4,1
1470 DRAW 0,-2
1480 DRAW 3,-3
1490 DRAW 0,-7
1500 DRAW 2,-2
1510 DRAW 2,-10
1520 DRAW 2,-1
1530 DRAW 11,-13
1540 DRAW 0,-5
1550 DRAW 3,0
1560 DRAW 0,2
1570 DRAW 3,-1
1580 DRAW 0,-4
1590 PLOT 210,71
1600 DRAW -6,37,1.5
1610 PLOT 210,71
1620 DRAW -15,-26
1630 DRAW -3,-6
1640 DRAW -2,-1
1650 DRAW -8,-3,1.2
1660 DRAW -2,-1
1670 DRAW -1,-3
1660 DRAW -2,3
1690 DRAW -7,4
1700 DRAW -5,-3
1710 DRAW -11,4
1720 DRAW -2,5
1730 DRAW 1,1
1740 DRAW -5,7
1750 DRAW -2,-1
1760 DRAW 0,8,1
1770 DRAW -3,-5
1780 DRAW -1,5
1790 DRAW 3,3
1800 DRAW -2,3
1810 DRAW -9,-9
1820 DRAW -50,5,2.4
1830 DRAW -13,-2
```

```
1840 DRAW -6,-3
1850 DRAW -7,-1
1860 DRAW  6,4
1670 DRAW -1,-1
1880 PLOT 213,96
1890 DRAW 2,4,.5
1900 PLOT 142,55
1910 DRAW -2,0
1920 DRAW -1,-2
1930 DRAW 2,0
1940 DRAW 1,2
1950 PLOT 170,25
1960 DRAW 15,-1,7
1970 DRAW 1,2
1980 DRAW -1,3
1990 DRAW 3,-2
2000 PLOT 183,9
2010 DRAW 5,18,.5
2020 PLOT 183,9
2030 DRAW -3,0
2040 DRAW 0,-2
2050 DRAW -4,0
2060 DRAW 0,1
2070 DRAW -2,0
2080 DRAW -1,10
2090 DRAW -3,4
2100 DRAW 0,4
2110 RETURN
2500 REM города
2510 PLOT FLASH error;a,b
2520 PLOT FLASH error;a,b+1
2530 PLOT FLASH error;a+1,b
2540 PLOT FLASH error;a+1,b+1
2550 CIRCLE FLASH error;a,b+1,3
2560 PRINT FLASH error;AT Y,X;a$
2570 IF error=0 THEN RETURN
2580 PAUSE 50: PAUSE 200
2590 LET error = 0
2600 GO TO printcity
2700 REM test
2710 PLOT OVER 1; xcity, ycity
2720 IF ABS (xcity-a) < 4 AND ABS (ycity-b) < 4 THEN LET result=1
2730 IF result = 0 THEN LET tries=tries+1:LET fall=1
2740 LET xcity = 0: LET ycity = 0
2750 PAUSE 25
2760 RETURN
2900 REM  ************
          *          *
          *  Данные  *
          *          *
          ************
3000 REM Данные по городам
3010 DATA 172,39,17,22,"Мельбурн"
3020 DATA 199,56,15,25,"Сидней"
3030 DATA 53,71,13,7,"Перт"
3040 DATA 147,55,13,14,"Аделаида"
3050 DATA 110,165,1,14,"Дарвин"
3060 DATA 179,10,20,23,"Хобарт"
3070 DATA 212,81,10,23,"Брисбейн"
3080 DATA 0,0,0,0,"eof"
```

BETA BASIC

Продолжение.
Начало см. стр. 3,47

16. DO

или DO WHILE <условие>
или DO UNTIL <условие>

Клавиша D (Ключевое слово WHILE находится на клавише J, а ключевое слово UNTIL - на клавише K).

См. также LOOP, EXIT IF.

Конструкция DO - LOOP имеет ряд преимуществ по сравнению с обычным способом организации циклов FOR - NEXT стандартного БЕЙСИКа. Эти преимущества становятся еще более ощутимыми при использовании квалификаторов WHILE и UNTIL.

Без них DO и LOOP являются просто маркерами, отмечающими начало и конец цикла. После того, как программа встретит оператор LOOP, управление передается на оператор DO и т.д.

Пример:

```
10 DO
20   PRINT "HELLO"
30 LOOP
```

Эта программа будет печатать бесконечное число раз слово HELLO. Остановить ее можно будет только нажав BREAK.

Действие DO можно изменить с помощью квалификатора WHILE <условие>. Его действие таково:

Если условие стоящее после WHILE справедливо (имеет значение "истина"), то выполняются операторы стоящие после DO до тех пор, пока не встретится оператор LOOP, после чего управление вновь передается на DO и вновь проверяется справедливость условия и т.д. Если же условие "ложно", то вся часть программы, стоящая между DO и LOOP игнорируется и управление передается к оператору, стоящему за LOOP.

Таким образом, та часть программы, которая стоит между DO WHILE <условие> и LOOP выполняется раз за разом, пока <условие> справедливо.

DO UNTIL <условие> имеет прямо противоположное значение. Часть программы заключенная между DO и LOOP выполняется, пока условие "ложно" (иными словами до тех пор, пока условие не станет справедливым).

Пример:

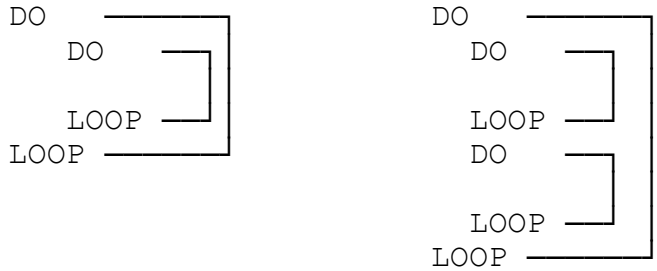
```
10 LET total = 0
20 DO UNTIL total > 100
30   INPUT "Введите число ";x
40   LET total = total * x
50   PRINT total
60 LOOP
70 PRINT "Получили число больше ста"
```

В этом примере строку 20 можно было бы заменить такой:

```
20 DO WHILE total < = 100.
```

Пары DO-LOOP могут вкладываться точно так же, как FOR - NEXT.

Например:



Компьютер запоминает адрес, по которому в программе хранится оператор DO. Для запоминания служит стек, поэтому нельзя выходить из цикла (перепрыгивать через LOOP), иначе как с помощью EXIT IF или POP. Иначе может произойти "закупоривание" стека и сбой работы в программе.

Если в программе для оператора DO не найден соответствующий LOOP, выдается сообщение об ошибке:

```
S, "Missing LOOP".
```

Аналогичные конструкции существуют и во многих других языках программирования и могут иметь другое написание, вот примеры аналогичных конструкций:

```
  REPEAT
     операторы
  UNTIL <условие>
```

аналогично:

```
  DO
     операторы
  LOOP UNTIL <условие>
```

```
  REPEAT
     операторы
  UNTIL FALSE
```

аналогично:

```
  DO
     операторы
  LOOP
  WHILE <условие>
     операторы
  ENDWHILE (ИЛИ WEND)
```

аналогично:

```
  DO WHILE <условие>
     операторы
  LOOP
```

17. DPOKE адрес, число

Клавиша: P

См. также DPEEK (адрес), число

DPOKE означает "двойной" POKE.

Этот оператор засылает двухбайтное число в две адресных ячейки. Аналогичная конструкция в стандартном Бейсике имеет следующий вид:

```
POKE a,n - INT(n/256)*256
POKE a+1,INT(n/256)
```

Здесь a - адрес

n - число (0…65535)

Другими словами, младший байт засылается по указанному адресу, а старший байт - в следующий за ним адрес. Поскольку многие системные переменные "Спектрума" имеют именно такой двухбайтный формат, то их очень удобно изменять с помощью DPOKE. Аналогично функция DPEEK представляет "двойной" PEEK.

18. DRAW TO x,y <,угол>

Здесь используются два ключевых слова стандартного Бейсика - DRAW и TO.

Этот оператор вычерчивает линию от текущей графической позиции до точки, заданной координатами x,y. Часто это более удобно, чем использовать стандартный DRAW, при котором задаются не абсолютные координаты, а величина относительного "смешения" по горизонтали и вертикали.

Пример:

```
10 FOR n=1 TO 100
20   DRAW TO RND*255,RND*175
30 NEXT n
```

Если Вы зададите и третий параметр (угол), то сможете изображать кривые.

После TO Вы можете поставить оператор цвета PAPER, INK или атрибут, например OVER и т.п.

```
DRAW TO 10,10
DRAW TO INK 2;20,30
DRAW TO 100,90,1
```

18. EDIT <номер строки>

Клавиша: 0 (в обычном, не графическом режиме).

Это не то же самое, что SHIFT + "1". EDIT - это ключевое слово. Оно предназначено для того, чтобы избежать утомительной последовательности: LIST - выбор строки - BREAK - SHIFT + "1".

Для того, чтобы быть по-настоящему полезным, EDIT должен вызываться без нажатия SHIFT-клавиши. Но все буквенные клавиши уже заняты, а цифровые клавиши нужны для ввода номеров строк. В то же время, номера строк не начинаются с нуля, поэтому эту клавишу можно использовать.

Вы закончили набор какой-либо строки. Нажали ENTER. Программа ждет ввода номера очередной строки. Вы нажимаете 0 и получаете ключевое слово EDIT (если даже оно и не появятся, все равно нуль в начале строки означает EDIT). Если после EDIT (или нуля) набрать номер нужной Вам строки и нажать ENTER, эта строка мгновенно будет помешена в нижнюю часть экрана для последующего редактирования.

Если номер строки не указан, то на редактирование поступит текущая строка. Дополнительно для упрощения редактирования введена возможность перемещать курсор по экрану "вверх"/"вниз" в рамках программной строки. Курсор не может вставать внутри ключевых слов, а занимает ближайший пробел. Если Вы попытаетесь поднять его выше, чем вершина строки или опустить ниже нижней строки экрана, он автоматически "прыгнет" в конец программной строки. Самый простой способ добавить что-то к концу строки - вызвать EDIT и затем нажать "курсор вверх".

9. EDIT строковая переменная

или EDIT ; числовая переменная

Клавиша: SHIFT + "5"

EDIT может применяться не только для удобного редактирования строк, но и для простого изменения программных переменных. Для этих целей нельзя использовать нуль в начале строки и введена возможность вызова EDIT нажатием SHIFT + "5" в графическом режиме. Можно также набрать слово EDIT по буквам в режимах KEYWORDS 3 или 4. Типичное применение - внесение изменений в строковые переменные, например для изменения фамилии, имени, отчества, адреса в массиве данных, представляющем из себя базу данных.

Такой массив мог быть заполнен, например с помощью INPUT. Если теперь надо в нем что-то изменить, то это трудоемкая задача для обычного БЕЙСИКа, а здесь с помощью EDIT a$(n) Вы получаете содержимое строки "n" в области редактирования. Рассмотрим пример:

```
10 LET a$ = "John Brown"
20 EDIT a$
30 PRINT a$
40 LET num=365.253
50 EDIT ; num
60 PRINT num
```

В строке 50 применен знак ";" для того, чтобы отличить режим EDIT <числовая переменная> от режима EDIT <номер строки>. Можно было бы, однако, вместо точки с запятой использовать и запятую, но в этом случае редактируемая переменная была бы изображена, начиная с 16-ой позиции экрана.

EDIT имеет синтаксис, очень похожий на INPUT. Вы можете использовать точку с запятой, запятую, AT, TAB, LINE и строковую подсказку точно так же, как обычно это делают с оператором INPUT. Отличие в том, что редактировать можно только одну переменную. Если Вы попробуете редактировать несколько, то все прочие, кроме первой, будут восприняты, как операторы INPUT. Нижеследующий пример показывает, как можно создавать массивы и редактировать их:

```
10 DIM a$(10, 15)
20 FOR n=1 то 10
30   INPUT a$(n)
40 NEXT n
50 PRINT "Редактирование"
60 FOR n=1 TO 10
70 EDIT ("запись ";n;" ");a$(n)
80 NEXT n
```

Если переменная, подлежащая редактированию в EDIT не существует, то команда полностью эквивалентна INPUT.

20. ELSE <оператор>

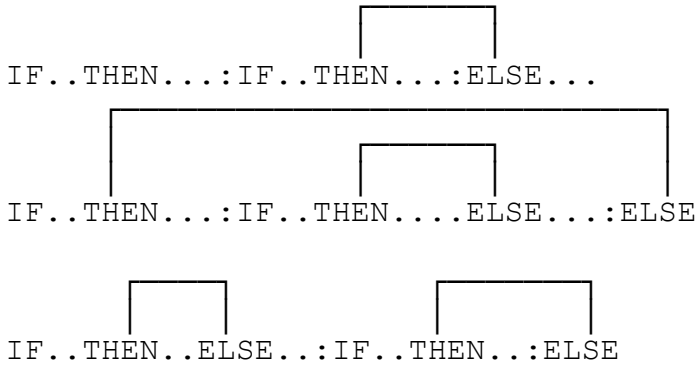
Клавиша: E

ELSE - один из элементов конструкции IF - THEN. Обычно, если условие, стоящее после if несправедливо (ложно), то следующей выполняется строка, стоящая за строкой, содержащей этот IF. Этот порядок можно изменить. Если IF и THEN содержат еще и ELSE, то в случае, когда условие не выполняется, управление передается оператору, стоящему после ELSE. С другой стороны, если условие справедливо, то строка IF исполняется только до ELSE и затем управление передается следующей строке (если THEN не передал его в иное место).

Например:

```
10 INPUT "Задайте число ";x
20 PRINT "Это число равно 1?"
30 PAUSE 50
40 IF x=1 THEN PRINT "ДА": ELSE PRINT "НЕТ"
50 GO TO 10
```

Однако, ситуация может быть осложнена, если на одной строке у Вас есть несколько IF THEN ELSE. Тогда может быть трудным определить, что же к чему относится. Приведенные ниже примеры помогут Вам разобраться.



Если ELSE используется без IF, например, когда ELSE стоит первым оператором строки, то и ELSE и остальная часть строки игнорируются.

21. END PROC

Клавиша: 3

См. также раздел, посвященный процедурам PROCEDURES; DEF PROC.

Этот оператор отмечает конец процедуры. Благодаря этому компьютер не будет выполнять ничего, стоящего между DEF PROC и END PROC, если процедура не была вызвана. Он просто перепрыгнет через нее за END PROC. Во время же исполнения процедуры, END PROC служит концом ее исполнения. Удаляются локальные переменные (LOCAL) восстанавливаются исходные значения глобальных (GLOBAL) переменных, если они есть. Если в процедуру какие-то параметры передавались по ссылке (перед формальным параметром стояло REF), то текущее значение формального параметра присваивается фактическому параметру, задействованному при вызове.

Если формальный параметр не найден, Вы получите сообщение "Variable not found".

После исполнения процедуры управление передается оператору, стоящему за вызовом процедуры.

Применение END PROC без соответствующего DEF PROC дает сообщение об ошибке:

```
W, "Missing DEF PROC".
```

22. EXIT IF <условие>

Клавиша: I

См. также DO, LOOP

Этот оператор является элементом конструкции цикла DO-LOOP (см. соответствующий раздел). EXIT IF используется для того, чтобы выйти по своему желанию из середины цикла DO-LOOP, если условие выполняется. После выхода из цикла управление передает оператору, следующему за LOOP.

Если условие не выполняется, ничего не происходит.

Пример:

```
100 DO
110   PRINT "строка 110"
120   PAUSE 20
130 EXIT IF INKEY$ = "" STOP
140   PRINT "строка 140"
150   PAUSE 20
160 LOOP
170 PRINT "Выход из цикла"
```

Программа будет работать до тех пор, пока не будет нажата клавиша "STOP" (SYM. SHIFT + A).

Если в программе опущен оператор LOOP, выдается сообщение ошибке:

S, "Missing LOOP"

23. FILL x,y

или FILL <INK число; > x,y

или FILL <PAPER число;> x,y

Клавиша: F

Команда FILL или FILL INK закрашивает область экрана, окрашенную цветом PAPER в цвет INK, начиная с координат x,y.

Команда FILL PAPER наоборот закрашивает область экрана, окрашенную цветок INK в цвет PAPER.

Закрашивание происходит от точки с координатами x,y во все стороны до тех пор, пока не встретятся участки, уже окрашенные в INK для команд FILL INK и FILL или в PAPER для команды FILL PAPER. Если исходная точка и ее окрестности окрашены в требуемый ничего не происходит.

В отличие от обычного Бейсика параметр цвета после INK или PAPER может не указываться, в этом случае идет закрашивание текущим цветом. Например, FILL PAPER; x,y - сработает.

Пример:

```
10 FOR n=1 TO 6
20    CLS
30    CIRCLE INK n;128,88,n*10
40    FILL INK n;  128,88
50 NEXT n
```

Возможно и использование сложных конструкций FILL, таких как:

```
FILL INK 2; PAPER 1; FLASH 1; X,Y
```

В таких случаях первое ключевое слово после FILL указывает на то что используется при заполнении (INK или PAPER), а остальные просто меняют атрибуты заполняемой области.

Поскольку, как Вы знаете, в пределах одного знакоместа невозможно использование более двух цветов одновременно, техника работы с цветом должна быть очень хорошо продуманной. Очень легко получить неприемлемый результат, особенно в тех местах, где контактируют два цвета INK. Обычно это обходят за счет того, что стык областей с разными цветами проводят по границам знакомест.

```
10 LET x=128, y=88, rad=70
20 LET g= (SQR 2)*rad/2
30 CIRCLE  x,y,rad
40 PLOT x,y : DRAW -g,-g
50 PLOT x,y : DRAW g,-g
60 PLOT x,y : DRAW 0,rad
70 FILL INK 2; x-5,y
80 FILL INK 4; x+5 y
```

FILL работает на областях любой геометрической формы. Для примера попробуйте заполнить весь экран, за исключением того, что находится внутри букв "Q".

```
PRINT STRING$ (704, "Q"): FILL 0,0
```

Метод работает быстро, если есть большое количество доступной свободной памяти. Если Вы заполняете большую область, например весь экран, то сможете увидеть очевидные паузы в те моменты, когда компьютер проверяет не забыл ли он окрасить какие-то участки. Можно рекомендовать делить сложную область на несколько более простых и заполнять их порознь.

Можно установить, какое количество пикселей было закрашено по последней команде FILL, для этого служит функция FILLED().

Прервать заполнение можно в любое время нажатием BREAK.

24. GET числовая переменная

или GET строковая переменная

Клавиша: G

Это то же самое, что GET от клавиатуры. Как и INKEY$ это способ чтения клавиатуры без использования ENTER. Отличие от INKEY$ в том, что GET ждет нажатия клавиши. При использовании со строковой переменной, GET вводит один символ:

```
10 GET a$: PRINT a$;: GO TO 10
```

Работа этой строки похожа на работу пишущей машинки. Обычным путем Вы можете переключаться на печать прописными и строчными буквами. Курсор можно перемещать в нужном направлении, работает стирание символов. ENTER дает переход к началу следующей строки. Более изощренная версия выглядит так:

```
10 GET a$
20 PRINT CHR$ 8; " "; CHR$ 8; a$; INVERSE 1; "B"; INVERSE 0;
30 GO TO 10
```

Может быть, Вы захотите изменить размер символов, воспользовавшись командой CSIZE.

Если GET применяется с числовой переменной, то при нажатии клавиш от "1" до "9" вводится число от 1 до 9. При нажатии "A" или "a" вводится число 10, при нажатии "B" или "b" вводится число 11 и т.д. Это очень удобно для организации разного рода экранных меню при написании меню-управляемых программ (см. также ON).

25. GET строковая переменная, x,y <ширина, длинах><;тип>

Это как бы GET с экрана. Применяется для того, чтобы какой-то прямоугольный блок экрана присвоить некоей строковой переменной и впоследствии печатать его на экране с помощью PRINT или PLOT в тех позициях экрана, в каких захотите.

(Если Вы будете применять PRINT, то необходимо учесть, что установка CSIZE не должна быть нулевой. Так, например, CSIZE 8 даст печать сохраненного в строковой

переменной изображения в натуральную величину.)

Размер задается в знакоместах, а координаты - в пикселах. В простейшем случае за именем строковой переменной следуют координаты левого верхнего угла.

```
10 PRINT "QWE"
20 GET a$,4,175
30 PLOT 100, 100; a$
```

Т.к. размеры снимаемого с экрана блока не указаны, то по умолчанию предполагается, что длина и ширина равны по одному знакоместу.

В этом примере на экране будет напечатано "QWE", а затем правая половина буквы Q и левая половина буквы W будут сохранены в переменной a$. С помощью усовершенствованной команды Бета-Бейсика PLOT Вы сможете напечатать полученное изображение в любом месте экрана (см. PLOT), причем сделать это можно с разным увеличением (см. CSIZE).

Здесь есть один нюанс. Как может строковая переменная содержать часть графики экрана? Если Вам это не интересно, пропустите несколько следующих абзацев.

Поставим маленький эксперимент. Узнаем длину строковой переменной a$. Это можно сделать PRINT LEN a$. Вы увидите, что длина этой переменной равна 9 символам. Первый содержит управляющий код, который говорит о том, что в последующих 8 символах идет графический образ. Если Вы сняли блок большего размера, а не одно знакоместо, то в него вставлены там, где это надо, коды управления курсором (см. раздел Управляющие Коды).

Нижеприведенный пример изображает на экране блок размером 3x3 знакоместа, запоминает его в переменной a$ и затем печатает в случайно выбранных позициях экрана, причем делает это гораздо быстрее, чем это можно было бы сделать повторным его перестроением.

```
10 CIRCLE 10,165,10
20 CIRCLE 13,165,5
30 FILL 5,165
40 GET a$,0,175,3,3
50 PLOT RND*230, RND*150+20; a$
60 GO TO 50
```

Если Вы используете OVER 0, то увидите, что поля рисунка затрут цветом PAPER то, что лежит под ними. Для получения других результатов попробуйте использовать OVER 1 или OVER 2 и запустите пример еще раз (OVER 2 это режим не стандартного БЕЙСИКа, а БЕТА-БЕЙСИКа, позволяющий Вам рисовать и без затирания и без инвертирования того, что лежит под Вашим изображением.)

Если Вы хотите нарисовать много окружностей одного размера, то гораздо быстрее работают GET и PLOT, чем CIRCLE. Попробуйте удалить строки 20 и 30 и введите OVER 2 (или используйте в строке 50 PLOT OVER 2; RND * ...)

Эти рассмотренные примеры относятся к нулевому типу. Если Вы не указываете при команде GET параметр <тип>, то предполагается, что он равен нулю. Нулевой тип команды GET - "бесцветный".

То есть рисунок снимается с экрана без цветовых атрибутов и воспроизводится командой PLOT в текущих установленных цветах или в локальных цветах, указанных в команде PLOT или PRINT.

Вы можете использовать команду GET и другого типа. Это тип-1. Для его использования после команды поставьте точку с запятой и поставьте 1. В этом случае изображение снимается с экрана в строковую переменную вместе с цветовыми атрибутами. Точно так же оно будет и изображено по команде PLOT или PRINT. Принципиально важно, что если графический блок, с которым Вы работаете, имеет несколько цветов, то все их можно будет воспроизвести только при работе с первым типом GET. Ни глобальные, ни локальные установки цвета не повлияют. Изображение будет воспроизведено так, как было снято.

```
10 PRINT INK 2; "ABC"
20 PRINT: PRINT INK 4; "DEF"
30 GET s$,0,175,3,3; 1
```

```
40 PLOT 100,100; s$
```

Как обычно, Вы не должны забывать, что для "Спектрума" в пределах одного знакоместа возможны только 2 цвета (INK и PAPER), поэтому тщательно планируйте, куда Вы помещаете изображение, снятое по GET вместе с цветами. При неаккуратной печати его на экране Вам не избежать проблем с атрибутами ("клэшинг" атрибутов).

26. JOIN <номер строки>

Клавиша: SHIFT + 6 (то же, что и "&")

См. также SPLIT.

Объединяются вместе две строки. Первая строка - номер которой задан (если не задан - то та строка, на которой стоит курсор) и следующая за ней строка.

Совместно используя SPLIT и JOIN Вы получаете возможность "отрезать" часть строки и "пристегнуть" ее к другой.

27. JOIN строковый массив или числовой массив.

Клавиша: SHIFT + 6 (то же, что и "&")

См. также главу "Обработка данных", с. 8.

Эта команда позволяет переместить часть символьной строки или массива в любую позицию другой символьной строки или числового массива. С командой JOIN тесно связана еще команда COPY. Поскольку они очень похожи друг на друга, то и обсуждаться здесь будут совместно. Разница их действия состоит в том, что если команда JOIN перемещает данные, то команда COPY - копирует их. Разница очевидна. При копировании исходный материал не уничтожается, а при перемещении он пропадает.

а) Работа с символьными массивами.

Образец синтаксиса:

```
JOIN a$ <n TO m> TO b$ <k>
COPY a$ <n TO m> TO b$ <k>
```

Здесь:

a$ - исходная символьная строка;

b$ - строка назначения;

n,m - параметры выделения подстроки из строки;

k - позиция в строке назначения.

Пример:

```
10 LET a$ = "12345"
20 LET b$ = "ABCDEFG"
30 JOIN a$ TO b$
40 PRINT b$: REM печатается ABCDEFG12345
50 PRINT a$: REM: a$ не найдено.
```

Поскольку мы применяли JOIN и не указывали параметров n,m, то вся переменная a$ целиком перешла в b$ и печать a$ в строке 50 ничего не даст. Если же теперь Вы в строке 30 замените JOIN на COPY, то сможете убедиться в строке 50, что a$ не пострадало. Вариант применения COPY, тем самым, похож на то же, что мы имеем просто работая с LET:

```
LET b$ = b$ + a$
```

Но зато и JOIN и COPY работают даже тогда, когда a$ и b$ такие огромные, что заполняют всю память компьютера, а LET в этом случае работать не сможет. Например LET a$ = a$ + "x" не сможет работать, если a$ длиннее, чем одна треть свободной памяти компьютера.

С помощью параметров выделения подстроки из строки Вы можете работать не только со строками, но и с подстроками. Их можно вставлять в строку назначения в заданное место, указав параметр позиции вставки.

Если не заданы параметры подстроки, то принимается по умолчанию вся строка. Если не задан параметр позиции вставки в строку назначения, то вставка выполняется после последней позиции. Попробуйте поэкспериментировать с вышеприведенным примером, заменяя строку 30:

```
30 JOIN a$ (2) TO b$
      REM:
      a$="1345",
      b$="ABCDEFG2"
30 JOIN a$ (3 TO) TO b$
      REM:
      a$="12",
      b$="ABCDEFG345"
30 COPY a$ TO b$(3)
      REM:
      a$="12345",
      b$="AB1234CDEFG"
30 JOIN a$(2 TO 3) TO b$(LEN b$+1)
      REM:
      a$="145",
      b$="ABCDEFG23"
```

А теперь рассмотрим конкретный практический пример. Вам надо просмотреть символьную строку t$ и всякий раз, как в ней встретится слово "Spectrum" заменить его на "ZX". Для этого можно исполь зовать COPY, DELETE и INSTRING.

```
10 LET t$="The Spectrum is a versatile computer, but Spectrum owners may feel it should have
      better editing"
20 PRINT t$
30 LET n$="ZX", o$="Spectrum"
40 LET P=1
50 LET P=INSTRING (P,t$,o$)
60 IF P<>0 THEN
      DELETE t$(P TO P+LEN o$-1)
      COPY n$ TO t$(P)
      GO TO 50
70 PRINT t$
```

b) Работа с символьными массивами.

Образец синтаксиса:

```
JOIN a <n TO m> TO b <k>
COPY a <n TO m> TO b <k>
```

Здесь:

a - исходный массив;

b - массив назначения;

n, m - параметры выделения подмассива из массива,

k - позиция в массиве назначения.

Массивы - это удобный метод работы с большим количеством данных, но обычно они имеют много серьезных ограничений. Наверное самое существенное из них - то, что массивы имеют фиксированный размер, объявленный в момент назначения массива. Вы можете нерационально расходовать память, назначив (DIM) массив больше, чем Вам на самом деле необходимо, и Вам может не хватать памяти например для создания еще одного массива.

БЕТА-БЕЙСИК позволяет более гибко манипулировать с массивами. Команды JOIN и COPY позволяют перемещать или копировать весь массив или его часть. Пространство для нового материала создается внутри назначенного массива, поэтому ничего не будет перезатерто. Можно обслуживать не только одномерные, но и двумерные массивы, а этого бывает достаточно для большинства приложений. Одномерные массивы трактуются, как двумерные, имеющие вторую размерность, равную единице. Нет никакой необходимости чтобы совпадали размерности исходного массива и массива назначения.

Нет также никакой необходимости, чтобы оба массива имели строго одинаковое количество рядов и их длину. Когда Вы работаете со строковыми массивами, строки исходного массива "подрезаются" в соответствии с длиной строк массива назначения, если они длиннее или, наоборот, "подбиваются пробелами", если они короче. Числовые массивы обрабатываются так же, за исключением того, что вместо пробелов используются нули.

Предположим, что Вы имеете массив a$(100,30) и он полностью заполнен. Чтобы

добавить к нему еще 20 символьных строк, Вы можете действовать например так:

```
DIM b$(20,30): JOIN b$ TO a$
```

Поскольку мы использовали JOIN, все строки будут реально удалены из массива b$ и помещены в a$, а не просто скопированы, как это было бы при применении COPY. Поскольку никаких параметров вырезки после имени исходного массива не было указано, то все его символьные строки будут перемещены и массив перестанет существовать. При использовании COPY он остался бы нетронутым.

Позиция вставки в массив назначения была указана. По умолчанию будет принято, что это позиция номер 101, то есть следующая за последней. Первая символьная строка из массива b$ станет сто первой в массиве a$, а последняя станет строкой номер 120.

Т.к. в БЕТА-БЕЙСИКе массивы могут часто менять свой размер, то Вам может потребоваться неоднократное применение функции LENGTH чтобы определить размер массива. В вышеприведенном примере LENGTH (1,"a$") дало бы величину 120.

Если Вы примените DIM b$(20,3) или DIM b$ (20,50), а затем сделаете JOIN b$ TO a$, то в результате строки массива b$ будут "подбиты" пробелами или, наоборот "подрезаны" так, чтобы они укладывались в строки массива a$, имеющие 30-символьную длину.

Если у Вас уже есть готовый массив и Вы решите, что в нем надо сделать строки подлиннее, Вы можете это сделать, задав через DIM массив с желаемой длиной только с одним элементом, а потом переместить (JOIN) свой массив в него.

```
DIM b$(1,40): JOIN a$ TO b$
```

Единственный недостаток, к сожалению, состоит в том, что теперь все наши данные будут находиться в b$, а a$ не будет существовать. Чтобы восстановить его, можно сделать обратный ход:

```
DIM a$(1,40): JOIN b$ TO a$
```

До сих пор мы имели дело с полными массивами, но и JOIN и COPY могут работать много гибче. Вы можете указать какие символьные строки, или какие ряды исходного числового массива Вы хотите использовать с помощью выделяющих параметров, а также позицию, в которую произойдет вставка.

Следующие примеры написаны для JOIN, но соответственно могут работать и с COPY.

```
JOIN a$ TO b$(4)
```

- введет весь массив a$ в b$ таким образом, что первая символьная строка будет вставлена после четвертой символьной строки в увеличенном массиве b$.

```
JOIN a$(2 TO 5) TO b$(1)
```

- переместит 4 символьных строки из a$ в начало массива b$. После этого массив a$ станет на четыре строки короче, чем был, а массив b$ станет на четыре строки длиннее.

```
JOIN a$(10) то b$
```

- переместит десятую символьную строку из a$ в конец b$.

c) числовые массивы.

Одномерные и двумерные числовые массивы обрабатываются способом, похожим на символьные массивы. После имени массива должны обязательно стоять скобки, чтобы отличать массивы от простых переменных.

Нижеприведенный пример создает два массива, а затем объединяет их между собой.

```
10 DIM a(8)
20 FOR n=1 TO 8
30   LET a(n)=n
40 NEXT n
50 DIM b(5)
60 FOR n=1 TO 5
70   LET b(n)=n*10
80 NEXT n
90 JOIN b() TO a()
100 FOR n=1 TO LENGTH (1,"a(")
110   PRINT a(n)
120 NEXT n
130 REM печать 1 2 3 4 5 6 7 8 10 20 30 40 50
140 REM b() - не существует
```

28. KEYIN строковая переменная

Клавиша: SHIFT + 4

KEYIN вводит строковую переменную так, как если бы Вы впечатали ее с клавиатуры. Таким образом, есть возможность сделать программу самосоздающейся, хотя это и выходит за пределы, рассматриваемые в данной инструкции. Можно рассмотреть пример автоматического создания строк DATA.

