ZX - РЕВЮ
N11,12 1992

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

Дорогие друзья!

Мы заканчиваем печать ZX-РЕВЮ образца 1992 года. Самое время подвести некоторые итоги, обсудить возникшие проблемы и наметить планы на будущее.

Основным итогом, конечно, является то, что ZX-РЕВЮ благополучно пережило второй год своего существования и, несмотря на то, что эти последние номера Вы получаете с очевидным опозданием, мы с оптимизмом смотрим в будущее и полномасштабный выход в свет ZX-РЕВЮ-93 сейчас уже не вызывает сомнений.

К отрицательным итогам нам придется отнести тот печальный факт, что многократно повышены почтовые тарифы между Россией и другими странами СНГ. Это практически лишает многих наших постоянных читателей из этих стран возможности получать ZX-РЕВЮ и другие материалы.

Мы, наверное, никогда не поймем, почему почтовые отправления на Украину и в Беларусь должны оплачиваться в несколько раз дороже, чем в Россию, а наших постоянных читателей из республик Прибалтики вообще приравняли по тарифам к жителям Мадагаскара.

Мы надеемся, что когда-нибудь нормально мыслящие политики еще скажут свое слово о роли Министерств Связи стран СНГ в деле укрепления гуманитарных связей и воздадут им по заслугам, а пока будем как-то выкручиваться и искать обходные пути.

Итак, начнем по порядку.

1. Каким будет "ZX-РЕВЮ" в 1993 году.

В основном мы оставим все без изменений. Судя по вашим письмам, содержание и структуру ZX-РЕВЮ, достигнутые в 1992 году можно считать более удачными, чем в 1991г., и ничего радикально мы менять не будем.

Единственно, чего будет больше - это раздела "Маленькие хитрости". Причем, мы уже сегодня можем сказать, что основной темой этого раздела будут разного рода манипуляции и операции с экраном (как правило, в машинных кодах).

Мы добьемся некоторого баланса, когда интересные приемы, приводимые в этом разделе, будут служить как бы дополнительными иллюстрациями к тем книгам, которые мы готовим и выпускаем по графике "Спектрума". Хотя это вовсе не означает, что для понимания и применения предлагаемых материалов необходимо эти книги иметь. В то же время, для тех, кто хочет разобраться с этими вопросами досконально, они лишними не будут.

Так же, как и в 1992 году, мы будем печатать и крупные материалы с продолжением. Одним из них станет сериал Дэвида Новотника, посвященный экспертным системам вообще и возможностям их реализации на "Спектруме" (перевод и редактирование - наше).

Вторым крупным материалом станет книга Стюарта Николса, посвященная вопросам создания игровых программ на АССЕМБЛЕРе. Перевод подготовлен нашим соавтором из г. Балашова Саратовской области - В.Паворинным. Книга заинтересует тех, кто еще не освоил машинный код, поскольку в ее методическую основу положен принцип постепенного "перевода" команд и операторов БЕЙСИКа в их машинно-кодовые аналоги. Книга очень

доступна для понимания и потому мы ее смело предлагаем массовому читателю, а не выпускаем отдельным выпуском для избранных.

В ближайшие месяцы мы закончим публикацию материалов, посвященных вопросам постановки и снятия защиты программ, но на очереди еще интересные статьи, содержащие разбор методов работы известных "хаккеров".

Готовится крупный материал о возможности применения "Спектрума" в астрологии. Только не пугайтесь, никакой мистики не будет. Речь пойдет о применении компьютера к анализу движений небесных тел. Кстати, первые эксперименты показали, что по производительности "Спектрум" почти не уступает в этой области IBM PC XT.

Мы по-прежнему будем уделять часть места исследованию игровых программ и будем экспериментировать с оригинальными литературно-художественными интерпретациями игровых программ. Основная цель этих экспериментов - выйти на новый жанр "компьютерной новеллы", которая должна, с одной стороны, давать пользователю полезную информацию о технике прохождения той или иной игры, а с другой стороны, делать это так, чтобы не отбивать, а наоборот пробуждать у него желание сыграть в эту игру.

Без изменения мы сохраним остальные рубрики "ZX-РЕВЮ".

Как обычно, в каждом номере будут статьи, посвященные применению "Спектрума" в школе. При наличии интересных материалов, поступающих от читателей, будет действовать раздел "Форум" и, конечно, для опытных программистов, которым есть чем поделиться, открыт раздел "Профессиональный подход".

2. Как можно будет подписаться на "РЕВЮ-93".

Большой разнобой в условиях поставки в разные республики СНГ не позволяет нам однозначно всем сообщить условия подписки. Пока можем сообщить только, что принято принципиальное решение о том, что прием подписки по почте мы все-таки проведем. Во-первых, непропорционально выросли цены на железнодорожные и авиабилеты, что не позволяет многим читателям посетить корпункт. Кроме того, не прекращается поток писем с просьбой о приеме подписки по почте в порядке исключения. Честно говоря, видя желание многих наших подписчиков получать "ZX-РЕВЮ" и в 1993 году, мы не имеем возможности им отказать.

С другой стороны, у нас есть определенные трудности в приеме подписки по почте, поэтому эти условия мы оговорим в прилагаемом к данному выпуску информационном листке, т.к. он попадает ТОЛЬКО К НАШИМ ЗАРЕГИСТРИРОВАННЫМ ЧИТАТЕЛЯМ и только они смогут им воспользоваться.

Наш постоянный корпункт продолжает действовать по адресу: г. Москва, ул. Новый Арбат (бывший просп. Калинина), д. 2, отделение связи Г-19. На первом этаже в операционном зале Вы найдете нас по вывеске "ZX-РЕВЮ". Корпункт работает все дни недели, кроме воскресенья с 10 до 17 часов (перерыв на обед с 14 до 15 часов).

Несмотря на то, что мы приняли решение о приеме подписки по почте, у Вас есть по крайней мере четыре причины, по который посещение корпункта было бы желательным:

- цены на корпункте на всю имеющуюся у нас литературу, включая и подписку на ZX-РЕВЮ-93, существенно ниже, чем при оплате по почте;
- при покупке мелкооптовых партий (от 10 экз.) действуют дополнительные скидки;
- только здесь можно приобрести некоторые материалы прошлых лет (описания языков программирования, различных прикладных программ и т.п.), которые у нас заканчиваются и потому из свободной продажи по почте изъяты;
- через корпункт мы начали распродажу остатков кассет с программными сборниками прошлых лет. Обращаем Ваше внимание на то, что по причине их малого количества, по почте они уже давно не высылаются, и заказы на них мы не принимаем.

Вам совсем не обязательно являться на корпункт лично. Вы можете попросить об этом кого-либо из своих друзей и знакомых, находящихся в Москве проездом.

Вторая возможность контакта с нами через радиорынки. На сегодняшний день в

Москве функционирует крупный радиорынок в Тушино (ст. метро Тушинская). По субботам и воскресеньям (с 9 до 13) там можно войти в контакт с нашими дистрибьюторами для решения оперативных вопросов. Местные органы власти планируют закрытие данного радиорынка. В этом случае он может быть перенесен в другие точки, где также можно будет найти наших дистрибьюторов. Есть предварительные сведения о планах переноса Тушинского радиорынка в район платформы "Трикотажная" (проезд на электричке от Рижского вокзала или от платформы "Тушино").

И, наконец, мы решаем вопрос об открытии представительств в республиках СНГ. В настоящий момент ведутся переговоры с украинской фирмой из г. Днепропетровска, которая возьмет на себя обслуживание жителей Украины нашими материалами. При положительном решении данного вопроса фирма-представитель войдет с Вами в контакт, предложит свои услуги и начнет обслуживание как тех, кто подписался у нас, так и своих подписчиков.

Надеемся, что и в других странах СНГ найдутся организации, имеющие желание и возможность стать нашими официальными представителями. Это позволят не только сохранить число наших читателей в этих странах, но и значительно его увеличить. Мы готовы к проведению подобных переговоров.

3. Наши новые разработки.

3.1. Выпущен годовой комплект "ZX-РЕВЮ-91" в виде одной аккуратной книжки размером 200 x 140 мм, 254 стр.

3.2. К тому времени, как вы прочтете эти строки, будет выпущена аналогичная книжка, содержащая полный комплект "ZX-РЕВЮ-92".

3.3. Вышло новое издание книги по программированию в машинных кодах. В новое издание вошли все три тома первого издания ("Первые шаги в машинном коде", "Практикум по программированию в машинных кодах" и "Справочник по программированию в машинных кодах".).

В отличие от первого издания, новое издание существенно расширено (более чем на 20%). Дополнение коснулось следующих тем:

- применение прерываний второго рода;
- концепция каналов и потоков;
- вопросы русификации компьютера;
- описание директив АССЕМБЛЕРа.

Общий объем новой книги – 272 стр.

3.4. На днях выходит из печати первый том четырехтомника, посвященного графике "Спектрума". Этот том называется "Элементарная графика" и содержит 208 стр. Готовится к передаче в печать второй том - "Прикладная графика".

4. Вопросы лицензирования.

Ранее мы объявляли нашим читателям о готовности передачи компьютерного текста наших книг для издания тиража в регионах на условиях ограниченной лицензии (см. стр. 133-134).

Сейчас мы можем сообщить, что такая основа для сотрудничества вызвала живейший интерес всей страны. Вместе с тем, вскрылись некоторые проблемы.

Так, например, все без исключения участники переговоров о покупке лицензии на право издания и распространения наших книг в своих регионах отметили низкий процент лицензионных отчислений, который мы получаем и который их полностью устраивает.

Основной проблемой оказались цены на услуги печати на местах. Так, например, в некоторых районах России и Украины только печать книжки объемом 200-220 стр. требует до 300-500 рублей за экземпляр. Многим покупать готовую продукцию у нас оказалось дешевле, чем печатать у себя. В итоге нами проданы пока три лицензии на печать книги "Элементарная графика" в следующие регионы: Чувашия, Нижний Новгород, Сахалинская

обл. Переговоры с еще 10-15 партнерами пока не закончены.

Мы очень рекомендуем тем, кто захочет получить от нас лицензию на издание и распространение наших книг стараться не пользоваться услугами крупных специализированных типографий, а сосредоточиться на небольших ведомственных типографиях или, еще лучше, принадлежащих предприятиям. С их помощью можно сделать печать более рентабельной.

Вместе с тем, мы продолжаем расширять перечень книг, доступных для издания по регионам. Сегодня мы предлагаем желающим получить на дискете и издать у себя нашу книгу, посвященную программированию в машинных кодах (расширенный и дополненный трехтомник).

В течение ближайшего времени будет предложен и второй том графики - "Прикладная графика".

Для проведения консультации и переговоров по вопросам покупки лицензий, открытия представительств в странах СНГ, а также для желающих приобрести литературу в среднеоптовых количествах (от 100 экз.) в прилагаемом информационном листке мы даем контактный телефон.

5. Новые разработки для IBM-совместимых компьютеров.

В ближайший месяц мы оповестим наших дилеров о новых разработках обучающих и деловых программ для IBM совместимых машин. Подготовлены программы для детей по арифметике, математике и информатике, а также выпущены две версии универсальной и очень удобной системы для создания баз данных - "DB-процессор", который может стать прекрасной основой для Вашего частного предпринимательства.

А сейчас мы прощаемся с Вами. До свидания и до новых встреч!

"ИНФОРКОМ"

BETA BASIC

Окончание.
(Начало см. на стр. 3, 47, 91, 135, 179).

6. DPEEK (адрес).

FN P(адрес).

См. также DPOKE.

Функция DPEEK - это то же самое, что и двойной PEEK. Эквивалентом этой функции в стандартном БЕЙСИКе является выражение:

```
LET a = PEEK (addr) + 256*PEEK (addr+1)
```

Таким образом, эта функция выдает содержимое двух следующих друг за другом байтов. Обратите внимание на то, что младший байт идет первым. Это очень удобно для проверки содержимого системных переменных и при анализе машиннокодовых процедур. Например:

```
10 LET nxt = DPEEK(23637):POKE nxt+5.65
20 REM XXXXX
```

В десятой строке будет прочитано содержимое системной переменной NXTLN, после чего в результате POKE первый символ, следующий после оператора REM будет изменен на символ "A". Сдвиг на +5 байтов в строке 20 необходим для того, чтобы пропустить номер строки (2 байта), длину этой строки (2 байта) и код самого оператора REM (1 байт).

Оператор DPOKE обеспечивает двойной POKE точно так же, как функция DPEEK обеспечивает двойной PEEK.

7. EOF (номер потока).

FN E(номер потока).

Название функции EOF происходит из сокращения End of File (конец файла). При работе с микродрайвом эта функция говорит о том прочитан или нет последний байт из загружаемого файла.

Поток, номер которого указан в функции, должен быть предварительно открыт (OPEN#) для данного файла, содержащегося на ленте микродрайва. Если этого не сделать, то Вы получите сообщение об ошибке "Invalid Stream". Этот файл должен, конечно, физически существовать и быть открытым для чтения, иначе вы получите сообщение об оошибке "Reading a "Write" file"

Функция выдает "1", если последний байт данных считан из файла или "0", если он еще не считан. Это помогает точно установить момент конца считывания данных и избежать попыток прочтения большего количества данных, чем в этом файле есть, что вызвало бы появление ошибки.

Предположим, что на микродрайве у вас есть файл "data". Наиболее элегантный путь использования функции EOF выглядел бы так:

```
10 OPEN #5,"m",1,"data"
20 DO UNTIL EOF(5)=1
30 INPUT #5;a$: PRINT a$
40 LOOP
50 STOP
```

Впрочем, строку 20 можно записать еще короче:

```
20 DO UNTIL EOF(5)
```

Если Вы еще не привыкли к использованию циклов DO...LOOP, то можете попробовать действовать так:

```
10 OPEN #5,"m",1,"data"
20 INPUT #5;a$: PRINT a$
30 IF EOF(5)=0 THEN GO TO 20
40 STOP
```

8. FILLED ().

FN F().

См. также команду FILL.

Эта функция дает количество пикселов, включенных по последней команде FILL. Например:

```
10 PLOT 0,0: DRAW 9,0: DRAW 0,9
20 DRAW -9,0: DRAW 0,-9
30 FILL 5,5
40 PRINT FILLED()
```

Длина стороны квадрата - 10 пикселов. (Мы давали число 9 в команде DRAW, но поскольку реальная линия на экране имеет толщину не менее одного пиксела, то и сторона квадрата у нас будет десять пикселов.) Внутренняя неокрашенная часть квадрата будет иметь линейный размер по 8 пикселов на сторону, и функция FILLED() даст нам значение 64.

Если Вы попробуете в строке 30 дать команду на стирание нарисованного квадрата:

```
30 FILL PAPER; 5,5
```

то функция FILLED() даст Вам результат - 100.

Разница между 100 и 64 - количество пикселов, ушедших на изображение периметра квадрата.

9. HEX$ (число).

FN H(число).

См. также функцию DEC(строка).

Эта функция конвертирует десятиричный числовой аргумент в шестнадцатиричную символьную строку. Строка имеет два символа, если число было в диапазоне от -255 до +255 или четыре символа, если абсолютная величина числа была больше. Если же число по абсолютной величине больше, чем 65535, то выдается сообщение об ошибке "Integer out of range".

```
HEX$ (32)       = "20"
HEX$ (255)      = "FF"
HEX$ (512)      = "200"
HEX$ (-64)      = "C0"
HEX$ (-1024)    = "FC00"
```

Возможность работы с отрицательными целыми числами будет полезна тем пользователям, которые работают с машинным кодом. В частности, это пригодится при расчете адресов обратных относительных переходов.

Эта функция очень удобна, если Вам надо распечатать содержимое памяти в шестнадцатиричном виде:

```
10 INPUT "Start address? "; addr
20 PRINT HEX$ (addr); " "; HEX$(PEEK addr)
30 LET addr=addr+1: GO TO 10
```

Если же Вы хотите и начальный адрес при вводе тоже задавать в шестнадцатиричном виде, то можете переделать строку 10 например так:

```
10 INPUT "start address? "; A$: LET addr = DEC (A$)
```

10. INARRAY (строковый массив (начальный элемент)<,границы>,искомая строка).

FN U (строковый массив (начальный элемент)<,границы>,искомая строка).

См. также функцию INSTRING.

Функция INARRAY осуществляет сканирование строкового массива в поисках заданной строки. Если она не найдена, то выдается 0. Если же она разыскана, то выдается номер той строки в массиве, в которой данная символьная последовательность встретилась впервые.

В принципе функция INARRAY - разновидность функции INSTRING, но предназначенная для работы с массивами, поэтому будет неплохо, если Вы сначала прочитаете про работу функции INSTRING.

Нижеприведенный пример показывает, как можно разыскать все символьные сочетания "howdy" в заданном массиве

```
10 DIM a$(20,10)
20 LET a$(RND*19+1) = "howdy"
30 LET num= 1
40 DO
50   LET num=INARRAY (a$ (num),"howdy")
60 EXIT IF num=0
70   PRINT num;" ";a$(num)
80   LET num=num+1
90 LOOP UNTIL num > 20
```

В строке 50 выражение a$ (num) при num=1 предполагает, что поиск начнется с первой строки массива. Когда первое искомое сочетание символов будет найдено, строка, в которой оно содержится, будет выдана на печать, после чего поиск будет продолжен с (n+1) строки.

Оператор EXIT IF выведет программу из цикла, когда уже не останется неразысканных сочетаний "howdy". Оператор LOOP UNTIL в строке 90 безусловно прекратит работу программы, когда все элементы исходного массива будут просмотрены.

Программа проверит каждую строку полностью, но Вы можете ограничить пределы поиска внутри каждой строки исходного массива, задав границы поиска.

```
50 LET num=INARRAY(a$(num,3 TO 7),"howdy")
```

Эта возможность позволяет Вам разрабатывать и манипулировать с базами данных. Например, вам надо найти в базе все адреса, относящиеся к городу Брест. Тогда вы проведете поиск именно по тем участкам информации, которые имеют отношение к полю "Город", и исключите тем самым жителей города Москвы, проживающих на Брестской улице, поскольку поле "Улица" в розыск не попадет.

При поиске вы можете использовать символ-заместитель "#", который обозначает "любой символ". Это же, кстати, относится и к функции INSTRING. Единственный случай, когда этот символ ничего не замещает, а принимается сам за себя, - это когда он стоит первым символом в искомой символьной строке.

Вы можете практиковать довольно сложные условия для поиска. Например, у Вас в базе данных содержится список лиц с именами, фамилиями, адресами, цветом волос и т.п. Если каждое поле начинается строго с фиксированной позиции, а так и должно и быть в структурированной базе данных, то Вы можете найти всех содержащихся в ней жителей Лондона, имеющих коричневый цвет волос. Для тех позиций, которые соответствуют городу, Вы введете London, для тех, которые соответствуют цвету волос - Brown, а для всех прочих поставите символы "#".

Нижеприведенный пример покажет как это можно сделать. Здесь первые 20 символов отведены под имя персоны, а последующие 15 - под название города.

```
10 LET namelen=20, townlen=15
20 DIM d$(10,namelen+townlen)
30 FOR n=1 TO 10
40   INPUT "name? ";n$
50   INPUT "town? ";t$
60 LET d$(n,1 TO namelen)=n$
70 LET d$(n,namelen+1 TO )=t$
80 NEXT n
90 PRINT "массив заполнен"
```

Мы подготовили исходный массив для дальнейшей работы. Следующий программный блок позволит Вам найти заданную комбинацию имени и адреса. Примененная здесь функция STRING$ служит для генерации заданного количества символов "#", служащих для отделения между собой полей имен и адресов.

```
100 INPUT "name=? ";n$
110 INPUT "town=? "; t$
120 LET s$=n$ + STRING$ (namelen-LEN n$,"#")+t$
130 LET loc=INARRAY(a$(1),s$)
140 IF loc=0 THEN
        PRINT "Not found "
    ELSE
        PRINT loc;" ";d$(loc)
150 GO TO 100
```

Примечание: функция INARRAY не может работать с массивами, размерность которых больше, чем 2.

11. INSTRING (старт, строка 1, строка 2).

FN I (старт, строка1, строка2).

См. также функции: MEMORY$, INARRAY.

Функция INSTRING просматривает строку 1 в поисках строки 2, начиная с символа "старт". Если такое вложение найдено, то функция выдает порядковый номер символа в строке 1, с которого начинается строка 2, в противном случае выдается ноль.

Первая строка может быть любой длины, а вторая - не более 256 символов, иначе будет выдано собщение об ошибке "Invalid argument". Если начальная позиция для поиска "старт" равна нулю, будет выдано сообщение об ошибке "Subscript wrong".

В тех случаях, когда длина второй строки больше, чем первой, а также когда "старт" больше, чем длина первой строки, функция выдает ноль.

В искомой строке можно использовать символ-заместитель "#", который служит вместо тех символов, которые при поиске не имеют значения. Например:

```
PRINT INSTRING (1,A$,"SM#TH")
```

найдет появление SMITH, SMYTH, SMATH и т.п. в символьной строке A$.

Единственный случай, когда символ "#" воспринимается буквально, т.е. служит вместо самого себя - когда он стоит первым символом в искомой строке.

Возможность задания начальной позиции для поиска будет полезной в тех случаях, когда Вы рассчитываете найти не одно, а большее количество вхождении искомой строки в исходную. Ниже приведен пример, который разыщет строку "TEST" в символьной строке A$.

```
100 DIM a$(1000)
110 FOR n=1 TO RND*10 + 3
130 LET pos = RND*995
130 LET A$ (pos TO pos + 3)
140 NEXT n
150 PRINT "Press any key"
160 PAUSE 0
170 LET loc=1
180 LET loc = INSTRING (loc,a$,"TEST")
190 IF loc <>0 THEN PRINT "Found"; loc: LET loc=loc+1:GO TO 180
200 PRINT "Finish"
```

Строка A$ начинает просматриваться с позиции 1 (loc=1), пока не будет найдена последовательность символов "TEST". После этого поиск будет продолжен со следующего символа (loc=loc+1). Когда функция выдаст 0, поиск будет завершен.

Обратите внимание на то, что строку 190 можно немного упростить:

```
190 IF loc THEN PRINT ...
```

Если Вам надо сделать поиск по оперативной памяти или по ее части, Вы можете воспользоваться функцией MEMORY$.

Функцию INSTRING можно с успехом использовать в обучающих и в некоторых игровых программах (в частности в адвентюрных). Предположим, что программа задала пользователю вопрос, правильный ответ на который хранится в переменной c$="NAPOLEON". Те, кто введут в качестве ответа "NAPOLEON "(обратите внимание на финальный пробел) или "NAPOLEON BONAPARTE", будут разочарованы, поскольку программа воспримет эти ответы, как неправильные. А это очень часто случается, если подпрограмма сравнения ответа и эталона не обладает достаточной гибкостью. Функция INSTRING поможет справиться с этой проблемой. Она оценит как правильный любой ответ, при котором контрольная строка входит в состав введенной пользователем.

```
INPUT a$: IF INSTRING (1,a$,c$)<> 0 THEN PRINT "Correct"
```

Вы можете решить при этом и проблему регистра, если заранее не знаете большими или малыми буквами будет набран ответ пользователя. Вам надо принудительно конвертировать его ответ в прописные буквы с помощью функции SHIFT$:

```
LET a$=SHIFT$(1,a$)
```

Еще одно возможное применение функции INSTRING - для упаковки нескольких строк в одну длинную строку. При этом Вы можете, например использовать символы с 1-го по 31-

ый в качестве "маркеров" входящих строк.

Так, символ CHR$ 1 будет отмечать начало первой подстроки в генеральной строке, символ CHR$ 2 - начало второй подстроки и так далее. Так хранить строки гораздо компактнее, чем в массиве, если они имеют неодинаковую длину. Найти строку с номером n в генеральной строке будет несложно:

```
PRINT a$(INSTRING(1,a$,CHR$ n)+1 TO INSTRING (1,a$,CHR$(n+1)-1)
```

Возможны и многие другие интересные пути использования функции INSTRING.

12. ITEM ()

FH T()

См. также раздел, посвященный процедурам.

Эта функция дает информацию о следующей единице данных, которые подлежат вводу через READ. Как правило, функция используется при работе с процедурами, но может быть применена и при обычной технике READ...DATA. Функция возвращает следующие значения:

0 - все данные из текущего оператора DATA прочитаны. Текущий оператор DATA может быть при этом и списком параметров, следующим за вызовом процедуры.

1 - следующий объект - символьный.

2 - следующий объект - числовой.

При работе с процедурами функция ITEM() может дать информацию о физической природе первого объекта данных, но во всех остальные случаях она будет выдавать 0 до тех пор, пока хотя бы один объект не будет считан из списка DATA с помощью READ. В нижеприведенном примере проверяется ITEM() после READ.

```
100 DO
110    READ x
       PRINT x
120 LOOP UNTIL ITEM()=0
130 DATA 1,2,3,4,5,6
```

Таким образом, строки 100...120 могут считывать строки DATA произвольной длины.

13. LENGTH (n, "имя массива")

FN L(n,"имя массива")

Функция LEHGTH выдает размер массива. Для Бета-Бейсика это особенно важно, поскольку этот язык программирования позволяет во время работы проводить изменения длин массивов без потери данных. Эта же функция может быть использована для определения местоположения числовой или символьной последовательности в оперативной памяти компьютера.

Параметр n определяет о какой размерности для двумерных массивов идет речь. Если n=1, то функция возвращает размер массива в первом измерении, а если n=2, то во втором измерении (или единицу, если массив одномерный). С массивами размерности больше, чем 2, эта функция работать не может.

В имени массива значимыми являются только первые два символа, поэтому следующие имена будут приняты, как правильные: a$, b$, C(, d(), a$QWERT. Если вместо имени массива ввести имя простой символьной переменной, то она будет интерпретироваться как одномерный массив, состоящий из односимвольных элементов, количество элементов при этом равно длине символьной строки.

Примеры:

```
10 DIM a$(10,20)
20 PRINT LENGTH (1,"a$"): (10)
30 PRINT LENGTH (2,"a$"): (20)
40 DIM b(5)
50 PRINT LENGTH (1,"b("): 15)
60 PRINT LENGTH (2,"b("): (1)
```

функция имеет еще одно замечательное свойство. Если вместо параметра размерности массива ввести 0, то она выдаст адрес, в котором в памяти компьютера расположен первый элемент массива или символьной строки:

```
LENGTH (0, имя массива)
```

Эта возможность может быть использована теми, кто программирует в машинных кодах - можно найти адрес, в котором хранятся те или иные данные и даже можно найти адрес, с которого начинается процедура, записанная в машинных кодах, если она оформлена, как массив данных. Для тех же, кто программирует на БЕЙСИКе, имеется тоже немало интересных приложений, особенно если этой функцией пользоваться совместно с оператором POKE и функцией MEMORY$. Так, например, можно продублировать массив a$ в массиве b$, что может быть полезным при реорганизации данных:

```
20 LET e=LENGTH(1,"a$")
30 LET f=LENGTH(2,"a$")
40 DIM b$(e,f)
50 LET start = LENGTH(0,"a$")
60 POKE LENGTH (0,"b$"),MEMORY$()(start TO start+e*f-1)
```

Примечание: если Вы перебрасываете не символьный, а числовой массив, то в строке 60 следовало бы вместо e*f подставить 5*e*f, поскольку каждое действительное число в "Спектруме" хранится в пятибайтной форме.

Приведенный выше алгоритм во многом отличается от команды Бета-Бейсика COPY для массивов. Он, конечно, менее удобен, но может служить неплохой основой для создания на его базисе быстро работающей процедуры копирования массивов. Добавьте для этого строки:

```
10 DEF PROC dup REF a$, REF b$
70 END PROC
```

И теперь такая команда, как dup r$,t$ продублирует символьную строку r$ в строке t$.

Замечание для опытных пользователей:

Если у Вас уже есть ранее созданный массив данных, который Вы хотели бы использовать вместе с программой, написанной на Бета-Бейсике, но который слишком велик для загрузки, Вы можете разделить его на части и представить в виде блоков кодов (CODE), а затем использовать функцию LENGTH для того, чтобы программно перебрасывать эти блоки кодов в массив.

Без Бета-Бейсика Вы можете найти начало своего массива или символьной строки, если дадите им такое имя, которое обеспечит им первое место среди переменных в соответствующей области. Сделайте их объявление первым в программе. Затем найдите искомый адрес:

```
PRINT PEEK 23627+256*PEEK 23628+d
```

Здесь d=3, если Вы ищете символьную строку: d=6, если Вы ищете начало одномерного массива и d=8, если массив - двумерный.

Так Вы получите адрес первого байта данных в области переменных, поскольку символы, строки и числовые ряды в памяти идут друг за другом в линейной последовательности. Вы легко сможете рассчитать стартовый адрес и длину блока своего массива. Помните, что символы занимают по одному байту, а числа - по пять.

Примечание: поскольку с помощью оператора LET могут создаваться новые символьные строки или изменяться ранее существовавшие, то значения, ранее полученные функцией LENGTH (0,"имя") могут устаревать.

14. MEM ()

FN M()

Эта функция выдает количество свободных байтов в оперативной памяти компьютера. В скобках ставить ничего не надо. Попробуйте, например:

```
PRINT MEM(): DIM a$(100):PRINT MEM()
```

Эта простая функция содержит в себе в основной только вызов процедуры из области ПЗУ. Если Вы работаете в стандартном Бейсике, то выполнить ту же операцию можете командой

```
PRINT 65535 - USR 7963
```

15. MEMORY$ ()

FN M$()

См. также POKE симв. строка.

Эта функция выдает содержимое оперативной памяти компьютера в виде символьной строки. Правда, при этом содержимое нулевой ячейки памяти не включается. Поэтому команда

```
CODE MEMORY$()(1)
```

- то же самое, что и PEEK 1. Последние три байта оперативной памяти также по техническим причинам исключены, поэтому эта функция имеет максимальную длину 65532.

Конечно, если Вы попробуете что-то типа:

```
LET a$ = MEMORY$ (),
```

то Вам не хватит оперативной памяти, чтобы разместить там переменную a$, но пользоваться этой функцией надо по-другому. Например:

```
LET a$=MEMORY$()(16364 TO 22527)
```

Если учесть способность Бета-Бейсика выполнять POKE для символьных строк, то эта функция дает программисту возможность манипуляций с большими блоками памяти при очень высокой скорости (в отличие от стандартного БЕЙСИКа). Более подробно возможности применения этой функции Вы найдете в разделе, посвященном оператору POKE.

Другое применение MEMORY$ - для сканирования оперативной памяти при использовании INSTRING. Вы можете просканировать заданную секцию в памяти, указав пределы в функции MEMORY$ () (23759 TO ...). Но функция INSTRING работает настолько быстро, что этим можно и не пользоваться, а сканировать по всей памяти целиком.

```
10 REM asdfg
20 PRINT INSTRING (1,MEMORY$(), "asdfg")
```

Эта программа найдет строку asdfg в операторе REM в строке 10. Если Вы опустите строку 10, то asdfg будет найдена в строке 20. Если Вы присвоите:

```
LET a$ = "asdfg",
```

то эта символьная строка будет найдена в области программных переменных. Таким образом, где-нибудь в памяти, но Вы найдете содержимое любой символьной строки.

Вместо "1" в функции INSTRING мы могли бы дать DPEEK (23635) , т.е. содержимое системной переменной PROG для того, чтобы начинать поиск не от начала ПЗУ, а с того места, откуда начинается БЕЙСИК-программа.

Если же Вы хотите найти все случаи повторения в памяти заданной последовательности символов (байтов), то можете делать так:

```
10 LET adr=1
20 LET adr=INSTRING(adr, MEMORY$ (), a$)
30 IF adr THEN PRINT adr: LET adr=adr+1: GO TO 20
```

Поскольку Бета-Бейсик позволяет выполнение операции POKE для символьных последовательностей, Вы можете легко организовать поиск нужной последовательности и ее замену, если хорошо представляете себе то, что задумали. Так, например, разрушительной будет операция замены группы байтов на другую группу, которая длиннее первой.

16. MOD (число, число)

FN V(число, число)

Эта функция дает остаток от деления первого числа на второе.

MOD (10, 3) = 1
MOD (66,16) = 2
MOD (125,35.5) = 18.5

Нижеприведенный пример показывает, как можно избежать попыток выполнения команды PLOT вне пределов экрана.

```
10 FOR n=0 TO 400
20   PLOT MOD(n, 256),MOD(n,176)
30 NEXT n
```

17. NUMBER (симв. строка)

FN N(симв. строка)

См. также CHAR$ (число)

Функция преобразует двухсимвольную строку в целое число от 0 до 65535. Эквивалентом в стандартном БЕЙСИКе является выражение:

```
LET num=256*CODE c$(1)+CODE c$(2)
```

Если символьная строка имеет более двух символов, выдается сообщение об ошибке "Invalid argument".

С помощью функции CHAR$ функция NUMBER может применяться для создания целочисленных массивов, в которых вместо чисел использованы их символьные эквиваленты.

18. OR (число, число)

FN O(число, число)

Эта функция произносится так же, как и обычное ключевое слово "OR", но имеет другое действие. В программе или при вводе отличается иным синтаксисом.

Функция дает результат побитной логической операции "ИЛИ" для двух чисел, каждое из которых находится в пределах от 0 до 65535. Если какой-либо бит включен в первом или втором числе, то в результате этот бит тоже включен (равен 1). Он будет равен 0 только если он выключен в обоих числах одновременно.

19. RNDM (число)

FN R(число)

Если "число" равно 0, то функция RNDM выдает случайное число от 0 до 1 - точно так же, как и функция RND стандартного БЕЙСИКа. Если же "число" не ноль, то функция выдает случайное целое число, лежащее в диапазоне от нуля до заданного "числа".

Эта функция работает в два с половиной раза быстрее, чем работало бы выражение RND*"число".

```
10 PLOT RNDM(255),RNDM(175)
20 GO TO 10
```

Оператор RANDOMIZE "число" устанавливает генерируемую псевдослучайную последовательность в определенное положение точно так же, как он делает это для функции RND в стандартном БЕЙСИКе.

20. SCRN$ (строка, столбец)

FN K$(строка, столбец)

Работает примерно так же, как и стандартная функция SCREEN$, за исключением того, что может распознавать и символы графики пользователя UDG. Кроме того, исправлена ошибка системного ПЗУ, которая отражается на работе функции SCREEN$. Об этой ошибке мы писали в недавно выпущенной "ИНФОРКОМом" книге "Элементарная графика". См. также статью "Ошибки ПЗУ" в этом номере "ZX-РЕВЮ".

Перед тем, как набрать нижеприведенный пример, дайте команду KEYWORDS 0. Программа распределит по экрану символы графики пользователя в виде случайного рисунка, а затем считает некоторые из них с экрана.

```
10 FOR a=USR "a" TO USR "u" + 7
20 POKE a, RND*255
30 NEXT a
40 PRINT "символы графики пользователя"
50 LET a$- =""
60 FOR c=1 to 31
70 LET a$ = a$ + SCRN$(0,c)
80 NEXT c
90 PRINT a$
```

Блочную графику "Спектрума" эта функция не распознает. Если Вам и это необходимо, то запрограммируйте некоторые символы графики пользователя так, чтобы они выглядели, как символы блочной графики.

Символы могут распознаваться только в том случае, если они изображены в

стандартном размере 8x8 пикселов (см. CSIZE).

21. SHIFT$ (число, строка)

FN Z$(число, строка)

SHIFT$ - многоцелевая функция для преобразования строковых переменных. Она имеет много разных режимов работы. Режим задается параметром "число" при вызове функции. Вот краткий обзор ее режимов.

1. Все символы строки преобразуются в верхний регистр (в прописные буквы).
2. Все символы преобразуются в нижний регистр (в строчные буквы).
3. Регистр всех символов меняется на противоположный.
4. Подавление управляющих кодов. Все символы, являющиеся управляющими кодами, за исключением символа CHR$ 13 (код ENTER) заменяются символом "точка" (".").
5. Подавление токенов ключевых слов. Символы CHR$ 128...255 заменяются символами 0...127. При этом управляющие коды, за исключением ENTER (CHR$ 13) заменяются символом ".".
6. Подавление токенов ключевых слов. Символы CHR$ 128...255 заменяются символами 0...127, при этой все управляющие коды заменяются символом ".".
7. Все ключевые слова преобразуются из токенизированной (однобайтной) формы в многобайтную (по байту на каждый символ).
8. Все ключевые слова преобразуются из формы с полным написанием в однобайтные токены. Регистр символов роли не играет. После каждого ключевого слова должен стоять небуквеный символ.
9. То же, что и предыдущий режим, но после ключевого слова может стоять любой символ.
10. То же, что и предыдущий режим, но не все ключевые слова должны быть набраны прописными литерами.
11. То же, что и режим 8, но все ключевые слова должны быть набраны прописными буквами.

SHIFT$1...SHIFT$3

Преобразования регистров.

Рассмотрим примеры:

SHIFT$ (1,"Basic") = "BASIC"
SHIFT$ (2,"Basic") = "basic"
SHIFT$ (3,"Basic") = "bASIC"

Обычное применение этих режимов - преобразование символьных строк, вводимых пользователем, перед сравнением с контрольной строкой в диалоговых программах.

```
100 INPUT i$: IF SHIFT$(1,1$) ="Y" THEN GO TO 200
```

Это поможет вам уйти от целой последовательности сравнений, таких, как

```
IF i$ = "Y" OR i$ = "y"
```

При работе с базами данных, эта функция может использоваться для того, чтобы предварительно конвертировать массив записей пользователя в верхний регистр, прежде чем давать команду SORT.

SHIFT$4...SHIFT$6

Подавление управляющих кодов и токенов ключевых слов.

Эти режимы, по-видимому, найдут широкое применение у тех пользователей, которые программируют в машинных кодах. Так, при просмотре содержимого памяти компьютера, Вам может быть захочется распечатать содержимое ячеек командой

```
PRINT CHR$(PEEK address)
```

Очень скоро по этой команде Вы получите сообщение об ошибке "Invalid colour". Это произойдет как только вы попытаетесь распечатать непечатный символ.

Например, последовательность 17, 200 будет интерпретироваться, как CHR$17; CHR$200, а это в переводе с машинного языка на БЕЙСИК означает PAPER 200. Компьютер отреагирует сообщением об ошибке.

Сам формат печати при этом будет выглядеть весьма неопрятно, поскольку символы выше 127 могут быть распечатаны, как символы UDG, как символы блочной графики и как токены ключевых слов, имеющие самую разную длину.

Функция SHIFT$ позволяет справиться с этой проблемой, подавляя неприятные эффекты. В течение нескольких минут Вы сможете "прощупать" память компьютера в поисках таблиц данных, сообщений и списков ключевых слов.

```
100 FOR n=1 TO 65535 STEP 704
110 PRINT SHIFT$(6,MEMORY$())(n TO n+703)
120 PAUSE 0: CLS
130 NEXT n
```

Если область памяти, которую вы сканируете, содержит не машинный код, а БЕЙСИК-программу, то может быть Вам нецелесообразно отключать изображение токенов ключевых слов и достаточно только подавить управляющие коды режимом SHIFT$4.

SHIFT$7

Преобразование токенов в полную символьную запись.

Начнем с примеров:

```
10 LET a$ = " THEN NOT":
      REM это токены
20 PRINT a$, LEN a$:
      REM LEN=2
30 LET t$ = SHIFT$(7,a$)
40 PRINT t$, LEN t$:
      REM LEN=9
```

Эта функция должна быть полезной для тех, кто работает с принтером. Если принтер подключен не через стандартный "Синклеровский" интерфейс, то он не сможет воспроизводить на печать токены ключевых слов БЕЙСИКа, поскольку он о них ничего не знает.

Подобная конверсия поможет вам получать распечатки ваших БЕЙСИК-программ.

SHIFT$8...SHIFT$11

Преобразование ключевых слов из полной формы записи в токенизированную форму.

Эта функция преобразует все символьные последовательности, которые являются ключевыми словами БЕЙСИКа в токенизированную форму. Ключевое слово не будет распознано, если непосредственно перед ним стоит какая-либо буква. Что же касается символа, стоящего непосредственно за ключевым словом, то его влияние зависит от того, какой конкретно режим был избран (см. выше).

Эта функция может пригодиться в том случае, если вы примете БЕЙСИК-программу через внешний порт или через сеть от компьютера другой системы. Конвертировав записанные символами ASCII ключевые слова в токены и подправив синтаксис программы под свой "Спектрум", вы сэкономите массу времени, т. к. вам не придется набирать текст программы вручную.

22. SINE (число)

FN S(число)

Это модифицированная функция "синус". Она дает менее точный результат по сравнению с функцией SIN стандартного БЕЙСИКа, но зато работает в шесть раз быстрее.

Если для математических приложения она, может быть и не годится, зато для работы с векторной графикой, при расчете координат проекций на экране она будет хороша.

23. STRING$ (число, строка)

FN S$ (число, строка)

Эта функция дает строковую переменную, состоящую из параметра "строка", повторенного столько раз, каково значение параметра "число".

```
STRING$ (32,"-") = "--.....--"
                        (32 знака)
STRING$ (4,"AB") = "ABABABAB"
PRINT STRING$(704,"X")
```

- печатает экран, заполненный символами "X".

```
PRINT STRING$(3,"A"+CHR$ 13)
```

печатает: A
A
A

Если вам нужно сгенерировать строку, состоящую из более, чем 14 повторяющихся символов, то использовать STRING$ удобнее, чем вводить символы от руки. Кроме того, функция STRING$ работает быстрее, чем цикл FOR...NEXT, который тоже может быть применен для создания длинной регулярной строки.

Эта функция может применяться в Бета-Бейсике для заполнения блоков оперативной памяти информацией, например для установки необходимых экранных атрибутов. Для этой цели она используется совместно с функцией POKE.

24. TIME$ ()

FN T$()

См. также команду CLOCK.

Эта функция выдает текущие показания встроенных часов (если они были инициализированы в Бета-Бейсике). Если Вы несколько раз повторите команду PRINT TIME$(), то всякий раз получите разный результат. В программах невредно передать показания часов какой-либо переменной и, тем самым, "заморозить" полученный отсчет.

```
100 CLOCK 1
110 LET n$ = TIME$(): PRINT n$
120 PRINT "HOUrS= "; n$(1 TO 2);"Mins = ";n$(4 TO 5)
130 GO TO 110
```

Так можно встроить в программу контроль за временем исполнения программы пользователя. Подробности смотрите в разделе, посвященном описанию команды CLOCK.

25. USING$ (строка, число)

FN U$(строка, число)

См. также команду USING.

Функция конвертирует "число" в строковую переменную в формате, заданном параметром "строка". Вы можете задать количество изображаемых символов до десятичной точки и после. Ключевое слово USING, расположенное на клавише U обеспечивает то же самое в команде PRINT. В отличие от него, функция USING$ позволяет не только распечатать полученный результат, но и запомнить его. Она может быть использована не только с командой PRINT, но и с другими командами, допускающими работу со строковыми переменными. Более подробное описание смотрите в разделе, посвященном команде USING.

26. XOR (число, число)

FN X(число, число)

Функция выдает результат побитной операции "ИСКЛЮЧАЮЩЕЕ ИЛИ" для двух чисел, которые должны быть в пределах от 0 до 65535. Если какой-то бит и в первом числе и во втором равны между собой, то в результате этот бит будет равен нулю. Если же они противоположны, то в результате он будет равен единице.

ПРИЛОЖЕНИЕ 1

Ключевые слова Бета-Бейсика. Версия 3.0.

КОД	КЛАВИША	ТОКЕН
128	8	KEYWORDS
129	1	DEF PROC
130	2	PROC

131	3	END PROC
132	4	RENUM
133	5	WINDOW
134	6	AUTO
135	7	DELETE
136	Shift 7	REF
137	Shift 6	JOIN
138	Shift 5	EDIT
139	Shift 4	KEYIN
140	Shift 3	LOCAL
141	Shift 2	DEFAULT
142	Shift 1	DEF KEY
143	Shift 8	CSIZE
144	A	ALTER
145	B	-------
146	C	CLOCK
147	D	DO
148	E	ELSE
149	F	FILL
150	G	GET
151	H	-------
152	I	EXIT IF
153	J	WHILE
154	K	UHTIL
155	L	LOOP
156	M	SORT
157	N	ON ERROR
158	O	ON
159	P	DPOKE
160	Q	POP
161	R	ROLL
162	S	SCROLL
163	T	TRACE
164	U	USING

Примечание:

Для того, чтобы вернуться к стандартному для "Спектрума" значению вышеуказанных кодов, нужно перейти в режим работы KEYWORDS 0.

ПРИЛОЖЕНИЕ 2.

Сообщения об ошибках Бета-Бейсика, версия 3.0.

G No room for line

Заданные параметры при перенумерации строк приводят к тому, что в результате перенумерации появляется строки с номерами из диапазона, не подлежащего перенумерации или появляются строки с номерами больше 9999.

Ошибка проявляется при работе команды RENUM.

S Missing LOOP

Оператор цикла по условию DO WHILE, DO UNTIL или оператор выхода из цикла EXIT IF не могут найти оператора конца цикла LOOP.

Ошибка проявляется при работе операторов DO WHILE, DO UNTIL, EXIT IF.

T LOOP without DO

В программе присутствует оператор LOOP, но нет соответствующего ему оператора DO.

U No such line

В программе был использован оператор DELETE с указанием в качестве параметра номера строки, которой в программе нет.

Ошибка проявляется при работе оператора DELETE.

V No POP data

При попытке выполнить оператор POP оказывается, что стек GO SUB/ DO-LOOP/PROC - пуст. Это означает, что в данный момент времени не исполняются ни подпрограммы GO SUB, ни циклы DO-LOOP, ни процедуры PROC.

Ошибка проявляется при исполнении оператора POP.

W Missing DEF PROC

В программе была попытка исполнить процедуру, которая ранее не была определена оператором DEF PROC. То же происходит, если различаются имена процедуры при задании и при вызове. Ошибка может возникать, если встретился оператор END PROC, а процедура ранее не была объявлена через DEF PROC.

Ошибка проявляется при вызове процедур и при исполнении операторов END PROC и LOCAL.

X No END PROC

Программа во время работы пытается обойти объявление процедуры, но не может найти оператора END PROC, который соответствовал бы DEF PROC.

Ошибка проявляется при встрече оператора DEF PROC.

ПРИЛОЖЕНИЕ 3

Коды ошибок.

Ниже приведен список кодов, которые записываются в переменную ERROR по команде ON ERROR. Список состоит из трех разделов. В первом разделе перечислены состояния, связанные со стандартным БЕЙСИКОМ "Спектрума". Во втором разделе - ошибки БЕТА-БЕЙСИКА. В третьем разделе - ошибки, связанные с ИНТЕРФЕЙСОМ-1.

Примечание: Коды 0 и 9 (а они фактически ошибками не являются) не перехватываются оператором ON ERROR.

1. Для стандартного БЕЙСИКа.

Значение ERROR	Код ошибки	Сообщение
0	0	O.K.
1	1	NEXT without FOR
2	2	variable not found
3	3	Subscript wrong
4	4	Out of memory
5	5	Out of screen
6	6	Number too big
7	7	RETURN without GOSUB
8	8	End of file

9	9	Stop statement
10	A	Invalid argument
11	B	Integer out of range
12	C	Nonsense in Basic
13	D	BREAK-CONT repeats
14	E	Out of DATA
15	F	Invalid file name
16	G	No room for line
17	H	STOP in INPUT
18	I	FOR without NEXT
19	J	Invalid I/O device
20	K	Invalid colour
21	L	BREAK into program
22	M	RAMTOP no good
23	N	Statement lost
24	O	Invalid stream
25	P	FN without DEF
26	Q	Parameter error
27	R	Tape loading error

2. Для БЕТА-БЕЙСИКа

28	S	Missing LOOP
29	T	LOOP without DO
30	U	No such line
31	V	No POP data
32	W	Missing DEF PROC
33	X	No END PROC

3. Для ИНТЕРФЕЙСа-1.

43	b	Program finished
44	c	Nonsense in BASIC
45	d	Invalid stream number
46	e	Invalid device expression
47	f	Invalid name
48	e	Invalid drive number
49	h	Invalid station number
50	i	Missing name
51	j	Missing station number
52	k	Missing drive number
53	l	Missing baud rate
54	m	Header mismatch error
55	n	Stream already open
56	o	Writing to a "read" file
57	p	Reading a "write" file
58	q	Drive "write" protected
59	r	Microdrive full
60	s	Microdrive not present
61	t	File not found
62	u	Hook code error
63	v	CODE error
64	w	MERGE error
65	x	Verification has failed
66	y	Wrong file type

На этом мы заканчиваем печать инструктивных материалов, посвященных диалектам БЕЙСИКА, выпущенным фирмой BETASOFT. За два года мы рассмотрели BETA-BASIC (версии 1.0, 1.8 и 3.0.).

Существует и еще более мощная версия 4.0, которая поддерживает работу с дисковой операционной системой. К сожалению, технической документации по этой версии у нас пока нет, а потому мы будем очень признательны тем из наших читателей, которые смогут такую документацию предоставить (на языке оригинала). Условия приобретения обсуждаются, в этом случае мы сможем довести эту информацию до самых широких кругов.

ЗАЩИТА ПРОГРАММ

Сегодня мы заканчиваем печать третьего тома книги В.С. Михайленко, посвященной вопросам защиты программ для "Синклер"-совместимых компьютеров и переходим к четвертому тому, который написан в соавторстве с экспертом из Белорусского Государственного Университета (БГУ) А. К. Туровичем.

Полностью печать статей данного цикла будет завершена в 1993 году.

Продолжение.
(Начало: 9-16, 53-60, 97-104, 141-146, 185-192).

1.2 Смещение системной переменной PROG.

Многие из Вас, вероятно, не раз убеждались в справедливости принципа, "чем проще - тем надежнее". Этот принцип был известен издавна и не раз подтверждал себя на практике. Актуальным он является и для нас, потому что темой нашего разговора будет метод, основанный на смещении системной переменной PROG. Метод, на первый взгляд достаточно простой, но, тем не менее, достаточно эффективный. Рассмотрим более подробно принцип его применения для защиты компьютерных программ.

Как Вам уже, вероятно, известно, Бейсик, в стандартном "Спектруме (без подключенной периферии) начинается с адреса 23755. Об этой свидетельствует содержимое системной переменной PROG (23635). Таким образом, практически всегда Бейсик в компьютере начинается с одного и того же адреса памяти. Однако, такое положение вещей достаточно легко изменить, если осуществить изменение системной переменной PROG. Рассмотрим, что это нам дает.

Предположим, вы разработали новый загрузчик в кодах и хотели бы затруднить его прочтение и просмотр без Вашего ведома. Для этого достаточно хорошо подходит данный метод защиты.

Для начала вы создаете специальную программу в машинных кодах, которая осуществляет изменение системной переменной PROG. После этого Вам понадобится совместить ее с программой на Бейсике (о том, как это сделать, было подробно написано в т.1 гл.1). Конечно, можно достаточно просто изменить содержимое PROG и из Бейсика путем применения POKE, однако предпочтительней закамуфлировать выполненные операции, а еще лучше замаскировать адрес старта программы в кодах одним из методов, предложенных в т.1 гл. 4 (Напомню что основное предпочтение отдавалось там изменению содержимого числового значения, расположенного после управляющего кода 14). Заканчивает данную программу на Бейсике команда LOAD"".

Как Вы уже, вероятно, догадались, данная Бейсик-программа служит исключительно для подготовки к загрузке Вашей специальной программы-загрузчика. Причем Ваш загрузчик в машинных кодах ассемблирован таким образом, чтобы работать только в новой области (после изменения системной переменной PROG). Адрес старта загрузчика, вполне естественно, тоже будет замаскирован и будет указывать на точку расположения вашей исходной процедуры после изменения системной переменной. Теперь Вам осталось только совместить исходную программу в машинных кодах с новой Бейсик-программой и после выяснения адреса реального старта осуществить необходимые изменения чисел после функции RANDOMIZE USR.

Давайте теперь рассмотрим, как работает данная защита. После проведения вышеописанных работ мы имеем две Бейсик-программы, в каждую из которых встроен блок машинных кодов. Причем процедура в кодах, встроенная в первую Бейсик-программу, осуществляет изменение системной переменной PROG, в то время как вторая процедура в кодах, встроенная во вторую Бейсик-программу, является ничем иным, как исходной программой-загрузчиком, которую Вы и собираетесь защитить от несанкционированного

просмотра.

После загрузки первой Бейсик-программы, она автоматически запускается и изменяет системную переменную PROG так, чтобы следующая Бейсик-программа загружалась уже в новую область. Такое изменение дает нам исключительную возможность правильно запустить Вашу программу-загрузчик в машинных кодах. Это объясняется тем, что ваша программа-загрузчик ассемблирована под новое значение системной переменной PROG и не будет работать, если не выполнить вышеописанных изменений.

Карта памяти компьютера до и после изменения системной переменной PROG показана на рис. 1.

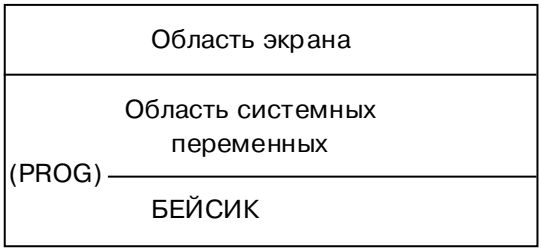


Рис. 1 а

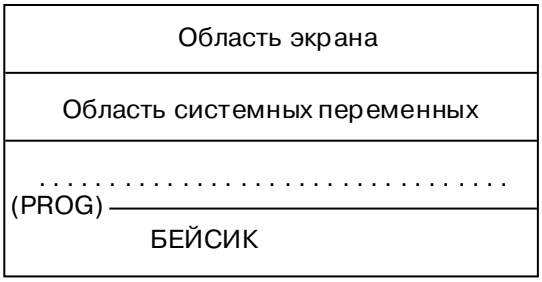


Рис. 1 б

Случай а) показывает вариант, когда Бейсик находится сразу после области системных переменных, т.е. начиная с адреса 23755. Случай б) рассматривает вариант изменения значения системной переменной PROG, т.е. между областью системных переменных и Бейсиком существует незаполненное полезной информацией пространство.

Примечание: Разумеется, изменение системной переменной PROG само по себе уменьшает объем памяти, доступной для пользователя, однако это бывает оправдано, например, когда мы строим на этом методе защиту.

Любопытно, что системная переменная PROG изменяется принудительно, в случае подключения INTERFACE 1. в этом случае между областью системных переменных и Бейсиком размещаются системные переменные, необходиные для работы 8K ПЗУ INTERFACE 1.

На рисунке взятая в скобки надпись системной переменной PROG означает, что Бейсик начинается с ячейки памяти, адрес которой находится в системной переменной PROG.

Теперь вы видите, что если попробовать загрузить второй Бейсик-файл не изменяя PROG, то программа в кодах, размещенная там, не будет работать, аналогично, как не будет работать и обыкновенная программа в кодах, перемещенная из места, для которого она ассемблирована, в какое-либо иное место памяти. Для программистов, работающих в машинных кодах причина неработоспособности в подобном случае очевидна - не совладают адреса переходов при обычной адресации.

Небольшая историческая справка.

Когда взлом компьютерных программ только начинался, во многих программах Билла Гильберта, да и других "хаккеров" можно было встретить строку приблизительно следующего содержания:

```
0 LIST USR (PEEK 23635 + 256*PEEK 23636 +17)
```

которая собственно и должна была запускать встроенную программу в машинных

кодах. Очевидно, что LIST USR - это лишь одна из разновидностей команды RANDOMIZE USR (об этом уже было записано в главе "новейшие достижения защиты"). Как видно, программа в кодах запускалась с адреса на 17 байтов большего, чем тот, на который указывала системная переменная PROG. Четыре байта уходили на номер и длину строки, еще один - на код оператора REM. А остальное место до начала программы Гильберт любил заполнять своими инициалами. Удобство такой записи адреса старта процедуры в кодах заключалось в том, что она сама находила, где находится Бейсик-программа. Это было необходимо ввиду широкого распространения в то время ИНТЕРФЕЙСА 1 и микродрайвов, которые изменяли значение PROG. Кроме того, такая запись затрудняла пользователю прочтение реального адреса старта машинокодовой процедуры.