```
10 LET a$ = "100 DATA"
20 FOR n=0 TO 9
30 LET a$ = a$+STR$(PEEK N) + ","
40 NEXT n
50 LET a$ = a$ (1 TO LEN a$ -1): REM удаление последней запятой
60 KEYIN a$
```

После запуска данного фрагмента Вы увидите, что к программе добавилась еще одна строка.

В режимах KEYWORDS 3 и 4 ключевые слова, набранные по символам, будут преобразованы в односимвольные токены.

29. KEYWORDS число.

Клавиша: 8

Оператор KEYWORDS управляет тем, как вводятся и как изображаются на экране ключевые слова стандартного БЕЙСИКА, БЕТА-БЕЙСИКа и символов графики пользователя (UDG).

KEYWORDS 0.

В этом режиме за клавишами в графическом режиме (курсор G) закреплены символы графики пользователя, как в стандартном БЕЙСИКе Вашего "Спектрума".

KEYWORDS 1.

В этом режиме за клавишами в графическом режиме закреплены ключевые слова БЕТА-БЕЙСИКА.

В исходном состоянии после загрузки система находится в режиме KEYWORDS 1. А если Вам надо воспользоваться символами графики пользователя, дайте команду KEYWORDS 0. Выбранный Вами режим KEYWORDS 1 или KEYWORDS 0 не влияет на то, как вводятся ключевые слова - по буквам или как токены, целиком, т.е. выбор режима 0 или 1 не влияет на установки режимов KEYWORDS 2, 3 и 4. После загрузки компьютер находится в состоянии KEYWORDS 3. Текущий режим ввода сохраняется вместе с Вашей программой, если Вы отгружаете вместе с ней код (CODE) самого БЕТА-БЕЙСИКА.

KEYWORDS 2.

В этом режиме все ключевые слова вводятся одним нажатием клавиш, при необходимости с одновременным нажатием шифтов. Ключевые слова стандартного БЕЙСИКа берутся, как обычно. Ключевые слова БЕТА-БЕЙСИКа вводятся в графическом режиме (курсор G), а функции БЕТА-БЕЙСИКа вводятся в порядке FN, буква, знак "$" или "(".

KEYWORDS 3.

Этот режим - тот же, что и KEYWORDS 2, за исключением того, что вводимая строка, прежде чем пойти в память компьютера, сначала проверяется на наличие в ней набранных по буквам ключевых слов и, если они найдены, происходит их замена на токены ключевых слов. По-видимому, это самый удобный режим, т.к. в нем можно работать и с вводом ключевых слов одним нажатием клавиш и со вводом их по буквам. Если нужно выйти из курсора "K", Вы можете использовать ведущий пробел.

KEYWORDS 4.

В этом режиме нет курсора "K", т.е. все ключевые слова вводятся только по буквам. Форсировать появление курсора "K", тем не менее, все же возможно. Это делается нажатием клавиш SYMBOL SHIFT и ENTER.

Этот режим, возможно, предпочтут те, кому может потребоваться совместимость с другими моделями персональных компьютеров.

d) Распознавание ключевых слов.

Есть небольшое ограничение в режимах 3 и 4, которое заключается в том, что Вы не можете использовать имена переменных и процедур, совпадающие по написанию с ключевыми словами. Поскольку при запоминании программной строки в памяти компьютера может произойти их замена на токен ключевого слова. Тем не менее, ключевое слово может быть частью имени переменной или процедуры без проблем.

Ключевые слова в этих режимах распознаются как набранные прописными, так и строчными буквами. Мы Вам рекомендуем использовать строчные буквы, тогда после конверсии этих ключевых слов в токены, Вы сможете наглядно увидеть, какие ключевые слова были преобразованы и, возможно, найдете синтаксическую ошибку.

Символы, стоящие непосредственно перед ключевым словом или за ним не могут быть буквами или символами подчеркивания "_".

Printa - не будет конвертировано в PRINT a, т.к. компьютер предположит, что это имя переменной или процедуры.

Print a - будет преобразовано в PRINT a. Ниже приведены несколько примеров возможных строк и показано их возможное преобразование в результате "токенизации".

```
print 10.....................PRINT 10
print fork, total............PRINT fork, total
alter to ink3, paper1.........ALTER TO INK 3, PAPER 1
print string$(10,"Plot").....PRINT STRING$(10,"Plot")
goto10.......................GO TO 10
go to x......................GO TO x
gotox..... .................gotox
defproc pink.................DEF PROC pink
mat_print....................mat_print
```

Пример с GO TO показывает, что внутренние пробелы в этот оператор можно и не включать. Точно так же и "gosub", "onerror", "defproc" будут распознаны, как полноценные ключевые слова.

Слово "ink", входящее как составная часть в "pink" и "print", входящее в "mat_print" распознаны не будут, т.к. перед ключевым словом не должно быть буквы или знака "_".

30. LET переменная = число <, переменная = число>...

Маленькое усовершенствование старой команды LET позволяет выполнять серию присвоений, разделяя их запятыми. Это сокращает объем занятой памяти (по одному байту на каждый опущенный LET), делает ввод более удобным и листинг более читаемым. Так, запись

```
10 LET x=1,y=2,z=3,a$="y",b$="n"
```

может заменить:

```
10 LET x=1: LET y=2: LET z=3: LET a$="y": LET b$="n"
```

31. LIST <номер строки> TO <номер строки>

или LLIST <номер строки> TO <номер строки>

Вы видите по синтаксису, что здесь есть небольшое добавление к стандартному БЕЙСИКу. Вы можете выводить на экран или принтер заданный Вами блок программы. Если первый номер строки не указан, то по умолчанию принимается строка, следующая за нулевой. Если второй номер строки не указан, предполагается по умолчанию последняя строка программы. Если оба параметра опущены, Вы получаете эквивалент обычной команде LIST.

```
LIST 20 TO 100
LIST TO 200
LLIST 100 TO 180
```

Если строка с первым номером существует, то при листинге она будет изображена с курсором ">", т.е. она готова к вызову на редактирование.

Если оба номера совпадают, то будет изображена только одна строка.

32. LIST DATA

или LIST VAL
или LIST VAL$
Эти разновидности команды LIST позволяют распечатать сводку переменных:
LIST DATA - все переменные
LIST VAL - числовые переменные
LIST VAL$ - строковые переменные.
"Спектрум" имеет 6 типов переменных. Команда LIST VAL распечатывает из них 4 типа в следующем порядке:

1. Числовые массивы.
2. Переменные циклов FOR-NEXT.
3. Переменные с односимвольными именами.
4. Переменные с многосимвольными именами.

Команда LIST VAL$ распечатает остальные два типа переменных

5. Символьные массивы.
6. Обычные символьные переменные.

Команда LIST DATA распечатает все 6 типов переменных.
Переменные каждого типа распечатываются в алфавитном порядке (для переменных с многосимвольными именем в расчет принимается только первая буква). Пример того что может дать LIST DATA приведен ниже:

```
d(10,4)
k(3,3,4)
n STEP 1    500          LN 200
g           3.5
j           100
s           23.1

applen      1
number      9999
xos         0
xrg         256
yos         0
yrg         176

t$(100,10)

a$ LEN 5    "Hello"
b$ LEN 40   "Too long too..."
e$ LEN 5    "Bang!"
```

Для массивов изображается только их размерность, но не содержание. Переменные циклов FOR-NEXT можно отличить от прочих благодаря присутствию параметра STEP и параметра LN (looping number - номер строки, из которой производится возврат в голову цикла). Для длинных строковых переменных изображаются только первые 15 символов.

33. LIST DEF KEY

Здесь DEF KEY располагается на клавише SHIFT + 1.
См. также DEF KEY.
Эта команда позволит распечатать список клавиш, определенных пользователем. Сначала распечатывается клавиша, а затем символьная строка или оператор, которые за ней закреплены. Если в назначении клавиши последним символом является двоеточие, то полагается, что при нажатии этой клавиши то, что за ней закреплено, сопровождается последующим ENTER, т.е. сразу после нажатия содержимое появляется в нижней части экрана в системном окне.

Михайленко В.С.

ЗАЩИТА ПРОГРАММ

Продолжение. (Начало см. стр. 9-16, 53-60)

Глава 4. Прочие приемы защиты.

4.1 Запуск программ в кодах.

О том, как запустить программу в машинных кодах, наверное, знают все пользователи "ZX SPECTRUM". Эта информация изложена во всех справочниках по данному типу компьютеров.

Вкратце напомним основные положения данной системы команд.

Для вызова подпрограмм в машинных кодах используется функция USR, составленная из ключевых слов английского языка:

```
User SubRoutine
```

В компьютерах типа ZX SPECTRUM эта функция может использоваться двояко. Во-первых, она применяется в случаях, когда необходимо вызвать подпрограмму, написанную машинными кодами и расположенную в памяти по известному адресу.

Она также может применяться для записи данных графики, устанавливаемой пользователем, в зарезервированное место в конце памяти.

Нас интересует случай применения USR для вызова подпрограмм в машинных кодах. Для запуска данной подпрограммы USR используется с ключевым словом Бейсика RANDOMIZE или PRINT. После комбинации этих слов указывается цифровая величина, например:

```
90 RANDOMIZE USR 30000
100 PRINT USR 45000
```

Если вместо определенного цифрового значения используется выражение, то оно должно быть заключено в скобки. Указанная величина округляется до ближайшего целого числа - это адрес памяти, с которого и должна стартовать записанная машинными кодами подпрограмма. В общем случае адрес начала подпрограммы и адрес, с которого она стартует, могут не совпадать, хотя очень часто они совпадают.

Результат функции USR - значение, находящееся в паре регистров BC микропроцессора. Комбинация ключевых слов RANDOMIZE USR или RESTORE USR только запускает подпрограмму, тогда как PRINT USR еще и индицирует содержимое пары регистров BC на экране.

Фактически, это достаточно широко известная информация и на ней не стоило бы останавливаться, если бы не одно но:

- современные системы защиты, используя встроенные функции компьютера, создали новые системы команд для запуска подпрограмм в машинных кодах.

То, что из этого следует, очевидно. Засекретив начало Вашей подпрограммы в кодах, Вы сбиваете с толку "хакеров", а это в свою очередь защищает Вашу программу от несанкционированного просмотра. Рассмотрим более подробно эти новые ухищрения.

Выше были рассмотрены команды, достаточно широко известные и наиболее часто применяемые. Чуть реже встречается комбинация

```
LET A = USR 30000
```

Фактически эта команда полностью аналогична рассмотренным выше - она запускает подпрограмму в кодах с адреса, указанного в функции USR - в данном случае, с адреса 30000. Любопытно, что эта комбинация достаточно широко используется при работе со

встроенным калькулятором "СПЕКТРУМа". (Для тех, кто интересуется этим более подробно, рекомендуем изучить трехтомник "ИНФОРКОМа" по программированию в машинных кодах, где это применение описано более подробно).

Но, если читатель мог встретить подобную комбинацию в некоторых программах, то описанные ниже выражения встречаются достаточно редко. Я имею ввиду применение для запуска подпрограмм в кодах некоторые функции Бейсика.

В частности, если Вы подадите команду:

```
GO TO USR 30000
```

то выполнение команды осуществится аналогично

```
RANDOMIZE USR 30000
```

Среди таких модификаций команды RANDOMIZE USR следует отметить еще ряд команд, в частности:

```
LIST USR
```

и

```
CLOSE# USR
```

Однако следует учитывать, что применение данной системы команд будет иметь весьма незначительный эффект, если не учитывать одного нюанса, а именно: несмотря на вариации первого ключевого слова, наличие команды запуска подпрограммы в кодах достаточно очевидно. Вот почему эту комбинацию следует использовать совместно с другими приемами.

Для пояснения вышеизложенного рассмотрим применение подобной системы команд у Billa Gilbert-a (следует отметить, что он берет на вооружение все новые системы защиты и поэтому изучение его методов и приемов значительно повысит Ваш профессиональный уровень. В самом деле, он взламывает программу и должен поставить на нее такую защиту, чтобы другому вскрыть ее уже не удалось. Причем следует отметить, что он использует и достаточно оригинальные методы, те свои собственные разработки, которые не встречаются ни в каких других программах).

Когда мы просматривали дампинг одной из программ, вскрытых Биллом Гилбертом, то обнаружили достаточно любопытную комбинацию символов. После номера строки следовало:

```
CLOSE # USR 0
```

и т.д.

Сначала мы приняли эту последовательность символов за начало программы в машинных кодах, размещенной в Бейсике. В самом деле, эта последовательность не несла в себе никаких информационных признаков о командах Бейсика. Но в то же время, если бы это было начало подпрограммы в кодах, то первым должен был бы следовать символ REM, поскольку именно после него размещается подобного рода код. Это заставило нас более внимательно проанализировать данную строку и секрет был быстро разгадан.

Естественно, эта комбинация свидетельствовала о начале подпрограммы в машинных кодах, но не думайте, что она была равнозначна команде

```
RANDOMIZE USR 0
```

Ноль в данном случае был ширмой, за которой скрывалось истинное значение числа, указывающего адрес начала программы в машинных кодах. Этот прием с подменой значений достаточно подробно описан нами в главе 11 данного пособия.

4.2. Защита, основанная на использовании вариаций числа PI вместо цифровых констант.

В главе I были описаны специальные POKE которые применяются в различных целях в том числе и для защиты от остановки командой BREAK во время работы Вашей программы. Таких методов существует несколько, но общие критерии их применения аналогичны - Вам необходимо изменить значение одной из системных переменных и тогда после нажатия BREAK (если Вы ввели это изменение до нажатия данной клавиши), у Вас не произойдет остановки программы, а произойдет сбой в работе компьютера и машина либо зависнет, либо произойдет сброс системы, аналогичный нажатию кнопки RESET.

Но с тем, чтобы скрыть от начинающего "хаккера", какую именно системную переменную Вы меняете и какое значение Вы туда засылаете, можно использовать следующий прием.

В частности, Вам необходимо изменить содержимое ячейки 30000, сделав его равным 150.

Традиционно данная команда выглядела бы следующим образом:

```
10 POKE 30000, 150
```

Но ваша задача запутать "хаккера". Вместо чисел желательно использование каких-либо переменных, причем таким образом, чтобы они не описывалась в этой же программе оператором LET.

Поясним вышеизложенное на примере, чтобы заменить адрес традиционным способом, нам пришлось бы вводить еще одну строку программы

```
10 LET A1 = 30000
20 POKE A1,150
```

Но этого можно избежать, если задать переменную A, подав команду с клавиатуры:

```
LET A = 30000
```

После этого значение переменной A будет занесено в память компьютера и мы сможем вызывать его по мере необходимости. Одно условие - это значение будет уничтожено, если Вы подадите команду RUN, т.к. эта команда полностью освежает буфер переменных. Для запуска Вашей программы необходимо использовать функцию GO TO. Следует отметить один нюанс - автозапуск программы по команде LINE N (со строки N) осуществляется функцией GO TO N, поэтому Ваше значение будет сохранено в памяти компьютера.

Другой аспект проблемы - это использование вместо числа 150 его кодового слова среди символов символьного набора компьютера. В частности, символу 150 соответствует графический код "G". Следовательно, если Вы используете команды:

```
LET A = 30000
10 POKE A, CODE "G"
```

- (используется символ графики), то это будет полностью аналогично первой строке программы в этой статье. Кроме функции CODE можно использовать и производные функции PI:

NOT PI - равноценно 0
SGN PI - равноценно 1
INT PI - равноценно 3

Значения, которые не находят эквивалента в производных PI могут быть образованы с использованием различных комбинаций, в частности:

```
6 = INT PI + INT PI
```

В заключение этой статьи натолкнем читателя на очень любопытную мысль. В частности, раз Вашей задачей является как можно более запутать "хаккера", то почему бы Вам не набрать вместо имени переменной (в нашем примере используется переменная A) имя, аналогичное одному из ключевых слов Бейсика. Смею Вас заверить, это будет иметь поразительный эффект. В частности, постарайтесь представить себе неопытного "хаккера", увидевшего на экране следующую комбинацию:

```
LET USR = 30000
10 POKE USR, SGN PI
```

Неправда ли, смотрится впечатляюще для тех, кто не знает, что здесь USR не ключевое слово, а имя переменной и набирается по буквам.

4.3 Ключевые слова Бейсика в "хэдере".

Одним из наиболее любопытных приемов защиты, шокирующим начинающих пользователей "СПЕКТРУМа", является использование управляющих кодов в хэдере. Но не менее интересным приемом является использование в хэдере ключевых слов Бейсика. Эта тенденция просматривается во многих программам и позволяет сэкономить и без того

скудное пространство хэдера с целью создания полноценного названия.

Поясним вышеизложенное на примере.

Тем, кто внимательно ознакомился с главой II данного пособия, наверняка известен тот факт, что в хэдере содержится всего 10 символов, отведенных под имя программы. При использовании управляющих кодов количество свободных символов сокращается. Однако, если Вам хочется иметь полноценную заставку - заголовок, то одним из методов, позволяющим совместить это желание с желанием использования управляющих кодов, является применение в хэдере ключевых слов Бейсика.

Естественно, ключевые слова не могут в полном объеме характеризовать Ваше название и вряд ли смогут передать всю гамму названий, набираемых отдельно по буквам. Но, в то же время, их применение создаст эффект высокой защищенности программы, а разного рода финты достаточно сильно действуют на непрофессионалов отталкивающим образом.

Рассмотрим любопытный пример - название программы "GAME OVER". Его можно было бы написать и по отдельной букве, но тогда это не дало бы возможности использовать в хэдере управляющие коды, которые в моем варианте уничтожают после загрузки на экране слово PROGRAM. Для экономии места здесь используется оператор Бейсика OVER. Среди остальных, известных мне названий программных файлов, приведу наиболее распространенные, использующие ключевые слова Бейсика:

"MVS FORMAT"
"MDS FORMAT"
"B.G. FORMAT" - используется Биллом Гилбертом.

Конечно, не все Ваши названия можно выразить подобным образом, но, в то же время, следует отметить, что можно использовать слова-синонимы, аналогичные ключевым словам Бейсика. В любом случае, возможность такой замены Вам следует иметь в виду.

Следует отметить, что использование ключевых слов Бейсика в хэдере несколько лет назад еще использовалось для защиты программ от копирования. Когда копировщик считывал название программы, то он не мог распознать символ ключевого слова Бейсика и это вызывало сбой в его работе. Однако, последние модели копировщиков уже избавлены от этого недостатка.

4.4 Мерцающий заголовок.

Многие из читателей наверняка пользовались программой COPY-COPY. И, вероятно, обращали внимание на тот факт, как интересно там организован хэдер.

После загрузки хэдера на экране начинают попеременно мигать символы COPY-COPY. Подобная комбинация не имеет принципиального значения для непосредственной защиты Бейсик-программ, но имеет косвенный эффект.

Организация защиты подобных программ воспринимается как очень хорошая, особенно среди начинающих "хаккеров". Этот косвенный эффект совместно с действительно хорошей защитой, основанной на приемах, описанных в этой работе, превратят Вашу программу в "неприступную" крепость для взломщиков.

Итак, вкратце рассмотрим, как работает данный хэдер. Для тех,[1]

В программе COPY-COPY символы в заголовке имеют следующую последовательность :

[1] В оригинале пропущена строка (Прим. OCR)

```
18        - управление FLASH 1
255 COPY
6         - PRINT запятая 20
1         - управление INVERSE
255 COPY
0
8
```

В данном случае набор этих символов следует рассматривать, как определенную программу, предписывавшую компьютеру определенные действия. Рассмотрим, какие команды будет выполнять компьютер в данном случае.

Теоретически наш хэдер должен был бы появиться на экране (имеется в виду название программного файла) в виде печатных символов, но в данном случае здесь нет таковых. Первая пара символов 18 и 1 - это под управление FLASH и соответственно значение, устанавливающее режим мерцания. Далее следует символ COPY, который распечатывается на экране и начинает мерцать. После этого следует управляющий код 6 - т.н. PRINT-запятая, которая перемещает курсор в следующую половину экрана, аналогично запятой, использующейся в операторе PRINT. Следующая пара символов устраивает значение INVERSE - в данном случае включает инверсный режим. Теперь мерцание второго символа "COPY" будет находиться в противофазе с мерцанием первого, что и создает эффект попеременного мерцания.

Чтобы создать такой хэдер на компьютере, Вам необходимо создать строку выгрузки в Вашей программе. В ней должны находиться операторы SAVE "название" и т.п. Причем вместо управляющих кодов должны стоять пробелы. После этого Вам надо получить дамп памяти по одной из предложенных в главе 2 программ и определить местонахождение заголовка. После этого Вы засылаете в эту область памяти последовательность цифр, которая приведена выше. Естественно, приноровившись, Вы сможете создавать и более оригинальные комбинации, в частности, например, попробуете убрать слово PROGRAM или разместить заголовок в другой точке экрана. Для этого Вам необходимо внимательно изучить раздел "Управляющие коды" главы II данной работы.

Поскольку подобные коррективы в хэдере не имеют принципиального значения, мы не будем останавливаться на заголовке другого цвета и в другом месте экрана (отличном от привычного). Надеемся, что читатель сам сможет сделать это.

4.5 Метод защиты, основанный на смещении программ в кодах при попытке взлома программ.

Если читатель внимательно изучил все материалы, изложенные выше, то он приобрел достаточно высокий уровень подготовки для защиты своих программ. Тем не менее, поработав с ними длительное время, он начинает осознавать, что многие из приемов, применяемых им, достаточно широко известны в сфере программистов для "ZX SPECTRUM".

Возникает необходимость создания новых приемов защиты, никем до этого не применявшихся, чтобы обеспечить надежность защиты собственных программ. К этой цели можно идти двояко:

1) Создавая свои сугубо оригинальные приемы;

2) Создавая новые комбинации из уже известных приемов, улучшающие качество и надежность систем защиты;

Первый путь, безусловно, более надежен, но он требует виртуозного знания всех тайников компьютера, что требует, в свою очередь, достаточно целенаправленной работы с компьютером в течение длительного времени. Большинство же читателей вряд ли обладают возможностью длительно и целенаправленно изучать компьютер и поэтому мы предлагаем им избрать второй подход. Если первый подход хорош только для опытного, квалифицированного специалиста, то второй может подойти любому пользователю, внимательно изучившему данную статью. Второй метод включает в себя комбинирование изложенных здесь приемов с целью повышения эффективности защиты. На первый взгляд выгода от такого подхода не столь очевидна, однако не стоит забывать, что порой

интересная комбинация уже известных методов может иметь любопытный косвенный эффект. В качестве примера рассмотрим комбинацию из известных Вам методов совмещения в Бейсик-строке машинных кодов и метода защиты от листинга с использованием управляющих кодов.

Когда мы с Вами рассматривали совмещение с Бейсиком машинных кодов, то было отмечено, что удобнее всего для Вас использовать первую строку программы для непосредственного совмещения, что связано с правильным нахождением адреса начала загрузки.

Безусловно для Вас такой метод является самым удобным, но не менее удобным он является и для взломщика. Ведь фактически в такого рода программах ему даже не приходится искать адрес начала подпрограммы в кодах - первая строка Бейсика начинается с фиксированного адреса оперативной памяти, на который указывает системная переменная PROG и без подключенной периферии и умышленных изменений значение, находящееся в этой паре байтов указывает на адрес 23755. Соответственно, для нахождения адреса размещения такой подпрограммы достаточно к этому значению прибавить 5 (эти 5 байтов включают в себя номер и длину строки - по два байта на каждый параметр соответственно - и код оператора REM). Кроме этого значение адреса начала подпрограммы можно узнать из значения, стоящего после RANDOMIZE USR или его вариаций.

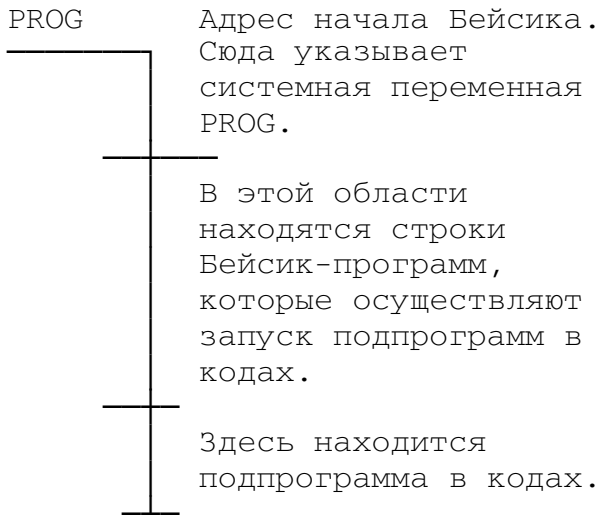
Одним из выходов является создание "плавающей" подпрограммы в кодах и задание "фальшивого" адреса старта данной подпрограммы.

Рассмотрим более подробно каждый из методов.

Как Вам уже вероятно известно, Бейсик-интерпретатор размещает строки программ последовательно. Т.е. если у Вас есть строки с номерами 5 и 10 и Вы хотите ввести строку с номером 7, то интерпретатор Бейсика разместит эту строку между строками 5 и 10. Когда он будет делать это, то строка с номером 10 переместится в более старшие адреса области ОЗУ, причем ее старое местоположение будет отличаться от нового на длину строки 7.

Теперь представьте, что строка с номером 10 - это строка, в которой расположена наша подпрограмма в кодах, а строки 5 и 7 на начальном этапе отсутствуют. Таким образом, после формирования данной строки с машинными кодами после оператора REM нам необходимо сместить ее в верхние адреса ОЗУ. Для этого все команды управления на Бейсике размещаем в строках, находящихся выше строки с подпрограммой в машинных кодах. (Для того, чтобы найти адрес начала подпрограммы в кодах, Вам придется использовать метод распечатки дампинга памяти, который будет подробно рассмотрен в конце этой статьи).

Если Вы начнете придерживаться изложенных выше рекомендаций, то можно сказать, что первый недостаток нам удалось преодолеть. В самом деле, теперь у Вас нет фиксированного адреса, с которого начиналась бы подпрограмма в кодах и этот адрес зависит от длины строк Бейсика, в которых бы осуществлялся запуск данной программы.



Вторая часть нашей программы будет создаваться временно, для получения дампинга и будет располагаться после подпрограммы в кодах.

Теперь перейден ко второй части проблемы. Нам необходимо скрыть адрес начала подпрограммы в кодах. Здесь мы используем комбинирование уже известных методов для создания очень интересного эффекта.

В целях более полной защиты, применим двойную защиту, которая при правильном использовании превратится в тройную защиту. Для этого нам придется, во-первых, использовать управляющие коды для слияния цветов INK и PAPER с целью сокрытия информации, а во-вторых, создать фальшивый адрес старта подпрограммы в кодах с использованием управляющего кода 14. Оба эти метода подробно описаны в главе 2 и поэтому Вам необходимо подробно ознакомиться с ними. Ниже будет приведен лишь алгоритм создания защиты и поэтому, если Вы не уяснили себе информацию, изложенную там, Вам будет достаточно сложно сориентироваться в происходящих процессах.

Итак, для начала Вам необходимо полностью оформить аппарат Бейсика, т.е. создать строки программы, которые бы выполняли необходимую Вам операцию. Естественно, все они должны иметь номер меньший, чем строка, в которой находится подпрограмма в кодах. Среди команд, помещенных в этих строках, обязательно должна быть команда запуска подпрограммы в кодах, причем номер, на который указывает RANDOMIZE USR (выгоднее применять разновидность этой команды - раздел 4.1) может быть произвольным. Обязательным условием является лишь то, чтобы он содержал не менее 5 знаков (и не более), т.е. например, запуск осуществлялся бы командой RANDOMIZE USR 25000. Это связано с тем, что при замене этого номера на истинный, который мы будем находить в следующей операции нашего алгоритма, программа в кодах может сместиться, из-за разного количества символов в номере.

После того, как Вы это сделали, нам необходимо будет зарезервировать место под управляющие коды, используя оператор REM. Всего у нас будет использоваться 4 байта INK CONTROL, значение цвета INK, PAPER CONTR и значение цвета PAPER.

Для этого в первой идущей в Вашей программе строке необходимо установить REM и три произвольных символа. Оператор REM устанавливается перед первым идущим в строке оператором.

После того, как мы установим точное расположение подпрограммы в кодах, мы заменим эти зарезервированные 4 байта на управляющие коды.

Следующим шагом мы как раз и установим точное местонахождение программы. Для этого в самом конце Бейсика разместим программу распечатки дампа памяти.

```
9997 FOR I = 23755 TO 64000
9996 PRINT I; TAB 7; PEEK I; TAB 12; CHR PEEK I
9999 NEXT I
```

Сразу же следует оговориться, что программа может останавливаться по ошибке

INVALID COLOR - неправильный цвет

NUMBER TOO BIG - большое целое число и т.п.

В этом случае Вам необходимо подать команду с клавиатуры:

```
NEXT I
```

и распечатка продолжится.

После того, как Вы стартуете данную программу командой GOTO 9997, Вы получите распечатку в следующем формате:

Номер ячейки памяти	десятичное значение содержащееся там	символьное значение содержащееся в этой ячейке памяти

Внимательно ее анализируя, Вы более подробно изучите структуру своей Бейсик-строки. Просматривая таким образом содержание памяти в области Вашей Бейсик-программы, Вам необходимо найти и записать номера следующих ячеек памяти:

1) Номер ячейки, где находится оператор REM, после которого следуют 4

зарезервированных под управляющие коды байта.

2) Номер ячейки, в которой содержится первая цифра числа, находящегося после RANDOMIZE USR.

3) Адрес ячейки, с которой начинается Ваша подпрограмма в кодах.

После того, как эта информация Вами получена, необходимо занести в строку, запускающую Вашу программу и содержащую RANDOMIZE USR, правильный адрес расположения подпрограммы в кодах, полученный в пункте 3.

Когда это Вами сделано, Вам необходимо это число "фальсифицировать".

Как Вам уже вероятно известно, в "ZX SPECTRUM" в памяти числа хранятся дважды - первый раз значение, которое печатается на экране, а второй раз - в пятибайтном формате, расположенное после кода 14. В данном случае нам необходимо изменить именно первое значение, которое будет распечатываться на экране.

Первый байт этого значения расположен по адресу, который мы получили в пункте 2. Вы можете убедиться в этом снова, запустив программу дампинга и увидев в этом же месте новое значение числа. Для того, чтобы изменить это значение, Вам необходимо заслать в эти ячейки памяти значения ASCII кодов "фальшивого" числа с помощью оператора POKE.

Следующим пунктом нам необходимо скрыть листинг данной программы. Для этого сделаем одинаковым цвет чернил и бумаги с помощью управляющих кодов. Например, мы хотим, чтобы это был белый цвет. Тогда нам необходимо последовательно заслать в ячейки памяти, адреса которых мы получили в пункте 1, следующие значения:

16 - изменение INK
7
17 - изменение PAPER
7

После того, как Вы выполните это командой POKE, на экране уже невозможно будет получить листинг.

Теперь Ваша программа защищена от просмотра командой LIST. Осталось удалить последние строки, осуществлявшие дампинг памяти. Сделайте это, подав команды

```
9997 ENTER
9998 ENTER
9999 ENTER
```

Теперь рассмотрим, какие новые качества получила наша защита после такой комбинации.

Если мы вызовем первую строку программы, в которой заложены управляющие коды, для редактирования и уничтожим эти коды командой DELETE, то теоретически мы увидим адрес начала программы в кодах - практически же Вы знаете, что это значение будет неправильным. Но, даже если Вам и удастся определить правильный адрес старта данной подпрограммы в кодах, просмотрев через PRINT PEEK то, что стоит после кода 14, Вам все равно не удастся ее правильно дисассемблировать, поскольку Вы уничтожили управляющие коды командой DELETE, то есть уменьшили размер первой строки на 4 байта и, тем самым, сместили адрес начала данной подпрограммы.

Таким образом, мы видим, что совмещение хорошо известных Вам приемов может приводить к совершенно новым результатам и порождать любопытные побочные эффекты, которые в данном случае тоже используются для защиты Вашей программы.

* * *

Том 2. "Техника взлома компьютерных программ ZX-SPECTRUM. "

Глава 1. Введение

Компьютеры "ZX-SPECTRUM" были разработаны в Англии и поэтому многие пользователи в нашей стране сталкиваются с трудностями освоения программ. Это касается не только проблем языкового барьера. Трудности, связанные со слабым знанием английского языка в большинстве случаев решаются за счет его изучения самим пользователем, потому что без определенного уровня достаточно проблематично осваивать интересное программное обеспечение. Но, кроме барьера языкового, существует и барьер программной защиты. Ведь многие пользователи, проработав определенное время с программой, желают залезть к ней в "нутро" и осуществить какие-либо изменения. Это могут быть изменения, направленные на русификацию программы, изменения, направленные на адаптацию программы под требования пользователя или же просто просмотр с целью изучения приемов программирования. Кроме этого, существует проблема с копированием некоторых защищенных программ, копирование которых невозможно осуществить без определенного уровня подготовки, а также исследования конкретной структуры защиты данной программы.

В преодолении этих трудностей Вам поможет изучение техники взлома защищенных программ, изложение которой и преследует своей целью данное пособие. Взлом компьютерных программ занятие достаточно увлекательное и, в то же время, многие используют это занятие в неблаговидных целях. В частности, очень многие взломщики оставляют после себя определенные надписи, т.е. используют взлом программ для саморекламы, но, как Вы скоро поймете, это не является самым интересным. Куда более любопытным нам представляется использование взлома компьютерных программ для повышения Вашей техники программирования, а также более детального изучения имеющегося у Вас компьютера, превращения его из обыкновенной игрушки в серьезный инструмент для решения Ваших задач.

Общие рекомендации.

Если Вы внимательно изучили т. 1, то теперь Вам известны основные принципы защиты компьютерных программ, и, несмотря на то, что техника защиты постоянно совершенствуется, все, что создается отвечает известному Вам принципу: "Все новое - это хорошо забытое старое". А это значит, что все новые приемы защиты являются модернизацией уже известных.

Но это происходит не всегда. Иногда появляются исключительно новые приемы, которые нельзя классифицировать ни по какой старой схеме. К счастью, это скорее исключение, чем правило. А это говорит о том, что Вам следует рассматривать защиту новой программы как набор уже известных Вам приемов или же их комбинацию, пока не станет ясно, что это что-то новое.

Второе, о чем Вам необходимо серьезно задуматься, так это о том, с какой целью Вы взламываете программу. Наиболее распространенным, на наш взгляд, является случай, когда Вы желаете изучить новые приемы программирования или же адаптировать программу для себя. Эти случаи предусматривают вариант, когда Вам придется досконально изучить содержание программы, чтобы понять принцип ее работы и внести необходимые изменения.

После того, как Вы определились в цели взлома, Вам необходимо подумать о том, как наиболее рационально загрузить программу, чтобы наиболее быстрым способом обезвредить защиту.

Для того, чтобы иметь хоть маломальское представление о типе защиты, примененном в данной программе, Вам необходимо загрузить ее и нажать клавишу BREAK (это необходимо узнать для того, чтобы впоследствии ликвидировать защиту от BREAK одним из предложенных далее способов) и если программа не "зависла", то испробовать, что получается при нажатии клавиши LIST.

Итак, Вы определились в целях взлома и осуществили предварительную подготовку. Мы хотим искренне надеяться, что Ваши цели будут такими, какими представляем их мы или аналогичными.

Структура фирменной программы.

Когда Вы загружаете какую-либо игровую программу, то перед тем, как пойдет загрузка, Вы набираете с клавиатуры команду LOAD "".

Подобная комбинация говорит о том, что первым будет загружаться Бейсик-файл, а уж дальнейшая загрузка пойдет в соответствии с командами данного Бейсик-файла. Рассмотрим более подробно, как происходит загрузка этого Бейсик-файла (фактически данная информация относится к загрузке любого файла в компьютере).

В течение первых 1-2 секунд считывания мы видим на экране широкие красные и синие полосы, а также слышим длительный однотонный звук. Это так называемый пилот (или пилоттон, или просто тон), который позволяет компьютеру синхронизироваться с сигналом с ленты, с которой он будет считывать программу, что обеспечивает надежность ввода и вывода кодов в память компьютера. (Как Вам уже вероятно известно, все программы состоят из кодов. Бейсик-программу также можно представить, как последовательность кодов), затем появляются мерцающие тонкие желтые и синие полоски, свидетельствующие о том, что компьютер считывает в память байты информации. Первый раз эти желто-синие полосы мерцают кратковременно, поскольку компьютер считывает в память заголовок. Этот заголовок (так называемый "хэдер") и содержит в себе всю информацию о последующем загружающемся блоке кодов и имеет размер всего 17 байтов. Поскольку у нас считывается Бейсик-файл, то появляется надпись "PROGRAM", но возможно появление и других надписей, в частности:

BYTES - байты (коды);

CHARACTER ARRAY - символьный массив;

NUMBER ARRAY - числовой массив.

После этой надписи будет стоять название программы.

Теперь после небольшого перерыва начнется второй пилоттон, а после него считывается собственно сама программа.

Если мы подавали команду LOAD "", то у нас загружался Бейсик-файл и после загрузки компьютер поступает в соответствии с данными, поступившими из хэдера. Для Бейсик-программы это может быть либо автостарт программы со строки n (если программа была записана на магнитофон командой:

```
SAVE "название " LINE
```

либо остановка в ожидании команды RUN.

В большинстве фирменных программ происходит автостарт и, в соответствии с командами Бейсика, выполнение программы продолжается, что позволяет осуществить дальнейшую загрузку блоков. Но теперь Вам достаточно очевидно, что если взломать Бейсик-файл и внести в него соответствующие изменения, то Вы сможете самостоятельно и целенаправленно отслеживать процесс загрузки и управлять им.

Именно Бейсик-файл наиболее серьезно защищается производителями и именно на нем Вам следует акцентировать свое внимание, поскольку именно там содержится первичная информация о процессе загрузки последующих блоков, изменив которую Вы сможете воздействовать на дальнейший процесс загрузки.

Естественно, следующим этапом нашим совместных исследований должно стать рассмотрение приемов вскрытия данного Бейсик-файла.

Небольшая историческая справка.

Поскольку данный раздел называется "Структура фирменных программ", мы постараемся Вам изложить в общих чертах эту структуру, несмотря на то, что могли только ограничиться информацией о структуре, необходимой для взлома.

Итак, как Вы уже теперь поняли, загружающийся в компьютер файл состоит из двух частей: хэдера и собственно загружающейся после него программы (для удобства будем впоследствии называть эту совокупность блоком кодов). В хэдере содержится вся

информация о загружающемся после него блоке.

Таким образом, структуру первых программ для "ZX SPECTRUM" можно представить, как совокупность Бейсика и машинных кодов. Машинные коды составляли саму программу, в то время как Бейсик осуществлял загрузку этих кодов в память компьютера, после чего осуществлялся их запуск из Бейсика с использованием функции USR. Таким образом, Бейсик состоял из совокупности команд LOAD "" CODE которую завершала функция USR (Автор не включает сюда все разнообразие текстовой информации, которая выводилась на экран с использованием оператора PRINT). Взлом таких программ с целью изучения вообще являлся личным делом. В последовательность команд Бейсика вводится оператор STOP между совокупностью команд LOAD "" CODE и функцией USR, после чего программа в необходимом для Вас месте останавливалась, и Вы могли беспрепятственно исследовать как структуру, так и содержание самой программы. Фактически мы можем сказать, что эти программы не были защищены, что трудно сказать о более поздних фирменных разработках. Несмотря на все разнообразие появившихся типов защит, мы можем выделить два направления защиты в области загрузки. Это загрузка с хэдером и загрузка без хэдера.

Как действует спектрумовская загрузка с хэдером, нам уже известно. Был, правда, вариант защиты с нестандартный хэдером, который отрабатывался специальной программой. Для загрузки программ этот метод не нашел широкого применения и был вскоре практически полностью вытеснен методом безхэдерной загрузки, однако впоследствии программисты вновь обратились к нему, когда для "Спектрума" вышла масса программ с догружающимися уровнями. Именно при догрузке уровней этот метод получил наибольшее распространение.

Второй метод - т. н. загрузка без хэдера применяется практически во всех фирменных программах последних годов выпуска. У него существует множество разновидностей, но фактически все они сводятся к созданию новой системы хранения данных о вводящимся в компьютер блоке кодов, то есть фактически, ту информацию, которую раньше передавал в компьютер хэдер, теперь передает специальная программа в кодах. Разновидностями же этого метода безхздерной защиты являются обращение либо ко встроенной в ПЗУ подпрограммы загрузки, либо обращение к одной из загружаемых в ходе работы программы подпрограмме, так называемой программе-загрузчику последующих блоков. Следует отметить, что применение своей программы-загрузчика преследует несколько целей:

- создание необычных эффектов в ходе загрузки;

- перемещение загруженного блока кодов в другое место памяти, чтобы обеспечить защиту;

- выполнение этих функции одновременно.

Наглядным примером первого типа являются программы MIKKI и KUNG FU второго типа GREEN BERET и третьего - BOMB JACK.

Разумеется, здесь перечислены не все цели, ради которых применяется конкретная программа-загрузчик собственного изготовления. Но в общем случае применение такой программы говорит о методе бесхэдерной загрузки.

Применение загрузки без хэдера становится возможным благодаря знанию встроенных процедур "ZX-SPECTRUM". В частности, чтобы вызвать встроенную подпрограмму загрузки, Вам необходимо внести в регистровые пары HL и BC соответственно адрес, с которого начинается загрузка кодов и их длину, а также соответствующий образом изменить содержимое аккумулятора, после чего вызвать подпрограмму в кодах.

(Очень подробно встроенные процедуры описаны в книге YAN LOGAN, FR. O'HARA "THE COMPLETE SPECTRUM ROM DISASSEMBLY", 1983, а также информацию о процедурах загрузки можно получить в ZX-РЕВЮ N5,6 1991г.)

Следует отметить, кроме всего вышеизложенного, что достаточно часто применяется комбинированная загрузка, когда наряду с блоками с хэдером загружаются блоки без хэдера.

Для того, чтобы читателю получить достаточно полное представление о структуре своей программы, рекомендуем загрузить ее в копировщик, который достаточно наглядно

покажет Вам из каких блоков состоит Ваша программа.

В копировщике ZX COPY 87, TF COPY и им аналогичным хэдер можно отличить достаточно легко потому, как именно в нем содержится информация о том, что это либо Бейсик, либо коды и т. д., а также само название файла. Разумеется, в копировщике Вы можете подробно изучить лишь стандартный хэдер, в то время как нестандартный подобному исследованию не поддается.

Структура хэдера.

Итак, теперь мы с Вами в общих чертах рассмотрели структуру фирменной программы. Конечно, каждая конкретная программа имеет свою структуру (в этом заключается ее оригинальность), но практически каждую из них можно классифицировать по способу загрузки - либо загружается файл с хэдером, либо загружающийся блок не имеет хэдера.

(Любопытно, что несколько лет назад появился оригинальный метод защиты программ от копирования. Он заключался в том, что в название программы наряду с обычными символами включались и ключевые слова Бейсика, либо управляющие коды. После загрузки этого "фальшхэдера" в копировщик, копировщик не мог распечатать эти символы и работа программы останавливалась с сообщением об ошибке. Великолепный комплект польских копировщиков ZX COPY 87, TF COPY, TF COPY 2, MR COPY и др. практически полностью свел на нет все усилия в данной области защиты. Интересным напоминанием этого направления защиты от копирования может служить программа GREEN BERET. Один из файлов которой длиной 17 байт как раз и должен был выполнять роль фальшхэдера.

Следует еще коротко сказать о том, как создатели копировальных программ вышли из положения. В последних разработках копировщиков вместо символов, не соответствующих ASCII кодам какой-либо литеры печатается вопросительный знак или иной заранее принятый символ.

Однако, несмотря на то, что метод, казалось бы, исчерпал себя, подробное его изучение может натолкнуть читателя на создание новых оригинальных разработок, базирующихся на этом или аналогичном принципах).

Теперь мы предлагаем Вам ознакомиться со структурой хэдера. Сделать это необходимо не только из чистого любопытства. Ведь невозможно создать что-то новое, не изучив досконально действующий старый образец.

Длина заголовка 19 байтов, а не 17, как написано в большинстве книг, но только 17 должны быть активны, поскольку "LOAD" (эта информация аналогична и для "SAVE") первый и последний байты определяют сами. Первый байт для заголовка всегда равен нулю, а последний - "байт четности" генерируется стандартной процедурой SAVE и потому нас не интересует.

Байт 2 содержит число, характеризующее тип записи, в зависимости от того, какое здесь содержится значение, компьютер определяет структуру файла, следующего за хэдером:

0 - это Бейсик-программа

1 - числовой массив

2 - массив символов

3 - блок кодов

Байты 3-12 содержат в себе имя программы

Байты 13 и 14 - длина основного блока (например для Бейсик программа это разность того, что содержится в системных переменных ELINE и PROG.

Байты 15 и 16 - начальный адрес загрузки кодов или номер строки автостарта для программ, написанных на Бейсике.

Байт 16 - для массива данных здесь располагается его имя в следующей форме:

Биты 0-4 - имя (от A=1 до Z=26)

Бит 5 - сброшен, если массив

Числовой бит 6 - активен, если массив

Строковый бит 7 - активен всегда

Байт 17 и 18 - длина Бейсик-программы, т. е. разность того, что содержится в системных переменных VARS и PROG.

Байт 19 - активизируется в ходе работы программ "LOAD" и "SAVE".

Для наглядности приведем схему заголовка (см. рис. 1):

Байты 1	2	3...12	13,14	15,16	17,18	19
Флаг	Тип	Имя	Длина	Старт	Длина для Бейсика	Четность
IX+...	0	1...10	11,12	13,14	15,16	17

Рис. 1 Структура "хэдера"

Поскольку хэдер анализируется интерпретатором с помощью индексной адресации, то внизу каждый байт дан как смещение относительно базового адреса, содержащегося в IX.

Небольшой комментарий по поводу некоторых байтов хэдера. В зависимости от считываемого блока байты 15 и 16 интерпретируются по-разному. В заголовках программ, написанных на Бейсике, эти байты содержат номер строки, с которой запускается программа, если она была записана с помощью оператора

```
SAVE "ИМЯ" LINE N
```

Если программа была записана без опции LINE и после считывания не запускается автоматически, то значение этого числа больше 32767. Любопытно, что одним из способов нейтрализации самозапускающихся программ на Бейсике является замена этих двух байтов на число большее 32767 (Подробно этот метод будет рассмотрен в одной из статей Главы 2 под названием "Блокировка автозапуска").

Байты 17 и 18 содержат число, указывающее длину самой программы на Бейсике, т. е. содержимое памяти от байта, указанного системной переменной PROG до байта, определяемого системной переменной VARS. Если мы от всей длины блока (байты 13 и 14) отнимем это число, то узнаем сколько байтов в этой программе занимают переменные Бейсика.

Теперь Вам достаточно хорошо известна структура хэдера и Вы уже представляете, чем является эта совокупность байтов. Однако, для того, чтобы исследовать хэдер более подробно, я рекомендую Вам использовать специальные программы, которые прозондируют загружающийся хэдер и выдадут информацию о нем.

Ниже в качестве примера приведена такая программа. Фактически она написана на Бейсике, но в своей работе обращается к процедурам в машинной коде. Программа загружает хэдер в определенное место памяти (которое, кстати, Вы можете изменить), после чего Бейсик-программа анализирует структуру хэдера и выдает всю информацию о ней на экран в виде письменных сообщений.

Сначала программа формирует процедуру в машинных кодах, используя блок DATA строки 30.

Фактически этот блок осуществляет запуск встроенной в ПЗУ подпрограммы-загрузчика, но делает это так, чтобы загружался только хэдер.

При загрузке хэдера в аккумуляторе процессора должен содержаться 0, а в регистре IX адрес, с которого начинает грузиться хэдер. Кроме этого, должен быть включен флаг переноса (флаг C регистра F).

```
1 REM TAPE EXAMINER
5 CLS:BEEP ,1,24:
      PRINT "LOAD A HEADER AND WAIT,PLEASE":
      PAUSE 150: BEEP ,1,24:CLS
7 RESTORE
10 CLEAR 32511
20 FOR A=32512 TO 32521:
       READ B:
```

```
        POKE A, B:
      NEXT A
30 DATA 175,55,221,33,16,127,205,86,5,201
40 LET B=32528:
      DEF FN A(X)=PEEK(B+X)*256*PEEK(B+X+1)
50 RANDOMIZE USR 32512
60 LET C=PEEK B
70 IF C>3 THEN GO TO 50
75 LET B$="":
      FOR A=B+1 TO B+10:
      LET B$=B$+CHR$ PEEK A:
      NEXT A
78 FOR A=10 TO I STEP -1
79 IF B$(A)=" "
      THEN LET B$=B$(TO A-1):
      NEXT A
80 IF B$=" " THEN LET B$=" "
85 PRINT "FILE NAME: ";INVERSE 1; B$
100 PRINT "FILE TYPE: "
110 GO SUB 1000+100*C
120 PRINT :PRINT
130 POKE B,255
140 GO TO 50
150 STOP
160 SAVE "TAPEXAN" LINE 5: CLS:
      PRINT "O. K. REWIND AND PLAY":
      VERIFY "TAPEXAN":
      CLS:
      PRINT "O.K. VERYFIED": STOP
1000 PRINT "PROGRAM"
1010 PRINT "TOTAL LENGTH: ":
      FN A(11); "BYTES"
1020 PRINT "PROGRAM LENGTH: ":
      FN A(15); "BYTES"
1030 IF FN A(13)>9999
      THEN PRINT "SAVED BY: "
      'TAB 5; "SAVE " " "; B$; " " " ":
      RETURN
1040 PRINT "SAVED BY:"
      'TAB 5; "SAVE " " ";
      B$; " " " LINE ";FN A(13)
1050 RETURN
1100 PRINT "NUMBER ARRAY"
1110 LET A$=" ":
      GOTO 1220
1200 PRINT "CHARACTER ARRAY"
1210 LET A$="$"
1220 PRINT "ARRAY LENGTH: ";FN A(11); "BYTES"
1230 LET D=PEEK (B+14)
1240 PRINT "ORIGINAL ARRAY NAME:";CHR$(64+32*(D/32-INT(D/32))) ;A$
1250 RETURN
1300 IF FN A(11)-6912 AND FN A(13)=16384 THEN PRINT "BYTES-SCREEN$": RETURN
1310 PRINT "BYTES"
1320 PRINT "START ADRESS: ";FN A(13)
1330 PRINT "LENGTH: ";FN A(11); "BYTES"
1340 RETURN
1350 CLEAR:
      SAVE "TAPEXAM" LINE 5:
      BEEP .1,24:
      PRINT AT 10,0: "O.K. REWIND AND PLAY TO VERIFY ":
      VERIFY "TAPEXAM":
      CLS:
      BEEP ,1,24:
      STOP
```

Листинг ассемблера программы в кодах, которая содержится в строке 30 DATA.

```
7F00  XOR A             Обнуление аккумулятора.
7F01  SCF               Принудительное включение флага переноса.
7F02  LD IX,7F10        В регистр IX поместили адрес с которого
                        начнется размещение хэдера в памяти компьютера.
7F06  CALL 0556         Вызов загружающей процедуры ПЗУ.
7F09  RET
```

Глава 2. Блокировка автозапуска.

Введение.

После внимательного ознакомления со структурой фирменных программ, читателю становится очевидно, что для взлома программы необходимо, прежде всего, исследовать Бейсик-файл (первичный загрузчик), этот вывод стал очевидным и для фирм-производителей программного обеспечения. Вот почему многие методы защиты направлены в основном на защиту Бейсик-файла.

Однако, следует заметить, что наряду с техникой программирования совершенствовалась и техника взлома, в этой области существует очень мало информации, но мой вывод подтверждает тот факт, что вскрытие хаккерами программ можно найти среди игр самых разнообразных годов выпуска.

В этой главе рассмотрены три концепции блокировки автозапуска программ. Естественно, каждую из этих концепций можно использовать независимо от других. Можно, однако применить и комплексный подход. Но основным является то, что каждый их предложенных Вашему вниманию подходов имеет свои достоинства, но он в то же время имеет и свои недостатки.

Мы думаем, что ознакомившись с этими концепциями более подробно. Вы выберете для себя ту, которая наилучшим образом удовлетворяет Вашим потребностям.

2.1 Загрузка Бейсика через блок кодов.

После того, как читатель ознакомился со структурой хэдера, ему становится ясно, что фактически между заголовком Бейсика и заголовком кодов разница небольшая. А разницы между непосредственным файлом Бейсика и непосредственным файлом кодов нет вообще никакой.

Примечание: В данном случае под файлом подразумевается та часть программы, которая загружается после хэдера. На самом деле отличия есть и заключаются они в следующем:

- Бейсик-файл загружается в память компьютера начиная с адреса, задаваемого системной переменной PROG, блок кодов же загружается с адреса, указанного в хэдере.

- Бейсик-файл обрабатывается интерпретатором как определенная последовательность символов-кодов Бейсика. Блок кодов обрабатывается процессором как последовательность кодов Z80.

На этом свойстве этих файлов и основан данный метод взлома. В самом деле, нам ничего не стоит обмануть компьютер - он думает, что загружает в память файл кодов, а на самом деле загружает файл Бейсика и при этом он не сможет установить никакой ошибки. Поскольку при той проверке, которую осуществляет "ZX SPECTRUM", такую ошибку установить просто-напросто невозможно.

Итак, рассмотрим этот метод более подробно.

Нам необходимо изучить структуру Бейсик-файла, при этом, чтобы избежать запуска команд и процедур, осуществляющих защиту, нам необходимо загрузить программу в память без автозапуска, т. е. так, чтобы она не стартовала, если в ней предусмотрена такая возможность.

Для того, чтобы решить данную задачу, нам необходимо прежде всего узнать длину Бейсик-файла. Это можно сделать, воспользовавшись вышеприведенной программой для анализа хэдера. То же можно сделать и воспользовавшись программой-копировщиком.

Обычно в копировщике мы можем наблюдать следующую картину (рис. 2):

Имя программы	Тип программы	Строка автозапуска	Длина
xxxxxxxxxxx	P	10	423
yyyyyyyyyyy	C	35000	22250
zzzzzzzzzzz	C	61200	1800

Рис. 2

После имени программы обычно появляется сообщение о типе данной программы: PROGRAM, CODE и т. п.

Строка автозапуска появляется только для программ Бейсика (для кодов же появляется номер ячейки памяти, с которого начинается загрузка). В разделе "длина" появляется длина загружаемого вслед за хэдером файла, это не длина хэдера.

Нам в нашей работе понадобится "реальная длина" (В большинстве случаев это значение не совпадает со значением "длина" в хэдере, но бывают и исключения). Кроме этого Вам необходимо запомнить номер строки автозапуска.

Следующим этапом нашей работы является создание нового хэдера, благодаря которому мы сможем загрузить нашу программу в произвольное место памяти. Как Вам уже вероятно известно, компьютер не в состоянии отличить файл Бейсика от файла кодов. Этим мы с Вами и воспользуемся. Мы создадим хэдер, который якобы должен загружать коды с определенного места памяти, но вместо файла кодов мы загрузим туда файл Бейсика для последующего изучения.

Для создания такого хэдера нам понадобится значение "реальной длины" файла, взятое из кодировщика. Теперь запишем на ленту этот хэдер кодов командой с клавиатуры: save "имя файла" code 30000, реальная длина

В графе "имя файла" Вам необходимо будет написать произвольное название файла по Вашему выбору, но не более 10 символов. Значение 30000 после CODE означает, что файл кодов, загружающийся после этого хэдера, будет загружаться с ячейки памяти 30000 (это же касается и записи информации т.е. после подачи данной команды, на ленту, должен будет записаться блок длиной "реальная длина", начиная с ячейки памяти 30000. Однако, поскольку нам необходим только хэдер, мы не будем дожидаться выгрузки на ленту файла кодов) и выключим магнитофон.

Теперь мы имеем следующие магнитофонные блоки:

Исходный хэдер (1) Бейсика	Файл Бейсика (1')	Специальный хэдер кодов (2)
Этот хэдер мы имели вместе с блоком Бейсик программы, длина хэдера 19 байт.	Этот файл описывался исходным хэдером и загрузить в компьютер автозапуском.	Этот хэдер мы создали таким образом, чтобы он описывал файл Бейсика как файл кодов. Длина=19 байт

Теперь нам необходимо непосредственно осуществить подмену. Для этого подадим команду с магнитофона:

```
LOAD "" CODE
```

после которой загрузим созданный нами хэдер. После этого хэдера Вам необходимо загрузить файл Бейсика (1'), однако этому файлу не должен предшествовать исходный хэдер (1). (В данном случае этот исходный хедер (1) не загружается вообще). Наилучшее техническое решение, позволяющее точно найти файл Бейсика (1'), пропустив исходный хэдер, является временное выдергивание шнура загрузки из магнитофона совместно с использованием клавиши ПАУЗА магнитофона (если такая имеется).

Итак, Вы осуществили подмену и загрузили Бейсик-файл под видом кодов. Теперь Вашей задачей является просмотреть этот файл и изучить его структуру.

(Продолжение следует.)

40 ЛУЧШИХ ПРОЦЕДУР

Продолжение.
Начало см. с. 17-28, 61-70.

7. ПРОЦЕДУРЫ ОБРАБОТКИ ПРОГРАММ.

7.1 Удаление блока программы.

Длина: 42
Количество переменных: 2
Контрольная сумма: 5977
Назначение: Эта программа удаляет блок BASIC-программы, находящийся между строками, определенными пользователем.
Переменные:
Имя - start line no
Длина - 2
Адрес - 23296
Комментарий: Номер первой строки, подлежащей удалению.
Имя - end line no
Длина - 2
Адрес - 23298
Комментарий: Номер последней строки, подлежащей удалению.

Вызов программы:

```
RANDOMIZE USR адрес
```

Контроль ошибок:
Если имеют место следующие ошибки, то процедура останавливается без удаления строк BASIC-программы:
- если последний номер строки меньше, чем первый номер строки;
- если между этими двумя строками нет программы на БЕЙСИКе;
- если один из номеров строк или оба равны 0.

Комментарий:
Эта программа довольно медленна для удаления большого блока программных строк, но, тем не менее, работать с ее помощью все же удобнее, чем удалять строки вручную.

Не вводите номера строк больше, чем 9999.

ЛИСТИНГ МАШИННЫХ КОДОВ

```
МЕТКА       АССЕМБЛЕР          ЧИСЛА ДЛЯ ВВОДА

            LD HL,(23296)      42    0     91
            LD DE,(23298)      237   91    2     91
            LD A,H             124
            OR L               181
            RET Z              200
            LD A,D             122
            OR E               179
            RET Z              200
            PUSH DE            213
            CALL 6510          205   110   25
            EX (SP),HL         227
```

```
            INC HL          35
            CALL 6510       205   110   25
            POP DE          209
            AND A           167
            SBC HL,DE       237   82
            RET Z           200
            RET C           216
            EX DE, HL       235
NXT_CH      LD A,D          122
            OR E            179
            RET Z           200
            PUSH DE         213
            PUSH HL         229
            CALL 4120       205   24    16
            POP HL          225
            POP DE          209
            DEC DE          27
            JR NXT_CH       24    243
```

Как она работает:

В пары регистров HL и DE загружаются начальный и конечный номера строк соответственно. Эти значения проверяются и, если одно из них или оба равны 0, программа возвращается в BASIC.

Затем вызывается подпрограмма ПЗУ по адресу 6510 - она возвращает адрес в памяти компьютера, с которого начинается первая строка. Эта же подпрограмма затем вызывается снова для определения адреса символа, стоящего после 'ENTER' в конечной строке.

В пару регистров HL помещается разность этих двух адресов и, если это значение равно 0 или отрицательно, программа возвращается в BASIC.

Содержимое пары регистров HL копируется в DE для использования DE в качестве счетчика. Если счетчик равен 0, то работа процедуры завершается, а если нет, то вызывается подпрограмма ПЗУ, расположенная по адресу 4120, которая удаляет один символ. После этого - возврат к 'NXT_CH'.

7.2 Обмен токена.

Примечание: под "токеном" подразумевается любое ключевое слово (команда, функция) из "словаря" БЕЙСИКа, которое рассматривается данной программой (как и интерпретатором компьютера) в виде определенного кода.

Длина: 46

Количество переменных: 2

Контрольная сумма: 5000

Назначение:

Меняет любое вхождение заданного токена в БЕЙСИК-программе на другой токен (например, все операторы PRINT могут быть изменены на LPRINT).

Переменные:

Имя - chr old

Длина - 1

Адрес - 23296

Комментарий: Код заменяемого токена.

Имя - chr new

Длина - 1

Адрес - 23297

Комментарий: Код замещающего токена.

Вызов программы:

```
RANDOMIZE USR адрес
```

Контроль ошибок:

Если БЕЙСИК-программы нет в памяти или один из двух заданных токенов имеет код меньше 32, то процедура возвращается в БЕЙСИК.

Комментарий:

Эта процедура очень быстра, но чем длиннее БЕЙСИК-программа, тем медленнее она работает.

ЛИСТИНГ МАШИННЫХ КОДОВ

```
METKA       ACCEMБЛЕР          ЧИСЛА ДЛЯ ВВОДА

            LD BC,(23296)      237   75    0     91
            LD A,31            62    31
            CP B               184
            RET NC             208
            CP C               185
            RET NC             208
            LD HL,(23635)      42    83    92
NXT_CH      INC HL             35
            INC HL             35
            INC HL             35
CHECK       LD DE,(23627)      237   91    75    92
            AND A              167
            SBC HL,DE          237   82
            RET NC             208
            ADD HL,DE          25
            INC HL             35
            LD A,(HL)          126
            INC HL             35
            CP 13              254   13
            JR Z,NXT_CH        40    237
            CP 14              254   14
            JR NZ,COMP         32    3
            INC HL             35
            JR NEXT_CHR        24    230
COMP        DEC HL             43
            CP C               185
            JR NZ,CHECK        32    229
            LD (HL),B          112
            JR CHECK           24    226
```

Как она работает:

В регистры B и C загружаются новый и старый токены соответственно. Если любой из токенов имеет код меньше, чем 32, то программа возвращается в BASIC.

В пару HL заносится адрес начала БЕЙСИК программы. Пара регистров затем увеличивается и сравнивается с адресом области переменных. Если HL не меньше чем адрес начала области переменных программа возвращается в БЕЙСИК.

Пара HL увеличивается указывая на следующий символ. Код этого символа загружается в аккумулятор и HL увеличивается вновь. Если значение в аккумуляторе равно 13 или 14 (ENTER или NUMBER) подпрограмма возвращается к 'NXT_CH' а HL увеличивается указывая на следующий символ. Если аккумулятор не содержит 13 или 14, хранимое значение сравнивается с 'chr old'. Если пара найдена, этот символ замещается на 'chr new'.

Затем возврат к проверке ('CHECK') на конец обрабатываемой программы.

7.3 Удаление REM строк

Длина: 132

Количество переменных: 0

Контрольная сумма 13809

Назначение:

Эта программа удаляет все комментарии из БЕЙСИК программы.

Вызов программы:

```
RANDOMIZE USR адрес
```

Контроль ошибок:

Если в памяти нет БЕЙСИК программы процедура вернется в БЕЙСИК без каких либо действий.

Комментарий:

Процедура ПЗУ, которая используется для удаления символов, выполняется не очень быстро и поэтому процедура "Удаление REM" работает медленно.

ЛИСТИНГ МАШИННЫХ КОДОВ

```
МЕТКА       АССЕМБЛЕР          ЧИСЛА ДЛЯ ВВОДА

            LD HL,(23635)      42    83    92
            JR CHECK           24    31
NEXT_L      PUSH HL            229
            INC HL             35
            INC HL             35
            LD C,(HL)          78
            INC HL             35
            LD B,(HL)          70
NXT_CH      INC HL             35
            LD A,(HL)          126
            CP 33              254   33
            JR C,NXT_CH        56    250
            CP 234             254   234
            JR NZ,SEARCH       32    26
            INC BC             3
            INC BC             3
            INC BC             3
            INC BC             3
            POP HL             225
DEL_L       PUSH BC            197
            CALL 4120          205   24    16
            POP BC             193
            DEC BC             11
            LD A,B             120
            OR C               177
            JR NZ,DEL_L        32    248
CHECK       LD DE,(23627)      237   91    75    92
            AND A              167
            SBC HL,DE          237   82
            RET NC             208
            ADD HL,DE          25
            JR NEXT_L          24    214
SEARCH      INC HL             35
            LD A,(HL)          126
            CP 13              254   13
            JR NZ,N_ENT        32    8
ENT_FD      POP HL             225
            ADD HL,BC          9
            INC HL             35
            INC HL             35
            INC HL             35
            INC HL             35
            JR CHECK           24    231
N_ENT       CP 14              254   14
            JR NZ,N_NUMB       32    7
            INC HL             35
            INC HL             35
            INC HL             35
```

```
            INC HL            35
            INC HL            35
            JR SEARCH         24     231
N_NUMB      CP 33             254    33
            JR C,SEARCH       56     227
            CP 34             254    34
            JR NZ,N_QUOT      32     8
FD_QD       INC HL            35
            LD A,(HL)         126
            CP 34             254    34
            JR NZ,FD_QT       32     250
            JR SEARCH         24     215
N_QUOT      CP 58             254    58
            JR NZ,SEARCH      32     211
            LD D,H            84
            LD E,L            93
FD_ENT      INC HL            35
            LD A,(HL)         126
            CP 13             254    13
            JR Z,ENT_FD       40     209
            CP 33             254    33
            JR C,FD_ENT       56     246
            CP 234            254    234
            JR NZ,N_QUOT      32     236
            LD H,D            98
            LD L,E            107
DEL_CH      PUSH BC           197
            CALL 4120         205    24     16
            POP BC            193
            DEC BC            11
            LD A,(HL)         126
            CP 13             254    13
            JR NZ,DEL_CH      32     245
            POP HL            225
            INC HL            35
            INC HL            35
            LD (HL),C         113
            INC HL            35
            LD (HL),B         112
            DEC HL            43
            DEC HL            43
            DEC HL            43
            JR CHECK          24     160
```

Как она работает:

В пару регистров HL загружается адрес начала БЕЙСИК области и делается переход к подпрограмме проверки конца обрабатываемой программы. Если конец достигнут, происходит возврат в БЕЙСИК.

Программа переходит к обработке следующей строки "NEXT_L". Адрес, имевшийся в HL сохраняется на стеке для дальнейшего использования, а в BC загружается длина обрабатываемой БЕЙСИК строки. Подпрограмма 'NXT_CH' увеличивает адрес в HL и загружает в аккумулятор очередной символ. Если его код меньше чем 33, т. е. это пробел или управляющий символ, процедура возвращается, чтобы повторить эту часть вновь. Если встретившийся символ не является символом REM, делается переход к 'SEARCH'.

Если оператор REM найден, регистр BC увеличивается на 4 (он может быть теперь использован в качестве счетчика), а HL восстанавливается из стека. Затем удаляется количество символов, определенное в BC, начиная с адреса определенного в HL. Для удаления используется процедура ПЗУ, расположенная по адресу 4120. Затем процедура снова переходит к процедуре 'CHECK'.

Если происходит переход к процедуре 'SEARCH', HL увеличивается указывая на следующий символ, и это значение загружается в аккумулятор. Если это символ ENTER, т.е.

достигнут конец строки, то HL восстанавливается из стека и увеличивается, указывая на начало следующей строки, а затем происходит переход к 'CHECK'.

Если аккумулятор хранит символ NUMBER (код равен 14), то HL увеличивается, указывая на первый символ после обнаруженного числа, и процесс поиска повторяется.

Затем делается проверка для символов, код которых меньше 33. Если такой символ обнаружен делается возврат к 'SEARCH'.

Если обнаружен символ "кавычка" (34), процедура выполняет цикл до тех пор, пока не будет обнаружена вторая кавычка, а затем поиск будет продолжен. Если найденный символ не двоеточие, определяющее строку с несколькими выражениями, поиск повторить.

HL копируется в DE, чтобы сохранить адрес двоеточия, а затем HL увеличивается, указывая на следующий символ. Если это символ ENTER, делается переход к 'ENT_FD', иначе, если это управляющий символ или пробел, программа возвращается к 'FD_ENT'.

Если символ не является кодом REM, происходит переход к 'N_QUOT'. Если же код REM найден, в HL загружается адрес последнего встретившегося двоеточия, а затем все символы от адреса, находящегося в HL до следующего символа ENTER удаляются. Указатели для этой строки корректируются, HL устанавливается в начало этой строки, и происходит переход к 'CHECK'.

7.4 Создание REM строк

Длина: 85

Количество переменных 3

Контрольная сумма 9526

Назначение:

Эта процедура создает выражение REM с заданным количеством символов в заданной строке. Символ выбирается пользователем.

Переменные:

Имя - line number

Длина - 2

Адрес - 23296

Комментарий: номер строки в которую надо вставить выражение REM.

Имя - number chr

Длина - 2

Адрес - 23298

Комментарий: Количество символов после REM.

Имя - chr code

Длина - 1

Адрес - 23300

Комментарий: Код символов после REM

Вызов программы:

```
RANDOMIZE USR адрес
```

Контроль ошибок:

Если номер строки равен 0, больше 9999, или строка с тем же самым номером уже существует, программа возвращается в BASIC.

Комментарий:

Эта программа не проверяет достаточно ли свободной памяти для добавления новой строки. Следовательно это должно быть сделано перед стартом с помощью программы "Размер свободной памяти" из нашей статьи. Символы для ввода после REM должны иметь коды больше чем 31, т. к. управляющие символы (0 - 31) могут сбить с толку подпрограмму LIST в ПЗУ.

Подпрограмма ПЗУ, которая вызывается для вставки символов, довольно медленно выполняется, занимая много времени.

Созданное выражение REM может быть использовано для хранения машинного кода или данных.

ЛИСТИНГ МАШИНЫХ КОДОВ

```
МЕТКА       АССЕМБЛЕР          ЧИСЛА ДЛЯ ВВОДА

            LD HL,(23296)      42    0     91
            LD A,H             14
            OR L               181
            RET Z              200
            LD DE,10000        17    16    39
            AND A              167
            SBC HL,DE          237   82
            RET NC             208
            ADD HL,DE          25
            PUSH HL            229
            CALL 6510          205   110   25
            JR NZ,CREATE       32    2
            POP HL             225
            RET                201
CREATE      LD BC,(23298)      237   75    2     91
            PUSH BC            197
            PUSH BC            197
            LD A,13            62    13
            CALL 3976          205   136   15
            INC HL             35
            POP BC             193
NXT_CH      PUSH BC            197
            LD A,B             120
            OR C               177
            JR Z,INSERT        40    11
            LD A,(23300)       58    4     91
            CALL 3976          205   136   16
            INC HL             35
            POP BC             193
            DEC BC             11
            JR NXT_CH          24    240
INSERT      POP BC             193
            LD A,234           62    234
            CALL 3976          205   136   15
            INC HL             35
            POP BC             193
            INC BC             3
            INC BC             3
            LD A,B             120
            PUSH BC            197
            CALL 3976          205   136   15
            POP BC             193
            INC HL             35
            LD A,C             121
            CALL 3976          205   136   15
            INC HL             35
            POP BC             193
            LD A,C             121
            PUSH BC            197
            CALL 3976          205   136   15
            POP BC             193
            INC HL             35
            LD A,B             120
            JP 3976            195   136   15
```

Как она работает:

В пару регистров HL загружается заданный номер строки. Это значение сравнивается с нулем и если он равен нулю, программа возвращается в БЕЙСИК. Если HL содержит число больше 9999 (максимально возможный номер строки), также происходит возврат в БЕЙСИК.

Вызывается подпрограмма ПЗУ (CALL 6510), которая возвращает в HL адрес строки, номер которой был в HL перед вызовом этой подпрограммы. Если флаг нуля установлен, то строка уже существует и в этом случае программа также возвращается в БЕЙСИК.

Если флаг нуля не установлен, делается переход к 'CREATE'. В BC загружается количество символов для вставки после 'REM' и это число запоминается на стеке. В аккумулятор загружается число 13 - код символа ENTER. Затем вызывается подпрограмма ПЗУ из адреса 3976 для вставки символа ENTER. Регистр BC восстанавливается из стека.

После повторного сохранения BC на стеке, пара BC тестируется чтобы определить есть ли еще символы для вставки. Если нет, то делается переход к 'INSERT'. Если есть еще символы для вставки, в аккумулятор загружается заданный нами код и подпрограмма по адресу 3976 (из ПЗУ) используется для его вставки. Счетчик BC уменьшается и программа возвращается к проверке BC на 0.

Как только процедура достигает 'INSERT', код REM вставляется, используя эту же подпрограмму. Затем в BC загружается длина созданной новой строки, и указатели длины для этой строки созданы. Номер строки затем извлекается из стека, и это значение вставляется перед возвратом в БЕЙСИК.

7.5 Сжатие программы.

Длина: 71

Количество переменных: 0

Контрольная сумма: 7158

Назначение:

Эта программа удаляет все ненужные управляющие символы и свободные пространства из БЕЙСИК-программы, увеличивая таким образом количество доступной памяти.

Вызов процедуры:

```
RANDOMIZE USR адрес
```

Контроль ошибок:

Если в памяти нет БЕЙСИК-программы, процедура возвращается в БЕЙСИК.

Комментарий:

Эта процедура предполагает, что все выражения REM уже удалены из БЕЙСИК-программы. Если же это не так, произойдет сбой. Время выполнения пропорционально длине БЕЙСИК программы.

ЛИСТИНГ МАШИННЫХ КОДОВ

```
METKA       ACCEMБЛЕР          ЧИСЛА ДЛЯ ВВОДА

            LD HL,(23635)      42    83    92
NEXT_L      INC HL             35
            INC HL             35
CHECK       LD DE,(23627)      237   91    75    92
            AND A              167
            SBC HL,DE          237   82
            RET NC             208
            ADD HL,DE          25
LENGTH      PUSH HL            229
            LD C,(HL)          78
            INC HL             35
            LD B,(HL)          70
NEXT_B      INC HL             35
LOAD        LD A,(HL)          126
            CP 13              254   13
            JR NZ,NUMBER       32    8
RESTOR      POP HL             225
            LD (HL),C          113
            INC HL             35
            LD (HL),B          112
```

```
            ADD HL,BC          9
            INC HL             35
            JR NEXT_L          24     227
NUMBER      CP 14              254    14
            JR NZ,QUOTE        32     7
            INC HL             35
            INC HL             35
            INC HL             35
            INC HL             35
            INC HL             35
            JR NEXT_B          24     231
QUOTE       CP 34              254    34
            JR NZ,CONTR        32     12
F_QUOT      INC HL             35
            LD A,(HL)          126
            CP 34              254    34
            JR Z,NEXT_B        40     221
            CP 13              254    13
            JR Z,RESTOR        40     223
            JR F_QUOT          24     244
CONTR       CP 33              254    33
            JR NC,NEXT_B       48     211
            PUSH BC            197
            CALL 4120          205    24     16
            POP BC             193
            DEC BC             11
            JR LOAD            24     204
```

Как она работает:

В пару регистров HL загружается адрес БЕЙСИК-программы. HL затем увеличивается дважды, указывая на два байта, хранящих длину следующей строки. В пару регистров DE загружается адрес области переменных. Если HL не меньше, чем DE, подпрограмма возвращается в BASIC, т. к. достигнут конец программной области.

Адрес из HL сохраняется в стеке, в BC загружается длина строки, и HL увеличивается, указывая на следующий байт в строке. Байт в HL затем загружается в аккумулятор. Если это значение не равно 13, происходит переход к 'NUMBER'.

Для достижения 'RESTOR' должен быть найден конец строки. Адрес указателей длины строки загружается из стека в HL и вставляется длина существующей строки. Длина строки добавляется к HL, HL увеличивается и программа возвращается к NEXT_L.

Если программа достигает 'NUMBER', проверяется, содержит ли аккумулятор символ NUMBER (код 14). Если да, HL увеличивается на 5 - т. е. выполняется прыжок через следующее за NUMBER число, и происходит переход к 'NEXT BYTE'.

Если аккумулятор не содержит кода кавычки, программа переходит к 'CONTR'. Если код кавычки найден, циклы программы будут выполняться до тех пор, пока не будет достигнут конец строки или найдена другая кавычка. В первом случае происходит переход к RESTOR, в последнем - к NEXT_B.

В процедуре CONTR происходит проверка символа. Если он имеет код не меньше, чем 33, подпрограмма возвращается к 'NEXT_B'.

Если свободное пространство или управляющий символ найдены, вызывается подпрограмма ПЗУ по адресу 4120 для его удаления. Длина строки, которая хранится в BC, уменьшается и происходит переход к 'LOAD'.

7.6 Загрузка машинного кода в DATA-строку.

Длина: 179

Количество переменных: 2

Контрольная сумма: 19181

Назначение:

Эта программа создает выражение DATA в первой строке программы на БЕЙСИКе, а

затем заполняет его данными из памяти.

Переменные:

Имя - data start

Длина - 2

Адрес - 23296

Комментарий: Адрес данных для копирования.

Имя - data length

Длина - 2

Адрес - 23298

Комментарий: Количество байтов для копирования.

Вызов программы:

```
RANDOMIZE USR адрес
```

Контроль ошибок:

Если число байтов для копирования равно 0, или первая строка уже есть, программа возвращается в БЕЙСИК. Программа не проверяет, достаточно ли памяти для новой строки. Программа требует 10 байтов на байт данных плюс пять байтов для номеров строк, указателей длины и т. д. Кроме того, используемая подпрограмма ПЗУ также использует большую рабочую область - принимайте это в расчет. Если доступной памяти недостаточно, указатели строки будут установлены неправильно, и листинг БЕЙСИКа будет недостоверен.

Комментарий:

Время, занимаемое работой этой программы, пропорционально объему памяти для копирования.

ЛИСТИНГ МАШИННЫХ КОДОВ

```
METKA       ACCEMБЛЕР          ЧИСЛА ДЛЯ ВВОДА

            LD DE,(23296)      237   91    0     91
            LD BC,(23298)      237   75    2     91
            LD A,B             120
            OR C               177
            RET Z              200
            LD HL,(23635)      42    83    92
            LD A, (HL)         126
            CP 0               254   0
            JR NZ,CONT         32    6
            INC HL             35
            LD A,(HL)          126
            CP 1               254   1
            RET Z              200
            DEC HL             43
CONT        PUSH HL            229
            PUSH BC            197
            PUSH DE            213
            SUB A              151
            CALL 3976          205   136   15
            EX DE,HL           235
            LD A,1             62    1
            CALL 3976          205   136   15
            EX DE,HL           235
            CALL 3976          205   135   15
            EX DE,HL           235
            CALL 3976          205   135   15
            EX DE,HL           235
            LD A,228           62    228
            CALL 3976          205   135   15
            EX DE,HL           235
NEXT_B      POP DE             209
            LD A,(DE)          26
            PUSH DE            213
```

```
          LD C,47          14    47
HUNDR     INC C            12
          LD B,100         6     100
          SUB B            144
          JR NC,HUNDR      48    250
          ADD A,B          128
          LD B,A           71
          LD A, C          121
          PUSH BC          197
          CALL 3976        205   136   15
          EX DE,HL         235
          POP BC           193
          LD A,B           120
          LD C,47          14    47
TENS      INC C            12
          LD B,10          6     10
          SUB B            144
          JR NC,TENS       48    250
          ADD A,B          128
          LD B, A          71
          LD A,C           121
          PUSH BC          197
          CALL 3976        205   136   15
          POP BC           193
          EX DE,HL         235
          LD A,B           120
          ADD A,48         198   48
          CALL 3976        205   136   15
          EX DE,HL         235
          LD A,14          62    14
          LD B, 6          6     6
N_ZERO    PUSH BC          197
          CALL 3976        206   136   15
          POP BC           193
          EX DE,HL         235
          SUB A            151
          DJNZ N_ZERO      16    247
          POP DE           209
          PUSH HL          229
          DEC HL           43
          DEC HL           43
          DEC HL           43
          LD A,(DE)        26
          LD (HL),A        119
          POP HL           225
          INC DE           19
          POP BC           193
          DEC BC           11
          LD A,B           120
          OR C             177
          JR Z,ENTER       40    10
          PUSH BC          197
          PUSH DE          213
          LD A,44          62    44
          CALL 3976        205   136   15
          EX DE,HL         235
          JR NEXT_B        24    173
          LD A,13          62    13
ENTER     CALL 397         205   136   15
          POP HL           225
          LD BC,0          1     0     0
          INC HL           35
          INC HL           35
          LD D,H           84
          LD E,L           93
```

```
            INC HL              35
POINT       INC HL              35
            INC BC              3
            LD A,(HL)           126
            CP 14               254     14
            JR NZ,END?          32      12
            INC BC              3
            INC BC              3
            INC BC              3
            INC BC              3
            INC BC              3
            INC HL              35
            INC HL              35
            INC HL              35
            INC HL              35
            INC HL              35
            JR POINT            24      237
END?        CP 13               254     13
            JR NZ,POINT         32      233
            LD A,C              121
            LD (DE),A           18
            INC DE              19
            LD A,B              120
            LD (DE),A           18
            RET                 201
```

Как она работает:

В пару регистров DE загружается адрес байтов для копирования, а в пару регистров BC загружается число байтов для копирования. Если BC содержит 0, программа возвращается в БЕЙСИК.

В пару регистров HL загружается адрес БЕЙСИК-программы. В аккумулятор загружается байт из адреса установленного в HL. Это старший байт номера строки. Если он не равен 0, первая строка не существует и программа переходит к 'CONT'. Если старший байт содержит 0, в аккумулятор загружается младший байт. Если это значение установлено в 1, первая строка уже существует и программа возвращается в БЕЙСИК.

Адрес старшего байта номера строки сохраняется на стеке. Сохраняется число байтов для копирования, затем адрес данных. В аккумулятор загружается 0 (старший байт нового номера строки). Вызванная подпрограмма ПЗУ, находящаяся по адресу 3976, вставляет символ, имеющийся в аккумуляторе, по адресу установленному в HL.

В HL восстанавливается значение, хранившееся там перед этой операцией. В аккумулятор загружается 1 и это значение вставляется 3 раза. Первая единица - это младший байт номера строки, следующие две - указатель длины строки. В аккумулятор затем загружается код символа 'DATA' и это значение вводится в строку.

Адрес следующего байта данных восстанавливается из стека и загружается в DE. Сам этот байт загружается в аккумулятор, а содержимое DE вновь сохраняется на стеке. В C-регистр загружается значение, на 1 меньшее, чем код символа "0". C-регистр увеличивается, а в B-регистр загружается значение 100. B-регистр вычитается из аккумулятора и, если результат не отрицательный, подпрограмма возвращается к 'HUNDR'.

B-регистр прибавляется к аккумулятору. Таким образом аккумулятор содержит положительное значение. Это значение затем загружается в B-регистр. В аккумулятор загружается содержимое C-регистра, а BC сохраняется на стеке. Вызовом подпрограммы ПЗУ (3976), вставляется символ, хранящийся в аккумуляторе, по адресу, содержащемуся в HL. Пара регистров BC восстанавливается из стека, а в аккумулятор загружается значение B-регистра. Вышеупомянутый прием затем повторяется для B=10. Аккумулятор затем увеличивается на 48 и полученный в результате символ вставляется в строку.

Вышеописанная подпрограмма делала вставку десятичного представления встречающегося байта данных в выражение DATA. Теперь должно быть вставлено двоичное представление. Это значение маркируется с помощью кода NUMBER (CHR 14), который

вводится первым, а за ним 5 нулей. Значение копируемого байта помещается в ячейку, чтобы заместить третий ноль. DE увеличивается, указывая на следующий байт данных. Число байтов для копирования снимается из стека в BC, и это значение уменьшается. Если результат равен 0, делается переход к 'ENTER', иначе содержимое пар регистров BC и DE вторично помещается в стек, в выражение DATA включается запятая, а программа возвращается к 'NEXT_B'. В данной процедуре код ENTER (конец строки) добавляется для маркировки конца выражения DATA. В HL загружается адрес начала строки, а BC устанавливается в 0. HL увеличивается, указывая на младший байт указателя строки, и этот новый адрес копируется в DE. HL увеличивается, указывая на старший байт указателя строки. HL и BC затем увеличиваются, а в аккумулятор загружается символ, находящийся по адресу в HL.

Если аккумулятор содержит код 14, число найдено, то HL и BC увеличиваются на 5, перепрыгивая через число и указывая на символ, следующий за ним. Подпрограмма затем переходит к 'POINT'.

Если аккумулятор не содержит значений 14 и 13, происходит переход к 'POINT'.

Завершение формирования строки выполняется помещением содержимого BC в ячейку указателя длины строки. На необходимый адрес указывает DE. По окончании работы процедура выполняет возврат в БЕЙСИК.

8. ИНСТРУМЕНТАЛЬНЫЕ ПРОГРАММЫ

8.1 Определение размера свободной памяти.

Длина: 14

Количество переменных: 0

Контрольная сумма: 1443

Назначение:

Дает количество свободного пространства ОЗУ в байтах.

Вызов программы.

```
PRINT USR адрес
```

Контроль ошибок: Нет

Комментарий:

Эта программа должна вызываться перед использованием любых подпрограмм, которые могут увеличивать длину программы, чтобы быть уверенным в том, что в ОЗУ достаточно свободного пространства.

ЛИСТИНГ МАШИННЫХ КОДОВ

МЕТКА	АССЕМБЛЕР	ЧИСЛА ДЛЯ ВВОДА			
	LD HL,0	33	0	0	
	ADD HL,SP	57			
	LD DE,(23653)	237	91	101	92
	AND A	16			
	SBC HL,DE	237	82		
	LD B,H	68			
	LD C,L	77			
	RET	201			

Как она работает:

В пару регистров HL загружается 0 и это значение суммируется с адресом конца свободной области ОЗУ (адрес хранится в SP). Пара регистров DE загружается адресом начала свободной области ОЗУ и вычитается из HL. HL копируется в BC и программа возвращается в БЕЙСИК.

8.2 Определение длины БЕЙСИК-программы

Длина: 13
Количество переменных: 0
Контрольная сумма: 1544
Назначение:
Дает длину БЕЙСИК-программы в байтах.
Вызов программы:

```
PRINT USR адрес
```

Контроль ошибок: Нет
Комментарий: Нет

ЛИСТИНГ МАШИННЫХ КОДОВ

```
МЕТКА       АССЕМБЛЕР          ЧИСЛА ДЛЯ ВВОДА

            LD HL,(23637)      42    75    92
            LD DE,(23635)      237   91    83    92
            AND A              167
            SBC HL,DE          237   82
            LD B,H             68
            LD C,L             77
            RET                201
```

Как она работает:
В пару регистров HL загружается адрес области переменных, а в DE загружается адрес БЕЙСИК-программы. DE вычитается из HL, чтобы получить длину программы. HL копируется в BC и программа возвращается в БЕЙСИК.

8.3 Определение адреса БЕЙСИК-строки.

Длина: 29
Количество переменных: 1
Контрольная сумма: 2351
Назначение:
Возвращает адрес первого символа после кода 'REM' в заданной строке.
Переменные:
Имя - line number
Длина - 2
Адрес - 23296
Комментарии: номер строки, которая должна содержать 'REM'.
Вызов программы:

```
LET A = USR адрес
```

Контроль ошибок:
Если заданная строка не существует, или нет выражения REM, программа возвратит значение 0.
Комментарии:
Эта программа может быть использована для нахождения адреса ячейки, в которую может быть помещен машинный код. После выполнения программы переменная A (может быть использована любая переменная) устанавливается по адресу, или в 0, если имеет место ошибка.
Не вводить номер строки более 9999!

ЛИСТИНГ МАШИННЫХ КОДОВ

```
МЕТКА       АССЕМБЛЕР          ЧИСЛА ДЛЯ ВВОДА

            LD HL,(23296)      42    0     91
```

```
            LD A,H              124
            OR L                181
            JR Z,ERROR          40      5
            CALL 6510           205     110     25
            JR Z,CONT           40      4
ERROR       LD BC,0             1       0       0
            RET                 201
CONT        INC HL              35
            INC HL              35
            INC HL              35
            INC HL              35
            LD A,234            62      234
            CP (HL)             190
            JR NZ,ERROR         32      243
            INC HL              35
            LD B,H              68
            LD C,L              77
            RET                 201
```

Как она работает:

В пару регистров HL заносится определенный номер строки. Если этот номер равен 0, делается переход к 'ERROR', иначе вызывается подпрограмма ПЗУ по адресу 6510. По возвращении из этой процедуры HL содержит адрес строки. Если флаг нуля установлен, делается переход к 'CONT'. Если флаг нуля не установлен, эта строка не существует, и подпрограмма переходит к 'ERROR', где в BC загружается 0, и затем происходит возврат в BASIC.

Если программа достигает метки 'CONT', HL увеличивается на 4, чтобы указать на первый оператор в данной строке. Если этот оператор не имеет кода 234, происходит переход к 'ERROR'. Если же это оператор 'REM', HL увеличивается, указывая на следующий символ. Значение HL затем копируется в BC и программа возвращается в БЕЙСИК.

8.5 Копирование данных в памяти.

Длина: 33

Количество переменных: 3

Контрольная сумма: 4022

Назначение:

Эта программа копирует область памяти с одного места в другое.

Переменные:

Имя - start

Длина - 2

Адрес - 23296

Комментарий: адрес источника для копирования.

Имя - destination

Длина - 2

Адрес - 23298

Комментарий: Адрес, в который происходит копирование.

Имя - length

Длина - 2

Адрес - 23300

Комментарий: длина блока, подлежащего копированию.

Вызов программы:

```
RANDOMIZE USR адрес
```

Контроль ошибок: Нет

Комментарий:

Эта подпрограмма может быть использована для создания "мультипликации" с помощью следующего метода:

- создание первого экрана информации;
- копирование экрана выше RAMTOP;

- повторить для других экранов;
- копирование экранов в обратном направлении по одному в быстрой последовательности.

ЛИСТИНГ МАШИННЫХ КОДОВ

```
МЕТКА       АССЕМБЛЕР          ЧИСЛА ДЛЯ ВВОДА

            LD HL,(23296)      42    0     91
            LD DE,(23298)      237   91    2     91
            LD BC,(23300)      237   75    4     91
            LD A,B             120
            OR C               177
            RET Z              200
            AND A              157
            SBC HL,DE          237   82
            RET Z              200
            ADD HL,DE          25
            JR C,COPY          56    3
            LDIR               237   176
            RET                201
COPY        EX DE,HL           235
            ADD HL,BC          9
            EX DE,HL           235
            ADD HL,BC          9
            DEC HL             43
            DEC DE             27
            LDDR               237   184
            RET                201
```

Как она работает:

В пару регистров HL загружается адрес первого байта памяти для копирования, в DE загружается адрес, в который копируется память, а в BC загружается количество байтов для копирования. Если BC=0 или HL=DE, то подпрограмма возвращается в БЕЙСИК. Если HL больше, чем DE, часть памяти копируется, используя инструкцию 'LDIR', и программа возвращает в БЕЙСИК.

Если же DE больше, чем HL, то к обеим парам регистров прибавляется по BC-1 и память копируется, используя инструкцию 'LDDR', после чего программа возвращается в БЕЙСИК.

Окончание следует.

ВОЗВРАЩАЯСЬ К НАПЕЧАТАННОМУ

КАНАЛЫ И ПОТОКИ

В 12-м номере "ZX-РЕВЮ" мы подняли вопрос о концепции каналов и потоков. Среди наших читателей есть мнение о том, что при всей нужности и важности этого вопроса, сама статья грешила определенной академичностью и, если читать ее от начала и до конца, то не все могли с ней разобраться. Большинство взяло то, что лежало на поверхности, а остальное отложили для разбора на долгое время.

В связи с этим мы с радостью принимаем предложение, высказанное нашим постоянным корреспондентом из г. Балашова Саратовской области Пашориным В. Г. о введении постоянной рубрики "Возвращаясь к напечатанному", и начинаем с того, что дадим рекомендации, которые предлагает наш уважаемый автор для тех, кто хотел бы не только в теории ознакомиться поближе с каналами и потоками, но и применить свои знания на практике.

Это достойное развитие начатой темы, и нам будет очень приятно, если полезное начинание будет подхвачено и новая рубрика вызовет появление новых публикаций.

* * *

Для начала небольшое уточнение. Информация о существующих каналах хранится в ПЗУ и только после включения компьютера и инициализации системы эта информация копируется в область, о которой указывалось в статье - непосредственно перед областью Бейсик-программы. А этот участок памяти, как известно, является уже частью ОЗУ. Такое дублирование области информации о каналах из ПЗУ в ОЗУ и одновременное существование двух одинаковых областей не является непродуманной расточительностью памяти. Это глубоко продуманное решение, ориентированное на активных пользователей и разработчиков программного обеспечения, после инициализации система работает только с областью, находящейся в ОЗУ. А как известно, из ОЗУ можно не только читать информацию, но и записывать туда свою информацию. Это значит, что внося продуманные изменения в существующую область информации о каналах или расширяя ее для создания новых, пользовательских, каналов, можно получить интересные эффекты. Для примера выберем канал S. Информация об этом канале находится в ячейках с 23739 по 23743 (см. Табл. 1)

Таблица 1.

Адрес	Содержимое	
23739	244	9*256+244=2548 - адрес процедуры
23740	9	обслуживания канала при выводе
23741	196	21*256+196=5572 - адрес процедуры
23742	21	обслуживания канала при вводе
23743	83	код литеры S

Во многих фирменных программах для того, чтобы не искажать изображение на экране, полученное после дисплейного файла, перед загрузкой последующих блоков программы меняют адрес процедуры обслуживания канала S при выводе путем записи простых команд:

```
POKE 23739,82
POKE 23740,0
```

При этом загрузка блока будет выполняться, но на экране не будет сопровождающих надписей:

```
Program...
Byte...
```

и т.д.

Использование же команды CL0SE#2 для этих целей неприемлемо. После загрузки

всех блоков программы адрес процедуры вывода восстанавливается:

```
POKE 23739,244
POKE 23740,9
```

Этот нехитрый прием могут применять пользователи и для своих целей.

Другое, не менее интересное применение концепции каналов и потоков для практических целей - это изменение функции канала R. Основное его назначение это ввод данных из буфера редактора. Канал нельзя обслуживать, программируя на языке Бейсик, но его можно изменить так, чтобы редакция строк была невозможна. Это будет интересно тем, кто работает над защитой своих программ от несанкционированного вмешательства, достаточно записать:

```
POKE 23744,124
POKE 23745,0
```

и редакция строки будет невозможна.

Аналогичные эффекты можно получить, манипулируя не с данными в области информации о каналах, а с данными в системной переменной STRMS (23568). Так как каналы связаны с определенными потоками, то переподключая потоки к другим каналам, получим новые эффекты. Ну, например, чтобы запретить вывод на экран информации о программе при загрузке ее с магнитофона, достаточно записать в ячейку 23570 через команду POKE вместо числа 6 число 16, переподключив поток-2, связанный с каналом S к каналу P, обеспечивающему вывод на принтер. Переподключать стандартные каналы к потокам можно и с помощью hash-символа #. Попробуйте ввести программу (табл. 2)

Таблица 2

```
10 PRINT # 0; "Текст в нижней части экрана": PAUSE 0
20 LPRINT # 2; "Текст в верхней части экрана": PAUSE 0
30 LPRINT # 3; "Текст на принтере": PAUSE 0
40 LIST # 1
```

и Вы убедитесь, что знакомые Вам Бейсик-инструкции работают несколько необычно.

Кроме системных переменных CHANS(23631/2) и STRMS(23568), о которых упоминалось в статье, существует еще одна системная переменная (а о ней ничего не упоминалось), сохраняющая информацию о каналах. Эта переменная CURCHL (23633/4) Current Channel, в которой записан адрес активизированного в данный момент канала. Именно значение этой переменной используется инструкцией RST 16 для вывода информации. Кроме того, системные переменные FLAGS (23611), FLAGS 2(23658) и TV FLAG (23612) также имеют некоторое отношение к каналам и потокам.

Внимательный пользователь заметил, что для каналов K, S и P процедура вывода имеет один и тот же адрес - 2548. Так вот, бит 1 переменной FLAGS указывает на то, должен ли символ выводится на принтер (бит равен 1) или на экран (бит равен 0).

Нулевой бит переменной TV FLAG информирует о том, используется ли верхняя часть экрана (бит равен 0) или нижняя (бит равен 1). Ну и бит 4 переменной FLAGS 2, находясь в различных состояниях, (сброшен - установлен), определяет является ли канал к рабочим (бит установлен) или нерабочий в данный момент (бит сброшен).

Прежде, чем перейти к процедурам, демонстрирующим возможность использования концепции каналов и потоков на компьютерах с 48K ОЗУ, еще одно замечание. Поскольку, как отмечалось выше, информация о каналах хранится и в ПЗУ и в ОЗУ, то можно, например, увеличить область Бейсик-программы, исключив область информации о каналах из ОЗУ. Для этого надо переписать значение системных переменных:

```
10 POKE 23635, PEEK 23631
20 POKE 23636, PEEK 23632
30 POKE 23631, 175
40 POKE 23632, 21
```

Здесь в строках 10 и 20 меняется значение системной переменной, указывающей на начало Бейсик-программы. Теперь она будет начинаться не с адреса 23755, а с адреса 23734. В строках же 30 и 40 в системную переменную CHANS записывается начальный адрес области информации о каналах, находящейся в ПЗУ. Таким образом, участок памяти

для хранения Бейсик-программы увеличивается на 21 байт.

Весьма полезной, особенно для создателей обучающих программ, может быть следующая процедура, которая несколько меняет действие команды PRINT. При использовании этой процедуры по команде PRINT информация на экран будет выводиться с временной паузой между отдельными символами и с генерацией короткого звука после вывода каждого символа. Причем величина временной паузы между символами может быть эффективно использована при создании обучающих программ, например, по чтению или скорочтению. Эффект имитации работы пишущей машинки или телетайпа, кроме того, вносит разнообразие в работу программы (вспомните, например, игровую программу Black Hawk).

А вот и сама процедура:

```
10    BEEP       EQU   949
20    CURCHL     EQU   23633
30    TIME       EQU   65300
40               ORG   55301
50    PRINT      DEFW  2548
60    START      PUSH  AF
70               LD    A,I
80               JR    PO,NOPAUSE
90               LD    A,(TIME)
100              LD    B,A
110   PAUSE      HALT
120              DJNZ  PAUSE
130   NOPAUSE    POP   AF
140              CP    33
150              JR    C,NOBEEP
160              PUSH  AF
170              PUSH  IX
180              LD    DE,50
190              LD    HL,100
200              CALL  BEEP
210              POP   IX
220              POP   AF
230   NOBEEP     LD    HL,(PRINT)
240              CALL  111
250              LD    HL,(CURCHL)
260              LD    BC,START
270              LD    E,(HL)
280              LD    (HL),C
290              INC   HL
300              LD    D,(HL)
310              LD    (HL),B
320              LD    A,B
330              CP    D
340              JR    NZ,CHNG
350              LD    A,C
360              CP    E
370              RET   Z
380              LD    (PRINT),DE
390              RET
```

Запуск этой процедуры может показаться несколько необычным, так как для этого не используется функция вызова программы в машинных кодах USR. Для того, чтобы эта процедура выполнялась, достаточно записать в область информации о каналах (в ячейки, отведенные для адреса процедуры вывода, обслуживающей канал S), адрес созданной процедуры. Стандартно тем записан адрес 2548, а мы с помощью команд POKE записываем туда стартовый адрес созданной процедуры - 65303:

```
POKE 23739, 23
POKE 23740,255
```

Теперь по команде RST 16 (вызов процедуры вывода символа на экран) будет вызываться не процедура из ПЗУ, а из адреса 65303. Временная пауза между выводимыми символами задается путем записи с помощью команды POKE соответствующего числа (от 1

до 10) в ячейку 65300.

Пояснения к процедуре: в строках 50...80 выполняется проверка на разрешение прерывания. Если прерывания запрещены, то выполняется переход к строке с меткой NOPAUSE и обход инструкции HALT, приводящей к зависанию компьютера в этом случае.

- В строках 90...120 формируется требуемая временная пауза перед выводом на экран очередного символа.

- в строках 130...150 выполняется проверка кода выводимого символа.

Если код менее 33, то это не печатаемый символ, а управляющий код и выполняется переход к строке с меткой NOBEEP, минуя строки 160...220, обеспечивающие выдачу звукового сигнала вместе с выводом символа на экран. Поскольку обработка управляющих кодов происходит с использованием других процедур, то прямой вызов CALL 2548 может привести к ошибке или потере управляющего кода. Поэтому выполняется вызов CALL 111. По адресу 111 записана команда JP (HL). Таким образом мы реализуем несуществующую в ассемблере процессора Z-80 команду CALL (HL) и выполняем переход по адресу, указанному в регистровой паре HL (строка 130).

Внутри процедур обработки управляющих кодов имеются команды, которые меняют содержимое байтов области информации о каналах. Поэтому в строках 250...310 выполняется проверка записанного для канала S адреса процедуры вывода и восстановление адреса 65303, а затем восстановление, при необходимости, переменной PRINT (строки 320...390).

Предложенная Вашему вниманию процедура прекрасно работает только до момента, пока не будет выполнена команда CLS, так как эта команда восстанавливает в области информации о каналах стандартные адреса процедур вывода. Чтобы вернуть прежнее действие команде PRINT, необходимо опять же с помощью команд POKE записать в ячейки 23739/40 адрес 65303. И так поступать после каждой команды CLS или нажатия клавиши ENTER.

Естественно, это не совсем удобно, поэтому лучшим решением будет открытие нового пользовательского канала, адреса процедур ввода-вывода которого будут оставаться незыблемыми, пока включен компьютер. Для этого необходимо расширить область информации о каналах на 5 байтов, в которых записать адреса соответствующих процедур ввода и вывода и имя нового канала. Выполняется это с помощью процедуры ПЗУ, находящейся по адресу 5717. При этом перед вызовом этой процедуры в регистровую пару HL необходимо записать начальный адрес резервируемой области, а в регистровую пару BC - количество резервируемых байтов. При использовании этой процедуры автоматически переписывается содержимое системных переменных, определяющих адреса отдельных блоков Бейсик-области. Ниже представлена программа, позволявшая создать новый канал:

```
10 FOR N=23296 TO 23304: READ A: POKE N,A: NEXT N
20 DATA 1,5,0,33,202,92,195,85,22
30 RANDOMIZE USR 23296
40 RESTORE 60
50 FOR N=23754 TO 23758: READ B: POKE N, B: NEXT N
60 DATA 23,255,196,21,83
70 STOP
```

После выполнения этой программы останется только подключить канал к потоку 2 командой POKE 23578,21. Теперь можно не беспокоиться о необходимости восстанавливать адрес созданной процедуры в области данных о каналах.

Алексеев А. Г.

ПРОФЕССИОНАЛЬНЫЙ ПОДХОД

Уважаемые читатели, если Вы читаете "ZX-РЕВЮ" не первый год, то должны быть знакомы с серией статей Стива Тернера под названием "Профессиональный подход", печатавшихся в прошлом году.

При всем уважении к английской школе программирования для "Спектрума" мы должны сказать что и у нас немало хороших программистов. Только скромность не позволяет честно признать что для "Спектрума" у нас их сейчас может быть уже и больше, а не видно их работы только потому что нет инфраструктуры многочисленных фирм способных приобретать их разработки и тиражировать массовыми тиражами.

Нет пока у нас и средств массовой информации, способных объединить миллионы людей, охваченных единой страстью Мы прекрасно понимаем что роль ZX-РЕВЮ в этом деле конечно какая-то есть, но это не более, чем капля в море. Мы ведь не охватываем и сотой доли процента тех, кто в этом нуждается.

В нашем портфеле с каждым днем становится все больше и больше интересных статей от отечественных программистов, которым есть чем поделиться с многочисленными любителями, как с начинающими, так и с имеющими некоторый опыт.

Сегодня мы возвращаемся к рубрике "Профессиональный подход", но уже на другом уровне. Теперь мы открываем ее для наших отечественных специалистов, которым есть что сказать своим коллегам. Если и у Вас есть идеи и Вы можете их изложить доступно для понимания широкими массами, мы открыты и для вас. Работа оплачивается.

ОБРАБОТКА ОШИБОК В БЕЙСИКЕ

Среди программистов более или менее освоивших Бейсик, прошитый в ПЗУ "Спектрума", бытует мнение, что невозможно на Бейсике создать хорошую "качественную" программу, которая к тому же выглядела бы достаточно "фирменно". Хорошая программа - на 5 баллов – это, как правило, программа в машинных кодах. Существует также достаточно много расширений возможностей обычного Бейсика. Это LASER-BASIC, BETA-BASIC, MEGA-BASIC и другие. Однако все их надо догружать с ленты, при этом результирующая программа получается достаточно громоздкой. Эти расширения Бейсика хороши и оправданы в том случае, если Вы создаете например серьезную игровую программу, заведомо претендующую на "фирменность". Зачастую же для повседневных задач не нужны богатые возможности расширений Бейсика. Прелесть Бейсика-ПЗУ в том, что он всегда сразу же готов к работе.

Что такое "повседневные задачи"? Ну, например я применяю компьютер для обучения детей устному счету, обучения азбуке. Одно дело, когда этому учит мама или папа, и совсем другое дело для ребенка вести диалог с КОМПЬЮТЕРОМ! Эффект обучения здесь на порядок выше. Компьютер может применяться в школе с первого до последнего класса. (На эту тему можно говорить бесконечно.) А студентам очень пригодится, например, при расчетах курсовых проектов. Это и есть "повседневные задачи". Вы освоили Бейсик ПЗУ и в состоянии написать нехитрую программу для той задачи, которая сейчас нужна. Задача решена и программа, может быть, больше никогда не понадобится. А может быть Ваш труд и не пропадет даром, а заинтересует других. Но это произойдет вероятнее в том случае, если программа выглядит "фирменно". Помните: встречают по одежке. Я применяю несколько простых приемов, которыми готов поделиться. Освоив их, Вы сможете придать "фирменный" вид любой самой простой и примитивной Бейсик-программе. Да Вы и сами получите удовольствие от пользования такой программой.

Итак, первый "фирменный" прием.

ПРЕДОТВРАЩЕНИЕ ОСТАНОВКИ БЕЙСИК ПРОГРАММЫ

Мало кому нравятся Бейсик программы, которые останавливаются при первой встретившейся ошибке или случайном нажатии клавиши "BREAK" Кроме неудобства в работе, такие программы выглядят "нефирменно", в отличие от программ в машинных кодах. Однако существует способ, при помощи которого можно устранить остановку программы от клавиши "BREAK" или при встретившейся ошибке, причем в зависимости от ошибки и места в программе, где эта ошибка произошла, возможна различная, заранее запланированная реакция программы.

Этот способ, по-моему, незаслуженно недостаточно распространен среди программистов, хотя дает отличные результаты. Речь идет о программе в машинных кодах "ON ERROR GO TO". Те, у кого есть фирменная программа "SUPERCODE" или ее более поздние версии, возможно знают об упомянутом блоке кодов, но не применяют его из за недостаточного удобства в работе. Методика, изложенная ниже, позволит достаточно легко и эффективно пользоваться этой программой, а для тех, у кого нет программы "SUPERCODE", приводится блок кодов "ON ERROR GO TO". Для тех, кто интересуется программированием в машинных кодах, подробно описывается его работа, а если Вы не интересуетесь машинными кодами, то можете пропустить описание работы блока "ON ERROR GO TO". Прочитав раздел о примере использования этого блока, Вы сможете "пристыковать" его к любой имеющейся у Вас Бейсик-программе. Итак, сначала о блоке кодов "ON ERROR GO TO".

1. Блок кодов "ON ERROR GO TO"

Этот блок имеет длину 73 байта. Он является релоцируемым, то есть его можно загружать в любое место памяти. Вызывается он обычным способом, то есть при помощи RANDOMIZE USR ADDR, где ADDR - это начальный адрес загрузки блока кодов.

Рассмотрим работу блока, рас положив его для примера в буфере принтера начиная с адреса 23296 (#5B00).

В листинге блока кодов справа число в скобках - это ссылки на комментарии по работе блока, описание которой дано ниже.

Текст программы "ON ERROR GO TO"

```
5B00 CD7C00       CALL  #007C        (1)
5B03 3B           DEC   SP           (2)
5B04 3B           DEC   SP
5B05 E1           POP   HL           (3)
5B06 010F00       LD    BC,#000F     (4)
5B09 09           ADD   HL,BC        (5)
5B0A EB           EX    DE,HL        (6)
5B0B 2A3D5C       LD    HL,(#5C3D)   (7)
5B0E 73           LD    (HL),E       (8)
5B0F 33           INC   HL
5B10 72           LD    (HL),D
5B11 C9           RET   (9)
5B1E 3B           DEC   SP           (10)
5B13 3B           DEC   SP
5B14 CD8E02       CALL  #028E        (11)
5B17 7B           LD    A,E          (12)
5B18 FEFF         CP    #FF
5B1A 20F8         JR    NZ,#5B14     (13)
5B1C 3A3A5C       LD    A,(#5C3A)    (14)
5B1F FEFF         CP    #FF
5B21 2821         JR    Z,#5B44      (15)
5B23 FE07         CP    #07          (16)
5B25 2B1D         JR    Z,#5B44
5B2T FE08         CP    #08          (17)
5B39 2819         JR    Z,#5B44
5B2B 3C           INC   A            (18)
```

```
5B2C 32815C       LD    (#5C81),A    (19)
5B2F FD3600FF     LD    (IY+0),#FF   (20)
5B33 210000       LD    HL,#0000     (21)
5B36 22425C       LD    (#5C42),HL   (22)
5B39 AF           XOR   A            (23)
5B3A 32445C       LD    (#5C44),A
5B3D FDCB01FE     SET   7,(IY+1)     (24)
5B41 C37D1B       JP    #1B7D        (25)
5B44 33           INC   SP           (26)
5B45 33           INC   SP
5846 C30313       JP    #1303        (27)
```

Блок "ON ERROR GO TO" состоит из двух частей. Собственно программа по обработке ошибки начинается с адреса #5B12 (23314). А с адреса #5B00 (23296) расположена процедура привязки блока кодов к тем адресам, в которых он находится. Привязка осуществляется следующим образом.

Сначала вызывается подпрограмма из ПЗУ (1). По указанному адресу #007C расположена одна единственная команда: RET. При выполнении инструкции CALL на стек компьютера помещается адрес ячейки, в которой находится следующая за CALL команда, то есть в нашем случае #5B03 (23299), а указатель стека - регистр SP - уменьшает свое значение на два.

Встречая инструкцию RET, выполнение программы продолжится с адреса, взятого со стека - это #5B03, а указатель стека увеличит свое значение на два. Но число #5B03 не стирается из памяти. Чтобы возвратить его, производится уменьшение указателя стека на два (2) и число со стека записывается в регистр HL (3). (При этом указатель стека опять увеличивается на два, то есть возвращается к своему прежнему значению.) Теперь в регистре HL находится число #5B03 (23299).

Далее (4) в регистр BC заносится величина смещения #000F (015) байт и добавляется к содержимому регистра HL (5). Теперь в регистре HL будет #5B03+#000F=#5B12 (23299+15=23314). Это адрес начала программы по обработке ошибки.

Если расположить блок кодов "ON ERROR GO TO" с любого другого адреса, то в результате действий (1)...(5) в регистре HL будет адрес начала программы по обработке ошибки. Далее этот адрес пересылается из регистра HL в регистр DE (6) а в регистр HL загружается (7) двухбайтное число из ячеек #5C3D, #5C3E (23613, 23614). Это адрес системной переменной ERR SP. При ошибке происходит переход на адрес, который содержится на стеке в ячейке, адрес которой находится в ERR SP. До работы программы "ON ERROR GO TO" там был адрес 4867 (#1303) - это в ПЗУ подпрограмма вывода сообщения об ошибке. Этот адрес перехода подменяется на новый, то есть на #5B12 (23314), записываемый на стек побайтно из регистров E и D (8). После этого происходит возврат в Бейсик-программу (9).

На этом подготовительная часть закончена. Если ошибки при работе Бейсик-программы не происходит, то все работает обычным порядком, как и без программы "ON ERROR GO TO". А если происходит ошибка, то компьютер передает управление на адрес, записанный на стеке в паре ячеек, указанных в системной переменной ERR SP, то есть на #5B12. При этом происходит следующее.

Вначале указатель стека уменьшается на два, чтобы не испортить стек при работе блока кодов "ON ERROR GO TO" (10). Затем вызывается подпрограмма из ПЗУ "KEY-SCAN" (11). Результат действия этой подпрограммы отражается в регистре DE. Точнее, в регистре E - номер нажатой клавиши. Если не нажато ни одной клавиши, то в регистре E - #FF (255) (аналогично оператору IN в Бейсике).

Сравнивая содержимое регистра E с #FF, предварительно переслав его (12) в аккумулятор A, определяется факт нажатия на какую-либо клавишу. Если результат сравнения не ноль, то есть нажата какая-нибудь клавиша (это может быть, например, клавиша "BREAK"), то зацикливается подпрограмма "KEY-SCAN", приостанавливая дальнейшие действия. Как только клавиша будет отпущена, в регистре E появится число #FF и продолжится работа программы "ON ERROR GO TO"

Далее анализируется произошедшая ошибка, путем проверки содержимого ячейки системной переменной ERR NR, расположенной по адресу #5C3A (23610). Это код ошибки минус 1. Для этого содержимое ERR NR пересылается в аккумулятор (14). Если нет ошибки, а это может произойти в случае логичного завершения программы и ее запланированной остановки в конце (0 OK), то есть код ошибки равен нулю, то содержимое ячейки 23610 будет 0-1 (что для одного байта эквивалентно 256-1) то есть 255 (#FF). В этом случае управление будет передано (15) на адрес #5B44 (23364). Здесь (26) возвращается прежнее значение указателю стека SP и осуществляется переход на подпрограмму вывода сообщения об ошибке (27) (в нашем случае это будет сообщение 0 OK), как это было бы при обычной работе Бейсик-программы.

Аналогичные действия будут и в случае появления ошибки с кодом 8 (8 END of file), (то есть PEEK 23610=7), такая ошибка может произойти при работе с внешней памятью, а также в случае появления ошибки с кодом 9 (9 STOP statement), (то есть PEEK 23610=8), если в тексте программы стоит оператор STOP. Сравнивается содержимое аккумулятора с числом 7 и с числом 8 и в случае совпадения, управление передается (16), (17) на адрес #5B44 (23364).

Дальнейшие действия являются подготовкой к запуску Бейсик-программы с заданной строки. Они необходимы для корректной дальнейшей работы интерпретатора Бейсика.

Для этого прежде всего в ячейку, отведенную для системной переменной ERR NR, заносится число #FF, имитируя отсутствие ошибки (20). Здесь следует пояснить, что при работе интерпретатора Бейсика в "Спектруме" в регистровой паре IY находится число #5C3A (23610) - адрес системной переменной ERR NR.

Далее в два байта системной переменной NEW PPC (ее адрес - #5C48) записывается номер Бейсик-строки, на которую должен быть сделан переход. Для того, чтобы осуществить это, необходимо предварительно номер строки для перехода занести (21) в регистр HL и затем (22) содержимое HL переслать по адресу #5C48. Здесь отметим для себя, что номер Бейсик строки для перехода находится в ячейках #5B34, #5B35 (23348-23349). К этому моменту мы еще вернемся позже.

Затем обнуляется ячейка системной переменной NS PPC - это номер оператора в строке, на которую будет сделан переход. Сначала обнуляется аккумулятор A (23) и затем содержимое A пересылается в NS PPC - ячейку #5C44 (23620).

Теперь надо установить флаг синтаксиса: 7-й разряд системной переменной FLAGS - управляющие флаги Бейсика, имитируя положительный результат проверки Бейсик-строки на синтаксис (24). Адрес FLAGS - #5C3B (23611) (или IY+1).

Теперь подготовка Бейсик-системы закончена и можно запускать интерпретатор Бейсика (25). Таким образом, Бейсик-программа будет запущена со строки, номер которой задан двухбайтным числом по адресу #5B34 (23348).

Некоторые моменты в работе блока кодов могут показаться излишними, но благодаря им блок "ON ERROR GO TO" устойчиво и надежно работает в разных режимах.

Для того, чтобы получить блок кодов "ON ERROR GO TO", наберите следующую Бейсик-программу:

```
10 LET n=23296: LET s=0
20 FOR x=n TO n+72
30 READ y
40 POKE x,y
50 LET s=s+y
60 NEXT x
70 IF s<>7298 THEN PRINT FLASH 1;"ERROR": STOP
80 SAVE "on err" CODE 23296,73
100 DATA 205,124,0,59,59,255,1,15,0,9,235,42,61,92,115,35,114,201
110 DATA 59,59,205,142,2,123,254,255,32,248,58,58,92,254,255,40,33,254,7,40,29,254,8,40,25
120 DATA 60,50,129,92,253,54,0,255,33,0,0,34,66,92,175,50,68,92,253,203,1,254
130 DATA 195,125,27,51,51,195,3,19
```

После старта, если Вы все набрали правильно, программа сформирует блок кодов и

выдаст его для записи на магнитофон.

Для того, чтобы показать работу программы "ON ERROR GO TO", нам понадобится демонстрационная Бейсик-программа. Это будет программа под названием "BEEPER". Задача, решаемая ей, примитивна и не представляет практического интереса, но на этом примере будут продемонстрированы возможности программы "ON ERROR GO TO" и методы ее практического использования.

2. Программа "BEEPER"

Текст программы:

```
1 GO TO 100
2 BORDER 7: PAPER 7: INK 0: CLS
3 LOAD "on err" CODE 23296
4 RUN
5 SAVE "BEEPER" LINE 2
6 SAVE "On err" CODE 23296,73
7 STOP
20 INPUT ;
22 PRINT #0; TAB 8; FLASH 1;"press any key"
24 PAUSE 0
26 RETURN
100 BORDER 1: PAPER 7: INK 0: CLS
110 PRINT AT 12,8; PAPER 2; INK 7; BRIGHT 1;"  B E E P E R  "
120 GO SUB 20
130 BORDER 7: CLS: PRINT AT 21,0;
1000 REM "beeper"
1010 INPUT "Enter tune (-60...69): ",t
1020 PRINT "Tune=";t
1030 PRINT #0;TAB 8;INVERSE 1;"    B E E P    "
1040 FOR n=1 TO 50
1050 BEEP .07,t
1060 NEXT n
1070 GO SUB 20
1080 GO TO 1000
```

После того, как Вы наберете программу, запустите ее со второй строки RUN 2 и загрузите блок кодов "on err" CODE. После окончания загрузки нажмите "BREAK" и выгрузите готовую программу на ленту, выполнив RUN 5.

Теперь можете командой RUN запустить программу сначала. После появления "заставки", нажмите любую клавишу. В ответ на запрос о величине тона введите число от -60 до 69 и нажмите любую клавишу. Вы услышите прерывистый звук, продолжающийся около пяти секунд, затем программа зацикливается, запрашивая следующую величину тона.

Теперь несколько слов о ситуациях, при которых происходит остановка программы.

Остановка с сообщением "L BREAK into program" произойдет при нажатии на клавишу "BREAK" в режиме ожидания, когда появляется мигающая надпись "press any key" или в момент звукового сигнала. Остановка с сообщением "D BREAK - CONT repeats" произойдет, если нажать "BREAK" в момент ожидания загрузки блока кодов (строка 2). Если при загрузке блока кодов произойдет ошибка магнитофона, то программа остановится с сообщением "R tape loading error". Программу можно остановить, если на запрос о величине тона вместо числа ввести букву, например a,b,... В этом случае остановка произойдет с сообщением "2 Variable not found". Если введенная величина тона будет ниже 60 или выше 69, то программа остановится при попытке выполнить оператор BEEP с сообщением "B integer out of range".

Чтобы запустить программу "ON ERROR GO TO", изменим строку 1:

```
1 RANDOMIZE USR 23296 GO TO 100
```

Теперь при любой ошибке, произошедшей после команды RUN, программа будет запускаться со строки, номер которой задан двухбайтным числом по адресу 23348. Так как там 0, то программа перезапустится с начальной строки (в нашем случае с первой строки). Теперь остановить программу невозможно, Вы можете убедиться в том, что при любой попытке остановки программа просто перезапустится заново.

Это не совсем тот эффект, который нам необходим, но на этом этапе становится ясно, что остановить программу после старта невозможно, можно только перезагрузить ее с ленты, а это неудобно при отладке программы. Поэтому надо позаботиться о возможности остановки программы в процессе отладки. Это можно сделать, например, введя такой "жучок":

```
25 IF INKEY$ "q" THEN STOP
```

Теперь, когда на экране появляется мигающая табличка "press any key", то программу можно остановить, нажав клавишу "Q".

Для того, чтобы инициировать программу "ON ERROR GO TO ", надо, чтобы было выполнено: RANDOMIZE USR 23296. Эту команду можно подставить в начало строки, с которой происходит запуск программы (строка 1) Далее, на разных этапах выполнения программы надо изменять содержимое ячеек 23348,23349, чтобы управлять переходом по ошибке согласно задуманной логике. Например, добавим строки:

```
115 POKE 23348,24 POKE 23349, 0
1000 POKE 23348.242 POKE 23349.3
1065 POKE 23348.0 POKE 23349,0
```

Строка 115 блокирует клавишу "BREAK", исключая остановку программы в режиме "заставки", возвращая ее все время на строку 24 - ожидание нажатия какой-нибудь клавиши. Строка 1000 определяет возврат на строку 1010 при ошибке ввода данных (PEEK 23348+256*PEEK 23349=1010). Строка 1065 определяет возможность перезапуска программы сначала при нажатии клавиши "BREAK", когда появляется мигающая надпись "press any key".

Теперь можно считать, что мы добились требуемой логики работы программы. Предотвращается остановка программы при ошибке ввода данных. Кроме того, можно прервать звуковой сигнал в момент исполнения (когда появляется табличка "BEEP"). А когда исполнение закончится можно вообще перезапустить программу, нажав "BREAK".

Однако следует отметить еще один момент. Определенное неудобство доставляет перевод номера строки в двухбайтную форму. Это приходится делать вручную. К тому же если придется переделывать программу в процессе отладки, изменяя нумерацию строк, то опять приходится пересчитывать данные для POKE 23348, POKE 23349.

Задачу можно значительно упростить если вспомнить о том, что двухбайтный конвертер в "Спектруме" уже есть. Это оператор RANDOMIZE, который задает начальное значение для функции случайной величины RND. Подав команду RANDOMIZE n, мы устанавливаем системную переменную SEED, используемую для вычисления очередного значения функции RND. SEED расположена по адресу 23670 и занимает два байта. В ячейке 23670 находятся младший, а в ячейке 23671 - старший байты числа n, которое используется с оператором RANDOMIZE. Подайте команду RANDOMIZE 1010. Теперь сделайте:

```
PRINT PEEK 23670
PRINT PEEK 23671
```

- получите 242 и 3

Теперь усовершенствуем нашу программу, добавив строки:

```
10 POKE 23348, PEEK 23670: POKE 23349, PEEK 23671: RETURN
```

Изменим строки

```
115 RANDOMIZE 24: GO SUB 10
1000 RANDOMIZE 1010: GO SUB 10
1065 RANDOMIZE 1: GO SUB 10
```

Теперь, с точки зрения программирования, задача упростилась. Здесь только надо отметить, что нельзя подавать команду RANDOMIZE 0, так как в этом случае переменная SEED принимает не значение 0, а становится равной другой системной переменной FRAMES - счетчика кадров, обеспечивая практически случайное число. Вместо RANDOMIZE 0 можно подавать RANDOMIZE 1, так как программа все равно обычно начинается со строки 1. (Если же у Вас присутствует нулевая строка и Вы хотите стартовать именно с нее, то просто подставьте в требуемое место, как и раньше, POKE 23348,0: POKE 23349, 0)

Сейчас можно удалить строку 25, которая нужна была для отладки программы. Теперь единственным путем, позволяющим остановить программу, является нажатие "BREAK" при загрузке с ленты, в тот момент, когда Бейсик-программа уже загружена и ожидается ввод блока кодов "on err" (строка 2).

При желании можно ликвидировать и эту возможность. Для этого блок кодов "ON ERROR GO TO" можно расположить внутри Бейсик-программы в нулевой строке. Наберите:

```
1 REM
```

а после REM - 73 пробела или любых других символов. Затем сделайте POKE 23756,0. Первая строка стала нулевой. Теперь выполните LOAD "on err" CODE 23760.

Затем измените строки:

```
2 RANDOMIZE 1: GO SUB 10: GO TO 100
10 POKE 23812,PEEK 23670: POKE 23813, PEEK 23671: RANDOMIZE USR 23760 RETURN
```

Теперь, когда блок "ON ERROR GO TO" расположен в новом месте (начиная с адреса 23760), ячейками в которых задана строка для перехода по ошибке, будут 23812, 23813.

Строки 1, 3, 6 - удалите, они больше не нужны, так как теперь наша программа состоит из одного блока вместо двух и "горячий" старт (RUN 2) совпадает с "холодным" стартом (RUN). В этом варианте изменен также способ инициирования блока кодов "ON ERROR GO TO". Команда RANDOMIZE USR 23760 помещена теперь не в первую (стартовую) строку, как раньше, а в строку 10. Поэтому включение блока в работу происходит теперь в любом случае, когда выполняется команда GO SUB 10, независимо от того, с какой строки Вы запустите Вашу программу. Такой вариант более универсален.

Мы рассмотрели работу блока кодов "ON ERROR GO TO" на примере маленькой программы. В больших программах может потребоваться более сложная логика перехода при ошибке. Для этого используйте ячейку системных переменных 23681, куда заносится код ошибки при работе блока "ON ERROR GO TO". Анализируя PEEK 23681, Вы можете организовать переход на нужную строку. Для примера измените строку 1000 программы "BEEPER".

```
1000 RANDOMIZE 2000 GO SUB 10
```

Добавьте строки:

```
2000 IF PEEK 23681=11 THEN INPUT ;: PRINT #0; PAPER 2; INK 7; BRIGHT 1; " W A R N I N G :
      60<Tune<69  ": BEEP 1,0: PAUSE 100
2010 GO TO 1010
```

Теперь в том случае, если вводимая величина будет выходить за пределы –60...+69, появится предупредительная табличка со звуковым сигналом. Конечно, проверку на допустимые пределы логичнее организовать при помощи обычного Бейсика, просто этот пример показывает, как можно использовать код ошибки. Хотя надо сказать, что в своих программах мне еще ни разу не приходилось этим приемом пользоваться.

В заключение хочу сказать, что при отладке программ с блоком "ON ERROR GO TO" почаще делайте RUN 5 (сохранение программы на ленте). В результате ошибочных действий может случиться так, что Вы не сможете остановить программу. Тогда останется только загрузить предыдущий вариант. Если предполагаются значительные изменения в программе, то временно в начало строки 10 подставьте RETURN: это отключит блок "ON ERROR GO TO". Кроме того всегда предусматривайте "жучок" для возможности остановки программы типа строки 27 в программе "BEEPER".

* * *

В следующий раз мы поговорим о структуре Бейсик программ, о том, с чего практически начинать написание новой программы, то есть когда в уме или на бумаге план уже достаточно "созрел" и Вы включили компьютер, чтобы набивать текст программы, а также о том, как облегчить себе жизнь, предусмотрев элементарный сервис для себя.

FORUM

В 11-12 номере "ZX-РЕВЮ" (стр. 254) за прошлый год мы писали о существовании "циклических" защит программ от копирования и упоминали защиты класса ALKATRAZ LOADER.

В ответ на эту заметку пришло несколько писем. Победы, одержанные нашими читателями над этим загрузчиком с помощью аппаратных средств мы рассматривать не будем, считая этот путь не относящимся к теме, а вот частный случай, с которым разобрался Д.Коронцвит из Г. Жуковского, Моск. обл., мы рассмотрим.

Почему нас не интересуют аппаратные пути взлома и копирования программ? Дело в том, что с программистской точки зрения взлом ради взлома, копирование ради копирования не интересны. Нас интересуют программистские приемы, новые методы, короче интересует все то, на чем можно учиться и набирать опыт. И, если быть до конца честными, то надо признаться, что любые статьи, посвященные вскрытию программ и снятию защит в основном служат не тем, кто их снимает, а тем, кто их ставит и тем, кто учится программировать.

Итак, по порядку. Просматривая загрузчики, написанные Сергеем Скоробогатовым для дискофицированных им программ Winter Edition, Chase H.Q., Agent X и др., наш читатель столкнулся с листингом, который выглядел примерно так (см. листинг 1, комментарий к листингу - наш, "ИНФОРКОМ"):

Декодирование вручную, с помощью MONS3 показало, что то, что было АБРАКАДАБРОЙ, содержит блок, очень похожий на то, что Вы видите на листинге. Изменился только "ключ" и конечно изменился адрес, загружаемый в регистровую пару HL. Дальнейшее декодирование открыло еще один аналогичный блок и т.д. Длина и структура всех трех просмотренных блоков были одинаковыми.

Конечно, это лишь частный случай, но этот факт позволил написать программу, которая все последующее декодирование выполнила автоматически. Конечно, в более общих случаях она не сработает, но сам принцип будет полезен начинающим, а по мере необходимости они смогут адаптировать метод к конкретным условиям.

Пример декодирующей процедуры приведен в листинге 2.

ЛИСТИНГ 1

```
        LD HL,nn    - загрузили адрес, с которого начинается декодируемый блок.
        LD BC,nn    - длина этого блока (организовали счетчик).
 LOOP   LD A,(HL)   - приняли байт для декодирования.
        XOR "ключ"  - само декодирование.
        LD (HL),A   - заслали декодированное значение на место АБРАКАДАБРЫ.
        INC HL      - перешли к очередному байту.
        DEC BC      - уменьшили счетчик байтов на единицу.
        LD A,B      - подготовка к проверке счетчика на ноль.
        OR C        - проверка счетчика на ноль.
        JR NZ,LOOP
        ...................
        АБРАКАДАБРА
        ...................
```

ЛИСТИНГ 2

```
AGAIN   LD IX (aa)    -  aa - адрес ячейки, в которой организовано хранение адреса начала
                         "взламываемого" блока.
        LD L,(IX+1)   -  второй и третий байты исследуемого блока
        LD H,(IX+2)      содержат адрес декодируемого куска
        LD C,(IX+4)   -  пятый и шестой байты исследуемого блока
        LD D,(IX+5)      содержат длину декодируемого куска.
LOOP    LD A,(HL)     -  приняли байт для декодирования.
        XOR (IX+8)    -  само декодирование.
        LD (HL),A     -  заслали декодированное значение на место.
        INC HL        -  перешли к очередному байту.
```