Однако, позднее на основе этого появилось достаточно любопытное направление защиты программ, описание которого было приведено выше. Я надеюсь, что мои рекомендации не только помогут читателям лучше прояснить работу своего компьютера, но и пригодятся в повседневной работе при разработке своих программ.

1.3 Кодирование и декодирование блоков машинных кодов.

Одной из разновидностей защиты программ в кодах является кодирование этих процедур. Под кодированием понимается изменение истинных значений программы в кодах с целью дестабилизации ее работы в нераскодированнои виде. Для обеспечения нормального функционирования программы в кодах ее необходимо подвергнуть раскодированию.

Одним из самых простых методов кодирования является изменение содержимого программы с использованием команды LDIR. Рассмотрим, как работает процедура, обеспечивающая правильное выполнение данной защиты.

Не вдаваясь в подробности, хочу напомнить читателям, что инструкция LDIR осуществляет перенос блоков кодов из одного места памяти в другое. Все необходимые значения для выполнения данной операции задаются заранее и заносятся в регистры микропроцессора. Для тех, кто интересуется работой данной команды процессора Z80 более подробно рекомендую книгу "ИНОФОРКОМа" "Первые шаги в машинных кодах".

Итак, каким же образом действует данная защита. Предположим, что исходная программа содержит два блока в машинных кодах. Один из них достаточно большой (порядка нескольких десятков килобайт), а второй небольших размеров (порядка нескольких килобайт или даже нескольких сот байтов).

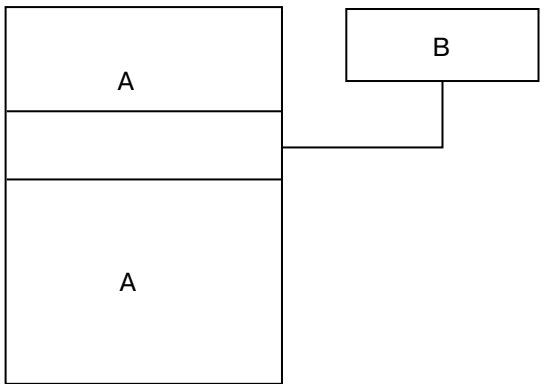


Рис. 2

На рисунке изображены блоки кодов, из которых состоит исходная программа.

Стрелкой показано, что после загрузки второй блок занимает заранее отведенное ему место в памяти, после чего программа может нормально функционировать. Перемещение блоков кодов из одного места памяти в другое осуществляется с использованиеи инструкции LDIR.

На рис. 2 изображена технология осуществления защиты данного типа защиты. Сразу после запуска программы осуществляется перенос блоков кодов из одного места памяти в другое, после которого программа может нормально функционировать.

Разумеется, в каждом конкретном случае эта защита может иметь определенные разновидности. В частности, исходный программный файл может состоять не из двух, а из гораздо большего числа блоков и перенос может осуществляться не для целого блока, а лишь какой-то его части. Если Вам ясен сам принцип защиты, то Вам не составит труда разобраться с его каждым конкретным применением.

Одним из примеров программы, использующей данный принцип, защиты может служить программа GREEN BERET, загрузчик которой был достаточно подробно описан в четвертой главе второго тома.

Следующий тип кодирования, который я хочу предложить Вашему вниманию, тоже получил распространение. Это объясняется тем, что в данном случае нам не требуется догружать какие-либо дополнительные блоки, чтобы восстановить правильное содержимое исходной программы - достаточно использовать специальную программу декодирования, которая используя специальный алгоритм воссоздаст из загруженного блока кодов содержимое исходной программы.

Рассмотрим более подробно, как этого можно добиться.

Одним из наиболее широко используемых приемов является самая обыкновенная инверсия содержимого ячеек памяти.

Как Вы уже вероятно знаете, любое значение, содержащееся в определенной ячейке памяти можно представить десятиричным значением от 0 до 255, шестнадцатиричным значением от 0 до FF или двоичным значением 00000000 до 11111111. Для программиста все эти системы счисления альтернативны, хотя известно, что компьютер оперирует двоичными значениями числа, в то время как при программировании принято использовать шестнадцатиричную систему счисления (шестнадцатиричная система счисления - HEX система - является профессиональной системой счисления для программистов). Так вот, любое шестнадцатиричное число можно представить в двоичном виде, используя специальные таблицы. А процесс инверсии очень легко понять, используя двоичное представление. Инвертировать байт - это значит изменить содержимое каждого его бита на противоположное.

Например, байт 00 после инверсии превращается в байт FF:

0000 0000 - байт 00H представленный в двоичном виде;

1111 1111 - после инверсии все значения данного байта заменяются на противоположные. Шестнадцатиричное значение данного двоичного числа FF.

Аналогично, байт 01010101 после инверсии преобразуется в 10101010 и т.д.

Как видите, кодирование инвертированием может применяться достаточно успешно, поскольку байты загружаемого блока не смогут правильно обрабатываться микропроцессором без соответствующего преобразования. Для того, чтобы программа заработала правильно, необходимо перед ее запуском снова инвертировать эти байты, чтобы все стало на свои места.

Однако кодирование инверсией, как Вы могли убедиться, является достаточно примитивным и поэтому настало время рассмотреть систему защиты, применяемую в большинстве фирменных программ (ART STUDIO, ATIC ATAC, NIGHT SHADE, THE WORD).

Рассмотрим эту систему зашифровки на примере программы "ART STUDIO".

В данном случае кодирование представляет собой систему достаточно простого типа, делающего невозможным правильную работу программы. Для расшифровки применяется специальная декодирующая процедура, которая находится в той же программе и, что очень важно, не закодирована. В данном случае (впрочем, как и во всех рассмотренных ранее) кодирование просто затрудняет доступ к тексту программы, после того, как все предшествующе защиты устранены и программа считана без автозапуска.

Несмотря на то, что бейсиковская часть программы практически не защищена, необходимо достаточно внимательно изучить систему декодирования прежде, чем осуществлять эксперименты с запуском процедуры на выполнение.

Поскольку запуск программы начинается с адреса 26000, исследуем команды, расположенные начиная с этого адреса:

```
26000 JP 26024
```

Как видим, сразу осуществляется переход к процедуре декодирования. (В дальнейшем мы рассмотрим с Вами и процедуру кодирования, размещенную в данной программе в адресах 26003...26021).

```
26034 LD HL,26049
26027 PUSH HL
26028 LD HL,25049
26031 LD DE,26719
26034 LD A (HL)
26035 SUB 34
26037 RLCA
26038 XOR #CC
26040 LD (HL),A
26041 INC HL
26042 OR A
26043 SBC HL,DE
26045 ADD HL,DE
26046 JP NZ,26034
26048 RET
```

Данная процедура сначала помещает в стек значение адреса 26049 - сюда программа перейдет после выполнения команды RET.

После этого и начинается процесс декодирования; в регистр HL снова загружается адрес 26049 - как начало декодируемого блока. Затем в цикле декодируются последовательно байты загружаемого блока, причем инструкции:

```
SUB 34
RLCA
XOR #CC
```

являются ключем, с помощью которого расшифровывается эта часть программы. В ходе работы проверяется условие достижения адреса 27719, как последнего декодируемого байта (это значение содержится в DE).

Выполнение инструкции RET приводит не к возврату в Бейсик, а к переходу на адрес, который был последним занесен в стек - в нашем случае 26049. Следовательно, по этой инструкции происходит запуск на исполнение расшифрованного блока.

Следует отметить, что инструкции, осуществляющие дешифрацию, в ходе своей работы не теряют ни одного бита. Это необходимое условие для данного типа программ, поскольку кодированию и декодированию может подвергнуться практически любой байт от 00 до FF.

Теперь рассмотрим, каким образом можно осуществить декодирование данной программа, чтобы после возврата в Бейсик подробно изучить новообразующуюся часть программы. Для этого нам понадобится перестроить работу исходной программы так, чтобы на стек не попало значение 26049, а там оставалось последним значение возврата в бейсик-программу.

Для выполнения поставленной цели существуют два пути:

- либо уничтожить команду PUSH HL, заслав вместо нее код NOP, т.е. ноль.

- либо запустить программу после помещения на стек вышеуказанных значений, командой RANDOMIZE USR 26028. В этом случае мы возвратимся в Бейсик по выполнении инструкции RET.

Теоретически оба метода взаимозаменяемы и могут быть применены равноправно. Однако, на практике оказывается, что это не так. В частности, в программе ART STUDIO существует блок программы, проверяющий сохранность значений ячеек памяти в месте расположения декодирующей процедуры. Если он обнаружит изменение содержимого ячеек памяти, то программа зависнет.

Процедура проверки находится начиная с адреса 26283:

```
26263 LD H,A
26284 LD L,A
26285 PUSH HL
26292 LD A,(26024)
26295 CP #21
26297 RET NZ
```

```
26298 LD HL,(26025)
26301 OR A
26302 LD DE,26049
26305 SBC HL,DE
86307 RET NZ
26308 LD A,(26027)
26311 CP #E5
26313 RET NZ
26314 POP HL
26315 RET
```

Рассмотрим более подробно, как работает данная процедура.

Для начала мы помещаем 0 на стек, используя то, что содержимое аккумулятора равно 0. После этого осуществляется проверка байтов 26024-26027 (включительно) и, если обнаруживается несовпадение с тем, что там ожидал найти автор программы, то происходит возврат на 00, а это ничто иное, как перезапуск компьютера. Если же проверка прошла успешно, мы снимаем ноль со стека и возвращаемся в вызвавшую данную подпрограмму процедуру.

Теперь Вы видите, что менять содержимое ячейки POKE 26027,0 достаточно рискованно. Поэтому в данном случае лучше воспользоваться вторым предложенным вариантом, а на будущее запомнить, что возможность проверки работы защиты - достаточно частое явление в компьютерных программах.

Рассмотренные нами три примера, разумеется, не исчерпывают всех возможных вариантов кодирования. Каждый программист старается разработать свой собственный метод, как можно более изощренный. Однако, при подобной разработке Вам необходимо учитывать, что инструкции, осуществляющие шифрацию и дешифрацию, не должны терять в ходе своей работы ни одного бита. В программе ART STUDIO вычитание производится по модулю 256 и для двух разных входных данных результаты тоже различны. RLCA заменяет значение битов 7,6,3,2 на противоположные.

Другими операторами, имеющими аналогичные свойства, являются ADD, INC, DEC, RRCA, CPL и др.

Однако, в данном случае нельзя использовать функции OR или AND, поскольку Вам не удастся правильно восстановить содержимое исходной программы.

1.4 Новые POKES.

Многие программисты, длительное время работающие с компьютером, наверняка изучили большинство POKES, обычно применяемых для защиты компьютерных программ. Многие считают малоперспективным разрабатывать новые приемы защиты, основанные на этом направлении и пытаются создавать более изощренные приемы. Тем не менее, несмотря на свою давнюю историю, этот метод защиты компьютерных программ не следует предавать забвению.

Несколько новых адресов, благодаря использованию которых можно защитить информацию от несанкционированного просмотра, помогли данному методу подняться на новые рубежи и составить конкуренцию традиционно используемым в современных игровых программах приемам защиты.

Метод засылки в определенные ячейки памяти измененных значений (в большинстве случаев этими ячейками являются системные переменные) начал применяться с самого начала появления защиты компьютерных программ для "Спектрума". Он применялся как для защиты от нажатия клавиши "BREAK", так и от произвольного листинга. (См. т.1). Следует отметить, что в дальнейшем использование данной методики было сильно ограничено ввиду незначительного количества ячеек памяти, которые приводили к защитному эффекту. Это привело к тому, что практически вся информация о POKES быстро стала широким достоянием большого числа "хаккеров". По этой причине информация о новых POKES, имеющих "защитные" функции является тем козырем, который даст вам возможность охладить неумеренный пыл юных взломщиков. В этой статье я приведу информацию о двух ячейках памяти, изменяя содержимое которых, можно получить либо оригинальную защиту от листинга, либо защитить работающую программу от непредусмотренной остановки

нажатием клавиши "BREAK". Несмотря на то, что эти методы различны по принципу действия, их объединяет то, что изменение содержимого ячеек памяти происходит в области системных переменных. Рассмотрим каждый из них более подробно.

1. POKE 23743,80 приводит к тому, что мы не можем получить на экране никакой информации, в том числе и листинга. Это объясняется тем, что мы изменили адрес программы вывода на основной экран компьютера. Тем не менее, адрес программы вывода с клавиатуры в нижнюю часть экрана остался прежним и поэтому мы можем вызвать в командную строку любую из строк исходной программы командой EDIT. Такая возможность несомненно является достаточно значительным дефектом эти защиты, однако это дело поправимое, поскольку для защиты от подобных ухищрений Вы можете аналогичным образом изменить и адрес программы вывода для этого канала.

В заключение лишь следует добавить, что для нормализации работы компьютера Вам необходимо набрать:

```
POKE 23743,83
```

Эти изменения нормализуют работу канала и теперь Вы можете без труда ознакомиться с листингом.

Данный метод защиты был обнаружен мной в программе MASBLAST -1990.

2. POKE 23613, PEEK (23700)-5 создает защиту от нажатия клавиши "BREAK" и приводит вначале к зависанию компьютера, а через несколько секунд к самосбросу.

Данный метод основан на изменении содержимого одной из системных переменных, ответственных за адрес в аппаратном стеке, используемый как адрес возврата при ошибке. Изменив предлагаемым образом содержимое ячейки, Вы при нажатии клавиши "BREAK" попадаете на мнимую подпрограмму обработки ошибки, которая и приводит ко всем вышеописанным результатам.

Этот метод защиты, к сожалению, нельзя разблокировать во время его работы и для успешного его преодоления вам придется загружать программу без автозапуска одним из предложенных ранее методов.

Как видите, данные методы защиты основаны на использовании хорошо известных приемов POKE и я уверен, что возможности данной методики еще далеко не исчерпаны и в будущем свое слово здесь скажете Вы, дорогие читатели.

1.5 Метод нулевых строк - новые возможности.

В данной статье разобран оригинальный метод защиты компьютерных программ к "Спектрум"-совместимым компьютерам. Подробные комментарии позволяют использовать этот материал не только как полезную процедуру, но и как пособие тем, кто самостоятельно изучает программирование на языке АССЕМБЛЕРА.

В первом томе (см. стр. 13) Вашему вниманию была предложена статья "Универсальная система защиты - метод нулевых строк". Несмотря на определенные достоинства, эта программа обладает целым рядом недостатков, которые в некоторых случаях могут нарушить работу Вашей исходной программы на Бейсике. Постараюсь объяснить причину возникающих в ходе работе программы "зануления" неточностей.

Как уже вероятно убедились пользователи, работавшие с моей программой, она практически всегда справляется со своей задачей. Однако, необходимо заметить, что программа не учитывает некотрых особенности Бейсик-строки "Спектрума" и поэтому существует вероятность ошибочного "зануления". Несмотря на то, что в большинстве случаев программа работает нормально, теоретически нельзя не учитывать возможность ошибки. Попробуем разобраться, почему это может произойти.

Вам Вам уже, вероятно, известно, все числа в Бейсике "Спектрума" представлены в пятибайтной интегральной форме. Это достаточно своеобразное представление и необходимо оно для правильной работы встроенного калькулятора. Существует несколько различных форм данного пятибайтного представления, в зависимости от того, является ли исходное число действительным либо целым (для целых чисел форма представления также различна - все зависит от того, является ли оно положительным или отрицательным). Для

тех, кто хочет более подробно разобраться в этом вопросе, рекомендую читать "Первые шаги в машинных кодах".

Я же, не вдаваясь в подробности, хочу отметить, что в этой пятибайтной последовательности чисел вполне может встретиться и число "13", однако в случае наличия его в пятибайтном числовом стринге, оно не должно анализироваться программой, как код "ENTER ", поскольку данный стринг является единым целым (компьютер определяет это по обнаружению управляющего кода "14"). Во всех остальных случаях число "13" должно восприниматься компьютером как код "ENTER".

Предложенная в первом томе программа не учитывала, что число "13" может содержаться в такой пятибайтной форме чисел и поэтому вполне могло произойти ошибочное "зануление". В самой деле, программа, обнаружив подобное число, обязательно превратило бы в ноль следующие за этим числом два байта, в то время как делать этого не следует.

Второй обнаруженный недочет состоит в том факте, что моя программа начинает анализ Бейсика с адреса 23755, в то время как это не всегда корректно, поскольку в некоторых случаях область Бейсика может сдвигаться "вверх" (например при подключении периферии). Точно определить адрес начала области Бейсика нам поможет содержимое системной переменной "PROG".

С момента опубликования вышеуказанной статьи я продолжал работу над данной темой и пришел к ряду выводов, с некоторыми из которых хочу поделиться с читателями. Разработанная мной программа не "зануляет" первую строку Вашей программы, что несомненно осложняет ее использование. Кроме этого, стоит отметить, что решение, когда программа "зануления" располагается после исходной программы на Бейсике, не совсем удачно, так как:

- по-первых, выполнение программы "зануления" осуществляется на Бейсике, что существенно сказывается на ее быстродействии;

- во-вторых, почти всегда придется "занулять" все строки программы, а это не всегда необходимо;

- в-третьих, бывают случаи, когда предпочтительней использовать процедуру в машинных кодах, поскольку после работы Бейсик-программы "зануления" ее необходимо уничтожить, в то время, как программа в машинных кодах в этом не нуждается.

Вышеописанные недостатки делают проблематичным использование моей исходной программы в некоторых случаях. Однако, мне удалось составить программу, которая не имеет вышеописанных недостатков. Рассмотрим ее более подробно.

Основным требованием к такого рода программе было то, чтобы она могла загружаться в любую область памяти без потери работоспособности. Это объясняется тем, что в некоторых случаях область памяти, для которой данная программа была ассемблирована, бывает занята и, чтобы обеспечить удобную эксплуатацию подобной процедуры, необходимо иметь возможность загружать ее в произвольную область ОЗУ с сохранением всех выполняемых функций. Подобный эффект становится возможным благодаря использованию относительной адресации. Этот вид адресации может использоваться командами как условного, так и безусловного переходов. В таком случае они состоят из двух байтов: первый байт содержит код операции, а второй - смещение в двоичной дополнительной системе. Действительный адрес получается прибавлением смещения к текущему показанию программного счетчика. Преимуществами относительной адресации перед абсолютной являются:

- команда занимает в памяти меньше места на один байт;

- программа становится перемещаемой, то есть не зависит от своего места расположения в памяти.

Полученная программа приведена в Листинге 1. Рассмотрим принцип ее действия.

Листинг 1.

```
Programing by Michailenko Vadim.
All rights reserved. Mensk 1991.
Michailenko Vadim driver system
```

```
for "EDITAS-48" files special for "INFORCOM".
10                ORG 64130
20                LD BC,9990
30                LD HL,(23635)
40                XOR A
50                LD HL,(A)
60                INC HL
70                LD (HL),A
80    ;     ---------------------------
90    NEXT        INC HL
100               LD E,(HL)
110               INC HL
120               LD D,(HL)
130               ADD HL,DE
140               LD A,(HL)
150               CP 13
160               RET NZ
170               INC HL
160               LD A,(HL)
190               CP B
200               JR Z,FINAL
210   ;     ---------------------------
220   CONT        XOR A
230               LD (HL),0
240               INC HL
250               LD (HL),0
260               JR NEXT
270   ;     ---------------------------
280   FINAL       INC HL
290               LD A,(HL)
300               CP C
310               RET Z
320               DEC HL
330               JR CONT
340               END
```

Сразу после начала работы в регистр BC заносится номер строки программы, до которой необходимо вести "зануление". Это значение можно изменять. Таким образом, Вы сможете осуществить "зануление" не всей исходной программы, а лишь определенной ее части. О том, как заменить число 9990 на другое, будет написан ниже.

После этого, по содержимому системной переменной PROG мы узнаем начало области Бейсика и осуществляем "зануление" первой программной строки.

Команда XOR A - простейший способ обнуления аккумулятора, ставший стандартным.

Это был подготовительный этап в работе программы - далее следует работа в циклическом режиме. По содержимому двух байтов, следующих за номером строки, программа определяет длину данной строки и, следовательно, сразу же может определить начало следующей. Теперь нобходимо проверить, не имеет ли данная строка программы номер 9990, который свидетельствовал бы об окончании работы, и, если номер совпадает, то процедура заканчивает свою работу и осуществляет возврат в Бейсик. В случае, если необходимый номер еще не достигнут, программа "зануляет" номер текущей строки и осуществляет возврат к началу цикла, после чего процесс повторяется.

Данная процедура имеет подстраховку для забывчивых пользователей. Если Вы обратили внимание, то возврат в Бейсик предусмотрен и после команды сравнения CP 13. Это может пригодиться, если Вы забыли создать строку с контрольным номером. В этом случае процедура "занулит" все строки программы и возвратится в Бейсик, когда обнаружит окончание исходной программы. При такой форме работы компьютер не "зависнет" и вам не придется перезагружать процедуру "зануления". Однако, свою исходную программу вам все же придется перезагрузить, так как процедура в машинных кодах внесет непоправимое изменение в номера строк Вашей исходной программы на Бейсике.

Сведущий пользователь без труда сможет внести коррективы в текст исходной программы в машинных кодах с тем, чтобы она избегала нежелательных изменений при

любых вариантах ее использования, однако я не стал этого делать с тем, чтобы программа имела как можно более простой вид и даже начинающий в программировании на ассемблере мог легко в ней разобраться. Тем не менее, если Вы с самого начала будете внимательны и не забудете задать строку с контрольным номером, то Вам не придется надеяться на подстраховку.

Наиболее простым вариантом ввода такой строки было бы набрать:

```
9990 REM ENTER.
```

Однако, как уже упоминалось, Вы можете изменить номер контрольной строки и для этого вам необходимо ввести соответствующие изменения в исходную программу в машинных кодах, в случае, если нобходимый вам номер равен X, то достаточно подать команды:

```
LET a=INT(X/256): LET b=X-256*a:POKE adr+1,b: POKE adr+2,a
```

В данном случае adr - это значение ячейки памяти, начиная с которой вы загрузили процедуру "зануления".

Естественно, что вводить новый номер конечной строки надо перед запуском процедуры.

Для читателей, которые не имеют ни малейшего желания разбираться в программах в машинных кодах, однако желающих использовать данную процедуру, я привожу здесь программу на Бейсике, которая позволит Вам загрузить в память компьютера описанную выше процедуру "зануления" (см. Листинг 2).

Листинг 2.

```
5 REM programming by Michailenko Vadim Mensk 1992
10 CLEAR 63129
20 FOR I=62130 TO 62168
30 READ A: POKE I,A
40 NEXT I
50 REM RANDOMIZE USR 68130
60 DATA 1,6,39,42,83,92,175,119,35,119,35,94,35,86,25,126,254,13,192,35,136,164,40,8,175,
      54,0,35,5,0,24,234,35,126,185,200,43,24,241
```

Данная программа на Бейсике загружает блок машинных кодов, начиная с адреса 62130. Для того, чтобы записать этот блок машинных кодов отдельным файлом, необходимо подать прямую команду:

```
SAVE "имя файла" CODE 62130,39
```

Как уже упоминалось в статье, данная программа рассчитана для работы в произвольной области ОЗУ. Для того, чтобы загрузить ее в необходимую Вам область памяти, следует подать команду:

```
LOAD "имя файла" CODE adr
```

где adr - адрес начала загрузки.

Том 4. Методы защиты программ от копирования.

Введение.

Все, что мы до сих пор рассматривали, по сути было защитой программ от несанкционированного просмотра. Теперь настал момент перейти к более важным моментам - защите от копирования. Поскольку защита от копирования тоже делается программно, то это и есть та самая уязвимая точка, которую и надо защищать от просмотра и мы полагаем, что теперь Вы уже умеете это делать и готовы идти дальше.

Данная книга посвящается вопросам защиты программ от копирования. Этот материал сегодня наиболее интересен для наших программистов, поскольку до сих пор не действует закон об авторском праве на программные продукты. Таким образом, для того, чтобы защитить свои авторские права программисту приходится прибегать к защите от копирования, чтобы только зарегистрированный пользователь мог работать с программой.

Данная проблема сегодня характерна для всех типов компьютеров. "Спектрум" - не исключение. Скорее, даже наоборот и у все большего числа людей возникает стремление

написать свои собственные программы. У них есть для этого и желание и способности но, наступает момент и автор начинает задумывается: а стоит ли это делать это, если ему не удастся защитить свое авторское право на данную разработку.

Естественный выход в подобных ситуациях - ограничение количества неофициальных пользователей. Компьютерное пиратство приобрело в нашей стране в последние годы небывалый размах. Единственным способом хоть как-то сдерживать его оказалось введение защиты компьютерных программ от копирования.

Для "Спектрума" существует несколько различных способов записи программ. В первую очередь, они подразделяются по типу носителей информации: большинство программ записаны на обычных магнитофонных кассетах, в тоже время в последние годы у пользователей все больший интерес приобретает дисковая операционная система.

Мы рассмотрим здесь методы защиты программ, записаных на магнитофонные кассеты. С момента возникновения "Спектрума" данных методов было изобретено очень большое количество, продолжают они совершенствоваться и сегодня. Кроме этого начинают появляться новые методы защиты от копирования.

При использовании приводимого нами материала необходимо учитывать, что большинство программ, которые мы будем рассматривать, написаны в машинных кодах. Это еще раз говорит о том, что высокого уровня профессионализма можно добиться, только изучив программирование на языке Ассемблера. Несмотря на то, что большинство рассматриваемых процедур снабжены подробным комментарием, мы настоятельно рекомендуем Вам перед прочтением данного материала поближе познакомиться с программированием в машинных кодах. Одной из лучших книг, которые нам приходилось встречать по данной тематике, является разработка "ИНФОРКОМа" "Практикум по программированию в машинных кодах".

Мы рекомендуем вам для начала просто ознакомиться с текстом, получить представление об основных принципах и положениях статьи, а потом уже детально изучить проблему по разделам. Мы старались подготовить информацию таким образом, чтобы она не требовала специального запоминания, а усваивалась бы по ходу внимательного прочтения.

Необходимо учитывать, что данный материал тесно связан с той информацией, которая давалась в предыдущих томах. Впрочем, если вы и не читали предыдущих статей, то не отчаивайтесь, этот том отличается определенной автономией и не требует особой подготовки. Он может быть интересен как профессионалам, так и начинающим. Желаем удачи при его изучении!

1. Временные диаграммы и основные характеристики файловой структуры записи на магнитную ленту.

Как Вы уже знаете, роль внешней памяти в вашем компьютере выполняет магнитофонная кассета. Она обеспечивает приемлемую надежность системы при небольшой стоимости носителя. Дополнительным преимуществом является возможность взаимодействия с произвольным, в т.ч. бытовым магнитофоном. Основным недостатком является большое время передачи данных и последовательный доступ к файлам.

Рассмотрим примененную систему кодирования информации на магнитной ленте. Принятое здесь решение дает весьма гибкую систему с достаточной степенью надежности и защищенностью от разброса параметров бытовых магнитофонов.

Основа проста. К порту компьютера с адресом 254 подсоединяется посредством контроллера ULA микрофонное гнездо MIC и выходное EAR. Напряжение в микрофонном гнезде полностью зависит от третьего бита байта, вписываемого в этот порт. Если этот бит равен 0, то выходное напряжение составляет 0,75 Вольта, а если 1 - 1,3 В. Очередная смена установки этого бита на 0 или 1 в компьютерном порту 254 приводит к образованию прямоугольного сигнала, изображенного на диаграмме 1.

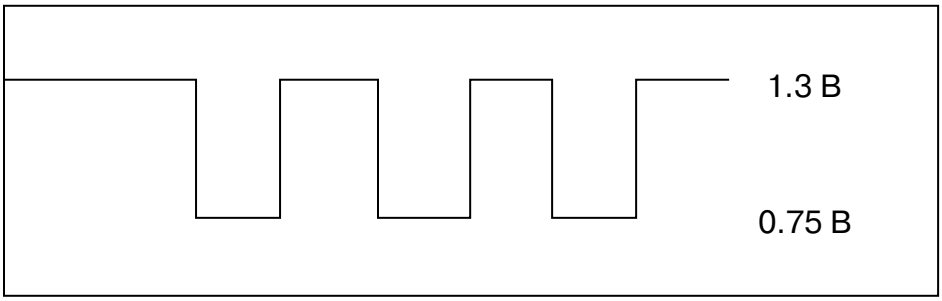


Диаграмма 1. Система кодирования информации на магнитофонной ленте в виде прямоугольных импульсов.

Показанный на диаграмме прямоугольный сигнал можно записать на ленту как звук с определенной частотой, зависящей от времени, в течение которого установка третьего бита есть константа. Рассмотрим это на примере программы, взятой из книги Анджея Кадлофа "Спектрум и магнитофон".

Данная программа попеременно включает и выключает магнитофонный бит компьютерного порта 254:

```
1 OUT 254,0
2 OUT 254,8
3 GOTO 1
```

Подключив усилитель низкой частоты к выходу MIC, мы услышим звук низкого тона. Это объясняется тем, что интерпретатор БЕЙСИКа работает очень медленно. На таком уровне скорости он не позволяет генерировать высокие тона. Поэтому собственные процедуры для записи и считывания с кассеты должны быть написаны только в машинном коде.

Кстати, попутно надо еще и пояснить причину, по которой бордюр телевизионного экрана во время выполнения предыдущей программы становился черным. Три младшие бита байта, выданного на порт 254, определяют действительный цвет бордюра. Это облегчает характерный только для "Спектрума" способ индицирования на экране операций с магнитофоном. Тот, кто пробовал работать с компьютерами других систем, не имеющих такого метода индикации, знает какое это благо.

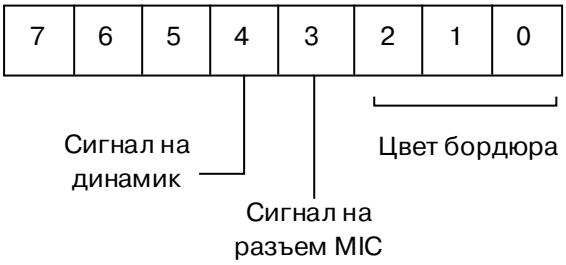
В машинном коде Z-80 существует обширная группа команд, позволяющая процессору получать данные от внешняя устройств и выдавать данные на эти устройства. Это делается по аналогии с загрузкой данных в регистры микропроцессора из ячеек ОЗУ и выдачей их в ОЗУ на хранение.

Как Вам уже, вероятно, известно, принцип внутренних и внешних устройств оценивается по отношению к микропроцессору Z-80. Именно поэтому такие части компьютерной системы, как клавиатура, магнитофон и звуковой динамик являются внешними.

Мнемоники команд АССЕМБЛЕРа, отвечающих за ввод/вывод байтов через внешний порт аналогичны операторам БЕЙСИКа IN и OUT. Однако, на языке АССЕМБЛЕРа передаваемые в порт данные рассматриваются как восьмибитные числа, и поскольку порт 254 служит для связи процессора не только с магнитофоном но и с клавиатурой, со звуковым динамиком и с бордюром экрана, у нас могут возникать побочные эффекты, если мы используем порт 254 из БЕЙСИКа.

Рассмотрим раскладку данного порта при записи и чтении информации. При вводе информации в порт 254 шестой бит указывает на наличие сигнала на магнитофонном разъеме (вход в компьютер), причем единице соответствует отсутствие сигнала, в то время как ноль показывает его наличие. Младшие пять битов определяют, какая из пяти клавиш каждого полуряда клавиатуры была нажата. Бит равен нулю, если клавиша была нажата и единице, если нет. Все это наглядно демонстрирует приведенная ниже диаграмма.

Изучим выходную раскладку сигналов компьютерного порта 254. Информация, которая находится в четвертом бите, передает сигнал на звуковой динамик "Спектрума". По третьему биту выдается сигнал на разъем MIC (запись информации на магнитофон). А по младшим трем битам, как мы уже упоминали, выдается сигнал на установку цвета бордюра. Цвет устанавливается в соответствии с номером одного из восьми стандартных цветов "Спектрума". Вся приведенная выше информация наглядно демонстрируется на диаграмме.



Информация, которую мы получаем с магнитофона, не является набором прямоугольных импульсов, а характеризуется сигналом близким к синусоидальному. Однако, такой сигнал тоже поддается компьютерному анализу. На этом принципе основана программа цветомузыки, когда в такт мелодии, игравшей на магнитофоне, экран начинал изменять свой цвет, приведенная в одном из номеров "ZX-РЕВЮ".

Для кодирования данных в "Спектруме" каждый записанный блок состоит из комбинаций четырех различных видов импульсов. Первым генерируется пилотирующий сигнал, при котором смена напряжения наступает регулярно через 619.4 микросекунды, что соответствует 2168 тактам частоты синхронизации микропроцессора Z-80. Генерируемый сигнал имеет частоту 807 Гц. Его продолжительность составляет порядка пяти секунд для заголовка ("хэдера") и около двух секунд для блоков данных. Конец пилотирующего сигнала характеризуется тремя фронтами, образующими так называемый импульс синхронизации (синхроимпульс). Интервалы между ними составляют соответственно 667 и 735 тактов частоты синхронизации микропроцессора. Далее без перерыва во времени пересылаются отдельные биты данных, причем единица представляется двумя фронтами, появляющимися с интервалом 1710 тактов (488.6 микросекунды), в то время как логический ноль представлен 855 тактами (244,3 микросекунды). Интервалы между передаваемыми байтами отсутствуют. Наглядно характеристики сигналов показаны на диаграмме 2.

Диаграмма 2

Очень часто защиту программ от копирования основывают на изменении каких-либо стандартных параметров загрузки или записи на магнитофон. Если Вам удалось изменить первичные параметры, то стандартный загрузчик уже не сможет правильно считать вашу программу с магнитофонной ленты. Это сможет сделать лишь специализированная

программа, написанная Вами специально для снятия такого типа защиты (т.е. загрузить эту программу сможет только тот, кто имеет переданный вами нестандартный загрузчик).

Вы можете изменять длительность импульсов пилотирующего сигнала, длительность синхроимпульса, а также длительность логического нуля и логической единицы. Все это приведет к описанному выше эффекту, то есть этим способом Вы сможете защитить свою программу от копирования.

Теоретически ознакомление с вышеописанными временными характеристиками сигналов уже достаточно Вам для написания программ записи и чтения с кассет в формате "Спектрума".

Однако, дело это достаточно сложное и требует очень хорошего знания языка АССЕМБЛЕРа. Еще для правильности интерпретации считываемых файлов необходимо подробно изучить, как производится контроль правильности блока данных. Ведь записывая каждый файл на ленту, "Спектрум" добавляет к нему два байта: один в начале и один в конце. Первый из них сигнализирует о том, является ли данный блок заголовком (значение байта в этом случае равно нулю) или же собственно блоком данных (в этом случае байт принимает значение 255).

Последний байт блока, так называемый байт четности, связан непосредственно с контролем правильности считывания. Его значение записывается в регистр и во время записи последовательного ряда байтов. Принцип контроля основан на применении команды "ИСКЛЮЧАЮЩЕЕ ИЛИ" (XOR). Здесь результат равен единице, если хотя бы один из операндов равен единице, но не оба вместе. В остальных случаях он равен нулю. Перед посылкой каждого восьмибитового байта из регистра L выполняются команды:

```
LD A,H
XOR L
LD H,A
```

В результате этого байт в регистре H содержит информацию о четности появления единицы на данной позиции во всех высланных байтах. Запись его в конце блока обеспечивает возможность контроля. В такте чтения проверяется имеет ли считываемый блок такое же свойство. Если результат последнего байта не совпадает с вычисленным в ходе загрузки значением, выдается сообщение об ошибке загрузки с ленты:

```
TAPE LOADING ERROR
```

Эта система на практике оказывается не только очень простой, но и очень эффективной.

Для того, чтобы научиться защищать собственные программы от копирования, вам необходимо не только представлять, каким образом информация кодируется на магнитной ленте, но и разобраться с основными принципами работы встроенных программ чтения и записи с магнитофона. Эти программы построены так, что они очень просты и непритязательны, если в одной упряжке используются "SAVE"и "LOAD", известна точная длина блока данных и вас не волнует возврат в БЕЙСИК вследствие ошибки или нажатия "BREAK".

Структурная схема заголовка.

Байты	0	1	2..11	12-13	14-15	16-17	-
	Флаг	тип	имя	длина	старт	бейсик-длина	паритет
IX+	-	0	1..10	11-12	13-14	15-16	17

Обычно, при загрузке "Спектрум" полагает что заголовок, говорящий компьютеру как работать, будет получен перед основным блоком и лишь затем последует сам блок. Однако, это характерно лишь для первых программ к данному типу компьютеров. Последние программы в большинстве случаев используют принцип записи блоков данных без заголовка. Это осуществляется при использовании машинных кодов и возможно только тогда, когда точно известны все параметры загружаемого блока. Однако, для того чтобы научиться работать на таком высокопрофессиональном уровне, Вам для начала необходимо

освоить основы, то есть изучить принципы чтения и записи с магнитофона "зашитые" в стандартном ПЗУ "Спектрума".

Для начала рассмотрим структуру заголовка. Его длина составляет 19 байтов, а не семнадцать, как написано в большинстве книг. Тем не менее, только семнадцать байтов должны быть активны, так как процедуры записи и загрузки первый и последний байты определяют сами.

Чтобы согласовать разночтения в литературе относительно длины заголовков давайте считать, что первый и последний байты являются служебными и к пользователю не имеют никакого отношения, а остальные семнадцать - информационными.

Тогда:

Байт 0 - флаговый байт, для заголовков всегда равен нулю, а для блоков данных равен 255.

Байт 1 - содержит число, характеризующее тип записи:

0 -- это БЕЙСИК-программа,

1 -- числовой массив,

2 -- массив символов,

3 -- блок кодов.

Байты 2-11 - содержат имя программы (блока данных).

Байты 12-13 - длина блока. Длина программы кодируется двухбайтным шестнадцатиричным числом. (Для БЕЙСИК-программы это, соответственно, разность между содержимым системных переменных "ELINE - PROG").

Байты 14-15 - хранит в себе начальный адрес загрузки (если это блок машинных кодов) или номер строки автостарта для БЕЙСИК-программ. Если же блок является массивом данных, то для него байт 15 кодируется специальным образом:

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 16-17 - это длина для БЕЙСИКа, то есть разность между содержимым системных переменные "VARS-PROG".

Последний байт - байт четности ("PARITY BYTE"). Он выдается при записи блока на ленту автоматически и при загрузке считывается и проверяется.

Поскольку при загрузке/выгрузке контроль за проходящими байтами ведут через регистр IX микропроцессора (в стандартных процедурах), то для систематизации информации принято отсчитывать эти байты смещением от базы, содержащейся в IX, как показано на структурной схеме заголовка.

Каждый пользователь "Спектрума" обратил внимание на то, что блок информации (файл) на магнитной ленте начинается с синхронизирующего сигнала длительностью две или пять секунд. Частота этого сигнала составляет около восьмисот Герц (период 1.25 миллисекунды). После этого сигнала идет один период специального синхросигнала для которого длительность нуля составляет около 0.19 миллисекунды, а единицы - 0.21 миллисекунды. Затем следуют байты данных передаваемые последовательно, начиная со старшего бита.

Итак, мы с Вами подробно рассмотрели структуру заголовка файла. Структура блока данных фактически ничем не отличается по частоте пилотирующего и длительности импульса синхронизирующего сигнала, однако байт типа принимает значение нуля для заголовков и 255 для блоков данных. Вы можете проверить это, если попытаетесъ загрузить два заголовка подряд, то есть загрузить первый заголовок, и перемотав ленту назад, попробовать загрузить его еще раз. У Вас ничего не получится, так как компьютер очень строго следит за типом вводимой информации.

На этом мы пока прервемся, а далее рассмотрим процедуры, которые отвечают за загрузку и выгрузку программ.

(Продолжение следует).

ОШИБКИ ПЗУ

Обзор по материалам зарубежной печати

Сегодня, уважаемые читатели, мы продолжаем разговор о вскрытых ошибках и неточностях стандартного ПЗУ "Спектрума". Начало статьи см. в предыдущем выпуске на стр. 209 - 210.

8. Ошибка CLOSE#.

Профессионалы считают эту ошибку наиболее серьезной из всех. С точки зрения рядового пользователя она, может быть, таковой и не является, поскольку ему редко приходится иметь дело с нестандартными каналами и потоками.

Ошибка проявляется в тех случаях, когда внешняя периферия, имеющая собственное ПЗУ для обслуживания каналов и потоков не подключена. В этой случае, если вы дадите команду на закрывание потока CLOSE #n, а сам поток #n никогда перед этим и не открывался, то Ваш "Спектрум" вместо того, чтобы предупредить Вас о том, что синтаксис неверен, зависает, а иногда (реже) сбрасывается.

Ошибка вызвана тем, что таблица данных, находящаяся в ПЗУ по адресу 1716H (5910 DEC) не заканчивается, как ей положено, нулевым байтом 00.

Интересно отметить, что и фирма "AMSTRAD", перекупив у К.Синклера права на производство "Спектрум"-совместимых машин, не исправила эту ошибку в ПЗУ для "Spectrum+2", хотя другие изменения в ПЗУ сделала. Казалось бы, уж если все равно меняешь ПЗУ (чего не делал сам К. Синклер дабы не снизить совместимость программного обеспечения и не огорчать простых пользователей), то можно было бы и исправить этот дефект.

9. Ошибка CHR$ 9.

Управляющий код CHR$ 9 должен был действовать противоположно коду CHR$ 8. Код CHR$ 8 называется BACKSPACE и вызывет перемещение курсора (текущей позиции печати) влево. Код же CHR$ 9 должен был бы по аналогии называться FORWARDSPACE и вызывать перемещение курсора или позиции печати на одно знакоместо вправо без изменения содержимого текущего знакоместа.

На практике он не делает ни того, ни другого. Более того, в результате его применения, содержимое текущей позиции окрашивается в текущие цвета INK и PAPER, т.е. в нем заложена двойная ошибка.

Листинг 1

```
2A655C      PR_FP_OK    LD HL,(STKEND)      ; HL хранит адрес вершины стека
                                            ; калькулятора.
225F5C                  LD (X_PTR),HL       : Запомнили вершину стека в си-
                                            ; стемной переменной X PTR.
CDE32D                  CALL 2DE3           ; Вызвали процедуру ПЗУ
                                            ; PRINT FP. при этом пятибайт-
                                            ; ное число снялось с вершины
                                            ; стека калькулятора.
2A5F5C                  LD HL,(X_PTR)       ; восстановили старый адрес вер-
                                            ; шины стека к-ра, т.е. теперь
                                            ; HL указывает на новый адрес
                                            ; вершины стека + 5.
11FBFF                  LD DE,FFFB          ; Число FFFBH равно -5 DEC (по
                                            ; правилам двоичной дополнитель-
                                            ; ной арифметики.
19                      ADD HL,DE           ; Теперь HL указывает на новую
                                            ; вершину стека калькулятора.
22655C                  LD (STKEND),HL      ; Запомнили ее в соответствующей
                                            ; системной переменной.
```

```
FD362600            LD (X_PTR),00      ; Погасили старший байт указателя X PTR.
C9                  RET                ; Возврат.
```

Первый недосмотр состоит в том, что процедура ПЗУ, выполняющая перемещение курсора вправо (0A3DH = 2621 DEC) должна бы заканчиватъся не командой возврата RET, а командой безусловного перехода JP 0ADC. Вторая же ошибка, связанная с цветом состоит в том, что когда эта процедура работает, надо запоминать состояние системной переменной MASK_T (5C8FH =23695 DEC), затем выставлять в ней число 0FFH, а по окончании работы процедуры восстанавливать запомненное значение.

10. Ошибка CHR$ 8.

Есть ошибки и в процедурах, выполняющих перемещение курсора влево. В большинстве случаев с кодом CHR$ 8 все в порядке, но, к сожалению, не всегда.

Так, если у вас текущей позицией печати является знакоместо с координатами AT 1,0; то BACKSPACE не работает.

Более того, есть возможность смещения влево из координаты 0,0; а это уже совершенная чепуха с неожиданными результатами.

Ошибка находится в процедуре, обслуживающей перемещение курсора влево (0A23H = 2595 DEC). Она проверяет номер экранной строки, на которой установлен курсор, но вместо того, чтобы "отловить" нулевую строку и заблокировать в ней перемещение курсора, делает это для первой строки. Очевидно, в команде программистов у К. Синклера была некоторая несогласованность. Конкретная причина - в том, что по адресу 0A33H = 2611 DEC должно быть число 019H вместо 018H.

11. Ошибка STR$.

Эта ошибка проявляет себя как в БЕЙСИКе, так и при программировании в машинном коде. Вызвать ее очень просто:

```
PRINT "KU-KU" + STR$ 0.5
```

По такой команде компьютер напечатает только 0.5.

Программисты, работающие в машинном коде, могут столкнуться с этой ошибкой при вызове часто встречавшейся процедуры PRINT_FP. Эта процедура находится по адресу 2DE3H = 11747 DEC и служит для того, чтобы выдать на печать по текущему подключенному каналу то действительное число, которое в данный момент находится на вершине стека калькулятора.

Процедура, обрабатывающая оператор STR$ в своей работе тоже обращается к процедуре FRINT_FP и таким образом эта ошибка проникает и в БЕЙСИК.

Эта ошибка происходит в тех случаях, когда число на вершине стека калькулятора находится в интервале от -1 до +1, исключая границы и число 0. Дело в том, что на вершине стека оставляется ошибочный ноль, который и вызывает все проблемы.

Для тех, кто работает на БЕЙСИКе, эту проблему обойти несложно. Достаточно ввести временную переменную, например так:

```
LET a$=STR$ 0.5
PRINT "KU-KU" + a$
```

Для тех, кто работает в машинном коде, в этом случае лучше не пользоваться процедурой PRINT_FP, а заменить ее какой-либо своей, например приведенной в Листинге 1.

12. Ошибки кодов управления цветом.

Если в качестве текущего канала для выдачи информации Вами выбран какой-либо нестандартный канал (иначе говоря, если вы работаете с пользовательским каналом), то операторы управления цветом, например такие, как PAPER 4 дадут сообщение об ошибке

```
C; Nonsense in BASIC.
```

Если подпрограмма, обслуживающая вывод информации в канал, возвращает после своей работы выключенный флаг C (флаг CARRY флагового регистра F).

Чтобы избавиться от ошибки, необходимо предусмотреть, чтобы все процедуры, обслуживающие вывод информации в каналы, возвращались с выключенный флагом CARRY по крайней мере для управляющих кодов от 10 до 15-го, а также для тех параметров,

которые следуют за кодами управления цветом.

С этой ошибкой связана и еще одна, касающаяся кодов управления цветом, но здесь она относится только к операторам временного изменения цвета, т.е. к тем случаям, когда оператор управления цветом является квалификатором оператора PRINT.

Итак, если в качестве текущего установлен канал, отличный от "S" или "K", т.е. последний оператор PRINT печатал не на экран, то команды установки временных цветовых атрибутов в операторе PRINT по ошибке выдадут цветовой код не в текущий, а в предыдущий канал.

Ошибка связана с неспособностью обслуживающей процедуры избрать канал "S" для этой цели по адресу 21E1H = 8673 DEC.

Обойти ошибку можно, если перед каждой командой временной установки цвета вставить пустой оператор PRINT:

```
PRINT;:INK 4
```

И еще одна смежная ошибка, которая относятся только к 128-килобайтным машинам, а точнее говоря - к тому порту RS232, который в них встроен.

На этих моделях команда:

```
LPRINT INK 4
```

вызовет сообщение об ошибке:

```
C; Nonsense in BASIC
```

Этому есть две причины. Во-первых, программа, которая обслуживает вывод на новый канал "p" (встроенный порт RS232) почему-то ошибочно полагает, что за кодом управления цветом должны идти два параметра, а не один. А во-вторых, соответствующая процедура ПЗУ включает флаг CARRY, а к чему это приводит Вы уже знаете.

В идеале на этих моделях по адресу 0086D (для Sp+128) и 0088C (для Sp+2) должен стоять байт 02 вместо 01, и, кроме того, по адресу 0087C (для SP+128) и 0089B (для Sp+2) вместо инструкции

```
CCF
```

должна стоять пара:

```
SCF
CCF
```

13. Ошибка SCREEN$.

Эта ошибка похожа на ошибку STR$. При расчете функции SCREEN$ из БЕЙСИКа на вершине стека калькулятора полученный результат ошибочно дублируется.

В результате такое выражение, как:

```
IF "x" = SCREEN$ (0,0) THEN PRINT "KU-KU"
```

всегда будет печатать "KU-KU", независимо от того, что имеется на экране в позиции с координатами (0,0). К счастью, эта ошибка характерна только для БЕЙСИКа. Соответствующая процедура ПЗУ при вызове ее из машинного кода работает нормально (о том, как ее использовать, мы писали в книге "Элементарная графика", т.1).

В БЕЙСИКе путь обхода этой ошибки прост и выполняется переприсвоением переменной:

```
LET a$ = SCREEN$ (0,0):
IF "x" = a$ THEN PRINT "KU-KU"
```

Ошибки в редакторе

Есть несколько оплошностей, связанных со встроенным редактором "Спектрума".

14. Ошибка Scroll?.

Когда в нижней части экрана при листинге программы появляется сообщение Scroll? или когда там появляется какое-либо сообщение, связанное с обслуживанием магнитофона, то компьютер ждет от вас нажатия какой-либо клавиши.

Проблема состоит в том, что есть несколько комбинации клавиш, которые нажимать при этом нельзя.

Это TRUE VIDEO, INV VIDEO, CAPS LOCK, GRAPHIC и EXTEND MODE.

Если Вы их нажмете, то вместо продолжения работы получите в нижней части экрана последнюю редактированную строку с включенным курсором. Самое интересное, что на 128-килобайтных машинах, работающих в режиме 128K этот курсор будет соответствовать режиму 48K.

Ошибка находится в процедуре, обслуживающей ввод с клавиатуры KEY_INPUT, которая находится в ПЗУ по адресу 10A8H = 4264 DEC. Эта процедура обрабатывает такие комбинации, как CAPS LOCK и пр., но этого не надо делать в данном случае, когда компьютер просто ждет нажатия какой-либо клавиши в ответ на свое сообщение.

15. Ошибка курсора текущей строки.

Эта ошибка проявляется только на 48-килобайтных моделях, поскольку редактор 128-килобайтных машин сильно отличается.

Наберите:

```
9000 PRINT\9001\EDIT
```

(здесь символ "\" обозначает нажатие клавиши "ENTER").

Если при этом у Вас в программе нет строк с номером, большим, чем 9000, то Вы увидите эту ошибку в нижней части экрана, и строке редактирования появится курсор текущей строки (символ ">").

Этой ошибки не было бы, если бы программа OUT_LINE, расположенная в ПЗУ по адресу 1855H = 6229 DEC, проверяла бы четвертый бит системной переменной FLAGS2 и, когда он включен, не печатала бы курсор ">".

16 Ошибка ведущего пробела.

Вы знаете, что операторы и функции БЕЙСИКа в "Спектруме" набираются не по буквам, а словами (токенами). Для того, чтобы отдельные ключевые слова на экране не сливались друг с другом, встроенный редактор автоматически вставляет между ними пробелы, но делает это не очень стабильно, попробуйте, например:

```
CLS: FOR i=1 TO 5: PRINT CHR$ 244: NEXT i
```

В принципе, для того, чтобы компьютер мог решить, когда надо давать ведущий пробел, а когда нет, существует системная переменная FLAGS, когда ведущий пробел не нужен, нулевой бит этой системной переменной должен быть включен.

Проблема была бы решена, если бы этот флаговый бит автоматически включался всякий раз после исполнения команды CLS или при печати управляющих кодов от 00 до 1FH.

17. Ошибка K-режима.

Если при работе с компьютером у Вас включен курсор "K", это означает, что машина находится в командном режиме и следующее нажатие клавиши должно будет восприниматься, как ввод ключевого слова оператора или функции. Например, нажатие на клавишу "p" в этом режиме даст ввод ключевого слова PRINT.

А что будет, если Вы задержите палец на клавише дольше, чем это необходимо? В этой случае, как положено начнется автоповтор, но K-режим останется принудительно включенным.

Одним словом, если Вам надо сделать например:

```
PRINT P
```

или

```
NEXT N
```

то Вы получите вместо этого

```
PRINT PRINT
```

или

```
NEXT NEXT
```

Ошибка находится в процедуре K_REPEAT, расположенной в ПЗУ по адресу 0310H = 784 DEC.

Чтобы ошибки не было, процедура должна вычитать число A5H из кода нажатой клавиши в тех случаях, когда этот код больше, чем E5H. Тогда повторное ключевое слово будет заменено на прописную литеру.

18 Ошибка проверки синтаксиса.

Эта ошибка проявляется только на машинах 48K, а на машинах 128K она мудро игнорируется.

Дело в том, что у Вас есть возможность ввести в программную строку такие ключевые слова, как ERASE, MOVE, FORMAT, CAT, например

ERASE симв. строка
MOVE строка, строка
FORMAT строка
CAT

Очевидно, что эти команды не могут быть выполнены, если у Вас не подключена соответствующая периферия, например INTERFACE ONE с микродрайвом.

И, конечно, при запуске программы на исполнение, она будет прервана с сообщением об ошибке. Спрашивается, почему же нельзя было отловить эту ошибку при проверке синтаксиса перед вводом строки в память?

На 128-килобайтных машинах тоже можно ввести такие ключевые слова, но при запуске программы они будут игнорироваться и восприниматься так, как воспринимается оператор REM.

Ошибки калькулятора

Теперь рассмотрим несколько ошибок, связанных со встроенным в ПЗУ калькулятором. О некоторых из них мы так или иначе уже упоминали в своих прочих работах.

19. Ошибка MOD_DIV.

Эта ошибка связана с работой кода калькулятора 32h. По этой команде со стека калькулятора должны сниматься два верхних пятибайтных числа, например x и y и вместо них на стек должны отправляться

```
x MOD y и
x DIV y
```

(именно в этом порядке).

Напомним, что x MOD y - это остаток от целочисленного деления x на y, а x DIV y - это целая часть частного от деления x на y.

Таким образом,

```
x MOD y = x - y*INT(x/y)
x DIV y = INT (x/y)
```

В своих расчетах процедура, обслуживающая эту функцию калькулятора, использует нулевую ячейку памяти калькулятора M0, а с этой ячейкой есть одна особенность. Дело в том, что при вычислении функции INT эта ячейка коррумпируется, если аргумент при INT меньше нуля. Таким образом, функция MOD_DIV калькулятора дает неверный результат, когда x/y число отрицательное.

Ошибки могло бы и не быть, если бы процедура, занимающаяся расчетом этой функции (а она расположена в ПЗУ по адресу 36A0H = 13964 DEC) использовала бы в своих расчетах не нулевую ячейку памяти калькулятора, а первую (M1).

20. Ошибка E_TO_FP.

В системе команд калькулятора есть команда с кодом 3C. Ее назначение - умножение числа, находящегося на вершине стека калькулятора на множитель, равный 10 в степени A, где A - содержимое аккумулятора микропроцессора.

Вся неприятность в том, что калькулятор после своего включения командой RST 28 не резервирует содержимое аккумулятора, в отличие от содержимого регистра B. Поэтому, к тому времени, как вы воспользуетесь командой калькулятора 3C, есть большая вероятность того, что в аккумуляторе будет не подготовленное вами число, а что-то совсем другое.

Единственный выход - выйти из калькулятора, прогрузить аккумулятор нужным Вам числом, выполнить нужное умножение вызовом процедуры ПЗУ E_TO_FP и снова вернуться в калькулятор:

```
        Endcalc
```

```
LD A, xx
CALL 2D4FH
RST 28
```

Процедура E_TO_FP находится в ПЗУ по адресу 2D4FH = 11599 DEC.

21. Ошибка INKEY$#0.

Обычно нулевой поток представляет собой клавиатуру, поэтому естественно предположить, что INKEY$#0 - то же самое, что и просто INKEY$ без номера потока.