```
DEC BC        -  уменьшили счетчик байтов на единицу
LD A,B        -  подготовка к проверке счетчика на ноль.
OR C          -  проверка счетчика на ноль.
JR NZ,LOOP    -  если не все биты декодированы, переход к декодированию очередного
                 байта.
LD BC,010H    -  длина декодирующего блоке - 16 байтов (010H).
ADD IX,BC     -  "перепрыгнув" через рассмотренный блок, вводим в IX адрес начала
                 следующего декодирующего блока.
LD (aa),IX    -  помещаем его адрес в ячейку с адресом aa.
LD A,0EEH     -  0EEH - это (238 в десятиричной системе) код операции XOR.
CP (IX+7)     -  сравниваем содержимое восьмого байта нового декодирующего блока с
                 кодом 238, ожидая, что там будет стоять XOR.
JR Z,AGAIN    -  если это так, то возврат к началу и декодирование следующего блока.
RET           -  если нет, то возврат в вызывающую программу. Отсутствие XOR в данной
                 позиции может говорить либо о том, что все декодирование успешно
                 завершено, либо о том, что защита сменила принцип кодирования и уже
                 не использует XOR или использует не только XOR. Тогда надо
                 остановиться и посмотреть, что же она использует и по-возможности
                 внести изменения в свою декодирующую процедуру.
```

Адвентюрные игры.

Heavy on the Magic.

Свое исследование этой увлекательной игры, выпущенной фирмой GARGOYLE GAMES в 1986 году прислал наш читатель из Грозного Каракашев А.Г. С его полезными советами, посвященными игре Sceptre of Bagdad Вы знакомы по прошлому выпуску ZX-РЕВЮ.

Ему удалось пройти все игровое программное пространство до конца, он посетил все комнаты замка, но не может считать, что разобрался с программой полностью. Есть еще нерешенные проблемы, может быть кто-то знает подходы и к ним? Наш корреспондент пользовался некоторыми ходящими по рукам непроверенными описаниями и ряд вопросов связаны именно с ними.

Начнем с проблем, а потом перейдем к достижениям.

В описаниях говорится о 21 типе монстров. Обойдя весь замок, насчитать такого количества не удалось, даже если к монстрам отнести демонов, огонь, воду и персонажей, неперемещающихся в пространстве и не нападающих на главного героя.

В описаниях говорится о 280 предметах, которые можно исследовать. Если считать все, включая двери, столы, камни, сталактиты, никак более 250 не получается.

В описаниях говорится о том, что надписи на стенах встречаются довольно часто, удалось увидеть только одну - в комнате слева от начальной. Более того, нет нигде обещанной комнаты со множеством дверей, где висит особый знак, прочитав который можно погибнуть.

Вскрыв игру с помощью COPY-COPY, удалось обнаружить следующие имена демонов: ASTABOT, ASMODEE, BELEZBAR и MAGOT, но ни в каких книгах заклятий о них не упоминается.

Таким же путей взлома удалось найти слова: ADEPTUS MAJOR, ADEPTUS MINOR, MAGISTR TEMPLI, MAGUS. И невооруженным глазом видно, что эти слова обозначают магический ранг. Но нигде эти ранги не участвуют. Открыв все двери и обойдя все, что

можно, герой повысил свой ранг от NEOPHYTE до PHILOSOPHUS через ZELATOR и PRACTICUS. Причем последняя дверь открывается только при ранге PHILOSOPHUS - не больше, не меньше.

Кто знает, где и как можно получить те ранги, которые были обнаружены при вскрытии программы? Отзовитесь! Может быть у кого-то есть фирменное описание игры?

Теперь несколько заметок для тех, кто еще не начал работать с этой увлекательной программой.

На самом первом экране Вы видите Аксила между двумя столами с книгами. Левая - отравлена, а вот правая - GRIMOIR ("Гримуар") содержит заклятья: BLAST, INVOKE, FREEZE.

С помощью заклятья BLAST можно уничтожать монстров, населяющих замок. Впрочем, для борьбы с самыми крепкими из них, придется прибегать еще и к помощи специальных предметов.

К дверям, около которых есть столы, надо подбирать ключи. Когда Аксил кладет на стол нужный ключ, дверь открывается.

К тем дверям, около которых есть знак в виде двух переплетенных колец, необходимо кроме ключа принести еще и мешочек с золотом (BAG OF GOLD) и положить его на стол. Мешочки с золотом взаимозаменяемы, т.е. любой такой мешочек (кроме отравленного) подойдет к любой такой двери.

Двери, возле которых есть охрана, открываются паролями.

Проход в следующие три двери повышает магический ранг:

ASK APEX - пароль "DOOR, SI LENCE" (комната ZELATOR).

SEEK FIRE BIRD TO ENTER DOOR - пароль "DOOR, LAZA" (комната PRACTICUS).

THE GREAT SIGN I IN FREE - пароль "DOOR, SORONOROS" (комната PHILOSOPHUS).

Прочие двери:

CRY AMD ENTER DOOR - пароль "DOOR, WOLF" - ведет на первый этаж.

TO ENTER IS MADNESS - пароль "DOOR, LUNACY" - открывает дальнейшие глубины замка.

SAY NUMBER OF MAGIC WORDS -пароль "DOOR, ELEVEN" - назначение неясно. Пройдя эту дверь, Аксил исполняет какой-то победный танец и получает сообщение, внушающее уверенность в собственных силах: Well done! Axil the able you can made it for an exit.

Аналогична ей и дверь: EYE FOR AN EYE TO ENTER PARADISE - она открывается паролем "DOOR, LONG".

Интересна дверь PILE TOMB NOKEY. Для того, чтобы ее открыть, надо привлечь на помощь демона: INVOKE ASMODEE-, а затем дать ему команду "ASMODEE, DOOR", - и он разрушит дверь.

Теперь несколько слов о демонах. Каждому демону соответствует свой талисман. Поэтому, чтобы вызвать демона, надо выложить этот талисман в комнате. Если его вызвать без талисмана, он жестоко наказывает, отправляя героя в адские печи.

Для демона BELESBAR нужен талисман MANTIS. Функции этого демона пока неясны, единственное, что удалось установить - он дублирует команду EXEMINE.

Демон MAGOT предпочитает SUNFLOWER. Он знает, где в замке расположен какой-либо предмет.

Предмет демона ASTAROT - меч (SWORD). Кажется, это самый полезный из демонов. Он выполняет роль телепорта и перенесет Вас в названную Вами область замка. Правда, есть недостаток - за мечом приходится возвращаться пешком, поскольку второй раз его уже не вызвать.

ASMODEE - самый злобный демон. Его можно вызвать только в комнате, в которой лежит рубин (RUBY). Он реагирует только на команду DOOR и разрушает дверь, если Вы имеете ранг PHILOSOPHUS, в противном случае с ним лучше дела не иметь.

Вызывая демона, Вы должны помнить, что он появляется точно над талисманом, поэтому лучше встать от него чуть подальше.

Теперь ряд полезных советов.

1. С помощью CLASP можно пройти через огонь.

2. NOUGAT можно положить на место NUGGET.

3. С помощью NUGGET можно убить оборотня (WEREWOLF).

4. С помощью зеркала (MIRROR) можно победить Медузу (MEDUSA).

5. Вампира побеждают чесноком (GARLIC).

6. SLAT побеждает циклопа.

7. BALL можно положить на место PELLET.

8. С помощью PELLET можно победить SLUG.

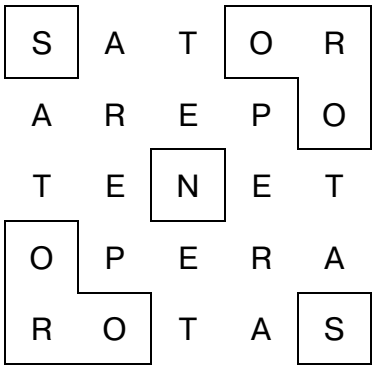
9. Вместо яйца (EGG) положить скорлупу (SHELL).

10. Яйцо используется следующим образом. В районе NIDUS на первом уровне замка, где живут циклопы, нужно положить яйцо в огонь и вызвать птицу-феникс "NEST,PHOENIX". Феникс должен Вам сообщить пароль LAZA.

11. Гидру (HYDRA) можно пройти с помощью SNAKE.

12. Компоненты ULNA, HEAD, THIGH нужно сложить в котел и произнести заклятие "CAULDRON, ACHAD". В котле возникает новый монстр: AU, холодный и мертвый. Он подсказывает пароль LAZA, который уже сообщал Феникс. Может быть это сбой в программе, связанный с неудачным снятием ее защиты? Может быть. AU должен сказать что-то другое, например пароль LONG, который был извлечен из программы просмотром COPY-COPY и подбором.

13. Так же, просмотром кода был получен и пароль SORONOROS, хотя можно предположить, что его можно было бы вывести и из надписи на стене:



14. Чтобы пройти через воду, служит заклятие "WATER, FALL".

15. Пройти через пропасть (CHASM) можно, имея FLASK.

16. В одной комнате двери не оказалось. Трое стражников сообщили, что южной двери не будет, пока не найдешь пароль с помощью ERLSTONE. Спросив об этом у MAROMa, получаешь ясный ответ, что искать его надо в районе PIT: Seek it in the pit. PIT находится в замке на четвертом этаже, но никакого ERLSTOME там нет. Демоны APEX и BELEZBAR на вопрос об этом просят показать им ERLSTONE.

Однако, с помощью хитрости, пройти за несуществующую дверь все же удалось. Просмотр в COPY-COPY дал слово LICHGATE, которое до этого нигде не использовалось. Сказав "ASTAROT, LICHGATE" в комнате с мечом, удалось попасть за эту дверь, так и не найдя ERLSTONE.

Таким образом, был пройден весь замок: 4 этажа, 8 X 8 комнат, но нет никакого выхода и нет естественного финала. Может быть, кто-то из читателей прошел игру до конца и сможет указать на ошибку в действиях, а может быть это сбой в программе, связанный с неудачным снятием защиты?

И еще: нигде не удалось использовать кость (RIB) и кости (BONES - три штуки).

В заключение наш читатель просит начать мозговой штурм игры "DUN DARACH", выпущенной той же фирмой GARGOYLE GAMES. "ИНФОРКОМ" присоединяется к этому пожеланию и обращает внимание на то, что есть еще две игры: TIR-NA-NOG и MARSPORT, не менее интересные и тоже пока никем не освещенные.

Полезные советы.

В прошлом году мы несколько раз писали об особенностях работы компьютеров "Дубна 48" (с. 115, 156). Мы просили тех читателей, которые найдут программные пути

повышения совместимости этой популярной модели, поделиться своими достижениями со всеми любителями "Спектрума".

Суть проблемы, если Вы помните, состоит в том, что из-за нехватки быстродействующих микросхем памяти в этой модели был применен процессор с пониженной частотой, для чего были изменены некоторые константы в процедурах ПЗУ, управляющих загрузкой программ. В итоге программы, снабженные спецзагрузчиками, обходящими ПЗУ, могут не загружаться.

Своим видением этой небольшой проблемы и своими подходами сегодня делится наш читатель из поселка Провидения Магаданской обл. Михаил Владимирович Лапырев.

Во-первых, есть копировщики, работающие на "Дубне", которые позволяют Вам откопировать программу.

1. Lady COPY.
2. COPY FM3
3. COPY-COPY (Pirate 02)

Эти копировщики, правда, не компрессирующие, но свою задачу выполняют хорошо.

И будет совсем прекрасно, если Вы достанете турбо-копировщики:

1 TURBO-COPY.
2 TURBO-CAC.

Эти копировщики - компрессирующие и на "Дубне" они работают, хотя надо знать один нюанс. После загрузки копировщика и выхода в стартовое окно, не спешите нажимать LOAD. Включите магнитофон с программой, которую Вы хотите переписать и дождитесь пилоттона, идущего перед блоком программы, и только потом нажимайте клавишу. Не беспокойтесь, если между блоками копировщик будет принимать посторонний сигнал и выдавать сообщение об ошибке. После начала следующего пилотсигнала снова нажмите LOAD и так далее. Всю загрузку в копировщик выполняйте в режиме "T".

А вот с выгрузкой придется немного помудрить. Если программа стандартная, то ее можно выгружать, как обычно, клавишей "A" (АВТО) в режиме "T". Если же у программы свой загрузчик, то первый блок БЕЙСИК-загрузчика и загрузчик машинного кода надо выгружать в режиме "T", а остальные блоки - в режиме "L". После такой переделки программа будет работать на компьютере "Дубна". Правда, возрастет примерно в 2 раза время загрузки и увеличится расход ленты. Надо также помнить, что переделанные таким образом программы уже не смогут работать на других машинах.

Таким приемом удалось адаптировать под "Дубну" многие из первоначально неработавших программ. Конечно, не все еще доведено до конца. Например, не поддается переделке программа SPOOKED, у которой нестандартный загрузчик выполняет как бы загрузку "с конца блока". Но это поле для новых экспериментов.

Советы и секреты.

CHRONOS - если в таблице результатов набрать YING IT BABY (причем обязательно заглавными буквами), то получите бесконечную энергию.

COBRA FORCE - если переназначить клавиши в игре на "SIMON", Вы получите бесконечную жизнь.

GEMINI WINGS - пароли:
уровень 2 - EYEPLANT
уровень 3 - WHATWALL
уровень 4 - GOODNITE
уровень 5 - SCULLDUG
уровень 6 - WIGMOUTH
уровень 7 - GREEPISH

PIPEMANIA - некоторые пароли:
уровень 5 - DISK

уровень 9 - NAIL
уровень 13 - ONCE
уровень 17 - ROPE
уровень 21 - PENS
уровень 25 - SLIP
уровень 29 - EACH
уровень 33 - RISE

AFTERBURNER: POKE 37934,0: 37935,0: 37936,0

RET STORM - POKE 37337,201

Секретами поделился Фильков Андрей Павлович из Москвы.

Письмо читателя.

Здравствуйте, "ИНФОРКОМ". Я прочитал Ваш трехтомник по изучению и работе с машинным кодом. Пока я не ознакомился с вашим изданием, я без успеха натыкался на барьер машинного кода. Мне всего тринадцать лет, но даже мне было нетрудно вникнуть в особенности процессора Z-80.

Я являюсь подписчиком "ZX РЕВЮ" на 91-92 годы. До того, как я не прочитал вашего издания, рубрику "Машинные коды" я просто пролистывал, уделяя ей ноль внимания.

Большое спасибо от всех людей, которые пользуются ПЭВМ "ZX-Spectrum".

Коллекция игр у меня небольшая и я очень люблю умные игры, такие как SHERLOCK, KNYGHT TYME и THE HOBBIT. Я очень хотел бы обратиться к экспертам, чтобы они замолвили слово об игре DUN DARACH. Сам же имею по ней очень мало информации.

P.S. Я прошу Вас напечатать в "ZX-РЕВЮ" немного информации.

Существует издание BEEP-SHOW, которое ежемесячно будет ко всем приходить, если Вы обратитесь по адресу: 416510, Астраханская об., г. Ахтубинск-6, Жуковского 24-63, UNICAL-STUDIO.

Этот журнал рассказывает о разных звуковых эффектах. Молчащая программа заговорит и запоет.

Король Юра, г. Свердловск.

* * *

Проблемы совместимости.

Нашим постоянным читателям знакомы работы Полубарьева С.В., направленные на повышение совместимости отечественных моделей Синклер-совместимых машин с их английским прародителем "ZX-Spectrum". "ИНФОРКОМ" традиционно рассматривает проблемы совместимости, как одни из самых основных и сегодня мы предлагаем Вашему вниманию две статьи, посвященные версиям "Ленинград-1" и "Пентагон-48".

Доработки портов ввода/ вывода в Sinclair ZX-Spectrum модели "Ленинград-1" (версия Зонова).

ZX-Spectrum-совместимый компьютер "Ленинград-1" (версия Зонова) создавался, по всей видимости, как максимально простой и дешевый вариант машины (или как самый доступный по элементам), о чем свидетельствуют многие примененные в нем схемотехнические решения (совмещенное поле ОЗУ 48 Кбайт и многие другие). Вероятно, именно по этой причине, логика адресации портов ввода/вывода в данной модели упрощена до такой степени, что это вызывает во многих игровых программах малоприятные эффекты мерцания бордюрной рамки экрана в такт с музыкой или звуком выстрела, а иногда и полную несовместимость, выражающуюся в "зависании" программ сразу после загрузки, или же в невозможности управления от KEMPSTON-джойстика. Однако, путем незначительных доработок схемы, эти недостатки можно полностью устранить.

Чтобы Вам полностью понять смысл вводимых изменений, здесь приводится краткая справка об устройстве аналогичных портов I/O фирменного ZX-Spectrum 48K, совместимости с которым мы добиваемся:

В фирменном компьютере для работы с периферийными устройствами используются следующие адреса:

254 (FE HEX)

- по вводу: клавиатура, ввод с магнитной ленты (далее МЛ);
- по выводу: цвет бордюрной рамки дисплея, звуковые эффекты, запись на МЛ.

251 (FB HEX)

- обслуживание фирменного узкопечатного устройства ZX-printer или аналогичных ему Timex-2040 Alphacom-32, Seicosha GP-50S.

31 (1F HEX)

- интерфейс джойстика типа Kempston.

При этом имеются следующие особенности:

1) При работе с портом 254 анализируется только сигнал на линии A0 адресной шины процессора. Поэтому с тем же самым успехом можно обращаться и к любому другому четному порту (252, 250, ... 0), поскольку при его выборке анализируется лишь разряд A0.

2) KEMPSTON-джойстик активируется в том случае, когда в адресе считываемого порта A5="0" и A0="1". При этом на шину данных передается байт, младшие 5 бит которого зависят от положения рукоятки и кнопки "FIRE" джойстика, а старшие 3 бита всегда равны "0". Таким образом, чтобы определить состояние джойстика типа KEMPSTON, программа может обратиться к любому порту с нечетным адресом в диапазоне 1...31.

3) Если ZX-Printer отсутствует в составе системы, то с порта 251 (#FB) должен считываться байт 255 (#FF). В противном случае компьютер будет неизбежно "зависать" при попытке выполнения операторов Бейсика LPRINT и LLIST, а также при загрузке некоторых игровых программ.

Теперь рассмотрим, как организована система ввода/вывода в версии "Ленинград-1":

1) По чтению порта 254 - совместимость с оригиналом полная, но запись в порт 254 выполняется при обращении к любому, а не только "четному" адресу порта (в этом причина мерцания бордюрной рамки). Причина - при выборке ИМС К555ТМ9, на которой выполнен порт вывода 254, адрес вообще не учитывается и не проверяется.

2) KEMPSTON-джойстик имеет сразу 2 дефекта: во-первых, считывается вообще по любому "нечетному" адресу, во-вторых, 3 старших бита установлены в "1" вместо "0".

3) Вытекает из предыдущего пункта: поскольку джойстик считывается по любым, в том числе и не "своим" адресам, то из порта 251 может иногда читаться не совсем то, что надо бы. В авторском варианте схемы это проходит незамеченным, поскольку "1" в старших битах означают отсутствие принтера в системе, но стоит только "занулить" их, и периодические "зависания" машины Вам надежно обеспечены.

Если Вы, прочитав вышеизложенное, узнали описание своих "бед", можете приступить к доработке. Остальным предлагаем убедиться в этом:

- выполните команду PRINT IN 31. На фирменной машине должен быть получен результат 0 (00000000 в двоичном коде). Вы, скорее всего, получите 224 или 255.

- теперь выполните PRINT IN 33 и PRINT IN 251. Если результат первой команды совпадает с предыдущим - нарушена адресация джойстика. Если то же относится и ко второй команде, да ПРИ этом еще и печатается не 255 - Вы уже наверное сталкивались, или скоро столкнетесь с проблемой загрузки игр, которые у Ваших приятелей работали нормально.

- выполните OUT 255,0 и OUT 255, 255. Если рамка экрана стала при этом менять цвет - и здесь нарушена адресация.

К счастью, разработчики платы этой машины предусмотрели в ее углу довольно большое макетное поле - "слепыш", на котором можно разместить все детали, необходимые для доработки.

Вам потребуются ИМС К555ЛЛ1 и К555ЛИ1 (по 1 шт. каждого типа) и некоторое количество тонкого изолированного провода (лучше всего марки МГТФ-0.07).

Дополнительные ИМС установите на свободное место на "слепыше" и подведите к ним питание (+5в) и "землю". Теперь приступим к собственно доработкам:

1) Замените нагрузочные резисторы джойстика на МЛТ-0.125,1кОм. Их общую точку отсоедините от "+5в" и подключите к "земле". С "землей" следует также соединить выводы 6, 10 и 13 мультиплексора D37 (D42) К555КП11. Общий провод джойстика при этом соединяется с "+5в". Это необходимо для использования джойстиков, изготовленных в стандарте "Atari", которые имеют нормально разомкнутые контакты ("Ленинград-1" спроектирован под самодельный джойстик с нормально замкнутыми контактами). Разумеется, что если Ваш джойстик уже подключен через инверторы, то резисторы менять не обязательно, а свободные входы мультиплексора заземлить все равно придется.

Примечание: автору известны 2 варианта чертежа принципиальной схемы, которые отличаются только нумерацией корпусов ИМС, поэтому приводится двойная нумерация. Вам следует определить, какая из них имеется у Вас. Для справки: упомянутый мультиплексор установлен на плате рядом с разъемом кабеля клавиатуры, ближе к краю платы.

2) Обеспечим стандартную адресацию порта вывода 254, выполненного на ИМС D39 (D40) К555ТМ9. Для этого произведем следующие изменения в схеме компьютера (рис. 1):

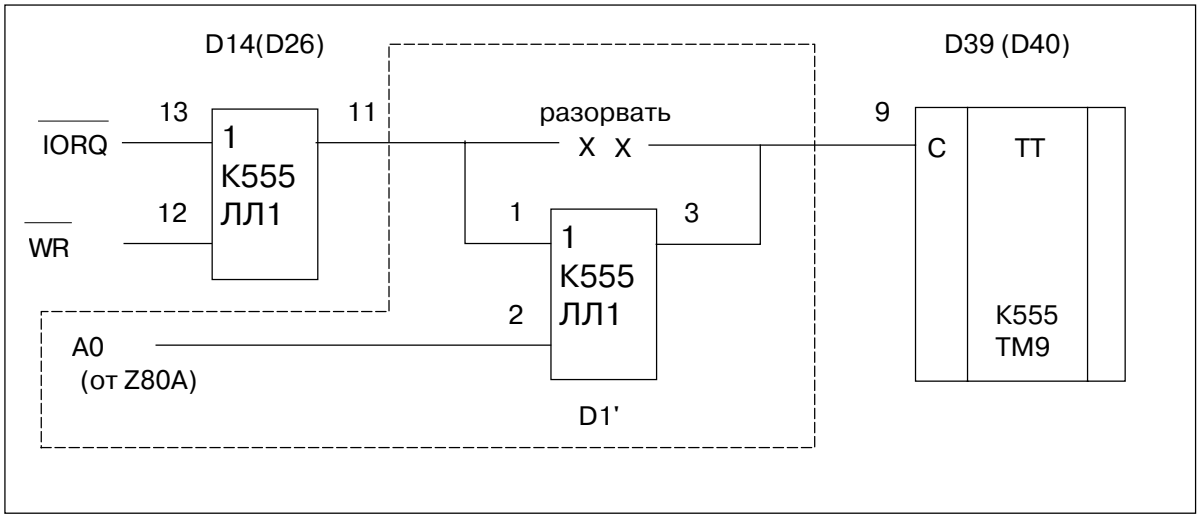


Рис. 1

Этим Вы блокируете запись в регистр К555ТМ9 при обращении по "нечетному" адресу, здесь и далее дополнительно введенные элементы имеют после номера значок " ' " чтобы отличать их от элементов базовой схемы компьютера.

3) Обеспечим нормальную адресацию интерфейса Kempston-джойстика. Для этого служит следующая доработка (рис. 2):

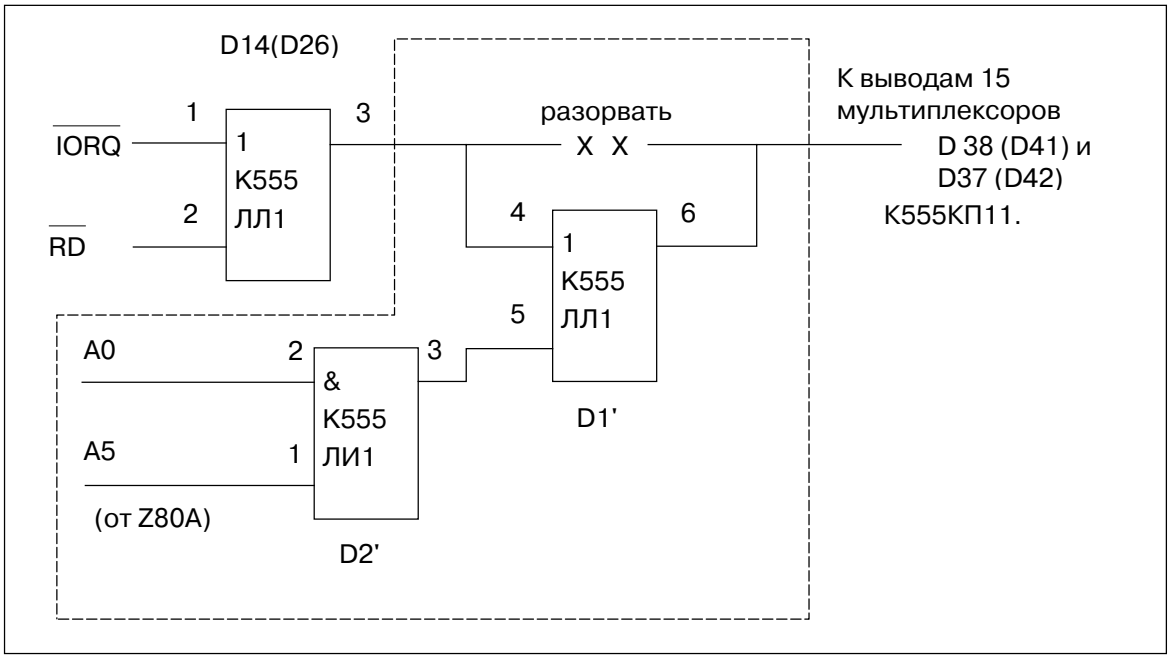


Рис. 2.

При этом обеспечивается "фирменная" выборка по следующему принципу:

- при A0="0" - клавиатура;
- при A0="1" и A5="0" - джойстик;
- при A0="1" и A5="1" - ничего не выбрано, при этом с шины данных считывается байт 255 (#FF).

Вышеприведенные доработки не обеспечивают 100% совместимости с фирменным ZX-Spectrum (следует отметить, что абсолютно совместимых моделей самодельных Спектрумов на сегодня не существует, - причины этого достаточно сложны и их обсуждение не является задачей данной заметки), но позволяет ликвидировать подавляющее большинство конфликтных ситуаций между аппаратной частью и программным обеспечением компьютера.

Еще несколько примечаний. Во-первых, помните, что "Ленинград- 1" - достаточно сложная машина, которую нетрудно вывести из строя неумелым вмешательством в схему и нелегко потом отремонтировать, поэтому не беритесь за доработку, если Вы не обладаете достаточным опытом работы с микропроцессорными системами, лучше обратитесь за помощью к более опытному человеку. Во-вторых, если Вас не волнуют проблемы с джойстиком, можете выполнить только пункт 2 (изменение выборки порта вывода 254). В-третьих, убедитесь перед началом доработки, что монтаж совпадает со схемой, и ранее до Вас никакого вмешательства в схему компьютера произведено не было.

Доработка "Pentagon-48K" для обеспечения совместимости с Sinclair "ZX-Spectrum"

На Spectrum в версии "Pentagon-48K", собранной в соответствии со схемой, не работают некоторые программы, например, "ELITE", копировщики COPY-86M, OutCopy. Одна из вероятных причин этого, возможно, заключается в следующем.

Известно, что микропроцессор Z80A имеет 3 режима обработки прерываний: IM0, IM1 и IM2. С обработкой прерываний в первых двух режимах проблемы не возникает, поскольку и в том, и в другом случае производится переход по фиксированному адресу памяти, начиная с которого размещен обработчик прерывания. При обработке прерывания в режиме IM2 периферийное устройство, выдавшее сигнал INT, должно установить на шине данных младший байт адреса, по которому в памяти находится двухбайтный вектор адреса перехода на обработчик прерываний этого устройства. Этот байт считывается процессором. Старший адресный байт, используемый для формирования полного адреса обработчика прерываний, берется процессором из регистра I. Таким образом, в памяти может быть организована таблица векторов (точек входа в подпрограммы-драйверы

внешних устройств), а сами драйверы располагаются произвольно, без привязки к жестко заданным аппаратной логикой адресам.

Хотя в компьютере ZX-Spectrum и не предусматривался режим IM2, он вполне может быть использован при соблюдении определенных ограничений. Это основывается на том, что в случае отсутствия на шине данных по заданному адресу устройства, которое эти данные выставляет, с нее считывается байт #FF (255), поскольку шина в фирменном компьютере "подтянута" резисторами 8.2 кОм к "плюсу" источника питания. Единственным стандартным источником прерываний в Spectrum является синхронизатор дисплея, выдающий 50 запросов на прерывание в секунду (в начале развертки каждого телевизионного кадра). Естественно, никаких данных на шину он при этом не выставляет, поэтому процессор примет #FF за младший байт адреса, по которому размещен вектор перехода на обработчик прерывания.

Таким образом, мы имеем возможность "перехватить" управление у стандартного драйвера прерывания, записанного в ПЗУ машины. Для этого необходимо выполнить следующее:

- разместить свой драйвер обработки прерывания в произвольно выбранном Вами месте памяти;

- вектор перехода, по которому производится передача управления драйверу, записать в ОЗУ, начиная с адреса #xxFF (т.е. старший байт адреса произволен, а младший всегда равен 255);

- записать старший байт (#xx) в регистр I процессора;

- выполнить команду IM2. Теперь при каждом прерывании будет запускаться не подпрограмма в ПЗУ, а Ваш собственный драйвер.

Такое решение может быть полезным, например, если Вы пишете программу, в которой необходима постоянная индикация текущего времени на экране (типа "электронных часов"), но параллельно с этим нужно выполнять и какие-то другие действия (скажем, редактировать текстовый документ), которые должны занимать основную часть процессорного времени. Возможно применение такого способа и при организации нестандартного драйвера клавиатуры и для других "фоновых" задач. Для любителей "покопаться" в коде фирменных программ не секрет, что немало игр могут использовать этот режим прерываний.

В "Pentagon-48K" шина данных не имеет нагрузочных резисторов, в результате чего при чтении байта с несуществующего устройства в некоторые моменты времени ее состояние может оказаться неопределенным. В режиме IM2 это может привести к "захвату" ложного адреса обработки прерывания, последствия чего очевидны. Не исключено, что в игре "ELITE" именно это и происходит.

Борьба с этим естественна - ввести в схему 8 резисторов, "подтягивающих" выводы D0...D8 процессора к шине "+5 вольт". Номинал этих резисторов для различных экземпляров машины может быть в пределах 3,3...10 кОм. Разумеется, не следует без достаточных оснований стараться установить резисторы "поменьше", поскольку это увеличивает коэффициент нагрузки и без того перегруженного процессора. Попробуйте сперва 7,5...10 кОм, и только если это не дает эффекта, уменьшайте сопротивление.

Для тех, кто имеет "Pentagon-48K" без собранного контроллера "Beta disk Interface", или с ним, но собранным по стандартной схеме (на плате установлены 4 ПЗУ К573РФ4, РФ6, 2764 или 2 ПЗУ 27128 и шинный формирователь КР580ВА86), переделки на этом заканчиваются.

Тем же, кто использует в качестве ПЗУ микросхемы 27256, заменив микросхему КР580ВА86 на перемычки, придется ее все же установить, иначе от перегрузки то и дело будет выходить из строя К561ЛН1, установленная в контроллере. Но, впаивая КР580ВА86, отсоедините ее вывод 9 (CS) от общего провода схемы и подайте на него сигнал IORQ от процессора Z80A. Это устранит конфликт на шине данных между 27256 и ВА86.

Вышеописанные доработки неоднократно проверены на практике и значительно улучшают совместимость "Pentagon-48K".

* * *

На необходимость доработки "Ленинграда" по дешифрации внешних портов указывают и другие авторы. Мы не будем здесь повторять все присланные рекомендации, но приведем несколько полезных советов от Иванова Н.Ю. из Якутии, касающихся других вопросов.

1. RGB-выход на "Ленинграде". (Рис. 3).

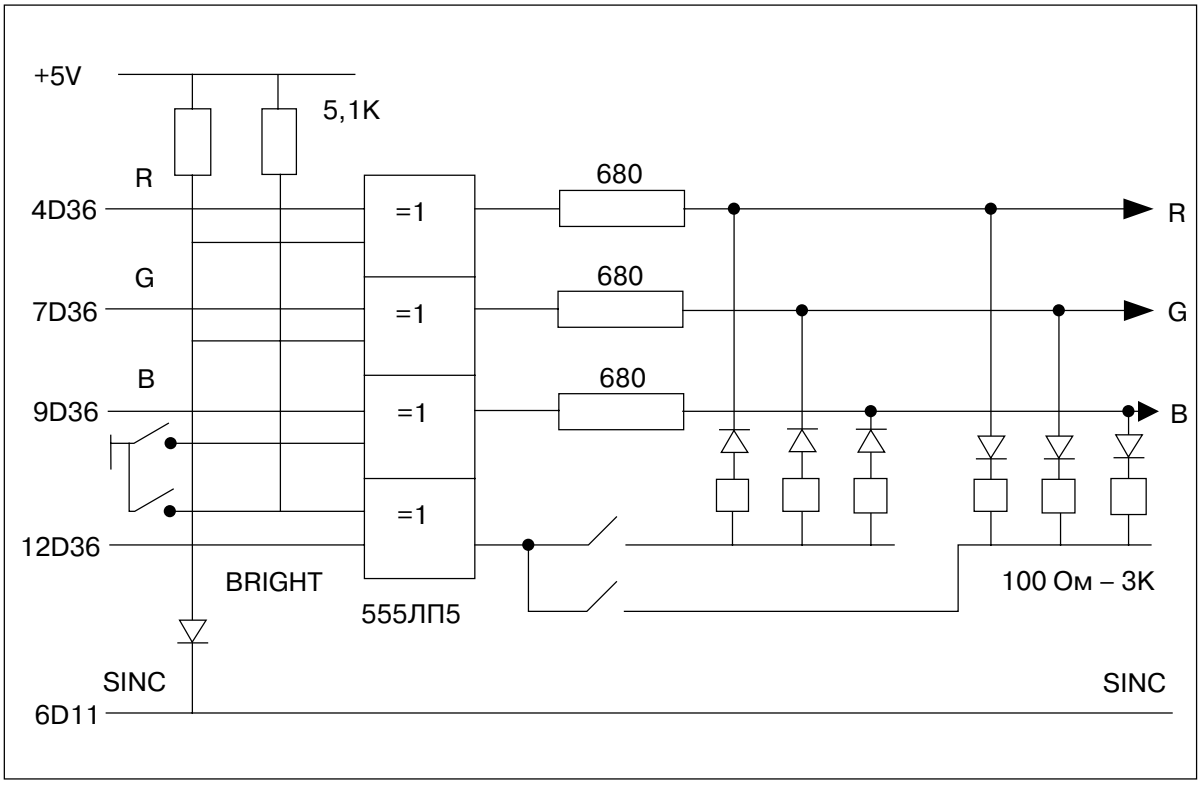


Рис. 3

Микросхема D11 (555ЛП5) желательно должна быть серии 555. Серии же 155 и 1533 работают не вполне хорошо, D40 тоже желательно серии 555.

2. Если на ножку 6 процессора подать 5V через резистор 1 кОм (обязательно по паспорту), то плата будет работать при пониженном напряжении от 4,5 до 5.5 вольт. Были даже случаи и от 3,8 до 4 В.

3. Сигнал INT можно настраивать по музыке из программ Savage или Saboteur. Эти программы очень чувствительны к длительности импульса, а недостаток этот присущ многим заводским машинам.

Проблемы "ELITE".

Мы уже писали о том, что есть возможность перехвата "Таргонов" в гиперпространстве. Многие пилоты пользуются этим для быстрого повышения игрового рейтинга. К сожалению, при этом могут возникать проблемы с нехваткой топлива на Вашем корабле.

Оригинальное решение проблемы нашли пилоты из Екатеринбурга Дремин А. (DEADLY), Кулик С. (DEADLY) и Киселев Д. А. (ELITE) из Казани. Вот их рекомендации:

Перед боем надо заправиться топливом у звезды и оттуда сделать гиперпереход. Следите, чтобы в этот момент надпись FUEL SCOOPS ON была на экране.

Эффект следующий. Во время боев в гиперпространстве все время будет гореть эта надпись, как будто дозаправка топливом не прекращалась. Вы сможете переходить от одной атаки к другой, не боясь "застрять" в космосе. Кстати, это мероприятия одновременно предохранит Вас от утечек горючего при повреждениях в бою (FUEL LEAK!).

* * *

Серьезному исследованию подвергли наши читатели поведение недокументированных космических объектов, впервые о которых сообщил нам Кислов Д.Н. из г. Челябинска (ZX-РЕВЮ-92, с.253). Напомним, что это крупные объекты неправильной геометрической формы, выпускающие в случае нападения на них несколько малых боевых аппаратов.

Семья потомственных художников Париловых из знаменитого города Палеха исследовала эти объекты вблизи и пришли к выводу, что они хоть и имеют не вполне регулярную форму, но зато этих форм три. Более того, они различаются даже в месторасположении боевого лазера (конечно это можно установить только побывав в ближнем бою с кораблем, почти вплотную).

Абсолютное большинство писем, которые пришли от пилотов, содержат мнение о том, что это астероид с "отшельниками". Наиболее глубокое исследование подготовил Шилин Д. П. из г. Симбирска. Он подтверждает данные Кислова и значительно их дополняет. Не претендуя на истину в последней инстанции, он систематизировал и обобщил данные и пришел к выводу, что "объект" обладает и свойствами астероида с поселившимися на нем "отшельниками" и свойствами "космической платформы", но факты, говорящие за первую версию, значительно более весомы. Д. П. Шилин высказал еще предположение, что авторы программы включили в нее корабль, обладающий и теми и другими свойствами для того, чтобы в условиях ограниченной памяти дать максимальный простор для фантазии играющих. Поскольку это не первый "фокус" фирмы, то в этой гипотезе что-то есть. Вспомните хотя бы о том, что три дополнительные миссии и связанное с ними оружие были недокументированны в официальной инструкции. Также были недокументированны и ряд управляющих клавиш. Совершенно ясно, что многое фирма "спрятала" для исследователей. Ход очень оригинальный, позволяющий поддерживать интерес к программе годами.