Тем не менее это не так, и почти необратимо INKEY$#0 выдает пустую символьную строку, что делает эту функцию полностью бесполезной.

Надо также заметить, что в системе команд калькулятора есть команда с кодом 1A, которая служит для расчета функции INKEY$#X, где X - число, содержащееся на вершине стека калькулятора. И эта команда калькулятора будет бесполезной, если поток X представляет клавиатуру.

Ошибка находится в подпрограмме ПЗУ по адресу 1634H=5684 DEC, которая устанавливает канал "X" - текущим каналом. В этой подпрограмме по адресу 1638K стоит ошибочная команда RES 5,(FLAGS), выключающая пятый бит системной переменной FLAGS. в результате этого ошибочно отбивается любое нажатие клавиши вместо того, чтобы быть принятый к рассмотрению.

Ошибку можно было бы исправить, если в подпрограмме READ_IN (3645H = 13893 DEC) сохранить значение системной переменной FLAGS на время вызова подпрограммы CHAN_OPEN (1601H = 5633 DEC).

На этом мы заканчиваем обзор ошибок и неточностей в ПЗУ стандартного компьютера "ZX-Spectrum". Конечно же это не все из того, что оттуда можно выудить, но очень экзотические ошибки, которые проявляются например только на машинах типа "ZX-Spectrum+2" и только при подключенном Интерфейсе-1 мы не рассматриваем, поскольку вероятность встретить среди миллионов наших пользователей подобную конфигурацию конечно есть, но она не более сотой доли процента.

Обзор подготовлен по материалам зарубежной печати; основные первоисточники:

1. Dr. Yan Logan, Dr. Frank O'Hara. "The Complete Spectrum ROM Disassembly".
2. Dr. FranK O'Hara "Understanding Your Spectrum".
3. Dr. Yan Logan "Understanding Your Spectrum".
4. Andrew Pennell "Master Your ZX Microdrive".
5. Tony Stratton "Understanding Your Spectrum".
6. Paul Harrison "Understanding Your Spectrum".
7. Stephen Kelly & others "Understanding Your Spectrum".
8. Chris Thornton "Understanding Your Spectrum".

Алексеев А.Г.

ПРОФЕССИОНАЛЬНЫЙ ПОДХОД

Сегодня - продолжение разговора о некоторых приемах, позволяющих придать "профессионализм" вашим программам. Мы поговорим о небольших кодовых блоках, позволяющих воспроизводить различные звуковые эффекты в ваших программах, такие, какие невозможно создать, непосредственно используя оператор "BEEP". Это, например, стрельба из лазерного пистолета, пуск ракеты, торпеды, получение приза и т.д. Также разговор будет о программе "SOUND", позволяющей создавать и редактировать такие звуки. Попутно, в качестве лирического отступления, остановимся еще на одной теме: о том, как усовершенствовать введение числовых параметров при помощи оператора "INPUT". Кроме того, рассматривая готовую программу, мы еще раз проследим все моменты, изложенные в предыдущие статьях: применение блока "ON ERROR GO TO", структура программы, управление программой при помощи меню и т.д.

За основу разработки взята программа "SPECSOUND" фирмы OZ SOFTWARE. Возможно, читателям известен еще один, более ранний прообраз этой программы "DZWIEKI". Взяв за основу принцип формирования кодовых блоков, воспроизводящих звуки, вся программа была изменена настолько, что стала возможна эффективная работа по созданию и редактированию звуков. Кроме того, устранены многие программные ошибки, и сделан перевод на русский язык. В общем, это стала практически другая программа, однако идея осталась той же. Структура самих кодовых блоков, получаемых в результате работы программы, изменена незначительно, лишь настолько, насколько это не повредило совместимости с прототипом программы.

В основе кодового блока одного звука лежит использование подпрограммы из ПЗУ "BEEPER", расположенной по адресу #03B5. При входе в эту подпрограмму, в регистре DE должно быть задано произведение частоты на время звучания звука, (то есть число периодов колебаний), а в регистре HL - величина, эквивалентная значению периода колебаний. Подробнее о работе подпрограммы "BEEPER" было рассказано на страницах ZX-РЕВЮ в разделе "СЕКРЕТЫ ПЗУ" см. N2 за 1991г. СТР. 28.

В общих словах, принцип формирования звука таков. Организуется цикл из последовательного обращения к подпрограмме "BEEPER", но при каждом следующем обращении период колебаний изменяется на определенную величину по сравнению с начальным тоном (может увеличиваться или уменьшаться). При этом, если величина смещения или ступеньки тона, незначительная, а число обращения к подпрограмме "BEEPER" (число ступенек тона) в цикле - большое, то на слух получится плавно изменяющийся звук - "скольжение" тона. Кроме того, получившееся "скольжение" входит в еще один цикл повторений, то есть большое число таких "скольжений" формируют результирующий звук. Изменяя исходные параметры в широких пределах и комбинируя их в различных взаимных сочетаниях, можно получать интересные звуковые эффекты. Имеется также возможность объединения двух или нескольких результирующих звуков в один (последовательное выполнение двух или более звуков), что открывает дополнительные возможности по созданию звуковых эффектов.

Для тех, кто интересуется машинными кодами, рассмотрим подробно блок кодов, воспроизводящий один звук. Если коды Вас не интересуют, то пропустите этот раздел и переходите непосредственно к Бейсик-программе "SOUND".

Блок кодов, воспроизводящий звук.

Длина блока - 32 байта. Его можно загружать в любое место памяти. Стартовый адрес блока равен адресу загрузки. Для примера, расположим его с адреса #AB00, звездочкой справа отмечены параметры, которые могут меняться.

```
AB00 0E00       LD C,#00          (1)
AB02 0605       LD B,#05          (2)  *
AB04 21F401     LD HL,#01F4       (3)  *
```

```
AB07 C5          PUSH BC           (4)
AB08 114600      LD DE,#0046       (5)  *
AB0B E5          PUSH HL           (6)
AB0C CDB503      CALL #03B5        (7)
AB0F E1          POP HL            (8)
AB10 112800      LD DE,#0028       (9)  *
AB13 ED52        SBC HL,DE         (10) *
AB15 C1          POP BC            (11)
AB16 10EF        DJNZ #AB07        (12)
AB18 3E02        LD A,#05          (13) *
AB1A 0C          INC C             (14)
AB1B B9          CP C              (15)
AB1C 20E4        JR NZ,#AB02       (16)
AB1D 00          NOP               (17)
AB1F C9          RET               (18)
```

Сначала (1) обнуляется счетчик числа повторений "скольжения" - (внешнего цикла) - регистр C. Затем (2) в регистр B заносится число ступенек тона (число повторений внутреннего цикла), а в регистр HL - период начального тона (3). Далее начинает выполняться внутренний цикл. На стеке запоминается (4) содержимое регистров B и C (так как оно будет изменено после отработки подпрограммы "BEEPER"). Потом (5) в регистр DE заносится число периодов колебаний, которые будут выработаны подпрограммой "BEEPER". Чем меньше это число, то есть чем меньшее число колебаний будет выполнено перед очередным изменением периода колебаний, тем более плавным на слух будет "скольжение" тона. Другими словами эту величину можно охарактеризовать как дискретность получающегося звука. Затем на стеке запоминается (6) содержимое регистра HL (по той же причине, что и BC), выполняется подпрограмма "BEEPER" (7) и возвращается со стека (8) содержимое HL. теперь надо изменить величину тона перед следующим вызовом подпрограммы "BEEPER". Для этого величина смещения (или ступеньки тона) заносится (9) в регистр DE, после чего (10) отнимается от содержимого регистра HL. (Если вместо SBC задан код команды ADC, тогда DE складывается с HL. Таким образом можно задать повышение или понижение тона "скольжения".) Теперь в HL - новое значение периода колебаний. Далее возвращается со стека (11) значение BC и проверяется величина B - счетчика внутреннего цикла. Если заданное число повторений внутреннего цикла не выполнено, то происходит (12) очередное повторение внутреннего цикла (с адреса #AB07) с измененным значением периода колебаний. А если счетчик внутреннего цикла обнулен, то продолжается выполнение программы. В регистр A заносится (13) число повторений "скольжений" (внешнего цикла). Так как один внешний цикл уже завершен, то происходит увеличение на единицу (14) счетчика внешних циклов - регистра C. Потом это значение сравнивается (15) с тем, что задано в A. Если число в A больше, чем в C (то есть разница не равна нулю), то происходит (16) возврат на повторение внешнего цикла с адреса #AB02 (выполнение следующего "скольжения"). Если заданное число внешних циклов выполнено, то (17), (18) завершение работы - возврат в вызывающую программу. Необходимость команды NOP в конце программы объясняется желанием сохранить совместимость с прообразом программы "SOUND".

Это то, что касается кодового блока. Теперь рассмотрим Бейсик-программу для редактирования кодовых блоков звуков.

Программа "SOUND"

Сначала несколько слов о том, что же может эта программа. Во-первых, она может продемонстрировать 20 готовых звуков, созданных для примера с ее помощью. Прослушав их, Вы сможете примерно оценить "диапазон" возможностей программы. Вы также можете получить для записи на магнитофон любой (или любые несколько, по выбору) или все 20 примеров звуков для дальнейшего использования в своих программах.

Во-вторых, программа позволяет создавать новые звуки и эффективно заниматься редактированием любого из созданных звуков. При создании нового звука можно, взяв за основу один из имеющихся 20 примеров, подробно "рассмотреть" параметры звука и по

мере необходимости изменять их, добиваясь требуемого звучания. Можно также объединить два и более готовых звука в один, чтобы получать комплексные звуковые эффекты. Готовые звуки можно записать на магнитную ленту. Можно также загружать с магнитной ленты для редактирования блоки кодов - звуки, полученные при помощи этой программы.

Область памяти для создания и редактирования звуков определена с адреса 43776 (#AB00) (может быть изменена по Вашему желанию). Максимальное число создаваемых звуков ограничено областью до конца памяти и составляет 680 звуков. Наверное, это заведомо превышает возможности любого пользователя, поэтому в программе не предусмотрена проверка на достижение конца памяти. При необходимости, Вы можете сами организовать такую проверку, если захотите.

При работе, программа отличается "деликатностью", а именно, она предлагает возможные варианты ответа, предупреждает, если результат неудовлетворительный и рекомендует, что надо сделать, чтобы добиться положительного результата.

Рассматривая текст программы, вы увидите, что структура программы - такая же, как была описана в ZX-РЕВЮ N 9-10 за этот год, когда речь шла о дебюте программы "PROG". Поэтому, если вам непонятны некоторые моменты, то еще раз прочтите о дебюте "PROG".

Итак, текст программы "SOUND", затем комментарии.

```
0 REM КОДЫ
1 GO TO 100
3 RANDOMIZE 3: GO SUB 10: GO SUB 7: GO SUB 8: BORDER 1: PAPER 0: INK 6: CLEAR 43775: PRINT
      BRIGHT 1;AT 2,10: "ПРОГРАММА ДЛЯ"; AT 3,6; "РЕДАКТИРОВАНИЯ ЗВУКОВ"; PAPER 2; INK 7;
      DIVERSE 1;AT 8,8; "                ";AT 9,8: "    S O U N D    "; AT 10,8; "
      "; PAPER 0; INK 2; INVERSE 0;AT 15,3; "OZ SOFTWARE + АЛЕКСЕЕВ А. Г. "
4 PRINT INK 4; BRIGHT 1;AT 18,12; "01.08.92": BEEP.1,26: BEEP .1,20: GO SUB 20: GO TO 100
5 SAVE "SOUND" LINE 2
6 VERIFY "SOUND": GO TO 5
7 RANDOMIZE PEEK 23635+256*PEEK 23536+773: POKE 23675,PEEK 23670: POKE 23676, PEEK 23671:
      RETURN : REM UDG
8 RANDOMIZE PEEK 23635+256*PEEK 23536-251: POKE 23606,PEEK 23570: POKE 23607,PEEK 23671:
      RETURN: REM RUS
9 POKE 23606,0: POKE 23607,60: RETURN : REM LAT
10 POKE PEEK 23635+256*PEEK 23636+993, PEEK 23670: POKE PEEK 23535+256*PEEK 23636+994, PEEK
      23671: RANDOMIZE USR (PEEK 23635+256*PEEK 23636+941): RETURN
20 INPUT ; : PRINT #0; PAPER 6; INK 2; BRIGHT 1; AT 1,5:" НАЖМИТЕ ЛЮБУЮ КЛАВИШУ "
22 RANDOMIZE 24: GO SUB 10: PAUSE 0
24 IF INKEY$="q" THEN GO TO 9999
26 RETURN
30 BORDER 1: PAPER 0: INK 6: CLS
31 FOR M=1 TO NN: READ M$: PRINT AT (Y0+(M-1)*DY), X0;N$: NEXT M: PRINT INK 4; AT Y1,3;
      "SPACE,DOWN,UP,0,ENTER,BREAK"
32 IF MM<1 THEN LET MM=NN
33 IF MM>NN THEN LET MM=1
34 PRINT AT (Y0+(MM-1)*DY),X0-DX; PAPER 7; INK 2; BRIGHT 1; OVER 2;S$: BEEP .03,2*MM+10
35 PAUSE 0: LET I=(INKEY$=" " OR INKEY$="6" OR CODE INKEY$=10)+(INKEY$="7" OR CODE
      INKEY$=11)*2+(INKEY$="0" OR CODE INKEY$=12 OR CODE INKEY$=13)*3+(INKEY$="q")*4: GO TO
      (35+I)
37 PRINT AT (Y0*(MM-1)*DY),X0-DX; OVER 1;S$: LET MM=MM-2*I+3:GO TO 32
38 BEEP .1,36: RETURN
39 GO TO 9999
40 RANDOMIZE 49: GO SUB 10: BORDER 1: PAPER 1: INK 7: CLS
42 PRINT AT 1,0;"   ЭТА ПРОГРАММА ПРЕДНАЗНАЧЕНА
      ДЛЯ СОЗДАНИЯ БЛОКОВ МАШИННЫХ КО-
      ДОВ, КОТОРЫЕ ВОСПРОИЗВЕДУТ  РАЗ-
      ЛИЧНЫЕ ЗВУКОВЫЕ ЭФФЕКТЫ  В ВАШИХ
      ПРОГРАММАХ  ( НАПРИМЕР, СТРЕЛЬБА
      ИЗ ЛАЗЕРНОГО ПИСТОЛЕТА, ПУСК РА-
      КЕТЫ, ПОЛУЧЕН ИЕ ПРИЗА И ДР. ) "
44 PRINT '"   БЛОКИ КОДОВ МОГУ Т БЫТЬ РАЗМЕ-
      ЩЕНЫ ЗА RAMTOP ИЛИ ИХ МОЖНО РАС-
```

```
      ПОЛОЖИТЬ В ПЕРВОЙ  (ИЛИ НУЛЕВОЙ)
      СТРОКЕ BASIC ЗА ОПЕРАТОРОМ REM. "
46 PRINT '"    БЛОК КОДОВ, ВОСПРОИЗВОДЯЩИЙ
      ОДИН ЗВУК, ИМЕЕТ ДЛИНУ 32 БАЙТА.
      ЕГО МОЖНО ПЕРЕМЕЩАТЬ  В  ПАМЯТИ.
      СТАРТОВЫЙ АДРЕС ДЛЯ  ВОСПРОИЗВЕ-
      ДЕНИЯ ЗВУКА БУДЕТ  РАВЕН  АДРЕСУ
      ЗАГРУЗКИ. "
48 BEEP .1,20: GO SUB 20: RETURN
50 RANDOMIZE 59: GO SUB 10: PRINT PAPER 6; INK 2; AT 2,6:" ЧИСЛО ЗВУКОВ: "; FLASH 1;B: PAUSE
      50
52 FOR J=1 TO B: PRINT PAPER 6; INK 2: AT 4,9;" ЗВУК НОМЕР ";J;" ": RANDOMIZE USR (U+32*(J-
      1)): PAUSE 60: NEXT J
59 RETURN
60 PRINT PAPER 2: INK 7; BRIGHT 1; FLASH 1;AT 2,6;" ВЫ УВЕРЕНЫ (Y/N)? ": PAUSE 0: RETURN
70 LET SEC=INT (H*C*(2*E+Z«(G-1)*F)*G/8E5): IF SEC<=10 THEN RETURH
72 LET T=INT(SEC/60): LET S=SEC-60*T
74 CLS : PRINT AT 5,0: "   ОБРАЩАЕМ ВАШЕ ВНИМАНИЕ НА ТО,
      ЧТО ВРЕМЯ ЗВУЧАНИЯ ЗВУКА БУДЕТ
      СОСТАВЛЯТЬ ПРИМЕРНО:"; AT 9,9; T: " МИН. ";S;" СЕК. "; AT 11,0;
     "   ВЫ ГОТОВЫ К ПРОСЛУШИВАНИЮ ?"''
     "   ЕСЛИ ДА, ТО НАЖМИТЕ [Y]; ЕСЛИ
      ВОЗВРАТ В МЕНЮ ТО ЛЮБУЮ КЛАВИШУ."''
     "   ПОСЛЕ ВОЗВРАТА  В  МЕНЮ НАДО
      УМЕНЬШИТЬ   ДИСКРЕТНОСТЬ  ЗВУКА.
      ЧИСЛО СТУПЕНЕК ТОНА  ИЛИ  ПЕРИОД
      НАЧАЛЬНОГО ТОНА."
76 BEEP .5,0: PAUSE 0: RETURN
100 LET P=0
110 LET B=20: LET U=PEEK 23635+256*PEEK 23636+1014
190 LET MM=1
200 RANDOMIZE 190: GO SUB 10: RESTORE 200: LET NN=4: LET Y0=6:LET X0=3: LET DY=2: LET DX=1:
      LET S$="                                        ":': LET Y1=18: GO SUB 30: GO TO 200+10*MM
210 LET U=43776: LET E=P: GO TO 300
220 GO SUB 50: GO TO 200
230 GO SUB 6000: LET MM=1: GO TO 200
240 GO SUB 40: LET MM=1: GO TO 200
299 DATA "РАБОТА С РЕДАКТОРОМ ЗВУКОВ","ДЕМОНСТРАЦИЯ ПРИМЕРОВ ЗВУКОВ","ЗАПИСЬ ПРИМЕРОВ
      ЗВУКОВ","ИНФОРМАЦИЯ О ПРОГРАММЕ"
300 RANDOMIZE 390: GO SUB 10: RESTORE 300: LET NN=9: LET Y0=6: LET X0=4: LET DY=1: LET DX=1:
      LET S$="                                    ": LET Y1=18: GO SUB 30
302 IF B=0 AND MM<>1 AND MM<>7 AND MM<>9 THEN PRINT PAPER 6;INK 2;AT 2,2:" СНАЧАЛА НАДО
      СОЗДАТЬ ЗВУК ": BEEP .5,0: PAUSE 50: LET MM=1: GO TO 300
304 GO TO 300+10*MM
310 GO TO 1000
320 GO то 2000
330 GO TO 4000
340 IF B<2 THEN PRINT PAPER 6; INK 2: AT 2,1;" СНАЧАЛА НАДО СОЗДАТЬ 2 ЗВУКА ": BEEP .5,0:
      PAUSE 50: LET MM=1: GO TO 300
345 GO TO 5000
350 GO SUB 50: GO TO 300
360 GO SUB 6000: LET MM=1: GO TO 300
370 GO TO 7000
380 LET MM = 1: GO TO 500
390 LET P=B: GO TO 190
399 DATA "СОЗДАНИЕ НОВОГО ЗВУКА","РЕДАКТИРОВАНИЕ ЗВУКА","ИЗМЕНЕНИЕ ЧИСЛА
      ПОВТОРЕНИЙ","СОЕДИНЕНИЕ ДВУХ ЗВУКОВ"," ДЕМОНСТРАЦИЯ ЗВУКОВ", "ЗАПИСЬ ЗВУКОВ",
      "ЗАГРУЗКА ЗВУКОВ", "УДАЛЕНИЕ ЗВУКОВ", "ВОЗВРАТ В ГЛАВНОЕ МЕНЮ"
400 RANDOMIZE 402: GO SUB 10: RESTORE 400: LET NN = 2: LET Y0=9:LET X0=8: LET DY = 2: LET DX
      = 2: LET S$="                    ": LET Y1=18: GO SUB 30
402 GO TO 400+10*MM
410 LET Z=-1: POKE A+20,82: RETURN
420 LET Z=1: POKE A+20,90: RETURN
499 DATA "ПОВЫШЕНИЕ ТОНА","ПОНИЖЕНИЕ ТОНА"
```

```
500 RANDOMIZE 504: GO SUB 10: RESTORE 500: LET NN=2: LET YO=9: LET XO=3: LET DY=2: LET DX=1:
      LET S$="                                ": LET Y1=18: GO SUB 30
502 GO SUB 60: IF INKEY$="y" OR INKEY$="Y" THEN GO TO 500+10*MM
504 LET MM=1: GO TO 300 510 LET B=B-1: GO TO 504 520 RUN 3
599 DATA "УДАЛЕНИЕ ПОСЛЕДНЕГО ЗВУКА","  ПЕРЕЗАПУСК  ПРОГРАММЫ"
1000 PAPER 1: INK 7: CLS
1010 PRINT AT 4,0;
     "   ЕСЛИ ВЫ ХОТИТЕ ЗА ОСНОВУ ЗВУ-
      КА ВЗЯТЬ ПРИМЕР, ИЗ ИМЕЮЩИХСЯ  В
      ПРОГРАММЕ, ВВЕДИТЕ НОМЕР ЭТОГО
      ПРИМЕРА." ' ' "  ЕСЛИ НЕТ, ТО НАЖМИТЕ (ENTER)."
1020 LET B=B+1: LET P=B: LET N=B-INT(B/20)*20
1030 RANDOMIZE 1030: GO SUB 10:INPUT ("НОМЕР ПРИМЕРА (1-20): ";N;" : "): LINE F$: IF F$<>""
      THEN LET N=VAL F$
1040 IF N<1 OR N>20 THEN GO TO 1030
1050 PRINT AT 13,0;"   ВАМ В ДИАЛОГЕ БУДУТ ПРЕДЛОЖЕ-
      НЫ ЧИСЛОВЫЕ ПАРАМЕТРЫ. ЕСЛИ ХО-
      ТИТЕ ИЗМЕНИТЬ ИХ,  ВВЕДИТЕ СВОИ
      ЗНАЧЕНИЯ." ' '
      "   ЕСЛИ НЕТ, НАЖИМАЙТЕ [ENTER].
1060 RANDOMIZE 1060: GO SUB 10:LET A=U+32*(B-1): LET Y=PEEK 23635+256*PEEK 23636+1014+32*(N-
      1)
1070 FOR R = 0 TO 31: POKE (A+R),(PEEK (Y+R)): NEXT R
1060 GO SUB 20: GO TO 3000
2000 PAPER 1: INK 7: CLS
2010 RANDOMIZE 2010: GO SUB 10:PRINT AT 21,0: "НОМЕР РЕДАКТИРУЕМОГО ЗВУКА": INPUT ("(1-
      ";B;"):";P;" : "); LINE F$: IF F$<>" THEN LET P=VAL F$
2020 IF P<1 OR P>B THEN GO TO 2010
2030 LET A=U+32*(P-1) 2040 GO TO 3000
2500 CLS : PRINT AT 7,0;"   ИЗМЕНЕНИЕ ТОНА ОЧЕНЬ ВЕЛИКО,
      ПОЭТОМУ В РЕЗУЛЬТАТЕ ТОН СТАНЕТ
      НИЖЕ МИНИНАЛЬНОЙ ВЕЛИЧИНЫ. "
2510 IF Z=-1 THEN PRINT AT 9,0;"ВЫШЕ МАКС";
2520 PRINT '''"   НАДО ИЗМЕНИТЬ НАЧАЛЬНЫЙ  ТОН,
      УМЕНЬШИТЬ ВЕЛИЧИНУ СТУПЕНЬКИ ТО-
      НА ИЛИ ЧИСЛО СТУПЕНЕК ТОНА."
2530 BEEP .5,0: GO SUB 20
3000 PAPER 1: INK 7: CLS
3010 LET C=PEEK (A+9)+256*PEEK (A+10): LET E=PEEK (A+5)+256*PEEK (A+6): LET F=PEEK
      (A+17)+256*PEEK(A+18): LET G=PEEK (A+3)
3020 RANDOMIZE 3020: GO SUB 10:CLS : PRINT AT 21,0; "ДИСКРЕТНОСТЬ ТОНА": INPUT ("(1-2000) :
      ";C; " : "); LINE F$: IF F$<>" THEN LET C=VAL F$
3030 IF C<1 OR C>2000 THEN GO TO 3020
3040 POKE A+10, INT (C/256): POKE A+9, INT(C-(PEEK (A+10)*256))
3050 RANDOMIZE 3050: GO SUB 10:CLS : PRINT AT 21,0:"ПЕРИОД НАЧАЛЬНОГО ТОНА": INPUT ("(1-
      20000): ";E;" : "); LINE F$: IF F$<>"" THEN LET E=VAL F$
3060 IF E<1 OR E>20000 THEN GO TO 3050
3070 POKE A+6, INT (E/256): POKE A+5,INT(E-(PEEK (A+6)*256))
3080 IF PEEK (A+20)=82 THEN LET Z=-1: LET MM=1
3090 IF PEEK (A+20)=90 THEN LET Z=1: LET MM=2
3100 GO SUB 100
3110 PAPER 1: INK 7: CLS
3120 RANDOMIZE 3120: GO SUB 10:PRINT AT 21,0;"ВЕЛИЧИНА СТУПЕНЬКИ ТОНА": INPUT ("(1-2000) :
      ";F;" : "); LINE F$: IF F$<>"" THEN LET F=VAL F$
3130 IF F<1 OR F>2000 THEN GO TO 3120
3140 POKE A+18,INT (F/256): POKE A+17,(F-(PEEK(A+18)*256))
3150 RANDOMIZE 3150: GO SUB 10:CLS : PRINT AT 19,0; "ЧИСЛО СТУПЕНЕК ТОНА"''"(ЕСЛИ 1 - ТО
      ПОСТОЯННЫЙ ТОН)": INPUT ("(1-255) : ":G;" : "); LINE F$: IF F$<>"" THEN LET G=VAL F$
3160 IF G<1 OR G>255 THEN GO TO 3150
3170 POKE A+3,G
3180 RANDOMIZE 3000: GO SUB 10: IF E+Z*F*(G-1)<1 OR E+Z*F*(G-1)>20000 THEN GO TO 2500
3190 LET H=1: GO SUB 70: IF SEC>10 AND INKEY$<>"y" AND INKEY$<>"Y" THEN LET MM=2: GO TO 300
3200 CLS : POKE A+24,201: PRINT AT 11,2; "СЛУШАЙТЕ: ТАК ЗВУЧИТ ВАШ ЗВУК": PAUSE 50:
      RANDOMIZE USR A
```

```
3210 PRINT AT 19,0:" НАЖМИТЕ BREAK ДЛЯ ПОВТОРЕНИЯ
     РЕДАКТИРОВАНИЯ ИЛИ ЛЮБУЮ КЛАВИШУ
     ДЛЯ ПРОДОЛЖЕНИЯ...": PAUSE 0
4000 PAPER 1: INK 7: CLS
4010 LET H=PEEK (A+25)-PEEK (A+1)
4020 RANDOMIZE 4020: GO SUB 10:PRINT AT 21,0: "ЧИСЛО ПОВТОРЕНИЙ": INPUT ("(1-255) : ";H;" :
     "); LINE F$: IF F$<>"" THEN LET H=VAL F$ 4030 IF H<1 OR H>255 THEN GO TO 4020
4040 POKE A+1,0: POKE A+24,62: POKE a+25,H
4050 RANDOMIZE 4000: GO SUB 10: GO SUB 70: IF SEC>10 AND INKEY$<>"y" AND INKEY$<>"Y" THEN
     LET MM=3: GO TO 300
4060 RANDOMIZE 4000: GO SUB 10: CLS : PRINT AT 8,11; INVERSE 1; "СЛУШАЙТЕ "; INVERSE 0; AT
     10,7;"ЗВУК НОМЕР ";P;" ГОТОВ": PRINT AT 12,2:"ЗАПУСК - RANDOMIZE USR ";A: PAUSE 50:
     RANDOMIZE USR A
4070 PRINT AT 19,0:"   НАЖМИТЕ BREAK  ДЛЯ  ИЗМЕНЕНИЯ
     ЧИСЛА ПОВТОРЕНИЙ  ИЛИ ЛЮБУЮ КЛА-
     ВИШУ ДЛЯ ПРОДОЛЖЕНИЯ...": PAUSE 0
4080 LET MM=2: GO TO 300 5000 PAPER 1: INK 7: CLS : LET D=1: IF B=2 THEN GO TO 5100 5010
     PRINT AT 10,0;"   СОЕДИНИТЬ МОЖНО ДВА СОСЕДНИХ
     ЗВУКА. ДЛЯ ЭТОГО ВВЕДИТЕ НОМЕР
     ПЕРВОГО ИЗ ДВУХ СМЕЖНЫХ ЗВУКОВ,
     КОТОРЫЕ НАДО СОЕДИНИТЬ."
5020 RANDOMIZE 5020: GO SUB 10: INPUT ("НОМЕР ЗВУКА (1-";B-1;") : ";D;" : "); LINE F$: IF
     F$<>"" THEN LET D=VAL F$
5030 IF D<1 OR D>B-1 THEN GO TO 5020
5100 LET A=U+32*(D-1): POKE A+31,0
5110 RANDOMIZE 300: GO SUB 10: CLS : PRINT INVERSE 1;AT 7,10;"СЛУШАЙТЕ "; INVERSE 0''" ЗВУК
     НОМЕР ";D;" + ЗВУК НОМЕР ";D+1''"ЗАПУСК - RANDOMIZE USR ";A; PAUSE 50: RANDOMIZE USR
     A
5120 RANDOMIZE 5200: GO SUB 10:PRINT AT 19,0;"   НАЖМИТЕ BREAK ДЛЯ РАЗДЕЛЕНИЯ
     ЗВУКОВ ИЛИ ЛЮБУЮ КЛАВИШУ  ДЛЯ
     ПРОДОЛЖЕНИЯ...":PAUSE 0
5130 LET MM=2: GO TO 300
5200 POKE A+31,201: LET MM=4: GO TO 300
6000 PAPER 1: INK 7: CLS 6010 LET O=1: IF B=1 THEN LET I= 32: GO TO 6100
6020 PRINT AT 8,0;"    СНАЧАЛА ВВЕДИТЕ НОМЕР  ЗВУКА,
     С КОТОРОГО ХОТИТЕ НАЧАТЬ ЗАПИСЬ,
     ЗАТЕМ НОМЕР ЗВУКА, КОТОРЫМ ХОТИ-
     ТЕ ЗАВЕРШИТЬ ЗАПИСЬ. "
6030 RANDOMIZE 6030: GO SUB 10: INPUT ("НАЧАЛЬНЫЙ ЗВУК (1-";B;" ) : ";O; " : "); LINE F$: IF
     F$<> "" THEN LET O=VAL F$
6040 IF O<1 OR O>B THEN GO TO 6030
6050 LET V=B: IF O=B THEN LET I=32: GO TO 6100
6060 RANDOMIZE 6060: GO SUB 10: INPUT ("КОНЕЧНЫЙ ЗВУК (";O;"-";B;") : ";V;" : "); LINE F$:
     IF F$<>"" THEN LET V=VAL F$
6070 IF V<O OR V>B THEN GO TO 6060
6080 LET I=(V-O+1)*32
6090 IF PEEK (U+(O-I)*32+(I-1))=0 THEN LET I = I+32: GO TO 6090
6100. RANDOMIZE 6200: GO SUB 10: CLS : PRINT AT 21,0; "ИМЯ ФАЙЛА : ": GO SUB 9: INPUT F$
6110 CLS : SAVE F$CODE U+32*(O-I),I
6120 GO SUB 8: CLS : PRINT AT 1,0; "   КОГДА  ВЫ  БУДЕТЕ  ЗАПИСАННЫЙ
     БЛОК  КОДОВ  ВСТРАИВАТЬ  В  ПРО-
     ГРАММУ, ПОМНИТЕ, ЧТО ЕСЛИ ВЫ
     РАСПОЛАГАЕТЕ ЕГО В ПЕРВОЙ (НУЛЕ-
     ВОЙ) СТРОКЕ BASIC, ТО ВЫ ДОЛЖНЫ
     В ЭТОЙ СТРОКЕ ПОСЛЕ ОПЕРАТОРА
     REM НАБРАТЬ "; FLASH 1;I; FLASH 0; " ПРОБЕЛОВ. "
6130 PRINT '"   ЗАГРУЗИТЕ ЗАПИСАННЫИ БЛОК КО-
     ДОВ: LOAD ""ИМЯ"" CODE 23760"'"   АДРЕСА ЗАПУСКА  ЗВУКОВ  БУДУТ
     РАВНЫ: 23760+32* (N-1 ), ГДЕ N -
     НОМЕР ЗВУКА ИЗ ЗАПИСАННОГО БЛОКА
     ЗВУКОВ."''"   ЕСЛИ ВЫ РАСПОЛАГАЕТЕ КОДЫ ЗА
     RAMTOP, T O: LOAD "ИМЯ" CODE ADR
     ГДЕ ADR >=RAMTOP+1. СТАРТОВЫЕ АД-
     РЕСА ЗВУКОВ БУДУТ ADR+32*(N-1)."
6140 BEEP .1,20: GO SUB 20
```

```
6200 GO SUB 6: RETURN
7000 RANDOMIZE 7100: GO SUB 10: GO SUB 9: PAPER 1: INK 7: CLS
7010 LOAD ""CODE (U+B*32)
7020 LET LEN = PEEK (PEEK 23649+256*PEEK 23650+28)+256*PEEK (PEEK 23649+256*PEEK 23650+29)
7030 LET P=B+1: LET B=B+INT (LEN/32)
7040 GO SUB 8: PRINT ' ' ' "ЧИСЛО ЗАГРУЖЕННЫХ ЗВУКОВ: "; INT (LEN/32)
7050 BEEP .1,20: GO SUB 20
7060 LET MM=2: GO TO 300
7100 GO SUB 8: PRINT ' ' 'TAB 4; FLASH 1; "   ОШИБКА  МАГНИТОФОНА  " 'FLASH 0' ' ' "
      ПОВТОРИТЬ ЧТЕНИЕ- [Y]," ' ' " ВОЗВРАТ В МЕНЮ - ЛЮБАЯ КЛАВИША"
7110 BEEP .5.0: GO SUB 20
7120 IF INKEY$="Y" THEN GO TO 7000
7130 LET MM=7: GO TO 300
9999 PAPER 1: INK 7
```

Переменные, используемые в программе.
P - текущий номер звука.
B - суммарное число звуков в блоке кодов.
U - начальный адрес массива кодов, отведенного под звуки.
N - номер звука из примеров, который берется за основу при создании новых звуков.
F$ - стринг для оператора INPUT, значение которого присваивается переменным.
A - начальный адрес звука, являвшегося текущим.
Y - начальный адрес выбранного примера звука.
R - счетчик перебрасываемых байтов при создании основы звука.
C - дискретность тона.
E - период начального тона.
F - величина ступеньки тона.
Z - признак увеличения или уменьшения тона.
G - число ступенек тона.
J - номер демонстрируемого звука.
H - число повторений "скольжений" тона.
SEC - примерное время звучания звука в секундах.
T, S - то же, но соответственно в минутах и секундах.
D - номер первого из двух смежных звуков, которые надо соединить.
O - начальный звук для записи.
V - конечный звук для записи.
I - длина блока кодов для записи на магнитную ленту.
LEN - длина блока кодов при загрузке звуков с ленты.

О нулевой строке и о кодах, размещенных там - чуть позже. А сейчас об остальных строках программы.

Автостарт программы - со строки 2. Так как для работы программы не нужны никакие кодовые блоки, кроме расположенных в нулевой строке, то строка 2 в программе отсутствует. Строка 3 - вывод на экран заставки. (При этом я считаю "делом чести" для себя указывать в заставке фирму, создавшую прототип этой программы.) После вывода заставки, строка 4 адресует на непосредственное начало программы - строку 100.

В строках 100, 110 устанавливаются начальные параметры. Обнуляется счетчик текущего номера звука (P), максимальное число звуков в блоке кодов (B) устанавливается равным 20 - именно столько готовых примеров находится в программе. Определяется начальный адрес (U) области кодов примеров звуков (он может измениться в случае работы с дисковой операционной системой TR-DOS или при подключении ИНТЕРФЕЙСА-1).

Строка 190 определяет начальное положение указателя главного меню на первый пункт. Затем со строки 200 запускается главное меню.

В главном меню 4 пункта:

```
[ РАБОТА С РЕДАКТОРОМ ЗВУКОВ ]
  ДЕМОНСТРАЦИЯ ПРИМЕРОВ ЗВУКОВ
  ЗАПИСЬ ПРИМЕРОВ ЗВУКОВ
  ИНФОРМАЦИЯ О ПРОГРАММЕ
```

Первый пункт - выделен, так как он наиболее вероятен для выбора. Работа меню подробно была изложена в ZX-РЕВЮ N 9-10 за этот год см. стр. 197. Там же подробно описаны переменные, касающиеся работы меню, поэтому эту часть - пропускаем. Отмечу только, что несколько изменена логика работы блока "ON ERROR GO TO", касающаяся переходов из одного меню в другое, но на этом мы еще остановимся. А также добавлен опрос клавиши "Q" и переход, в случае ее нажатия, на строку 9999, аналогично тому, как это делается в строке 24 для остановки программы.

При выборе первого пункта, строка 210 запускает второе меню. При этом принимает заданное значение начальный адрес области памяти, определенной под создаваемые звуки (U), а максимальное число созданных звуков (B) становится равным P, то есть пока что - нулю.

При выборе второго пункта выполняется (строка 220) подпрограмма GO SUB 50 - это демонстрация готовых звуков, затем опять переход в главное меню. Перед демонстрацией Вам показывается, сколько созданных звуков находится в памяти. Если в этот момент нажать "BREAK", то произойдет завершение подпрограммы путем перехода на строку 59. При помощи "BREAK" можно прервать подпрограмму демонстрации звуков в любом ее месте, но только если клавиша удерживается нажатой до завершения очередного звука, то есть до окончания программы в машинных кодах и возврата в Бейсик, когда возобновит свое действие блок "ON ERROR GO TO".

При выборе третьего пункта выполняется большая подпрограмма GO SUB 6000 - это запись звуков на магнитную ленту, после чего происходит возврат в главвое меню с выделением первого пункта.

При выборе четвертого пункта на экран выводится некоторая вступительная информация о программе (при помощи GO SUB 40).

Вернемся к действиям, происходящим в результате выбора первого пункта главного меню. Строка 210 адресует программу на строку 300 - это запуск второго меню с выделенным первым пунктом. (При этом нет необходимости задавать, как положено, LET MM=1: GO TO 300, так как MM и так равно 1.)

Второе меню выглядит следующим образом:

```
[ СОЗДАНИЕ НОВОГО ЗВУКА ]
  РЕДАКТИРОВАНИЕ ЗВУКА
  ИЗМЕНЕНИЕ ЧИСЛА ПОВТОРЕНИЙ
  СОЕДИНЕНИЕ ДВУХ ЗВУКОВ
  ДЕМОНСТРАЦИЯ ЗВУКОВ
  ЗАПИСЬ ЗВУКОВ
  ЗАГРУЗКА ЗВУКОВ
  УДАЛЕНИЕ ЗВУКОВ
  ВОЗВРАТ В ГЛАВНОЕ МЕНЮ
```

При работе второго меню после выполнения подпрограммы меню GO SUB 30 идет строка 302, в которой проверяется допустимость тех или иных действии. Так, если еще не создано ни одного звука, то не может идти речи о редактировании или демонстрации звуков или записи звуков и т.п. В этом случае на экран выводится табличка "СНАЧАЛА НАДО СОЗДАТЬ ЗВУК", затем предупредительный звуковой сигнал, после чего возобновляется работа второго меню.

А теперь небольшое лирическое отступление, необходимое для пояснения дальнейших строк программы. Кстати Вы можете использовать изложенный прием и в других своих программах.

Ввод параметров при помощи оператора INPUT.

Для того, чтобы сформировать звук, надо в диалоге задать несколько исходных

параметров: дискретность тона, период начального тона, величина ступеньки тона, число ступенек тона, число повторений. Для человека, первый раз запустившего программу, эти параметры - "китайская грамота". Что такое, например, дискретность тона? Даже если указать пределы, в которых может изменяться вводимый параметр, например так:

```
INPUT "ДИСКРЕТНОСТЬ ТОНА (1-2000) "; C
```

то это еще немногим поможет, так как все параметры взаимоувязаны между собой и, скорее всего, величина, взятая наобум из указанного диапазона, приведет к тому, что звука не получится, так как результирующие параметры выйдут за допустимые пределы. Чтобы не блуждать в темноте с завязанными глазами, программа должна подсказывать оператору возможные варианты ответа - такие, что если все время соглашаться с этими ответами (нажимая ENTER), то в конце получится что-нибудь осмысленное. Так должно быть в идеале.

Реализуются эти требования следующим образом. Например, если дискретность тона обозначить C, то процедура ввода может выглядеть так (номера строк даны условно):

```
3010 LET C=5
3020 PRINT AT 21,0;"ДИСКРЕТНОСТЬ ТОНА": INPUT ("(1-2000) : ";C; " : "); LINE F$: IF F$<>""
      THEN LET C=VAL F$
3030 IF C<1 OR C>2000 THEN GO TO 3020
```

При этом в строке 3010 или где-либо в другом месте, например, в начале программы, указывается то значение, которое будет принято программой по соглашению (если Вы просто нажмете ENTER, ничего не вводя). В строке 3020 на экран выводится наименование параметра, требующего ввода, далее идет информация о допустимых пределах и затем предлагается вариант для ввода. Если согласен с ответом - нажимай ENTER, если нет - то задай свое значение и нажми ENTER. Для реализации этого способа потребовалось вместо числового значения C вводить стринг F$, а чтобы при вводе не появлялись кавычки, как обычно при вводе стринга, то вместо INPUT F$ используется INPUT LINE F$.

Строка 3030 проверяет вводимый параметр на допустимые пределы, например, набирая "1000", у Вас случайно набрался лишний ноль, но Вы этого не заметили и уверены в правильности ввода. Если программа просто откажется принять введенное число, то это вызовет у Вас недоумение: "Как же так, ведь было сказано ввести число от 1 до 2000, я и ввел 1000, а что-то программа отказывается работать". В таком случае строка 3030 вернет программу на строку 3020, где Вам будет "показано" введенное Вами ошибочное число. Теперь Вы имеете возможность исправить ошибку, повторив ввод.

Заканчивая лирическое отступление отмечу, что на протяжении всей программы ввод параметров осуществляется такими процедурами. А теперь возвращаемся к работе программы, когда на экране второе меню.

При выборе первого пункта меню, строка 310 переводит программу на строку 1000. С этой строки выполняются действия по созданию в памяти основы нового звука. Строка 1020 увеличивает на 1 число созданных звуков (B), затем этот новый звук становится текущим (P=B), затем подготавливается значение (N) для оператора INPUT в следующей строке. N - это номер примера из набора демонстрационных звуков. Если Вы хотите выбрать какой-то конкретный понравившийся пример, то введите его номер, если нет, то нажмите ENTER, при этом будет взят один из примеров. Обратите внимание на активизацию блока "ON ERROR GO TO" при работе INPUT. Конструкция:

```
RANDOMIZE 1030: GO SUB 10
```

в строке 1030 вернет программу на эту строку в случае ошибки при вводе. Строка 1040 также вернет программу для повторного ввода на строку 1030 в том случае, если вводимая величина выходит за допустимые пределы.

После вывода на экран комментария, строки 1060 и 1070 произведут переброску блока кодов длиной 32 байта из области памяти примеров звуков в область памяти для создания и редактирования звуков. Конструкция:

```
RANDOMIZE 1060: GO SUB 10
```

"подстраховывает" процесс создания в памяти основы звука. Так как этот процесс занимает порядка 1 секунды, то можно успеть в это время нажать "BREAK". В этом случае процесс создания основы звука будет повторен сначала и работа программы не пострадает. Строка 1080 передает управление на строку 3000 - это основная часть по редактированию

звука. О ней - через один абзац.

В том случае, если находясь во втором меню, Вы выберете 2 пункт, то строка 320 передаст управление на строку 2000 - эта часть программы позволяет задать номер звука для редактирования из создаваемого блока звуков. По соглашению, редактируемым является последний созданный Вами звук, но Вы можете выбрать любой из тех, которые создали. Работа оператора INPUT аналогична рассмотренной выше. Строка 2030 определяет начальный адрес текущего звука. Строка 2040 передает управление на основную часть программы по редактированию звука - строку 3000.

В строке 3010 происходит присвоение параметров выбранного примера звука числовым переменным для дальнейшего использования их в качестве "предлагаемых по соглашению" параметров при работе оператора INPUT. Далее идет несколько запросов параметров. Строки 3020, 3030 - ввод "дискретности звука". Строка 3040 - занесение полученных значений в память. Строки 3050-3070 аналогично ввод "периода начального тона". Далее в строках 3060, 3090 происходит подготовка параметров для работы следующего, вспомогательного меню со строки 400, которое вызывается как подпрограмма, чтобы можно было не беспокоиться о номере строки для возврата из этого меню в программу редактирования звука. Это меню задает параметр Z - повышение тона или понижение тона. Если задано повышение тона (то есть уменьшение периода колебаний), то Z=-1 и в память вводится число 82, так как 237, 82 - это коды команды SBC - вычитание. Если задано понижение тона (то есть увеличение периода колебаний), то Z=1 и в память вводится число 90, так как 237, 90 - это коды команды ADC - сложение. После возврата по RETURN в строке 410 или 420 - продолжение программы со строки 3110.

Строки 3120-3140 - ввод значения величины "ступеньки тона". Строки 3150-3170 - ввод "числа ступенек тона".

Далее можно подвести некоторые предварительные итоги, проверив взаимоувязку параметров между собой. Для этого в строке 3180 подсчитывается величина тона, получающаяся при завершении звука, то есть на "последней ступеньке" тона. Изменение тона не должно приводить к тому, что тон выйдет за допустимые пределы. Если это все же происходит, то программа возвращается назад, на строку 2500. Здесь выводится сообщение о недопустимом изменении тона и рекомендуется, что надо сделать для корректировки. После предупредительного звукового сигнала и паузы в строке 2530, переходим опять на начало редактирования звука, на строку 3000. В процессе нового ввода Вам будут показаны введенные Вами значения и Вы сможете скорректировать их согласно указаниям строк 2500 - 2520.

Если проверка в строке 3180 прошла успешно, то программа переходит к следующей проверке в строке 3190. Здесь производится приблизительный подсчет времени звучания одного (H=1) "скольжения" тона. Для этого вызывается подпрограмма со строки 70. Если время звучания не превышает 10 секунд, то - возврат в программу, ничего не делая. Если превышает 10 секунд, то в строке 72 происходит перевод этого времени в минуты и секунды для вывода предупреждения. Оно выводится строкой 74. При этом даются рекомендации, что надо сделать, чтобы уменьшить время звучания. Однако, может быть, Вам специально нужно сделать звук такой длины. Поэтому Вы можете продолжить работу программы, дав согласие на демонстрацию звука, нажав "Y". В противном случае строка 3190 вернет программу во второе меню. Если вместо "Y" нажать "BREAK", то сразу окажемся на строке 3000 - то есть без возврата в меню начинается повторное редактирование. Это происходит благодаря установке в строке 3160 блока "ON ERROR GO TO" на строку 3000.

Теперь, когда пройдены все проверки, в строке 3200 Вы можете услышать результаты своего труда. Здесь Вам демонстрируется одно "скольжение" тона. Если Вам результаты сразу не нравятся, Вы можете нажать "BREAK" и, минуя меню, быстро вернуться на строку 3000 для повторного редактирования. (Как Вы помните, блок "ON ERROR GO TO" настраивается на эту строку в строке 3160). Если результаты вас устраивают, то продолжается редактирование звука со строки 4000.

Строка 4000. Сюда мы попадем, также, если во втором меню выбрать третий пункт (см. строку 330). Эта часть программы задает число повторений "скольжений". Строки 4010-

4040 - ввод значения в память - аналогично тому, как это делалось раньше. Далее, в строке 4050 еще раз производится проверка времени звучания заданного числа "скольжений" тона. И если оно больше 10 секунд, то, как и раньше, требуется подтверждение "Y" для демонстрации звука, иначе - возврат во второе меню. При этом, если вместо "Y" Вы нажмете "BREAK", то сразу же попадете на строку 4020 согласно последней установке блока "ON ERROR GO TO", которое было сделано в строке 4020. Если была нажата клавиша "Y", то в строке 4050 демонстрируется готовый звук, после чего процесс редактирования заканчивается и строка 4060 возвращает программу во второе меню.

Если во втором меню выбран четвертый пункт, то есть соединение двух звуков в один, то необходимо сначала убедиться в том, что эти два звука у нас имеются в наличии. Эта проверка выполняется в строке 340. Если число созданных звуков больше или равно двум, то разрешается дальнейшая работа, где строка 345 переводит программу на строку 5000, иначе - вывод предупредительной таблички со звуковым сигналом и возврат во второе меню.

Программа со строки 5000 - это соединение двух звуков. Соединение производится путем замены команды RET в конце кодового блока первого звука, командой NOP, в результате чего после завершения первого звука не произойдет возврат в Бейсик, а начнется выполнение следующего по порядку звука.

Естественно, что первый звук можно соединить только со вторым, второй - только с третьим и т.д. Если создано всего два звука (B=2), то других вариантов, кроме как соединить первый со вторым - не может быть и программа со строки 5000 переходит на строку 5100, а если звуков несколько, то следует запрос: надо указать номер первого из двух смежных звуков, которые надо соединить. Ввод этого значения - в строках 5010 -5030. После того, как в строке 5100 произойдет непосредственно замена команды RET командой NOP, вам демонстрируется получившийся звук. Если он Вам не понравится, Вы можете опять разъединить звуки, если в момент паузы в строке 5120, после вывода на экран комментария, нажать "BREAK". При нажатии "BREAK" программа переходит на строку 5200, где команда NOP опять заменяется командой RET. В завершение этой части программы запускается второе меню.

Если во втором меню выбрать пятый пункт - демонстрация звуков, то в строке 350, как и в первом меню (см. строку 220), вызывается подпрограмма демонстрации звуков GO SUB 50, после чего опять запуск второго меню. Этим режимом можно также воспользоваться, если Вы просто хотите посмотреть, сколько звуков уже создано. Для этого, как только на экране появится: "ЧИСЛО ЗВУКОВ:", прервите выполнение этой подпрограммы, нажав "BREAK".

При выборе шестого пункта второго меню - записи звуков - выполняется большая подпрограмма записи GO SUB 6000, аналогично тому, как она выполнялись из первого меню. Отмечу только один момент. Например у вас создано три звука. При этом Вы соединили между собой второй и третий звуки, заменив команду RET в конце второго звука командой NOP (при этом второй звук стал вдвое большей длины). Теперь у Вас имеется два звука: первый - обычной длины, и второй - двойной длины. Это необходимо учесть при записи звуков. Так, если Вы хотите записать результаты работы на магнитофон, то в ответ на запросы о номере начального и конечного звуков для записи, Вы логично ответите: 1 и 2. При этом программа сама должна разобраться в том, что в конце блока кодов длиной 32*2=64 байта отсутствует RET. Иначе записанный блок кодов окажется неработоспособным. То есть, если в конце отсутствует RET, то длина кодового блока для записи увеличивается на 32 байта, после чего опять должна быть выполнена проверка на наличие RET в конце блока. Эта проверка выполняется в строке 6090, после чего следует запись (строки с 6100). После записи на экран выводятся краткие комментарии по дальнейшему использованию звуков в Ваших программах. После завершения подпрограммы записи - возврат туда, откуда подпрограмма была вызвана - в первое или второе меню.

При выборе седьмого пункта второго меню - загрузки звуков, программа в строке 370 переводится на строку 7000. Перед началом загрузки блок "ON ERROR GO TO" в строке 7000

настраивается на переход на строку 7100 в случае ошибки магнитофона. Далее идет загрузка блока кодов с ленты в область памяти, непосредственно за последним созданным звуком (строка 7010). Далее определяется длина загруженного блока кодов (строка 7020). Текущим устанавливается первый звук из загруженного блока звуков, а общее число звуков увеличивается на число загруженных звуков (строка 7030), которое индицируется на экране (строка 7040). Далее - вызов второго меню. В случае ошибки магнитофона или нажатия "BREAK" при ожидании загрузки программа переходит на строку 7100. Можно возобновить загрузку, нажав "Y", иначе - переход во второе меню.

В том случае, если во втором меню будет выбран восьмой пункт - удаление звуков, то строка 360 вызывает новое меню, которое расположено со строки 500. Здесь два варианта: удаление последнего звука или вообще удаление всех звуков путем перезапуска программы. При этом в обоих случаях требуется подтверждение для удаления звуков, иначе строка 504 возвратит программу во второе меню. Строка 510 - уменьшение на 1 числа созданных звуков, строка 520 - перезапуск всей программы аналогично перезагрузке с ленты.

И, наконец, последний пункт второго меню - возврат в главное меню. Вообще говоря, этот пункт меню лишний, возврат в главное меню производится точно так же при помощи нажатия на клавишу "BREAK", однако помня об идеологии меню, то есть если Вы активно пользуетесь джойстиком, то этот пункт меню может придать большее удобство при работе.

Для того, чтобы реализовать возможность перехода из второго меню в главное меню, в программе "SOUND" применима несколько иная логика работы блока "ON ERROR GO TO", по сравнению с программой "PRIM". Там при нажатии "BREAK", находясь в первом или втором меню, можно было перезапустить программу сначала с выводом титульной заставки (аналогично перезагрузке с ленты), для чего в начале строки 30 - в начале подпрограммы меню стояла конструкция, активизирующая переход при ошибке (нажатий "BREAK") на строку 3. В программе "SOUND" такое нажатие "BREAK" (случайное, конечно) было бы крайне опасно, так как оно может в один миг уничтожить все созданные Вами звуки, то есть все результаты труда. Поэтому, когда на экране какое-нибудь меню, программа должна по разному реагировать на нажатие "BREAK" в зависимости от ситуации.

Рассмотрим подробнее реакцию программы в различных меню на нажатие "BREAK". Прежде всего, теперь в подпрограмме меню GO SUB 30 Вы не видите конструкции, активизирующей блок "ON ERROR GO TO". Эта активизация должна быть выполнена при входе в соответствующее меню до начала выполнения подпрограммы GO SUB 30. Это происходит в строках 200, 300 и т.д., то есть в начальных строках каждого меню. А теперь конкретно о каждом меню.

Первое меню. Строка 200. Конструкция RANDOMIZE 190: GO SUB 10 вызывает при нажатии "BREAK" опять же это первое меню, переходом на строку 190. То есть, нажимая "BREAK", невозможно никуда попасть дальше первого меню. Так предохраняются от уничтожения результаты редактирования. Кроме этого, появляется небольшое дополнительное удобство на практике - быстрый переход на первый пункт меню из середины меню: вместо многократного или длительного нажатия "ВВЕРХ" или "ВНИЗ" достаточно один раз нажать "BREAK".

Второе меню. Строка 300. При нажатии "BREAK" произойдет переход на строку 390, то есть возврат в первое меню.

Меню "повышение - понижение тона". Строка 400. Здесь при нажатии "BREAK" будет переход на строку 402, то есть та же реакция, что и на нажатие ENTER, то есть принятие параметра по соглашению.

Меню удаления звуков. Строка 500. Здесь при нажатии "BREAK", как и любой другой клавиши, происходит переход на строку 504, то есть возврат во второе меню. В этом меню какие-либо действия возможны только при подтверждении Вашего желания - нажатии "Y".

Немного изменена логика работы блока "ON ERROR GO TO" и в подпрограмме GO SUB 20 - "нажмите любую клавишу". В строке 22 RANDOMIZE 22 заменено на RANDOMIZE 24, то есть при нажатии на любую клавишу, в том числе и "BREAK", работа программы будет продолжена.