Наш корреспондент предлагает "хаккерам" включиться в исследование программы не только "снаружи", но и как бы "изнутри". Так, в качестве объекта первого удара им предлагается код программы где-то начиная с адреса 58500 и до конца ОЗУ. Здесь находятся текстовые сообщения, в которых, возможно, кто-то и найдет что-либо интересное.

Сообщения закодированы очень распространенным способом, который применяется во многих программах, особенно в адвентюрных.

Известно, что в английском языке многие слова состоят из типичных слогов, например таких, как OR, IR, EA и т.п. Это позволяет уменьшать размеры текстового блока от 1,5 до 5 раз. В ELITE закодированы даже целые слова, повторяющиеся много раз. К примеру, LASER, COMPUTER и другие. Каждому такому слову или слогу присваивается код. Например, присвоим слогу ER код 145, тогда в памяти к примеру вместо слова GLIDER можно будет разместить G, L, I, D, 145. Обычно кодируют 256 слов или слогов. Число 256 принято потому, что в этом случае код будет занимать 1 байт.

Просмотр таких закодированных сообщений может быть трудным. Это не гладкий текст, а обрывки каких-то фраз, череда разных букв. Поэтому этот способ не только сжимает текст, но еще и делает его нечитаемым. Д. П. Шилин исследовал текстовый блок "Элиты" и даже составил программу по его декодированию, но работу еще не закончил в связи с нехваткой времени. Если кто-то найдет там что-либо интересное, мы с удовольствием напечатаем. Сам же корреспондент приводит значения некоторых вскрытых им кодов:

128 - AL	129 - LE
131 - GL	133 - CE
134 - BI	137 - ES
138 - AR	139 - HA
140 - IN	141 - DI
142 - RE	145 - AT
144 - ER	146 - EB
148 - RA	149 - LA
150 - VE	151 - TI

152 - ED 153 - OR
154 - QU 155 - AH
156 - TE 158 - RI
159 - ON 165 - SYSTEM
174 - UNIT 187 - LASER
203 - ST 206 - CARGO
215 - COMPUTER 227 - LARGE

Следует также добавить, что обычно в программах такие коды начинаются с какого-то определенного числа, ведь надо же оставить еще место для таблицы символов. В стандартном знакогенераторе символы расположены, начиная с 32 по 127.

Очень часто в программах для увеличения скрытности текста этот отрезок смещается в какую-либо сторону, то есть коды символов специально делают несовпадающими со стандартными, однако, к счастью, создатели "Элиты" этого не сделали.

Примечание "ИНФОРКОМа":

За примерами того, как кодируется текст в игровых программах далеко ходить не надо. Этот прием чрезвычайно широко распространен. Но ведь кодируют не только текст, но и графику. Мы как раз подготовили статью о том, как это делают на примере программы JET SET WILLY. В одном из ближайших выпусков напечатаем и Вы сможете очень элегантно хранить графику своих программ с малым расходом памяти.

47-ая галактика: тупик или дорога к новый мирам?

Мы неоднократно упоминали о появлении 47-ой галактики в версии Родионова. Большинство наших читателей полагают, что это дефект программы, вызванный неаккуратным снятием защиты. На это указывают и неадекватные суммы кредов на счетах и сумасшедшее вооружение. Но если внешний дефект смог вызвать появление 47-ой галактики, то возникает вопрос: "А нельзя ли нормальным путем проникнуть в новые миры?". По крайней мере в двух письмах начато исследование этого вопроса. Но начнем все по порядку.

Несколько месяцев назад мы предложили читателям начать атаку на тот отгрузочный блок, который сохраняется на ленте при сохранении состояния программы. Многие уже в этом преуспели и мы об этом тоже писали. К чему же ведет это исследование дальше? Давайте посмотрим.

Сначала мы представим выводы, сделанные Балисом Линасом из Каунаса.

1...11 - имя пилота в символах ASCII. Байт, равный нулю информирует об окончании имени.
12 - правовой статус.
0 - clean
1...49 - offender;
50...255 - fugitive.

Как видите, правовой статус - величина не дискретная, т. е. может постепенно ухудшаться или улучшаться.

13...15 - боевой рейтинг. Значения этой трехбайтной переменной таковы:
0,0,0...255,8,0 - HARMLESS
0,9,0...255,16,0 - M. HARMLESS
0,17,0..255,32,0 - POOR
0,33,0..255,64,0 - AVERAGE
0,65,0..255,128,0 - AB. AVERAGE
0,129,0.255,255,2 - COMPETENT
0,0,3...255,255,10 - DANGEROUS
0,0,11..255,255,25 - DEADLY
0,0,26..255,255,255-ELITE
16...17 - ?
18 - номер карты галактики (0 -первая, 1 - вторая и т. д.).

19 - CASH
1…255 - 1000 - 255000 Cr.
20 - CASH
1…255 - 256000 - 65280000Cr
21 - CASH
1…255 - 0,1 - 25,5 Cr
22 - CASH
1…255 - 25,6 - 6528 Cr
Сумма денег образуется суммированием всех этих четырех байтов.
24…40 - наличие груза в корабле (см. ZX-РЕВЮ-92, с.254).
41 - грузоподъемность корабля. Если здесь содержится 0 и в байтах 24…40 тоже нули, то в корабле находятся беженцы (REFUGEES).
42 - FRONT LASER:
1 - pulse laser:
2 - beam laser;
3 - military laser;
4 - mining laser.
43 - REAR LASER
44 - LEFT LASER
45 - RIGHT LASER
46 - байт первой миссии. Если не 0, то дают миссию.
47 - FUEL
48 - количество ракет.
49 - LARGE CARGO BAY (В номере ZX-РЕВЮ-92, с. 254 здесь и далее ошибка).
50 - E. C. M. SYSTEM
51 - Дополнительный PULSE LASER
52 - Дополнительный BEAM LASER
53 - FUEL SCOOPS
54 - ESCAPE POD
55 - ENERGY BOMB
56 - ENERGY UNIT
57 - DOCKING COMPUTER
58 - GALACTIC HYPERDRIVE
59 - Дополнительный MILITARY LAS.
60 - Дополнительный MINING LASER
61…66 - данные по галактике, в которой Вы находитесь.
67 - ?
68 - Координата Y курсора на большой карте галактики.
0 - вверху; 255-внизу.
69 - ?
70 - Координата X курсора на большой карте галактики, 0 - слева; 255 - справа.
71 - 74 - ???
75 - 91 - содержат информацию о наличии товаров на станции.
92 - ?
93 - наличие дополнительного оружия на корабле:
64…127 - CLOAKING DEVICE;
128..191 - E.C.M. SYS. JAMMER;
192..255 - и то и другое.
94…102 - ??

Теперь рассмотрим внимательно байты 61… 66, содержащие данные по галактике, в которой Вы находитесь.

	1	2	3	4	5	6	7	8
61	74	148	41	82	164	73	146	37
62	90	180	105	210	165	75	150	45
63	73	144	33	66	132	9	18	36
64	2	4	8	15	32	64	128	1
65	83	166	77	154	53	106	212	169
66	183	111	222	189	123	246	237	219

Обратите внимание на то, что содержимое данных по каждой галактике можно получить логическим удвоением данных по предыдущей галактике. Мы говорим именно о логическом, а не об арифметическом удвоении именно потому, что если при удвоении образуется число, большее чем 255, то 255 от него отбрасывается, а в таблицу заносится остаток.

	41	42	43	44	45	46	47	48
61	129	3	6	12	24	48	96	192
62	161	67	134	13	26	52	104	208
63	245	235	215	175	95	190	125	250
64	72	144	33	66	132	9	18	36
65	80	160	65	130	5	10	20	40
66	241	227	199	143	31	62	124	248

Таким образом, облетев все восемь галактик и полетев дальше, Вы попадаете в ... первую галактику.

И вот что по этому поводу пишет наш читатель из Каунаса:

"Если изменить значение хотя бы одного байта, мы попадем в новую систему из восьми галактик и названия планет не будут повторяться. Возможно, что в игре есть какие-то условия, позволяющие перелететь в другие системы, но я их пока не обнаружил. Сколько всего галактик в игре, я даже не берусь ответить. Прошу всех о помощи! Надо найти способ перелета в другие галактики! Я уверен, что он есть, и большая вероятность, что там скрывается планета RAXXLA. "

Мы не случайно приводим слова нашего читателя из Литвы дословно. У Вас есть прекрасная возможность сравнить его выводы с выводами, полученными в то же время, но на несколько тысяч километров восточнее.

Семейный экипаж из села Гайтер Хабаровского края в составе Троеглазова Павла Герасимовича и его сына Геры провел огромную многодневную работу. Облетев восемь галактик, не уклоняясь ни от одного боя и отгружаясь на каждой планете они тщательно исследовали 102-байтный блок и тоже раскрыли циклический характер изменения байтов с 61 по 66 при переходе от 1-ой до 8-ой галактик. Но они пошли дальше и исследовали галактику N47, в которую на некоторых версиях "Элиты" можно попасть "нелегальным" путем. Как оказалось, с ней тоже связана группа из восьми галактик, данные по которым представлены ниже:

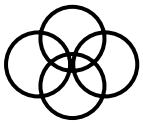
И вот, что пишут они:

"А где галактики между 8-ой и 41-ой? Они есть! Мы обнаружили за короткое время еще 36 галактик и прогулялись по ним. Мы также подозреваем, что галактик может оказаться больше, чем 48... В глубинах этих неизвестных еще галактик можно, по-видимому, повстречать и корабли поколений и пресловутую планету RAXXLA. Ищите и обрящите"

Вот как разыскивались эти новые галактики. В первоначальном варианте (система LAVE) исследователи обнуляли один за другим каждый байт из шести (61...66) и каждый раз попадали в новую галактику. Пока еще неясно является ли каждая из вновь открытых галактик "опорной" для генерации целой серии из 8-ми галактик или эти галактики - часть одной системы. Не выяснены также все вопросы, связанные с "устойчивостью" новых систем. Дело в том, что после перехода из первой во вторую и т. д. и вновь после возврата в первую могут наблюдаться сбои в работе программы. Возможно, есть какие-то законы, ограничивающие бесконечно возможное количество галактических систем, зато какие

открытия там возможны! Так, например, при 61-ом байте, равном нулю, в самом центре 5-ой галактики есть ЧЕТВЕРНАЯ ЗВЕЗДА:

RAGETI

ADIIS  TEAMA

BESOORRA

Обнулив 62-ой байт и прогулявшись по полученной "восьмерке", наши корреспонденты с интересом увидели, что во второй галактике созвездия неожиданно похожи на земные: М. Медведица, Цефей, Дракон, Рыба, Лира. Третья галактика - необычно высоко социально развита, в шестой есть созвездие, похожее на Северную корону, а восьмая - кишит бандитами всех мастей.

Вот к таким интересным открытиям может привести кропотливое исследование. Теперь дело совсем за малым - надо найти способ и условия возможности перемещения между галактическими системами. Но этот малый шаг можно пройти только объединив усилия всех пилотов и растянуться он может надолго.

Авторы исследования полагают, что они затронули только "краешек" глобальной проблемы доведения игры до логического конца.

"Новые" галактики - это еще не все интересное из того, что удалось установить семейному экипажу. Интерпретация байтов в основном та же, что и представленная выше, но зато есть некоторые любопытные комментарии.

1. Байты 24 - 40 - товары.

Здесь есть особенность. Максимальные цифры (255) можно вносить только в 37-ой, 38-ой и 39-ый байты. Сумма же остальных байтов (товаров) не должна превышать 20 (35, если у Вас есть LARGE CARGO BAY). Все же, что выше этой нормы, будет при попытке продажи умножено на существующую в данном месте цену и списано с Вашего банковского счета. А вот, что будет, если Вы доведете свой счет таким способом до нуля и будете продавать дальше, - узнайте сами.

2. Байты 42... 44. Засылая сюда число, большее чем 4, Вы получаете лазеры с самыми невероятными названиями. 99% их работают, как технологические и все имеют хорошую скорострельность и огромную силу удара. Замечено, что бортовые лазеры имеют примерно вдвое более высокую скорострельность, чем носовые и кормовые.

Интересно, что на этих "супер-лазерах" можно хорошо подзаработать. При замене такого оружия на стандартное (из меню) на Ваш счет в банке поступает немалая сумма (30 тыс. и более) компенсации, но бывает и наоборот. Проверьте!

3. Байт 46 - миссия "Сверхновая".

Обычное значение - 0. В зависимости от величины засланного сюда числа, Вы получите миссию сразу или после нескольких перелетов с планеты на планету.

4. Интересно поэкспериментировать с байтами из группы 47-60.

Засылая в 47-ой байт число 255, Вы получите на карте круг размером 25,5 световых лет и сможете перелететь на самую дальнюю планету этого круга за один обычный перелет, но если перелетите на ближайшую, излишки топлива у Вас отберут и останется обычный радиус в 7 световых лет.

Заслав в 48-ой байт число 100 или 200, Вы получите соответственно 100 или 200 ракет. При пуске ракет они могут появляться стаей на экране, заслоняя "пейзаж". Можно стрелять очередями по 5-6 ракет.

Во все байты, кроме 50-го можно засылать максимальное число -255.

5. С группой байтов 94...102, увы и этому экипажу справиться пока не удалось. Что ж, будем ждать новых открытий.

Кто может помочь с ремонтом японского дисковода TEAC - отзовитесь. Вышла из строя 40-ножечная микросхема на плате 15532097-05B.

745100, Туркмения, Балканская обл., г. Небит-Даг, кв-л 211, д. 70, кв. 14, Виннику.

СОВЕТЫ ЭКСПЕРТОВ

Дорогие друзья!

Много читателей с интересом следят, как развивается "раскрутка" программы ELITE на страницах ZX-РЕВЮ. Вместе с тем, они справедливо отмечают, что давно бы пора найти этой "программе десятилетия" достойную замену и дружно предлагают на эту роль известную программу "ACADEMY" (развитие программы TAU CETI).

Сегодня мы предлагаем Вашему вниманию необычную экспертную проработку, выполненную сразу двумя авторами. Они проживают в разных городах и, возможно, даже не знают друг друга, но их объединяет любовь к этой программе. Мы же взяли на себя скромную задачу компиляции представленных ими материалов в единый блок.

Фамилии экспертов приведены в порядке поступления материала.

ACADEMY (TAU CETI II)

Автор: Pete Cooke.
Фирма: CRL Group PCL.
Год: 1986.

Эксперты:
Хоминич Р.В., г. Киев
Жаров Р.Н., г. Херсон

Академия Галактической Корпорации по повышению квалификации пилотов скиммеров (GASP) была основана в 2213 году после несчастного случая на 61 Cygnus, когда пилот новичок, выбрав неисправный аппарат, ошибочно произвел стыковку с реактором и половина планеты погибла под расплавленной лавой. Галактическая Корпорация приняла решение создать специальный тренировочный центр подготовки элитного корпуса пилотов, летающих на новейших военных скиммерах для применения в колониях и аванпостах Вселенной.

Выпускники академии направлялись в ряды специального корпуса скиммеров, где кандидатам требовалось успешно выполнить двадцать заданий, состоящих из пяти уровней, по четыре задания каждый.

Будьте внимательны! И, возможно, Вас будет ждать награда Галактической Корпорации.

Итак, прочитав сценарий игры, Вы нажимаете "FIRE" и переходите в главное меню.

Работа с меню.

Выбор осуществляется путем перемещения стрелки курсора и нажатием "FIRE". Выбранная Вами опция окрашивается в белый цвет.

Главное меню имеет следующие опции:
Accept Mission - выполнение миссии;
Select Mission - выбор миссии;
Select Skimmer - выбор скиммера;
Progress Report - рапорт о выполнении уровня;
Tape Menu - меню работы с лентой;
Enter a New Cadet - прием нового кадета;
View/Redefine Keys - просмотр/выбор клавиш;
Кроме этого в меню указаны: 00:00:00 - часы;
Sound - звук: "V"-вкл. , "x"-выкл;
Mission - текущая миссия;
Skimmer - выбранный скиммер;
Cadet - имя и фамилия кадета.

Просмотр/выбор клавиш.

Клавиши управления:
O - влево; P - вправо;
S - вверх/ускорение;
X - вниз/торможение;
N (SPACE) - огонь лазера/выбор;
M - огонь ракетой;
A - огонь противоракетным снарядом;
F - огонь осветительной ракетой;
B - сбрасывать бомбу;
V - круговой осмотр;
H - увеличение высоты;
G - уменьшение высоты;
J - прыжок;
L - приземление;
I - инфракрасное видение;
R - доклад о состоянии систем корабля.
Alter Keys - выбор клавиш: Kempston Joystick - джойстик;
Return To Main Menu - возврат в главное меню;
"BREAK" - восстановление стандартных клавиш.

Если у Вас подключен джойстик, то он будет выбран автоматически. Не рекомендуется менять клавиши управления (кроме первых пяти) во избежание путаницы.

Прием нового кадета.

Прошение о приеме в группу подготовки пилотов скиммеров.
Форма: VV1B/6702 (3 экземпляра).
Дата прошения 7/11/2047.
Имя (имя и фамилия): ....
Дата рождения (день/месяц/год): ../../....

Начать с I уровня (да/нет): ...

Для заполнения бланка введите свои имя и фамилию, а затем дату рождения. Год, видимо, надо рассчитывать, учитывая сегодняшнюю дату: 2047 год. Поэтому, если Вам 20 лет, то Вы родились в 2027 году и т. д. Вы можете выбрать любой год с 1901 по 2040. После введения даты в скобках появится день недели Вашего рождения.

Последний вопрос имеет смысл, если Вы находитесь на II-ом уровне или выше.

Меню работы с лентой.

Load Game File - загрузить отложенную ситуацию.
Save Game File - записать текущую ситуацию.
Load Ship Designs - загрузить созданный корабль.
Save Ship Designs - сохранить созданный корабль.
Return To Main Menu - возврат в главное меню.

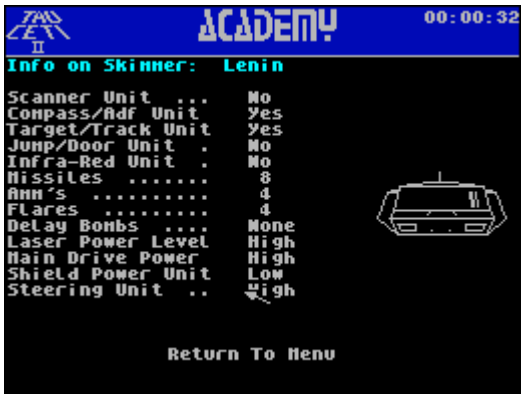
Рапорт о выполнении уровня.

В рапорте указаны имя кадета, уровень игры и степень выполнения каждой миссии.

Миссии считается выполненной, если Вы набрали не менее 90%. Уровень считается пройденный, если Вы набрали средний результат, близкий к 100%, выполнив все четыре задания.

Выбор скиммера.

Вы можете выбрать один из трех стандартных скиммеров:
GCS Lenin (Ленин),
GCS Lincoln (Линкольн),
GCS Wilson (Вильсон).



Кроме этого, Вы можете спроектировать скиммер по своему усмотрению (одновременно в памяти может быть три личных скиммера).

Тактико-технические характеристики стандартных скиммеров Вы можете просмотреть с помощью функций приведенных ниже:

Info on this Skimmer - информация о скиммере, отмеченном "V";
View Panel - просмотр панели скиммера;
Design New Skimmer - спроектировать новый скиммер;
Selection Complete - выбор завершен.

Проектирование скиммеров.

Для своего скиммера Вы можете выбрать любое оборудование, соблюдая два условия: его стоимость не должна превышать 100 MCr и его вес не должен превышать 100 единиц.

После выбора Design New Skimmer, перед Вами появится таблица:

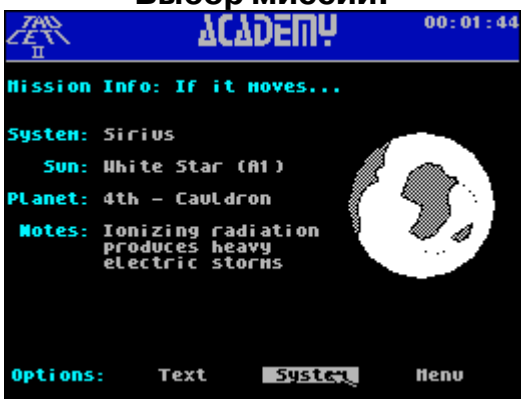
Scanner Unit - сканнер;
Compass/Adf - компас/азимут;
Target/Track Unit - цель/курс;
Jump/Door Unit - гиперпрыжок/стыковка;
Infra-Red Unit - инфракрасное видение;

Missiles - ракеты;
Amm's - противоракетные снаряды;
Flares - осветительные ракеты;
Delay Bombs - бомбы замедленного действия;
Laser Power Level - уровень мощности лазера;
Main Drive Power - главный энергетический привод;
Shield Power Level - энергия защиты;
Steering Unit - блок управления.

Отметьте знаком "v" нужный ответ:
No (нет); Yes (да);
None (не требуется); 4 (штуки);
8 (штук); Low (низкий);
Med (средний); High (высокий).
После чего выберите Design Complete (проектирование завершено).
Если Вы решили отказаться от проекта, то выберите Abandon Design (отказ от проектирования).

Вы перешли в режим проектирования панели. Выберите структуру панели и ее цвет, затем расположите приборы (Place Instruments).
Появятся сообщения: Put Viewscreen (разместить обзорный экран) и Undo Last Step (отменить последнее действие).
Выберите место для экрана и поместите туда белый прямоугольник, нажав "FIRE". После чего выберите Menu (меню) и повторите операцию с окном сообщений (Message window). Желтый прямоугольник - поле занято, белый - поле свободно.
Далее размещается выбранное Вами оборудование. Последними размещаются: Height Gauge (шкала высоты), Shield Gauge (уровень защитной энергии), Fuel Gauge (шкала запаса топлива), Laser Temp (температура лазера) и Speed Gauge (шкала скорости).
На этом проектирование завершено. Осталось только присвоить имя новому скиммеру. При желании, Вы можете записать на ленту созданные корабли (см. меню работы с лентой).

Выбор миссии.



Для выбора миссии отметьте ее знаком "v". Используя Info in this Mission (информация о миссии), Вы сможете ознакомиться с целью (Text) и планетной системой (System), в которой Вам предстоит действовать.
Функция Load in Next Level (загрузить следующий уровень) будет заблокирована до полного выполнения всех миссий этого уровня.
Прежде, чем приступить к выполнению выбранной миссии, Вы должны внимательно изучить предоставленную информацию по звездной системе.

Уровень I.

Миссия I: If it moves... (Если оно движется...).
Цель: Уничтожение вторгшихся роботов.

Счет очков: Процентное основание.
Примечание: Нет системы гиперпрыжков, нет системы поддержки, одна (1) G.L.V. база.
Рекомендуемый скиммер: GCS Ленин.
Система: Sirius (Сириус).
Солнце: Белая звезда (A1).
Планета: 4-ая Cauldron (Котел).
Примечание: Ионизирующее излучение тяжелых элементов, электрические бури.

Миссия II: Red Dawn (Красный рассвет).
Цель: Уничтожение автоматических заводов во всех квадратах.
Счет очков: Процентное основание.
Примечание: Полная система гиперпрыжков и центр снабжения, одна (1) G.L.V. база.
Рекомендуемый скиммер: GCS Вильсон.
Система: Betelguese (Бетельгейзе)
Солнце: Красный гигант (M2).
Планета: 6-ая Eventide (Вечер).
Примечание: Гигантское красное солнце, господствующее на небе, делает инфракрасную систему бесполезной.

Миссия III: Meltdown (Плавка).
Цель: Реактор в критическом состоянии - должен быть уничтожен.
Счет очков: 15 минут до расплавления.
Примечание: Нет системы гиперпрыжков, нет системы поддержки, одна (1) G.L.V. база.
Рекомендуемый скиммер: GCS Ленин.
Система: Звезда Ван Маанена.
Солнце: Желтый карлик (dG5).
Планета: Escot.
Примечание: Одноликий мир, маленькая полярная колония.

Миссия IV: Softly Softly (Тихо-тихо).
Цель: Определить местонахождение и возвратится на базу.
Счет очков: По времени.
Примечание: Зона недавно заминирована оборонным сектором Галактической Корпорации. Причина минирования - административная ошибка.
Рекомендуемый скиммер: GCS Линкольн.
Система: Rigel (Ригель).
Солнце: Бело-голубая звезда (B8)
Планета: 12-ая Ice-world (Ледяной мир).
Примечание: Аванпост обороны, нет гражданских сооружений.

Уровень II.

Миссия I: Cipher (Шифр)
Цель: Собрать и смонтировать кодовые устройства реакторов.
Счет очков: Четыре кодовых блока.
Примечание: Нет системы гиперпрыжков, нет системы поддержки, одна (1) G.L.V. база.
Рекомендуемый скиммер: GCS Линкольн.
Система: Vega (Вега).
Солнце: Белая звезда (A0).
Планета: 5-ая Homebase (Родная база)
Примечание: Маленькая, недавно сформированная колония.

Миссия II: At the OK Coral (В загоне все в порядке).
Цель: Уничтожение бродячих роботизированных систем.
Счет очков: Процентное основание.
Примечание: Нет системы гиперпрыжков, система поддержки, одна (1) G.L.V. база.
Рекомендуемый скиммер: GCS Ленин.
Система: Avior.
Солнце: Бело-голубая звезда (09)
Планета: 3-ая Corral (Кораль).
Примечание: Одноликий мир.

Миссия III: Where to Guv?
Цель: Пираты захватили гипертранспортную сеть - требуется их уничтожить.
Счет очков: Процентное основание.
Примечание: Полная система гиперпрыжков и система поддержки, одна (1) G.L.V. база.
Рекомендуемый скиммер: GCS Линкольн.
Система: Sirius (Сириус).
Солнце: Белая звезда (A1).
Планета: Greengage (Зеленый заложник).
Примечание: Озоновый слой создает необычные световые эффекты.

Миссия IV: Hide and seek (Прятки).
Цель: Уничтожить комплекс солнечных дисков и вернуться на G.L.V. базу.
Счет очков: Процентное основание.
Примечание: Уничтожение дроидов может быть полезным.
Рекомендуемый скиммер: GCS Вильсон.
Система: Beta Hydri (Бета Гидры).
Солнце: Желтая (G1).
Планета: Engels (Энгельс).
Примечание: Одна из первых 4-х колоний.

Уровень III.

Миссия I: Laserium (Лазериум).
Цель: Уничтожение роботов-захватчиков.
Счет очков: Процентное основание.
Примечание: Нет системы гиперпрыжков, небольшая поддержка, одна (1) G.L.V. база.
Система: Groombridge 34.
Солнце: Красная звезда (M2).
Планета: Единственная планета - Rayet (Лучистая).
Примечание: Малая новая колония.

Миссия II: Hades II (Гадес II).
Цель: Обнаружение и уничтожение вышедших из под контроля роботизированных систем.
Счет очков: Процентное основание.
Примечание: Небольшая система гиперпрыжков и центры поддержки, одна (1) G.L.V. база.
Рекомендуемый скиммер: GCS Вильсон.
Система: Procyon (Проционн).
Солнце: Белая звезда (F5).
Планета: 15-ая Hades II.
Примечание: Очень удалена от солнца и постоянно покрыта льдом.

Миссия III: The Sands of Time (Песок времени).

Цель: Сеть реакторов должна быть уничтожена до эвакуации планеты.
Счет очков: Временное основание (1 час).
Примечание: Нет системы гиперпрыжков, нет системы поддержки, одна (1) G.L.V. база.
Рекомендуемый скиммер: GCS Ленин.
Система: Fomalhaut (Фомальгаут).
Солнце: Белая звезда (A3).
Планета: Foma-3.
Примечание: Изолированный аванпост.

Миссия IV: Mission Improbable (Миссия "Невероятная").
Цель: Собрать и смонтировать кодовые блоки реакторов и возвратиться на G.L.V. базу.
Счет очков: Процентное основание.
Примечание: Deaf-фильтр применяется во многих реакторах.
Рекомендуемый скиммер: GCS Линкольн.
Система: Delta Pavonis (Дельта Павониса).
Солнце: Желтая (G1).
Планета: 3-я Dawnmist (Утренний туман).
Примечание: без примечаний.

Уровень IV.

Миссия I: Ceti Revisited (опять Тау Кита).
Цель: Сбор и монтаж кодовый блоков реакторов.
Счет очков: По мере сбора блоков.
Примечание: Система гиперпрыжков, небольшая поддержка, одна (1) G.L.V. база.
Система: Тау Кита.
Солнце: Желтая звезда (G8).
Планета: Третья планета.
Примечание: Одна из четырех первых колоний.

Миссия II: Out of the Frying Pan (Из огня …).
Цель: Обнаружение и уничтожение роботов вторжения.
Счет очков: Процентное основание.
Примечание: Небольшая система гиперпрыжков, полная поддержка, одна (1) G.L.V. база снабжения.
Система: Ригель.
Солнце: Бледно-голубая (O9)
Планета: 9-ая "Безымянная".
Примечание: нет примечаний.

Миссия III: Don't panic. (Без паники).
Цель: Уничтожение реакторов и солнечных дисков.
Счет очков: Процентное основание.
Примечание: Нет системы гиперпрыжков, небольшая поддержка, одна (1) G.L.V. база.
Система: Эпсилон Инди.
Солнце: Оранжевая звезда (K5).
Планета: Adams.
Примечание: Радиация делает систему ночного зрения бесполезной.

Миссия IV: Needle in a Haystack (Иголка в стоге сена).
Цель: Определить собственное месторасположение и возвратиться на G.L.V. базу.
Счет очков: По времени.
Примечание: Предбазовая зона усеяна суперминами.

Система: Денебол.
Солнце: Белая (A3).
Планета: "Пыльный шар".
Примечание: без примечаний.

Уровень V.

Миссия I: Coal Mine (Угольные копи).
Цель: уничтожение враждебных роботов.
Счет очков: Процентное основание.
Примечание: Нет системы гиперпрыжков и системы поддержки, одна (1) G.L.V. база.
Система: Epsilon Erandi.
Солнце: Красная звезда (K2).
Планета: Третья планета.
Примечание: Миссия на ночной стороне планеты.

Миссия II: PAZ!
Цель: Обнаружение и уничтожение роботов вторжения.
Счет очков: Процентное основание.
Примечание: Нет системы гиперпрыжков и системы поддержки, одна (1) G.L.V. база. Роботы могут иметь суперракеты.
Система: Процион.
Солнце: Белая звезда (F5)
Планета: 7-ая "Нью-Марс".
Примечание: нет примечаний.

Миссия III: Protector. (Защитник).
Цель: Уничтожение следящей системы на Дельте.
Счет очков: Процентное основание.
Примечание: Нет системы гиперпрыжков и системы поддержки, одна (1) G.L.V. база снабжения.
Система: Альфа Центавра.
Солнце: Желтая звезда (G2).
Планета: Дельта.
Примечание: Ближайшая колония к Солнечной системе.

Миссия IV: The Shepherd (Пастух).
Цель: Найти и выстроить сторожевые башни рядом с G.L.V. - базой.
Счет очков: Пять сторожевых башен по мере сборки.
Примечание: Все пространство минировано.
Система: Альтаир.
Солнце: Белая (A7).
Планета: "Плато".
Примечание: без примечаний.

Полезные советы по сборке скиммеров.

Стандартные скиммеры не всегда являются удобными, поэтому рекомендуется пользоваться личными кораблями. Для проектирования корабля принимается во внимание наличие системы гиперпрыжков, системы поддержки, класс звезды и, конечно, Ваша цель. Возможно, Вам придется сделать несколько разведывательных вылетов, прежде чем Вы найдете приемлемый вариант.

Ниже приводится описание отдельных систем корабля.

Scanner Unit (радар) - рекомендуется ставить на все корабли, значительно облегчает выполнение задачи. Позволяет обнаруживать объекты и проходы в минных полях. Голубой квадрат в центре - зона непосредственного контакта. Граница квадрата - предельно допустимая дистанция приближения к минным полям.

Compass/Adf (Компас/УНБ Указатель Направления на Базу) - обязательный прибор (хотя в некоторых ситуациях опытный пилот может обойтись и без него). В случае его повреждения вдали от базы, Ваша гибель почти гарантирована. Азимут всегда указывает на Вашу G.L.V. базу.

Targeting/Tracking/Drain: Первый блок - это электронный прицел (может быть заменен или дополнен механическим).

Второй - настройка на радиомаяк.

Третий - указывает на то, что Вы подверглись нападению. Красный цвет индикатора указывает на то, что прибор включен. Последний блок может стать зеленого цвета (близкая опасность).

Jump/Door Unit - блок необходим при наличии системы гиперпрыжков или необходимости стыковки с другими объектами.

Система JUMP активизируется только в непосредственной близости от Jump-площадок. Запускается командой JUMP, красный цвет индикатора Jump указывает, что Вы вошли в зону действия платформы для гиперпрыжков.

Система DOOR нужна для открывания проходов в силовых полях.

Работает автоматически. Для открывания базы она не требуется. Красный цвет индикатора Door указывает, что стыковка запрещена (зеленый - разрешена).

Infra-Red Unit - прибор ночного видения. Дает контуры окружающих объектов. Необходим на планетах с резкой сменой дня и ночи (может быть заменен на осветительные ракеты). Включается клавишей Infra-red.

Missiles - самонаводящиеся ракеты (эффективное оружие). Пуск возможен только при наличии захвата цели. Пуск выполняется клавишей Fire miss.

Amn's - система противоракетной обороны. Энергетический залп возможен после получения сообщения от бортового компьютера о ракетной атаке. Пуск клавишей Fire amm.