Нулевая строка и коды, размещенные в ней, в основном такие же, как и в дебюте программы "PROG". Отличие лишь в том, что вместо SOUND1 - SOUND2 находятся SOUND1 - SOUND20. Это демонстрационные примеры тех звуков, которые могут быть созданы при помощи этой программы.

Для получения блока кодов "sound1-20" можно воспользоваться программой:

```
10 CLEAR 43775: PRINT "PLEASE WAIT..."
20 LET a=43776: LET s = 0
30 FOR n=1 TO 20
40 RESTORE 200
50 FOR a=a TO a+31: READ y: POKE a, y: LET s=s+y: NEXT a
60 NEXT n
70 LET a=43776: RESTORE 300
80 FOR p=1 TO 8: READ m: LET s=s+m
90 FOR n= 1 TO 20: READ y: POKE (a+32*(n-1) +m),y: LET s=s + y: NEXT n
100 NEXT p: CLS
110 IF s<>57522 THEN PRINT AT 0, 0; FLASH 1; "ERROR" : STOP
120 SAVE "sound1-20"CODE 43776,640
130 STOP
200 DATA 14,0,6,0,33,0,0,197,17,0,0,229,205,181,3,225,17,0,0,237,0,193,16,239,62,0,12,
      185,32,228,0,201
300 DATA 3,5,100,40,50,40,30,25,30,5,50,15,20,20,40,5,7,8,20,4,5
310 DATA 5,244,200,5,1,144,232,32,136,232,20,136,16,200,32,232,160,16,136,32,40
320 DATA 6,1,0,0,0,1,3,3,19,3,0,19,39,0,3,3,15,39,19,78,0
330 DATA 9,70,10,5,1,3,3,5,5,50,15,3,5,9,1,100,4,1,5,8,3
340 DATA 17,40,2,10,100,10,20,7,5,50,50,15, 200,144,200,50,100,244,244, 5,5,40
350 DATA 18,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,0,0,0
360 DATA 20,82,90,9090,90,90,90,90,82,90,82,82,90,90,82,82,82,82,82,90
370 DATA 25,1,1,5,2,3,1,2,1,2,2,3,1,2,5,2,5,5,1,1,80
1000 FOR s=1 TO 20: RANDOMIZE USR (43776+32*(s-1)): PAUSE 20: NEXT s
```

Если все набрано правильно, то после того, как Вы сделаете RUN, программа выдаст для записи на магнитофон блок кодов "sound1-20" длиной 640 байт. Вы сразу же можете прослушать Получившиеся звуки, сделав RUN 1000.

Вернемся к нулевой строке. Всего за оператором REM должно быть:

Симв. набор	768 байт
UDG-графика	168 байт
ON ERROR GO TO	73 байт
SOUND 1-20	640 байт
Всего:	1649 байт

Если вы уже набрали дебют программы "PROG", то можно поступить следующим образом. Сначала сохраним на ленте необходимые коды из нулевой строки "PROG":

```
SAVE "code" CODE 23760,1009
```

Это будут коды нулевой строки, кроме SOUND1-2. Дальше надо сформировать нулевую строку заданной длины. Когда Вы будете создавать такую строку при помощи программы "REM FILL", (см. "ZX-РЕВЮ", стр.197) то на запрос о числе добавочных байтов после REM Вы должны ответить: 1649. После того, как нулевая строка заданной длины будет сформирована, загружаем в нее коды:

```
LOAD "code" CODE 23760
LOAD "sound1-20" CODE 24769
```

Теперь эту нулевую строку можно соединить с текстом программы "SOUND" при помощи MERGE.

Рассматривая в этой и предыдущих статьях различные приемы, при помощи которых можно усовершенствовать простые программы на Бейсике, мы невольно начали касаться программирования в машинных кодах. Пока что это небольшие кодовые блоки, вызываемые из Бейсика. Но таких кодовых блоков может быть достаточно много. Вы и в дальнейшем будете выискивать в других программах (в частности в "SUPERCODE") интересные эффекты, получаемые при помощи машинных кодов и все больше использовать их в своих разработках. В результате постепенно будет создаваться Ваша личная библиотека. Важно

то, что для Вас уже не существует непреодолимой границы между Бейсиком и машинными кодами. Ваша программа станет все больше состоять из машинных кодов, которые в конечном итоге заменят всю Бейсик-программу, останется только Бейсик-загрузчик. Такой путь возможен, если идея, заложенная в программе, оказалась удачной и вышла за рамки "повседневных задач", с которых все началось.

Рассматривая некоторые приемы, я поделился лишь одним из огромного множества направлений, таким, которое в итоге приведет вас к полному переходу на програмирование в машинных кодах. А может быть, Вы выберете другой путь, например применение "диалектов" Бейсика - MEGABASIC, BETABASIC и т.д. Или компилирование Бейсик-программ. Каждый свободен в своем выборе.

* * *

Уважаемые читатели!

"ИНФОРКОМ" уже не первый раз печатает работы, присланные нашим постоянным корреспондентом Алексеевым А.Г. Мы надеемся, что многие из предложенных автором идей помогли Вам или еще помогут взять максимум того, что может дать Ваш "Спектрум".

Творческий поиск не стоит на месте и мы, конечно же, не в состоянии опубликовать все возможные идеи и новые разработки, а стараемся останавливаемся на вопросах, имеющих, как нам кажется, массовый интерес и широкое применение.

Для получения более специфичной информации мы можем предложить Вам напрямую связаться с нашим корреспондентом по адресу:

141220, Московская обл., Пушкинский район, п/о Черкизово-2, ул. Б.Тарасовская, д.113, кв.21. Алексееву Андрею Георгиевичу

При обращении просьба вкладывать заполненный конверт с обратным адресом.

В ответном письме Вы получите список авторских разработок служебных программ и вспомогательных драйверов, которые могут быть приобретены по почте за весьма скромную цену (плюс почтовые расходы).

Среди них, например, программный драйвер для программы AST-STUDIO, который позволяет использовать этот графический редактор вместе с принтером для тех, кто не имеет стандартных интерфейсов, поддерживаемых этой программой, а пользуется компьютерами с программируемым портом KP580BB55A или с интерфейсом LPRINT II/III. Таких пользователей у нас очень и очень много и до сих пор использовать графический редактор с принтером им было проблематично.

Этот драйвер позволит взять от редактора максимум: печать разномасштабных копий, печать как поперек листа, так и вдоль, подключать принтеры, имеющие и 8 и 7 иголок, получать полутоновые картинки с имитацией цвета оттенками серого.

Драйвер имеет небольшую длину. Вы всегда сможете набрать его самостоятельно и подключить к программе ARTSTUDIO, следуя подробным инструкциям.

Если у вас при этом нет оригинальной (настроечной) версии программы "ARTSTUDIO", а есть уже инсталлированная и неперенастраиваемая программа, этот вопрос тоже может быть решен.

Среди прочих разработок, предлагаемых Вашему вниманию, стоит отметить универсальный программный драйвер, позволяющий реализовать все графические возможности Вашего принтера. Будучи подключенным к текстовому редактору, такой драйвер позволит Вам формировать и распечатывать документы достаточно сложной фактуры. Это могут быть, например, фирменные бланки, виньетки, визитные карточки и т.п.

Исчерпывающую информацию и условия оплаты Вы получите, обратившись письменно по указанному выше адресу.

ВНИМАНИЕ

Уважаемые читатели!

"ИНФОРКОМ" проводит исследование, посвященное игровым программам для IBM-совместимых компьютеров.

Мы знаем, что многие из вас имеют доступ к IBM-совместимой технике в учебных заведениях, на работе, в клубах, через друзей, а некоторые и дома.

Нам очень ценно ваше мнение о значении игровых программ и мы будем очень рады, если Вы сможете написать нам несколько слов о самых любимых играх. Если Вы напишете названия нескольких любимых игр, по-возможности названия фирм, их выпустивших, и год выпуска - этого будет вполне достаточно, чтобы составить представление о том, как распространены эти программы в вашей стране и какой популярностью они пользуются. Свои отклики Вы можете отправлять на простой почтовой открытке.

Мы будем особенно признательны, если Вы сможете указать свой возраст и род занятий, поскольку изучение приверженности разных возрастных групп к разным игровым программам - одна из задач этого исследования.

Ваше мнение поможет в проведении этого интересного и, надо сказать, уникального эксперимента.

Ждем ваших откликов по адресу 121019, Москва, Г-19, а/я 16.

Если же Вас лично интересуют те исследования, которые мы ведем по игровым программам для IBM-совместимых компьютеров (а любители игровых программ оценивают их как очень интересные), то Вы можете ежемесячно знакомиться с ними на страницах журнала "МОНИТОР", который предоставляет для этого центральные полосы каждого номера. Планируется и выпуск отдельной книги массовым тиражом.

КАК ЭТО ДЕЛАЕТСЯ!

RANARAMA

По своему жанру эта игра относится к аркадным адвентюрам. Пожалуй, ярко выраженный мотив блуждания по огромному многоэтажному лабиринту все-таки приближает ее к чисто аркадным играм, но возможность использования различных объектов, встроенные аркадные вставки и применение магических заклинаний все-таки делают ее аркадной адвентюрой.

Автор игры - знаменитый программист из фирмы "HEWSON CONSULTANTS" - Стив Тернер. Наши читатели знакомы с его работами, например по программе "Quazatron", освещенной в "ZX-РЕВЮ-92" на стр.79, а также по серии публикаций "Профессиональный подход" в "ZX-РЕВЮ-91".

Мы знаем, что игры лабиринтного типа явились одними из первых игр на "Спектруме" и их период расцвета относится к 1982 -1983 годам. В какой-то мере можно удивляться, что в 1987 году на рынок поступило одновременно несколько продуктов, выполненных в единой манере лабиринтной игры - "Dandy" (ELECTRIC DREAMS), "Gauntlet" (US GOLD), "Ranarama" (HEWSON CONSULTANTS) и некоторые другие. Сразу несколько фирм вернулись к старой лабиринтной идее на новом уровне. Что это - случайность или закономерность?

Дело в том, что бурное развитие техники программирования к этому периоду позволило совсем по другому представить старую идею. В лабиринтных играх нового поколения мы видим трехмерную растровую графику, применение оттеночных эффектов для придания изображению глубины, многокрасочную цветовую палитру, совершенные звуковые эффекты и элементы стратегического планирования, необходимые для успешного прохождения игры и, конечно, значительное увеличение размеров игрового поля. Тому, как удается "втиснуть" все это ограниченный объем памяти "Спектрума" нам сегодня расскажет Стив Тернер на примере программы "Ranarama", а пока несколько слов о самой игре.

* * *

Начинающий чародей Мервин доэкспериментировался с различными магическими снадобьями до того, что превратил себя в лягушку. Положение, можно сказать, почти безвыходное, но нет худа без добра. Благодаря этому ему удается скрываться от своры враждебно настроенных колдунов, жаждущих его погибели (нам неизвестно за какие грехи).

Теперь он может обыскать восемь этажей в подземелье, на каждом из которых находятся по двенадцать колдунов, победить их и, вооружившись захваченными магическими приемами, вернуть себе человеческий облик.

Как только он находит колдуна и входит с ним в контакт, игра переходит в режим аркадной вставки. В этом режиме надо решить головоломку типа анаграммы в ограниченное время. Злой колдун перепутал буквы в названии игры "R.A.N.A.R.A.M.A" и, оперативно манипулируя джойстиком или клавишами, Вы должны расставить их по своим местам (этот элемент напоминает схватку двух роботов за обладание системой управления в предыдущей программе С. Тернера "Quazatron"). Работа требует сообразительности, глазомера и четкой координации движений. На каждом уровне время, отведенное для решения головоломки, постепенно уменьшается. Кажется, что ввод аркадных вставок в аркадно-адвентюрную игру стал как бы визитной карточкой фирмы "HEWSON CONS." (см. например игру "Firelord").

Если Вы с этой задачей справляетесь успешно, то программа возвращает вас в главный экран, колдун исчезает, и на его месте появляются магические руны, которые Вы должны подобрать как можно быстрее. Сбор этих рун является необходимым элементом, поскольку Вы сможете впоследствии конвертировать их в магические способности.

В пол лабиринта местами встроены иероглифы (сокращенно "глифы"), которые обладают разнообразными функциями. Так, например, "голова" обозначает "глиф колдовства" (Glyph of Sorcery"). Задействовав этот глиф, Вы можете проскроллировать и исполнить те магические заклинания, к которым получили доступ в результате сбора магических рун.

Имейте в виду, что некоторые заклинания не могут быть использованы, поскольку магические силы, которые с их помощью вызываются, могут быть доступны только на более высоких уровнях.

Поскольку в игре Вам дано только две "жизни", а этого явно маловато, то очень полезно приобрести заклинания, способные перебросить Вас на более высокий уровень.

Каждый из восьми уровней имеет от 50 до 100 комнат. В начале игры вас помещают в случайно выбранной комнате. Проходя через дверь, Вы как бы "включаете свет" в очередной комнате и она становится видимой. Очень полезно воспользоваться глифом "зрения", который позволит вам увидеть все комнаты своего этажа. Правда, вам покажут только те комнаты, в которых вы уже были, но зато есть возможность увидеть невидимые двери, которые при простом обходе комнат вы не обнаружили. Впрочем, их можно оригинально обнаружить и при обычном проходе по лабиринту. Движущиеся существа могут на мгновенье "просунуть" голову сквозь стену в тех местах, где есть такая дверь.

Как и должно быть в играх подобного рода, подземелье кишит всякой нечистью. Здесь Вы найдете змей, злобных гномов, чудовищных насекомых. Уничтожение их требует стрельбы и уменьшает запасы Вашей боевой мощи, но существует специальный глиф, с помощью которого можно уничтожить всех монстров в пределах комнаты. Этот глиф, в отличие от остальных, расходуемый и не восстанавливается после применения.

Значительно большую опасность представляют такие неприятные духи, как вращающиеся мечи или, например, щелкающие челюсти. Жизнь усложняется тем, что они не поражаются Вашим огнем. Надо подумать, каким образом уничтожить тот генератор, который насылает их на Вас.

Даже когда Вы пройдете все комнаты на этаже, Вы найдете, наверное не всех колдунов (а должно их быть двенадцать). Уничтожить всех необходимо, поэтому придется разыскать тех, кого Вы пропустили. Так как они шляются, где пожелают, то может быть Вы просто разминулись с ними по пути. Есть специальное заклинание, которое поможет обнаружить их местоположение - можете воспользоваться им.

Вот, пожалуй и все. Собирайтесь в путь и настройтесь на увлекательную игру, требующую как логического мышления и наблюдательности, так и определенных спортивных навыков. А мы сейчас предоставим слово Стиву Тернеру для рассказа о том, как создавалась эта игра.

* * *

В прошлых статьях из серии "Профессиональный подход" я немного рассказал о том, что делается на той кухне, где программисты готовят новые программы для Вашего "Спектрума". В основном это были теоретические статьи, а вот сейчас на примере программы "RANARAMA" мы посмотрим, как это происходит на практике.

Техническое задание.

Поскольку я свободный программист, то работаю над тем, что мне нравится, и вроде бы нет никакой необходимости в подготовке технического задания на разработку игры. Тем не менее, это не так. Просто-напросто я сам ставлю себе техническое задание и сам же себе его утверждаю. Это дисциплинирует и позволяет на всех этапах работы сверять то, что получается с тем, что должно было быть.

Концепция игры пришла от программы "Paradroid", написанной для "Коммодора-64". В принципе, эта концепция была использована и при подготовке программы "Quazatron", но способ экранного представления был там совершенно иным. Во-первых, сам сценарий там

был научно-фантастическим, а мне казалось, что игра на тему магии и волшебства очень хорошо впишется в структуру программы типа "Квазатрона". К моменту, когда я в принципе обдумал сценарий, стоящий за игрой, у меня уже довольно четко вырисовались требования к программе.

1. Достижение трехмерного образа на экране за счет применения теневых эффектов. К тому времени, когда я задумал эту игру, такая графика уже была испытана на "Коммодоре", но на "Спектруме" в полной мере этот прием еще не был освоен.

2. Отказ от скроллинга экрана и за счет этого возможность использования всего цветового многообразия компьютера, т.к. отпадает существенная часть атрибутных проблем (проблема "клэшинга" атрибутов).

3. Двумерное представление игрового поля на экране (вид сверху). Отказ от принципа "один экран - одна комната". Одновременно на экране могут изображаться несколько комнат. В то время это было новым словом.

4. Структура игры - аналогична программе "Quazatron", но в основе сценария должна лежать магия и волшебство. Соответственно, меняется тема и содержание аркадной вставки.

5. Принцип разделения "монстров" на две категории.

Первостепенные - колдуны. Найти и победить их - необходимый элемент для успешного исполнения программы.

Второстепенные "монстры" не являются необходимыми - это просто объект для стрельбы. Их не надо искать, они нападают сами. Количество их разновидностей измеряется десятками. Общее количество уничтоженных "монстров" за время игры - порядка сотен.

6. Техника постепенного высвечивания игрового поля. Впервые эта идея появилась во время тестирования программы "Quazatron", но туда она не пошла. Суть состоит в том, что карта игрового поля хранится "нераспакованной" для тех участков, которые еще не были исследованы, а на экране эти области затемнены. Это была пионерная идея. В то время аналогов еще не было.

7. Обработка "скрытого" изображения. Эта техника требует пояснения. Дело в том, что "монстры", находящиеся в комнатах, соседних с той, в которой находится "герой", на экране показываться не должны, но, тем не менее, должны жить собственной жизнью, перемещаться по заданным законам, в общем, вести себя так, как если бы они были видны.

Эта идея была заимствована у программы "Paradroid", но значительно переработана с целью ускорения обработки данных.

Косвенный эффект от применения такой техники - ускорение перестроения изображения на экране, т.к. все перемещения выполняются только в пределах ограниченного окна экрана, соответствующего комнате, в которой находится герой.

8. Стопроцентное использование площади экрана. К тому времени, о котором идет речь, очень широко распространилась техника выделения в качестве динамического "окна" какой-то части экрана, например одной трети или двух третей. При этом остальная площадь экрана заполняется статичной, неизменяющейся графикой или закрашивается чёрным цветом или используется для вывода текстовых сообщений.

На мой взгляд, использование всей площади экрана должно было дать особую конкурентоспособность программе, по сравнению с прочими продуктами, имеющимися на рынке.

9. Вышеизложенное техническое требование означало для меня полный отказ от текстовых сообщений и всяких статичных локальных дисплеев. Это, в свою очередь, вызвало требование так организовать пользовательский интерфейс, чтобы всю информацию о ходе игры пользователь получал бы через динамическую графику и звук.

10. Требования к логике управления "монстрами" заключались в том, что они должны вести себя "разумно". Они должны искать героя, переходить из комнаты в комнату и выходить за пределы экрана. Их способы перемещения по игровому полю должны быть достаточно мотивированы. То есть, они должны входить и выходить из помещений через двери, а не материализовываться из воздуха, когда и где вздумается.

Когда я подготовил такой набор технических требований к программе, я почувствовал реальность выполнения этой задачи как с точки зрения объема занимаемой оперативной памяти, так и с точки зрения быстродействия процессора.

Предварительные исследования.

1. Дизайн экрана.

Исследовательскую часть я начал с самого важного, на мой взгляд, момента, ибо он во многом будет определять коммерческую ценность будущей программы. Я построил демонстрационный экран программы и оценил:

- художественное впечатление;
- объём расходуемой памяти:
- быстродействие.

Всеми результатами я остался доволен. Особенно привлекательными мне показались добротные, прочные стены, разделяющие комнаты.

Все графические элементы, присутствуюшие на экране, конструировались из специальных символов, для которых я создал знакогенератор.

2. Раскладка оперативной памяти.

Закончив с первым этапом, я перешел ко второму. Поскольку у меня уже был определенный опыт, я поначалу воспользовался той картой оперативной памяти, которая сложилась после завершения программы "Quazatron", благо структура у этих двух программ была похожа. Я выделил участки памяти для хранения карты игрового поля, для хранения нужной мне графики, для всевозможных таблиц, массивов, для области рабочих процедур и программных переменных.

Как обычно, вскоре я обнаружил, что мои амбиции заходят слишком далеко и не все, что я запланировал, можно "втиснуть" в спектрумовское ОЗУ. Так, например, я хотел иметь 24 типа различных движущихся "монстров", но пришлось урезать их количество до 14.

3. Упаковка данных.

Убедившись, что оперативной памяти мне не хватает, я начал думать об упаковке данных. В первую очередь, мне предстояло хранить несколько карт игрового поля (по одной для каждого уровня) - это наиболее емкие данные.

Я подготовил несколько различных методов, поэкспериментировал с ними и, наконец, остановился на наиболее оптимальном.

В принятом мною методе на задание комнаты на карте игрового поля расходовалось всего два байта и еще по два байта уходило на описание каждой двери. Вот как это было сделано.

Единицей измерения на карте я выбрал блок из четырех знакомест (размер 16x16 пикселов).

Для комнат в первом байте я решил хранить координату левого верхнего угла комнаты, я во втором байте - размер этой комнаты. Вы знаете, что поскольку байт не может принимать значение больше 255, то мне нелегко было бы сделать достаточно большое игровое поле. Максимум - 16x16 блоков по 16x16 пикселов, то есть, чуть больше одного экрана. Это, конечно, недостаточно, поэтому координата левого верхнего угла задается не как абсолютная, а как относительная, то есть это "смещение" начала N-ой комнаты относительно N-1 -ой комнаты. Тогда все стало на свое место (пример см. на рис. 1)

Пришлось "помудрить" и со вторым байтом, задающим размер комнаты. В итоге я остановился на том, что у меня в программе будет ограниченное количество разных типоразмеров комнат и второй байт будет задавать собственно не размер комнаты, а номер ее типоразмера. Соответствующая рабочая процедура потом по этому номеру найдет в таблице данных истинный размер комнаты (пример см. на рис. 1).

0	1	2	3	4	5	6	7	8	9	10	11
12	13	14	15	16	17	18	19	20	21	22	23
24	25	26	27	28	29	30	31	32	33	34	35
36	37	38	39	40	41	42	43	44	45	46	47
48	49	50	51	52	53	54	55	56	57	58	59
60	61	62	63	64	65	66	67	68	69	70	71

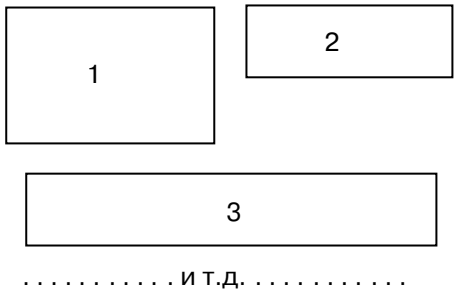
. . . . . и так далее . . . . . .

ОРГАНИЗАЦИЯ ДАННЫХ ПО КОМНАТАМ

N комнаты	смещение	типоразмер
1	0	1
2	6	2
3	24	2
4	18	3

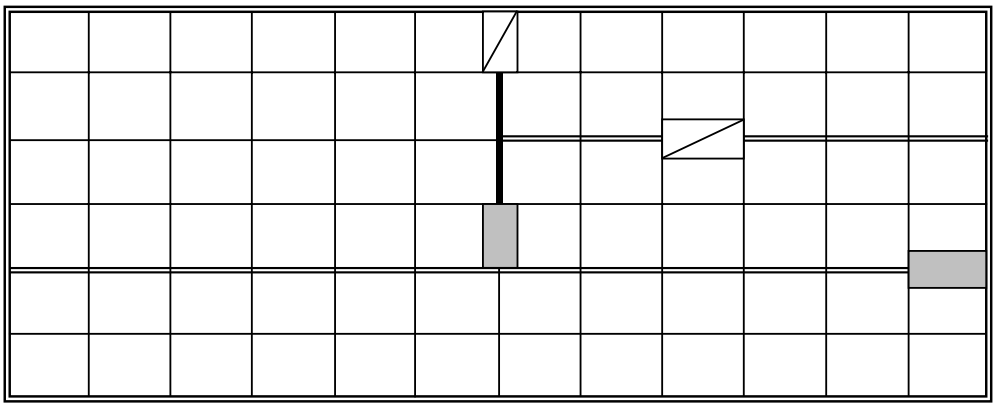
. . . . . и так далее

ТИПОРАЗМЕРЫ КОМНАТ



. . . . . и т.д. . . . . .

Рис.1



. . . . . и так далее . . . . . .

ОРГАНИЗАЦИЯ ДАННЫХ ПО ДВЕРЯМ

N двери	смещение	тип
1	5	1
2	15	2
3	21	1
4	6	4

. . . . . и так далее

ТИПЫ ДВЕРЕЙ

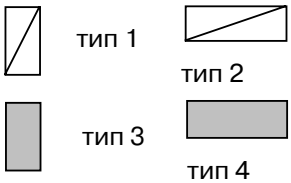


Рис.2

Аналогично было и с дверьми. Первый байт задает "смещение" координаты двери относительно координаты предыдущей двери. Второй байт задает тип двери. Двери могут быть четырех типов. Во-первых, они могут быть вертикальными (тип 1) или горизонтальными (тип 2), а во-вторых, они еще могут быть и невидимыми (типы 3 и 4 соответственно). Пример см. на рис. 2.

4. Специальные алгоритмы.

Для того, чтобы работать с упакованными, как было описано выше данными, пришлось разработать несколько специальных алгоритмов и на их основе создать несколько соответствующих процедур.

В основу была положена следующая логика работы (представим себе, что герой входит в новую комнату и ее надо "высветить" на экране):

а) По координате "героя" из упакованной карты игрового поля извлекаются координаты комнаты и дверей.

б) На экране строится соответствующая комната из символов, которые выдает специальный знакогенератор.

в) По номеру комнаты из специальной таблицы сценария извлекаются данные о наличии в ней объектов и предметов.

г) По их номерам и координатам они изображаются на экране.

д) Если в таблице программных переменных есть "монстры", находящиеся в данный момент в этой комнате, то изображаются и они.

Для создания трехмерного эффекта теневой графики я разработал специальный алгоритм, который назвал "шэдоу-процессором". После того, как комната и весь ее внутренний антураж построены, на изображение накладываются полутоновые горизонтальные и вертикальные тени. На рис. 3 доказано, как все тени изображаются с помощью всего лишь двух элементов.

"Шэдоу-процессор" позволил сэкономить еще изрядное количество памяти.

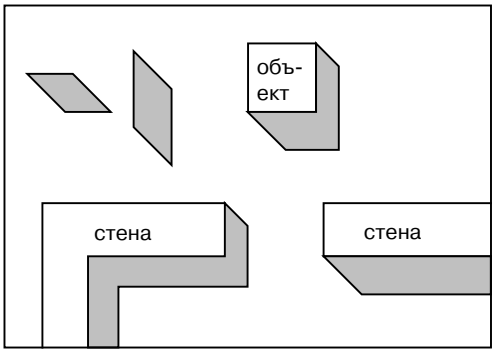


Рис. 3

5. Проверка концепции.

Предприняв такие меры, я успокоился относительно распределения оперативной памяти. Теперь я понял, что ее мне хватит, оставался открытым вопрос о достаточности быстродействия компьютера для того, чтобы обслуживать экран при работе по описанной выше логике.

Пришлось воплотить идеи в машинный код и проверить, как обстоит дело с быстродействием. Не сразу, но дело пошло. Работа оказалась неожиданно сложной. Только примерно через месяц я убедился в том, что экранное представление моих идей совпадает с тем, что я хотел получить. Теперь, закончив предварительные исследования и убедившись, что задача в принципе выполнима, я мог приступать к основному объему работ - разработке программы в целом. Надо сказать, что в этот момент мне пришлось преодолеть массу искушений добавить в программу что-то еще. Это было, конечно, возможно, но при таком подходе можно никогда не выйти из стадии предварительных исследований.

Дизайн программы.