Flares - осветительные ракеты. Пуск клавишей Fire Flar. Обеспечивают более хорошую видимость. По сравнению с инфракрасной системой, но имеют ограниченное время действия.

Delay Bombs - бомбы замедленного действия. Это чрезвычайно мощное оружие. Уничтожают все объекты в радиусе действия, в том числе и скиммер. После сбрасывания требуется срочно покинуть заминированный квадрат.

Laser - основное оружие Вашего скиммера. Поражает практически все объекты, но при стрельбе быстро перегревается, поэтому его не следует использовать бесконтрольно.

Main Drive Power – главный энергопривод. Этот блок определяет максимальную скорость Вашего корабля, а также время ее достижения.

Shield Power Unit - отсек энергетической защиты. Это генератор защитного поля. Определяет максимальную интенсивность поля и скорость ее восстановления до номинального уровня.

Steering Unit - блок управления. Определяет инерционность Вашего корабля в управлении.

Выполнение миссии.

Итак, цель определена, скиммер выбран и Вы приступили к выполнению миссии.

Перед вами появилась панель корабля, наверху указано положение скиммера, счет в % и бортовые часы. Компьютер дает Вам сообщение о том, что все системы готовы и просит ввести команду.

Наберите "HELP" для получения подсказки.

Сводка боевых команд

HELP - помощь;
LAUNCH - вылет скиммера;
PAUSE - пауза;
QUIT - выход в главное меню;
SIGHTS ON - прицелы включены;
SIGHTS OFF - прицелы выключены;
WAIT - ожидание (около 5 минут);
EQUIP - снаряжение скиммера;
STATUS - состояние систем корабля (аналог "R");
CODES - коды запирающей системы реактора;
DEAF - клавишный электронный звуковой фильтр;
LOOK – осмотр.

EQUIP - в этом режиме Вы можете пополнить запасы оружия и произвести дозаправку (Refuel) и ремонт (Repair). Режим работает только в состоянии стыковки с базой или ремонтным центром.

STATUS - информация о состоянии бортового оборудования.

LOOK - информация о местонахождении корабля.

DEAF - включение Цифрового Электронного Аудио Фильтра, потренируйтесь с этим устройством. Оно сделано специально для защиты от роботов, поскольку они страдают глухотой и дальтонизмом. Этот режим включается автоматически при попытке стыковки с объектом. Для нахождения пароля Вам необходимо нажать Play и затем с помощью курсора

повторить звуковую мелодию. В случае неудачи Вы можете запросить другую мелодию с помощью Reset.

CODES - этот режим предназначен для монтажа кодовых блоков к реакторам. Используется в трех миссиях (2-1, 3-4, 4-1). Облетев реактор, Вы получаете определенное количество кодовых фишек. Обойму можно просмотреть в правом окне командами "вверх"/"вниз". После просмотра Вы с помощью команд Undo и Place переносите какой-либо блок в левое окно. Ваша задача совместить три кода для создания устойчивого рисунка. Изменяя положение и цвет (с помощью Flip и Colour) левого блока, Вы пытаетесь совместить его с каким-либо блоком из правого окна. (Этот этап практически является решением головоломки). Конечным результатом должно стать получение рисунка двух цифр, после чего первая часть кода будет зафиксирована в памяти. Следует повторять операцию до полного нахождения всех цифр.

Если все сделано правильно, код займет соответствующее положение, в противном случае компьютер сообщит об ошибке и укажет на ее причину. Появившиеся цифры будут занесены в графу Locking System Code: _ _ _ _ _ _ _.

Миссия считается выполненной, если заполнены все прочерки.

(Кроме перечисленного компьютер реагирует на команду REACTOR, видимо уцелевшую от игры TAU CETI. При наборе команд, иногда допускаются грамматические ошибки).

Набрав команду LAUNCH, Вы оказываетесь на поверхности планеты и можете приступить к выполнению миссии. Во время боя компьютер информирует Вас о ракетной атаке, о применении роботами Amm's против Ваших ракет, об атаке камикадзе, о Ваших повреждениях, а также выдает некоторую другую информацию.

Итак, Ваша задача - получить звание пилота скиммера. Это произойдет, когда на экране Вашего компьютера появится надпись CADET QUALIFIED.

Если Вы более или менее научитесь управлять скиммером и перейдете к выполнению миссий, Вам могут пригодиться следующие полезные советы:

- КАТЕГОРИЧЕСКИ ЗАПРЕЩАЕТСЯ открывать огонь по платформам гиперпрыжков, центрам поддержки и реакторам! Автоматика ремонтных баз и реакторов, если Вы ее повредите, заблокирует проходы в силовом поле. JUMP-платформы легко повреждаются даже огнем лазера, но это сразу ставит под сомнение возможность выполнения миссии.

Практический опыт показывает, что без вреда (и без пользы) можно обстреливать только свои G.L.V. базы - они не повреждаются даже бомбой, но не пробуйте их таранить. База есть база. Стыковаться лучше на скорости не более, чем 1/4 от максимальной и ниже.

- не пытайтесь расстреливать минные поля - не хватит здоровья!

- не летите на полной скорости к неизвестному объекту (тем более к группе)!

- не забывайте своевременно возвращаться на базу для ремонта и заправки!

- при отмеченном большом количестве целей (радаром или визуально) желательно двигаться импульсами. Роботы обычно наступают шеренгами по 3-4 робота в каждой. Активизируйте шеренги по одной. Так Вы легко расправитесь с нападающими, но если Вы сразу активизируете 2-3 шеренги, Вам не удастся даже развернуться, чтобы отважно скрыться от погони.

- в некоторых миссиях при достаточном удалении от базы желательно фиксировать свое направление по компасу, а еще лучше - по Солнцу. Компас и радар - приборы очень хрупкие и в бою нередко ломаются, а без них, если Вы не знаете направления на базу, лучше начинать миссию заново.

- некоторые миссии Вы пройдете с первого раза, над некоторыми Вам придется поломать голову. Не отчаивайтесь и не ждите подсказок, из любой ситуации есть выход. Лежит он на поверхности, надо только его увидеть.

- помните, что при выходе в главное меню Ваши очки теряются!

Желаем Вам удачи!

SHERLOCK

Наши читатели помнят, как в шестом номере ZX-РЕВЮ за прошлый год мы давали полную фирменную инструкцию к программе "SHERLOCK". Программа заинтриговала многих наших читателей, но несмотря на дружные усилия до сих пор пока никому не удалось раскрыть это сложное и запутанное дело.

А дело действительно интересное. К счастью, совсем недавно, вытирая пыль в кабинете Холмса в доме на Бейкер-стрит 221 Б праправнучка миссис Хадсон случайно обнаружила пакет, датированный 1892 годом, на которой было написано "Вскрыть через сто лет в присутствии сэра К.Синклера и ответственного представителя "ИНФОРКОМа".

После вскрытия из конверта выпали пожелтевшие листочки. Давно выцветшие чернила и малоразборчивый почерк не помешали установить, что это заметки, которые велись по ходу расследования Лизерхэдской трагедии. К сожалению, судя по почерку, их писали не доктор Уотсон и не сам великий сыщик, так что никакой литературной обработки они не прошли. Сведения отрывочны, изложены небрежно. Кто этот неизвестный секретарь, посвященный в дела Холмса, еще предстоит разобраться биографам. Мы же спешим их опубликовать в надежде на то, что эти заметки помогут и Вам докопаться до сути самого захватывающего дела Холмса.

Понедельник, 8:00

Холмс и Уотсон в своей гостиной. По просьбе Холмса Уотсон раскрывает газету и находит в ней сообщение об убийстве молодой женщины в Лизерхэде (SAY TO WATSON "READ CHRONICLE"). Холмс сразу же со всей присущей ему энергией начинает готовиться к расследованию дела.

Пройдя в гардеробную (OPEN PLAIN DOOR), он берет с вешалки два комплекта маскировочной одежды (DISGUISE). Первый комплект - для маскировки под китайца, второй - под старика. Чтобы взять костюм с собой, ему приходится сначала его надеть (WEAR), а затем снять (TAKE OFF). Вернувшись в комнату и поговорив с Уотсоном, Холмс направляется на место преступления и просит Уотсона следовать за ним (FOLLOW ME).

Ближайший поезд в Лизерхэд, как выясняется из расписания, отходит со станции Kings Cross, в 9 часов 15 минут. Чтобы успеть на него, Холмс, выйдя на улицу, должен взять извозчика (HAIL A CAB), сесть в экипаж (CLIMB INTO CAB) и объяснить ему, куда надо ехать (SAY TO CABBY "GO TO KINGS CROSS ROAD").

На третьей платформе Холмс встречается с инспектором Лестрейдом, который тоже следует на место преступления. Подождав до 9:15, сыщики садятся в поезд и в 10:30 прибывают в Лизерхэд.

Здесь Холмс предоставляет Лестрейду вести дело так, как ему кажется нужным. Он ходит за ним по городу (FOLLOW LESTRADE) и внимательно слушает все разговоры.

Расследование привело их к небольшому мосту, сделанному из песчаника. Здесь, на этом мосту и произошло преступление. Тело жертвы еще не убрали в ожидании полиции. Как Холмсу и Лестрейду удалось уже установить, это труп миссис Браун. Более внимательный осмотр места происшествия позволяет найти скомканную записку, подписанную инициалами T.F.

Миссис Браун была убита с близкого расстояния, пуля вошла в правый висок. Возможно, миссис Браун держала оружие в руках, поскольку на правой руке есть следы пороховых газов.

Холмс начинает подозревать, что здесь имело место не убийство, а самоубийство, у Лестрейда же определенно складывается иное мнение.

Проследовав за Лестрейдом к дому покойной, Холмс становится свидетелем допросов близких и домашней прислуги. Эти допросы помогли установить следующее:

- миссис Браун была вдовой недавно умершего мистера Брауна;
- мистер Браун был крупным ученым. Последнее время он работал над неким

секретным военным проектом;

- после его смерти чертежи и прочая документация по проекту бесследно исчезли;

Кроме того, Холмс убеждается во время этих допросов, что мистеру Базилю Фиппсу доверять нельзя, несмотря на то, что он имеет твердое алиби на момент смерти миссис Браун (он весь вечер находился дома и играл на рояле этюды Шопена).

Кроме того, Холмс проводит беседу с майором Персивалем Фосом, проживающим на улице Sidmouth Street, который отказывается ему сообщить, где находился во время смерти миссис Браун.

Инспектор Лестрейд возвращается в Лондон. У него уже есть своя версия убийства и он подозревает в преступлении майора.

Обследуя дом, Холмс обнаруживает еще один труп. В библиотеке он находит тело миссис Джонс. Тщательно обследовав книжные шкафы, Холмс также обнаружил потайную комнату. В этой комнате хранилась женская одежда с пятнами крови. Изучение одежды и меток на ней показало, что принадлежит она Тришии Фендер (Tricia Fender).

При внимательном обследовании письменного стола (desk) в кабинете (study) Холмс обнаруживает ящик с двойным дном, здесь находятся банковские счета миссис Браун. Здесь же есть письмо от Тришии Фендер.

Судя по состоянию банковских счетов, миссис Браун в последнее время неоднократно снимала немалые суммы денег.

Теперь Холмс хочет поподробнее узнать о Тришии Фендер. Он возвращается в гостиную и продолжает допросы свидетелей (TELL ME ABOUT TRICIA FENDER).

Ему удается получить следующую информацию:

- Тришия Фендер была секретарем мистера Брауна;
- она проживает в Лондоне на улице Portman Street;
- она и миссис Джонс очень похожи друг на друга внешне.

Далее Холмс занимается проверкой свидетеля Базиля Фиппса. Он приходит к нему в квартиру на Cobden Lane и проходит наверх, в спальню. Здесь ему становится ясно, что ни на каком рояле свидетель не играл в ночь гибели миссис Браун. У него есть граммофон, на котором стоит пластинка с этюдами Шопена.

Из окна свисает простыня. Можно предположить, что кто-то использовал окно для выхода из дома.

Доверять Базилю Фиппсу нельзя, из свидетеля он превращается в подозреваемого.

Теперь Холмс имеет всю необходимую ему информацию и возвращается в Лондон. Его ближайшая задача - допросить отставного майора, но сначала он забегает к Лестрейду в Скотланд-Ярд (на улице Parliament Street) и решительно заявляет ему, что майор невиновен (THE MAJOR IS INNOCENT).

Лестрейд требует доказательств, ведь майор отказался сообщить где он был во время убийства. Вместе они едут к майору домой на Sidmoutn Street.

Хозяина дома не оказалось и сыщикам пришлось ждать до 11 часов вечера, когда появился майор. Не задержавшись дома, он тут же снова вышел и, наняв кэб, приказал екать на Slater Street. Холмс и компания последовали за ним.

Слежка привела их к притону, в котором собираются курильщики опиума. Майор прошел внутрь. Для того, чтобы пройти за ним, Холмс должен переодеться в костюм китайца.

Теперь Холмсу понятно, почему майор скрывал, где он находился во время трагедии на мосту. Выйдя из курильни, Холмс говорит Лестрейду "THE MAJOR IS IN AN OPIUM DEN". Вскоре появляется и сам майор. Лестрейд отпускает его и говорит Холмсу "WELL DONE, HOLMES".

Так закончился первый день следствия.

Вторник.

Холмс возвращается в Лизерхэд и сразу же направляется к Базилю Фиппсу. Находит там комнату, в которой установлен сейф, вскрывает сейф и обнаруживает письма от Тришии Фейдер к миссис Браун. Из содержания писем становится ясным, что здесь имел место

шантаж. По-видимому, Тришия похитила документы ученого и впоследствии шантажировала его вдову. На это же указывает и состояние банковских счетов миссис Браун.

Чтобы встретить Лестрейда, Холмс идет на станцию. Лестрейд прибывает около 9 часов утра. Здесь Холмс сообщает инспектору, что миссис Браун совершила самоубийство под влиянием шантажа: "MISSIS BROWN KILLED HERSELF".

Вместе они следуют на мост, ставший местом ее гибели. Тщательно обследовав ручей (CLOSELY EXAM THE DEAP STREAM), они находят пистолет, к которому привязан тяжелый камень. Очевидно, совершая самоубийство, миссис Браун хотела наказать шантажистку. Для этого она привязала пистолет к камню, чтобы после выстрела оружие исчезло навсегда. Для этого же она зажала в руке скомканную записку, наводящую на след T.F. То есть, она инсценировала убийство самой себя.

Мост сложен из мягкого камня, песчаника, и падая, пистолет оставил выщербину в нем. Именно это и побудило Холмса тщательно обследовать поток.

Теперь Холмс вновь идет на станцию, ему пора возвращаться в Лондон. Лестрейд соглашается, что миссис Браун совершила самоубийство и говорит: "WELL DONE, HOLMES".

Не позже, чем в 11:15 Холмс должен выехать в Лондон и прибыть на вокзал Кингс Кросс в 12:30. Наняв кэб, он едет на улицу Portman Road к дому, где живет Тришия.

Войдя в дом, он проходит в гостиную. У стены стоит сейф. Здесь его встречает хозяйка. Не обращая внимания на ее протесты, Холмс открывает сейф и находит папку с надписью "Военный Проект" и незаконченное письмо, обрывающееся словами: "Жди меня на мосту...".

Холмсу совершенно ясно, что именно Тришия Фендер шантажировала миссис Браун. Однако, в папке секретных документов не оказалось, очевидно они были похищены.

Холмс приказывает хозяйке дома следовать за ним, берет кэб и везет ее в Скотланд Ярд. Здесь они дожидаются прихода Лестрейда и Холмс приступает к допросу.

Подозреваемая сдается на вопросе об окровавленной одежде, найденной в доме миссис Браун (TELL ME ABOUT THE BLOODSTAINED CLOTHES) и начинает давать показания.

Сразу же открывается, что она вовсе и не Тришия Фендер, а настоящее ее имя - миссис Джонс. Она всегда была близкой подругой миссис Браун и, когда узнала, что та стала жертвой шантажа со стороны Тришии Фендер, убила шантажистку. Ее труп и был найден в библиотеке.

Теперь Холмс может сделать официальное заявление Лестрейду, как представителю власти: "MRS JONES KILLED TRICIA FENDER" ("миссис Джонс убила Тришию Фендер"). Лестрейду нужны неопровержимые улики и Холмс обращается к подозреваемой "TELL LESTRADE WHAT HAPPENED" ("расскажите Лестрейду о том, что произошло").

Если перед этим Холмс тщательно изучил все улики, не пропустил ничего важного, то она во всем сознается инспектору, придет дежурный полисмен и арестует преступницу. Если же это не произошло, значит что-то важное Холмс все-таки упустил.

После ареста миссис Джонс, Холмсу остается разрешить еще одну загадку - для кого же Тришия похитила секретные документы и где они сейчас находятся. После тщательного анализа всех собранных улик, Холмс приходит к выводу, что возможным преступником является Базиль Фиппс.

Из Скотланд Ярда Холмс отправляется на квартиру к Фиппсу. В Лондоне тот проживает в доме на улице Camden Street. Подождав под окнами до 10 часов вечера, можно дождаться, что кто-то изнутри откроет окно. Переодевшись в костюм старика, Холмс проникает в спальню через окно. Быстро осмотрев дом, Холмс остановил особое внимание на библиотеке. Здесь в мусорной корзине он находит клочки какой-то записки. На заднем дворе в мусорном ящике ему впоследствии тоже удается обнаружить порванную записку. Выбравшись из дома, переодевшись и изучив содержимое клочков, Холмс обнаруживает, что это обрывки одного зашифрованного послания. Недолго провозившись с простым шифром, он получает примерно следующее содержание:

"Чертежи у меня. Ваша цена меня устраивает. Прошу сообщить о времени покупки. Смерть миссис Браун вызвала вмешательство полиции. Базиль."

Для Холмса это неопровержимое свидетельство того, что похищенные документы

находятся у Фиппса. С этим он и направляется в Скотланд Ярд, где до 7 часов утра ожидает прихода Лестрейда.

Среда.

Встретив Лестрейда, Холмс сообщает ему, у кого находятся чертежи: "BASIL HAS THE PLANS". Если к этому времени Холмс прочитал содержимое зашифрованной записки и обнаружил пустую папку из-под секретных документов, то Лестрейд предложит ему захватить преступников в момент передачи документов.

Холмс полагает, что сейчас Базиль Фиппс ожидает ответа от неизвестного сообщника и возвращается к дому на Camden Street.

В 9:50 появляется мальчик-посыльный и приносит Фиппсу записку. Холмс должен добыть ее любой ценой. Прокравшись к окну, он заглядывает внутрь комнаты. (LOOK THROUGH WINDOW). Если ничего интересного он не видит, наблюдение надо продолжать (команда повторяется). Наконец, где-то в 10:06 ему удается разглядеть горящий клочок бумаги. Теперь все решает скорость. Быстро влезть в окно. Схватить горящий лист и вылезти обратно. Маскировочный костюм теперь ему уже ни к чему.

Позже ему удается разобрать на обгоревшем листе зашифрованный текст, но на этот раз шифр уже другой. Кроме того, текст, за исключением имени адресата, написан задом наперед. Текст гласит:

"Базиль! Я покупаю чертежи. Буду в два тридцать на дороге Old Mill Road около Лезерхэда." Подписано инициалами H.W.

Быстро вернувшись в Скотланд Ярд, Холмс сообщает о том, что ему удалось узнать. Он отвечает на вопрос Лестрейда о том, где состоится передача секретных документов и они бросаются за преступниками.

Прибыв в 1:30 в Лизерхэд, Холмс выходит из поезда раньше Лестрейда и немедленно идет на главную улицу городка. Здесь уже ждет полицейский кэб. Задача Холмса успеть расположиться в кэбе до того, как его займут Лестрейд с помощником, иначе Холмсу просто не хватит места.

Лестрейд велит кэбмену ехать на Old Mill Road. Они приезжают туда в 3:13, но это уже поздно. Базиль только что уехал. Лестрейд приказывет ехать обратно и в 4:47 они снова возвращаются на главную улицу. Здесь им удается заметить Базиля и рядом с ним немецкого агента. Преследование приводит Холмса на платформу номер 2, где злоумышленники вскакивают в отходящий поезд. Холмс же должен найти другой путь в Лондон. Сказав Лестрейду: "За мной!", (FOLLOW ME), он спешит в полицейский кэб. Уотсон также следует за ними. Холмс велит кэбмену ехать на King Cross Road.

Если все делать достаточно быстро, то Холмс имеет шанс увидеть, как Базиль и агент усаживаются в кэб. Удается и услышать, как Базиль приказывает кэбмену отправляться на Buckingham Palace Road.

Холмс мгновенно соображает, что соответствующая станция подземки - это Виктория (Victoria). Сыщики прыгают в поезд на платформе King Cross, который идет к Victoria, где и выходят.

Через некоторое время появляются преследуемые. Заметив Холмса, Базиль понимает, что он попался и пытается застрелить Холмса. Жизнь ему спасает доктор Уотсон (если он рядом). В последнюю секунду ему удается оттолкнуть Холмса в сторону.

Лестрейд производит арест.

Если Холмс сделает все это, то конечно он станет Величайшим сыщиком всего мира.

Так заканчивается одно из наиболее загадочный и интригующих расследований, которое вел когда-либо Шерлок Холмс. Почему оно не было до сих пор обнародовано, остается загадкой.

* * *

Мы напечатали эти заметки не только для тех, кто увлекается программой SHERLOCK.

Они относятся ко всем, кто любит адвентюрные игры, но пока не может похвастаться особыми достижениями. Пусть действия Холмса послужат примером того, что можно требовать от хорошей программы этого жанра. А для тех, кто еще не начал работу с адвентюрными программами, мы только укажем, что все изложенное здесь представляет кратчайший путь к победе, но и на 10% не исчерпывает всех возможных вариантов развития событий.

Мы также рекомендуем начинающим изучить цикл статей "ADVENTURE LESSONS" в номерах ZX-РЕВЮ 1991 года.

НАШ КОНКУРС

В номере 1-2 ZX-РЕВЮ за этот год, мы объявили конкурс по программе ELITE на самый рискованный маршрут.

Напомним условия: в одной из галактик надо облететь 20 планет, не побывав на одной планете дважды. Подсчет уровня опасности избранного маршрута производится по следующим критериям:

- анархическая планета - 5 очков;
- феодальная планета - 4 очка;
- любая другая - 1 очко.

Призы в конкурсе: пять первых мест получают ксерокопию повести DARK WHEEL на английском языке. Повесть написана по мотивам программы ELITE.

Как обычно, конкурс вызвал большой интерес среди наших читателей. По нему поступило около 100 писем с предложениями самого "крутого маршрута". Абсолютное большинство читателей пришли к выводу, что самым злачным местом во Вселенной является Галактика N4. Так получилось, что те читатели, которые пытались проложить маршрут по другим галактикам, не добрали очков и остались вне числа призеров.

Наилучший результат, достигнутый нашими пилотами - 100 баллов достигнут Ивановым А.И. из Читы. Мы, правда, должны сказать, что были еще такие же достижения, но получались они с нарушением правил. Например, когда пилот прокладывал маршрут по двум и более галактикам, пользуясь гипердрайвом. Такие решения мы не рассматривали.

На втором месте с результатом 92 балла оказалось сразу 12 человек.

Вот как выглядит наш Зал Славы:

Дзреев К.К. (Ростов на Дону)
Довженко В.П. (Киев)
Жаров Р.Н. (Херсон)
Игнатов А. (Новосибирск)
Коротков О.Е., Шилов А.В. (Ростов на Дону)
Минаков Р.С. (Калининград)
Минеев А.В. (Владивосток)
Мясников Г.Л. (Сыктывкар)
Радзевич А.А. (Нальчик)
Сергиенко Т.П. (Волжский, Волгоградской обл.)
Тошев В.В. (Норильск)
Хохлов Д.В. (Санкт-Петербург)

Нет числа разбитым пиратским кораблям, теперь они долго будут усеивать четвертую галактику, а перед нами новая проблема: как разделить 5 призов между 13 победителями (у нас просто больше нет ни одного лишнего экземпляра).

Мы приняли такое решение:

Иванову А. И. отправляем повесть "DARK WHEEL", путем жеребьевки она же отправляется Игнатову А., Короткову О.Е., Радзевичу А.А., Тошеву В.В.

Остальным же победителям мы в качестве завоеванного приза вышлем имеющиеся несколько экземпляров книги Яна Логана и Фрэнка О'Хары "Полный дисассемблер ПЗУ" тоже на английском языке. Эта книга является библией всякого настоящего "синклериста" и мы надеемся, что победители не будут огорчены произведенной заменой.

А что же касается самой повести "DARK WHEEL", то сотни наших читателей спрашивают, почему бы нам не перевести ее на русский язык и не опубликовать по частям в ZX-РЕВЮ? Действительно, такое решение выглядит наиболее рациональным и мы просто упустили его из виду, не ожидая что она вызовет массовый интерес. Так мы и сделаем в ближайших выпусках.

Сегодня мы предлагаем вниманию наших коммерческих представителей для проведения маркетингового исследования новую информационно-поисковую систему "DBP".

DATA BASE PROCESSOR - процессор баз данных.

Вы помните, мы представляли в N1-2 за этот год систему многоцелевого назначения "РЕГИСТРАТУРА". "ПРОЦЕССОР" обладает всеми теми же свойствами, но имеет ряд мощных дополнительный возможностей, делающих его самостоятельным коммерческим продуктом. Предполагается, что наибольший эффект он даст при совместном использовании с "РЕГИСТРАТУРОЙ".

I. НАЗНАЧЕНИЕ СИСТЕМЫ

Как и система "РЕГИСТРАТУРА", система DBF предназначена для ведения учета и проведения обработки любой информации, необходимой на Вашем предприятии (в Вашем учреждении) Например:

- регистрация входящих и исходящих документов;
- регистрация входящих и исходящих товаров;
- регистрация пациентов в лечебном учреждении;
- регистрация клиентуры и заказчиков; - учет движения материалов на складе;
- и многое, многое другое.

Система может удовлетворить потребности отдела кадров, планово-финансового отдела, отдела соцкультбыта, различных административных, муниципальных и хозяйственных служб и т.п., кроме бухгалтерии.

II. ВОЗМОЖНОСТИ СИСТЕМЫ

Не останавливаясь на всех возможностях системы "РЕГИСТРАТУРА", перечисленных в N1-2 (стр. 44), мы укажем на дополнительные инструментальные возможности, предоставляемые ПРОЦЕССОРОМ при создании и модификации баз данных.

- теперь нет необходимости при заказе базы приводить список необходимых полей с указанием их типа и размера. ПРОЦЕССОР позволяет пользователю самостоятельно создавать базы данных с любой необходимой ему информацией. Пользователь имеет возможность задать те поля информации, которые ему необходимы в данный момент.

- пользователь теперь имеет возможность изменять структуру даже ранее созданной базы. Можно в любое время создать новые поля, удалить ненужные, произвести замену. Вставка новых полей может происходить и в середину готовой и заполненной базы.

- удобным дополнением является поиск повторяющихся 8 записей по одному или нескольким полям.

- введена функция поиска и замены информации (по аналогии с текстовыми редакторами).

- при копировании и печати информации добавлена возможность пометки блока записей из заданного интервала.

- введен режим подведения суммарного итога по заданнному числовому полю.

III СТРАТЕГИЯ ЭКСПЛУАТАЦИИ

Система DBF полностью заменяет ранее выпущенную систему "РЕГИСТРАТУРА", но вследствие своих расширенных возможностей, требующих специальных знании при подготовке структуры баз данных, отличается:

а) более высокой ценой;
б) большей сложностью в освоении неподготовленным пользователем.

Поэтому наиболее целесообразно применять эти системы совместно так, если на предприятии несколько подразделений нуждаются в информационно-поисковых системах, целесообразно в каждом иметь систему "РЕГИСТРАТУРА", настроенную на потребности данного подразделения и одну центральную систему DBF для первичной подготовки и настройки информационных файлов для этих подразделений с возможностью последующего обслуживания и модифицирования.

IV. КОМПЛЕКТ ПОСТАВКИ

Система поставляется на одной дискете 5,25" (MS DOS, 360 K).
K системе прилагается подробная инструкция по эксплуатации.

V. СРОК ИСПОЛНЕНИЯ ЗАКАЗА.

Срок исполнения - 1 - 2 недели после поступления средств на наш р/с и письма-заказа.

VI. ТЕХНИЧЕСКИЕ ТРЕБОВАНИЯ К АППАРАТНО-ПРОГРАММНОМУ ОКРУЖЕНИЮ

1. Полная аппаратно-программная совместимость с IBM PC XT/AT. Надежность функционирования на отечественных модификациях не гарантируется и не обсуждается.
2. Наличие "жесткого" диска ("Винчестера") стандартного объема.
3. Дисковод гибких дисков 5,25" или 3,5"".
1. Операционная система - MS DOS не ниже 3.20.
5. Русификация компьютера в стандарте ГОСТ (кодировка альтернативная).
6. Требования к монитору - не специфицируются. Желательно - EGA.
7. Требования к принтеру - совместимость со стандартом EPSON.

VII. ГАРАНТИЙНЫЕ ОБЯЗАТЕЛЬСТВА

Гарантийным свидетельством при поставке программного продукта является картонный альбом, в который вложены дискеты с указанной на нем датой продажи. При его отсутствии поставка выполняется с заверенным гарантийным талоном.

Гарантиями обеспечивается: - бесплатная замена поставочных дисков, неработоспособных в состоянии поставки (в течение месяца после поставки);

- замена с минимальной оплатой при выходе программ из строя по вине пользователя (механическое или электромагнитное повреждение, поражение вирусом на машине пользователя и т.п.) или по истечении месяца после поставки. Минимальная оплата не превышает стоимость дисков + 5% текущей стоимости программого обеспечения + стоимость почтово-транспортных расходов и согласовывается с потребителем.

VIII. ПОРЯДОК ОФОРМЛЕНИЯ ЗАКАЗА.

а) направить в наш адрес письмо-заказ с указанием необходимого программного продукта и количества копии. Приложить копию платежного поручения.

Наш адрес: 121019, Москва, Г-19, а/я 16, "ИНФОРКОМ".

б) произвести предварительную оплату платежным поручением на наш р/с: N 500461778 во Фрунзенском коммерческом банке г. Москвы.

Стоимость системы на период июль-сентябрь 1992г. - 12500 рублей + 28%

"ЗЕЛЕНЫЙ ПАКЕТ" ДИСТРИБУТОРА

О том, что такое "зеленый пакет" Вы можете подробно прочесть в N11-12 "ZX-РЕВЮ" за 1991г.

Стоимость "зеленого пакета" по системе DBP составляет для частных лиц:

Рабочая версия программы - 10% * 12500 =	1250
Дискета -	100
Почтовые расходы -	15
Итого:	1365 рублей

Уточняем, что высокая стоимость дискеты вызвана тем, что нашим дистрибуторам поставляются дискеты особо высокого качества с защитным тефлоновым покрытием, имеющие особый представительский вид и высокую коммерческую стоимость.

Напоминаем, что затраты дистрибутора на "зеленый пакет" являются только залогом и возвращаются по требованию. Активно работающим дистрибуторам затраты возвращаются при выплате комиссионных и далее такие пакеты предоставляются бесплатно.

Содержание