Закончив с предварительными исследованиями, я перешел к проработке структурной диаграммы. В статьях "Профессиональный подход" (см. "ZX-РЕВЮ"-91) мы уделили серьезное внимание тому, как создаются структурные диаграммы, зачем они нужны и чем они отличаются от алгоритмических блок-схем. Там же мы упомянули и о том, что при всем многообразии игровых программ, структура их может оказываться удивительно похожей. Так случилось и в этот раз. Практически эта важная и ответственная работа превратилась в пустую формальность. Почти на 100 % подошла структура, ранее разработанная для

программы "QUAZATRON". Конечно мне помогло то, что я начал разработку новой программы не на пустом месте, а уже имея солидный опыт программиста игровых программ. Если же Вам когда-то придется начинать это дело "с нуля", то для справки я привожу структурную диаграмму на рис. 4.

Машинный код.

Этот этап я начал с того, что "перетащил" все процедуры, которые можно, из программы "QUAZATRON" в новую программу. Нет никакого смысла изобретать велосипед и если Вы абсолютно уверены, что та или иная процедура надежно работает и хорошо отлажена, то Вы не только сэкономите время на ее разработку, но и более того, она явится необходимой базой для создания работающих с ней совместно других процедур и для их отладки.

Сейчас у меня уходит на создание такой программы примерно восемь месяцев. Это много, но я абсолютно все делаю сам. Тем более для меня очень важна экономия времени за счет использования большого количества проверенных процедур. И, надо сказать, многое мне удалось использовать. Процедуры верхнего уровня структурной диаграммы были вообще перенесены с минимальными доработками.

Основную трудоемкость, как ни странно, составило не программирование машинного кода, а борьба с разного рода мелкими неприятностями. Примерно месяц ушел на то, чтобы выловить все "жучки" в ассемблирующей программе. Я пользовался ассемблером "OCP", а он регулярно разрушал мои таблицы меток. Еще столько же времени ушло на борьбу с механическими помехами (дребезг контактов на разъеме, через который подключались дисковод и принтер).

Но так или иначе, рано или поздно, наступает такой момент, когда перед Вами еще гора работы, а сделано уже так иного, что отступать нельзя. Вот на этой стадии мне и пришлось столкнуться с концептуальной проблемой, связанной с моим героем. Как оказалось, у меня на него осталось так мало места в оперативной памяти, что сделать приличного чародея я уже не мог. Как я ни экспериментировал, он меня не устраивал. Спас положение Эндрю Брейбрук, который предложил сделать героя лягушкой и тогда спрайт 16x16 получается намного лучше. В поисках приемлемой картинки для своей лягушки я перелистал несколько томов по биологии, неплохо ознакомился с жизнью земноводных и, самое главное, узнал, что лягушка по латыни звучит, как RANA. Так и родилось название игры RANARAMA.

Вторая проблема возникла, когда я уже перевалил за середину своего проекта. Я сделал аркадную вставку, в которой мой герой обменивался ракетными ударами со злыми колдунами. Выпущенная ракета могла взаимодействовать с другими, стационарными ракетами и инициализировать их. При правильной игре можно было вызвать что-то вроде цепной реакции и тогда героям приходилось бы непросто.

Все, кто видел мою работу на этой стадии, однозначно оценили эту аркадную вставку, как неудовлетворительную. У меня было еще очень много работы с графикой и я отложил решение этой проблемы в долгий ящик.

Однажды рано утром я сидел и экспериментировал с клочками бумаги на столе, пытаясь придумать оригинальное изображение титульного экрана с названием игры, выполненным крупными буквами. В этот момент меня осенило. Я понял, что аркадной вставкой может стать перестановка букв в слове RANARAMA. Как оказалось, название программы очень соответствует этой задаче. Поскольку в слове есть 4 буквы "A", то аркадная вставка получилась скорее динамичной, чем головоломной. Первые же эксперименты убедили меня в том, что это решение удачно. Труды предыдущего месяца были сняты с полки и полетели в мусорную корзину, а через пару дней проблема перестала существовать.

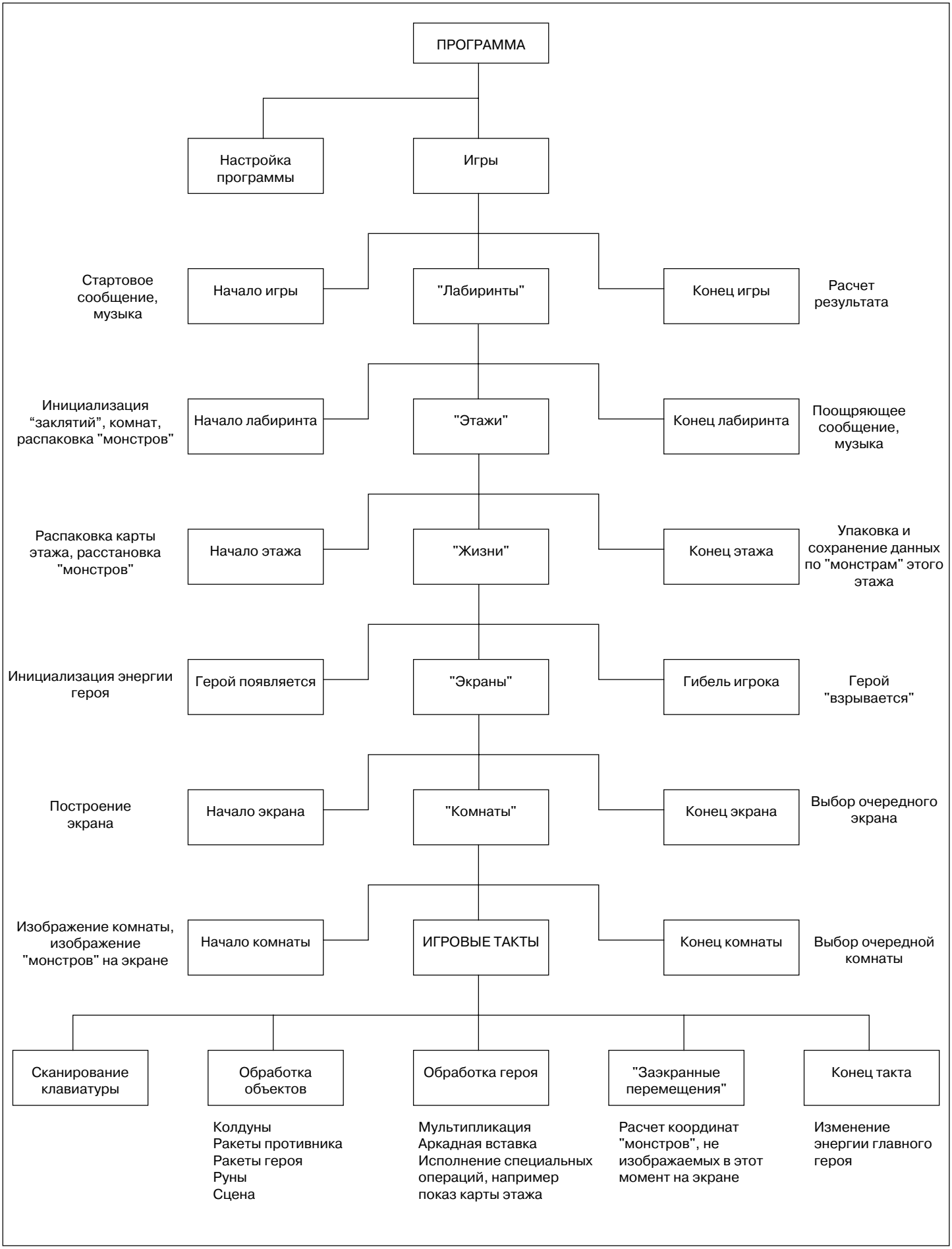


Рис.4 Структурная диаграмма программы RANARAMA

Звук и музыка.

Музыку в программу я добавляю в самый последний момент. Это происходит потому, что музыкальные и звуковые процедуры работают в режиме прерываний 2-го рода - IM2.

Примечание ИНФОРКОМа: В последнем издании книги, посвященной машинным кодам, мы широко рассмотрели применение прерываний 2-го рода. Это издание объединяет выпущенные ранее в 1990 году три тома и примерно на 20% дополнено. Общий объем - 271 стр. Заказы принимаются на приобретение отдельных экземпляров и на лицензированный тираж в регионе принимаются.

Поскольку меня очень часто спрашивают, как мне удается получать такие интересные звуковые эффекты в программах, я остановлюсь на этом вопросе поподробнее, хотя здесь не буду касаться работы с прерываниями 2-го рода, дабы не нарушить простоту изложения.

"Спектрум" имеет очень ограниченные звуковые возможности. У него есть всего лишь один звуковой канал, который может находиться в двух состояних - "вкл"/"выкл". Пульсирующий сигнал определенной частоты вызывает появление звукового тона соответствующей частоты (см. рис. 5).

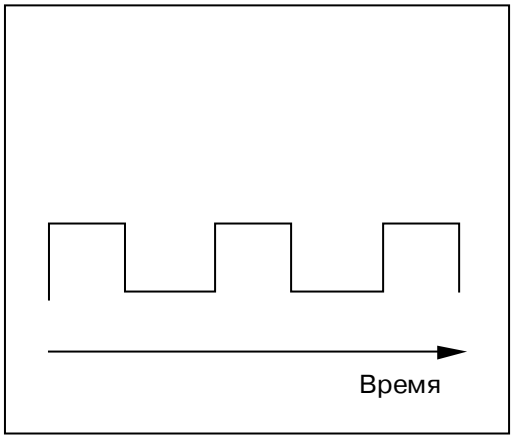


Рис. 5

Таким образом, создание сложных звуков сводится к управлению частотой.

Тот принцип, который мы здесь рассмотрим, может быть использован и владельцами 128-килобайтных машин при программировании встроенного звукового процессора. Я применял этот прием и для "Коммодора 64" при озвучивании программы "URIDIUM". Аналогичную технику используют и синтезаторы серии "Ямаха DX".

Предлагаемая Вашему вниманию программа выполняет линейную модуляцию частоты. Таким образом, частота изменяется вверх или вниз в соответствии с данными, взятыми из специальной таблицы. На рис. 6 и 7 показаны графики частот.

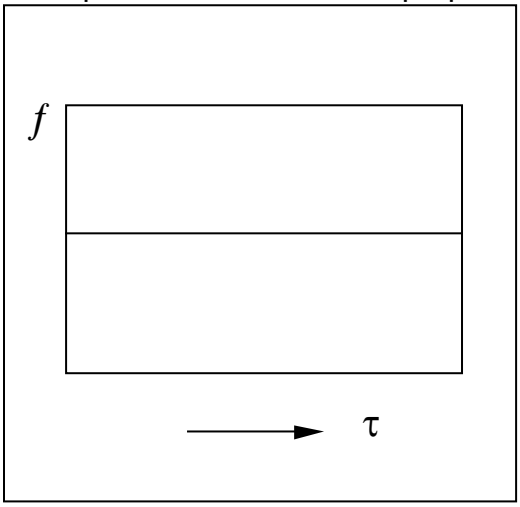


Рис. 6

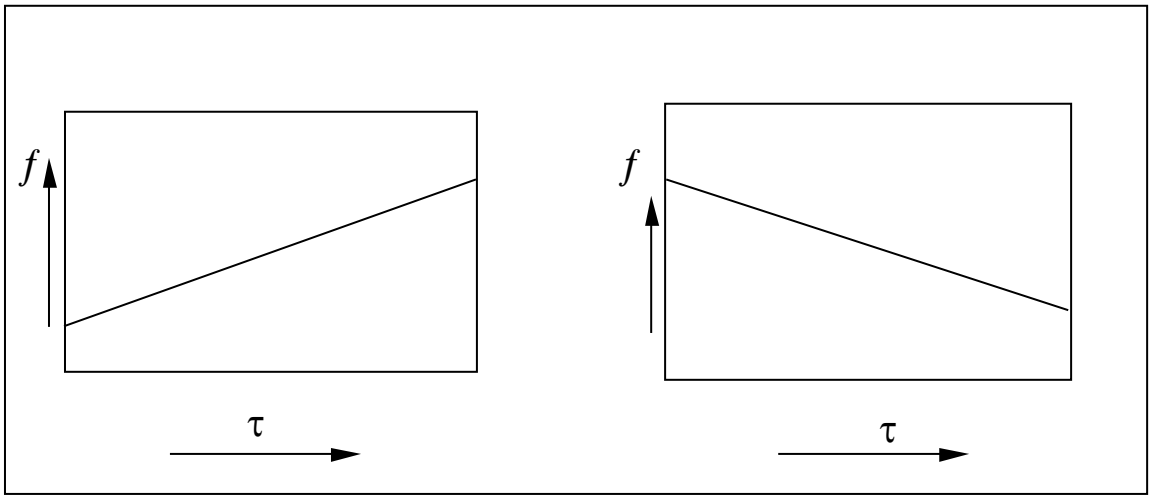


Рис.7

Одноступенчатая модуляция.

Итак, изменяя частоту звука вверх или вниз или и так и этак, мы можем сделать звучание более интересным. В приведенной ниже программе используется только изменение частоты по линейному закону. Это означает, что скорость изменения частоты остается постоянной. Вы можете поэкспериментировать и с другими формулами, меняя не только саму частоту, но и скорость ее изменения.

Пилообразная модуляция

(рис. 8) легко организуется путем создания в программе счетчика тактов. Всякий раз, как счетчик будет обнуляться, частота возвращается к своему исходному значению. Другой счетчик отсчитывает необходимое количество таких циклов. На рис. 8 видно, что пилообразная модуляция может быть как прямой, так и обратной.

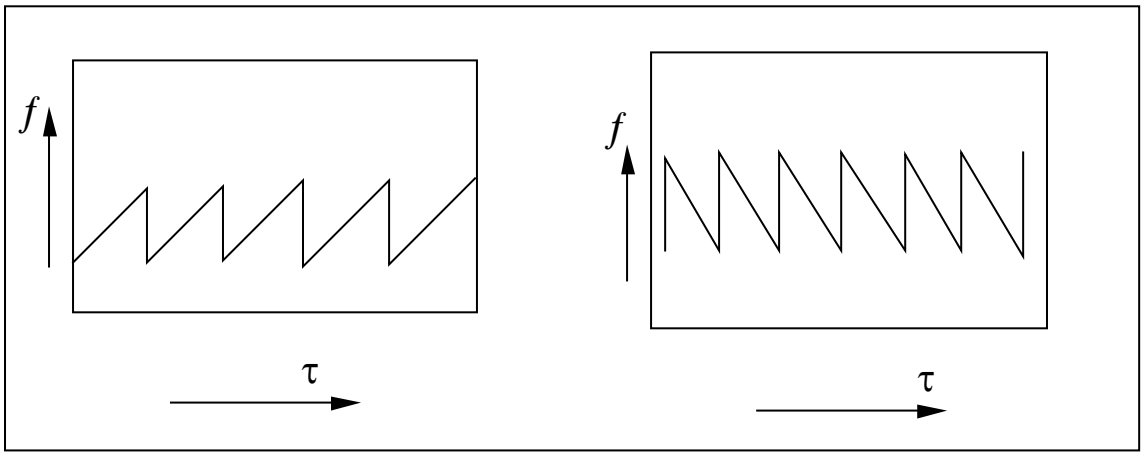


Рис. 8

Треугольная модуляция

(рис. 9) обеспечивается тоже с помощью счетчика тактов, но в этом случае при обнулении счетчика происходит не восстановление исходного значения частоты, а изменение знака ее приращения.

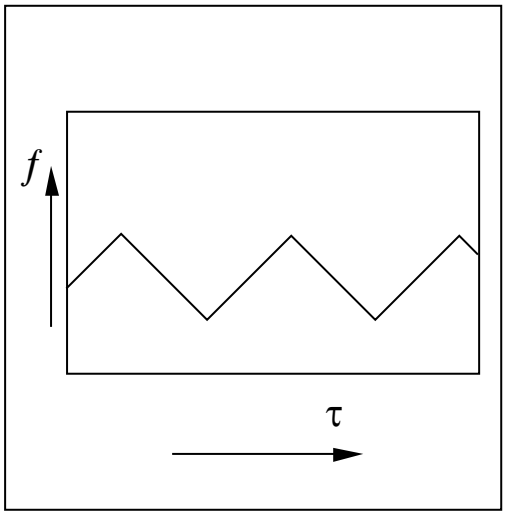


Рис. 9

Двуступенчатая модуляция

Пример такой модуляции показан на рис.10. Всякий раз, когда счетчик тактов обнуляется, возврат к исходной частоте происходит с некоторым смещением.

Все остальное - дело практики. Если идея Вам понятна, то можете приступать к экспериментам. Вы можете создавать собственные алгоритмы для управления частотой, можете хорошо поэкспериментировать с данными в таблице звуковых эффектов. Скоро Вы почувствуете диапазон Ваших возможностей.

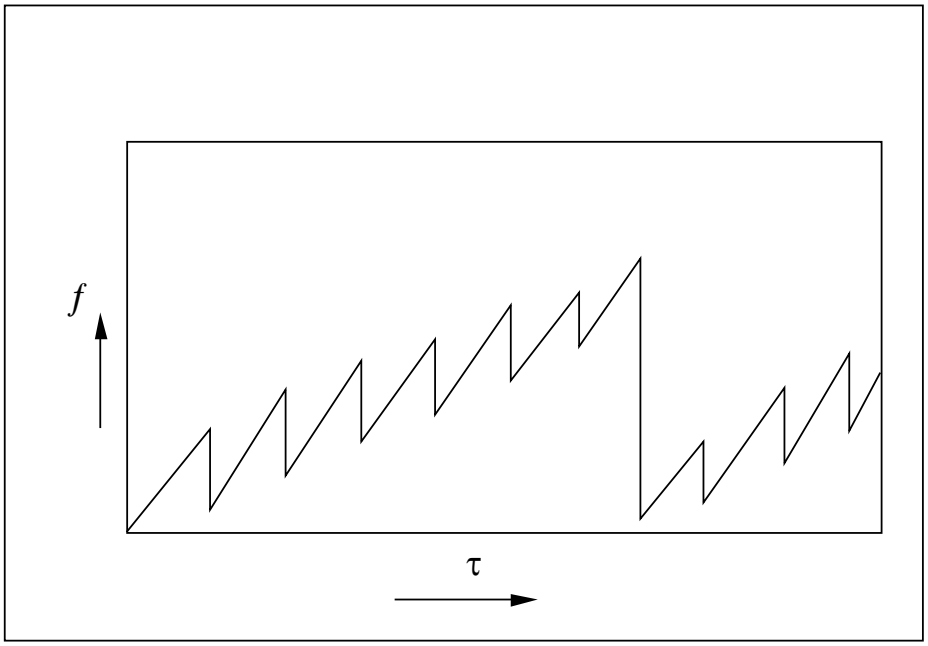


Рис. 10

Программа.

Вашему вниманию я предлагаю программу для испытания звуков. Назовем ее SOUNDTESTER. Она состоит из двух блоков. Первый блок - БЕЙСИК-загрузчик (Листинг 1). Он загружает второй блок и выполняет настройку параметров, которые Вы пожелаете изменить. Здесь Вы задаете закон, по которому будет происходить модуляция частоты, здесь же задаются настроечные параметры. Второй блок - в машинных кодах (Листинг 2). Он, собственно, и выполняет все необходимые расчеты и выдает сигнал на порт звукового динамика.

ЛИСТИНГ 1

```
10 CLEAR 40000
15 LET sontab = 45235: LET sonreq = 45234
20 LOAD "sound" CODE 45000
30 DIM a(10)
```

```
 40 FOR x=1 TO 9: LET A(x)=0: NEXT x
 1000 CLS: PRINT "          SOUNDTEST"
 1010 PRINT ' "1. SOUND NUMBER", a(1)
 1020 PRINT "2. START FREQ. ", a(2)
 1030 PRINT "3. FREQ. CHANGE". a(3)
 1040 PRINT "4. CHANGE TIMES". a(4)
 1050 PRINT "5. REPEAT TIMES". a(5)
 1060 PRINT "6. MODULATE TYPE",a(6)
 1070 PRINT "    0 = SAWTOOTH"'
       "    1 = 2nd MOD DOWN"'
       "    2 = 2nd MOD UP"'
       "    OTHERS = TRIANGLE"
 1080 PRINT "7. RESET FREQ", a(7)
 1090 PRINT "8. CHAHGE T RESET",a(8)
 1100 PRINT "9. CHAIN TO NO",a(9)
 1110 PRINT "10. DELAY", a(10)
 2000 INPUT "ENTER 0 TO FIRE SOUND OR NUMBER TO CHANGE VALUES"; i
 2010 IF i = 0 THEN GO TO 3000
 2020 IF i>10 OR i<1 THEN GO TO 2000
 2030 LET x=INT (i)
 2040 INPUT "NEW VALUE 0. . . 255 "; i
 2050 IF i<0 OR i>255 THEN GO TO 2040
 2060 LET a(x) = INT (i)
 2063 IF x=1 THEN GO TO 2500
 2065 REM Запись звуков в таблицу.
 2070 LET z=sontab + (a(1)*8)+x-2
 2080 POKE z,a(x)
 2090 GO TO 1000
 2500 REM вызов звука на редактирование
 2510 LET z=sontab + (a(1)*8)
 2520 FOR x=2 TO 9: LET a(x)=PEEK z: LET z=z+1: NEXT x
 2530 GO TO 1000
 3000 REM Вызов звука на прослушивание
 3005 LET x=a(10)
 3010 PRINT AT 20, 1; "PRESS 0 TO REPEAT SOUND OR 1 TO CHANGE. "
 3020 POKE SONREQ, a(1)+1
 3025 PAUSE 20
 3030 RANDOMIZE USR 45000: FOR y=0 TO x: NEXT y: IF INKEY$="" THEN GO TO 3030
 3060 IF INKEY$ ="0" THEN GO TO 3020
 3070 GO TO 1000
 9000 FOR x= 45227 TO 45400: PRINT PEEK x: NEXT x
```

КОММЕНТАРИИ

Строка	Содержание
1010	Номер звука a(1)
1020	Начальная частота a (2)
1030	Скорость изменения частоты a(3)
1040	Количество модуляций в звуке a (4)
1050	Количество повторений звука a(5)
1060	Вид модуляции a(6): 0 – пилообразная, 1 - двухступенчатая нисходящая, 2 - двухступенчатая восходящая, пр. – треугольная,
1070	Частота сброса a(7)
1080	Темп изменения частоты сброса a(8)
1090	К какому звуку "привязать" данный звук? a(9)
1100	Пауза a(10)
2000	Введите 0 для прослушивания звука или число от 1 до 10 для внесения изменений.
3010	Нажмите 0 для повтора или 1 для редактирования.

```
                        ORG 0AFC8H
AFC8  F3        SOUND   DI
AFC9  3AB2B0            LD A,(SONREQ)
AFCC  A7                AND A
AFCD  281F              JR Z,NONEW
AFCF  32B1B0            LD (SONNOW),A
AFD2  FE0A              CP 0AH
AFD4  2825              JR Z,NOISE
AFD6  21B3B0            LD HL,SONTAB       ; Адрес таблицы звуков
AFD9  3D                DEC A
AFDA  87                ADD A,A
AFDB  87                ADD A,A
AFDC  87                ADD A,A
AFDD  5F                LD E,A
AFDE  AF                XOR A
AFDF  32B2B0            LD (SONREQ),A
AFE2  57                LD D,A
AFE3  19                ADD HL,DE
AFE4  010800            LD BC,08
AFE7  11A9B0            LD DE,SONFRQ
AFEA  EDB0              LDIR
AFEC  1632              JR PROCESS
AFEE  3AB1B0    NONEW   LD A,(SONNOW)
AFF1  A7                AND A
AFF2  CA8EB0            JP Z,SONEX
AFF5  FE0A              CP 0AH
AFF7  E027              JR NZ,PROCESS
AFF9  1609              JP CNOIS
AFFB  3E0A      NOISE   LD A,0AH           ; Звук 10.
AFFD  32ACB0            LD (SONLEN),A
B000  AF                XOR A
B001  32B2B0            LD (SONREQ),A
B004  0630      CNOIS   LD B,30H
B006  CD90B0    GAIN    CALL RANDOM
B009  E610              AND 10H
B00B  D3FE              OUT (0FEH),A
B00D  0E02              LD C,02H
B00F  0D        MAKE    DEC C
B010  20FD              JR NZ,MAKE
B012  10F2              DJNZ GAIN
B014  21ACB0            LD HL,SONLEN
B017  35                DEC (HL)
B018  2074              JR NZ,SONEX
B01A  AF                XOR A
B01B  32B1B0            LD (SONNOW),A
B01E  166E              JR SONEX
B020  3AA9B0    PROCES  LD A, (SOFRQ)
B023  67                LD H,A
B024  3E10              LD A, 10H
B026  16FF              LD D,0FFH
B028  5C        SONLP   LD E,H
B029  D3FE              OUT (0FEH),A
B02B  EE10              XOR 10H
B02D  15        FREQ    DEC D
B02E  2805              JR Z,MOD
B030  1D                DEC E
B031  20FA              JR NZ,FREQ
B033  18F3              JR SONLP
B035  3AAAB0    MOD     LD A,(SONCFG)
B038  64                ADD H
B039  32A9BC            LD (SONFRQ),A
B03C  21AEB0            LD HL,SOKMOD
B03F  35                DEC (HL)
B040  C28EB0            JP NZ,SONEX
```

```
B043  21ACB0              LD HL, SONLEN
B046  35                  DEC (HL)
B047  2011                JR NZ,MODIFY
B049  AF                  XOR A
B04A  32B1B0              LD (SONNOW),A
B04D  3AB0B0              LD A,(SONNEX)
B050  A7                  AND A
B051  CA8EB0              JP Z,SONEX
B054  32B2B0              LD (SONREQ),A
B057  C38EB0              JF SONEX
B05A  3AAEB0    MODIFY    LD A,(SONRSF)
B05D  1F                  LD C,A
B05E  3AADB0              LD A,(SONTYP)
B061  A7                  AND A
B062  2820                JR Z,RESET        ; Треугольная
B064  3D                  DEC A
B065  2815                JR Z,TYP1
B067  3D                  DEC A
B068  E80A                JR Z,TYP2
B06A  3AAAB0    TYPOTH    LD A,(SONCFQ)     ; Пила.
B06D  ED44                NEG
B06F  32AAB0              LD (SONCFQ),A
B072  1614                JR MODE
B074  0C        TYP2      INC C             ; Двуст.восх.
B075  0C                  INC C
B076  79                  LD A,C
B077  32AEB0              LD (SONRSF),A
B07A  1606                JR RESET
B07C  0D        TYP1      DEC C             ; Двуст.нисх.
B07D  0D                  DEC C
B07E  79                  LD A,C
B07F  32AEB0              LD (SONRSF),A
B082  1600                JR RESET
B084  79        RESET     LD A,C
B085  32A9B0              LD (SONFRQ),A
B088  3AAFB0    MODE      LD A,(SONRND)
B08B  32ABB0              LD (SONMOD),A
B08E  FB        SONEX     EI
B08F  C9                  RET
B090  22A5B0    RANDOM    LD (RNHLST),HL
B093  2AA7B0              LD HL,(RNSEED)
B096  23                  INC HL
B097  7C                  LD A,H
B098  E603                AND 03
B09A  67                  LD H,A
B09B  32A7B0    ROK       LD (RNSEED),A
B09E  ED5F                LD A,R
B0A0  AE                  XOR (HL)
B0A1  2AA5B0              LD HL,(RNHLST)
B0A4  C9                  RET

B0A5  0000      RNHLST    DEFW 0000
B0A7  0010      RNSEED    DEFW 1000H
B0A9  00        SONFRQ    DEFB 00
B0AA  00        SONCFG    DEFB 00
B0AB  00        SONMOD    DEFB 00
B0AC  00        SONLEN    DEFB 00
B0AD  00        SONTYP    DEFB 00
B0AE  00        SOBRSF    DEFB 00
B0AF  00        SONRND    DEFB 00
B0B0  00        SONNEX    DEFB 00
B0B1  00        SONNOW    DEFB 00
B0B2  00        SONREQ    DEFB 00
B0B3  00        SONTAB    DEFS 256
B1B3                      END
```

СОВЕТЫ ЭКСПЕРТОВ

Сегодня в разделе, посвященном игровым программам, мы продолжаем печатать экспертные проработки нашего корреспондента из С.-Петербурга Фокина С.А., посвященные авиаимитаторам.

ATF

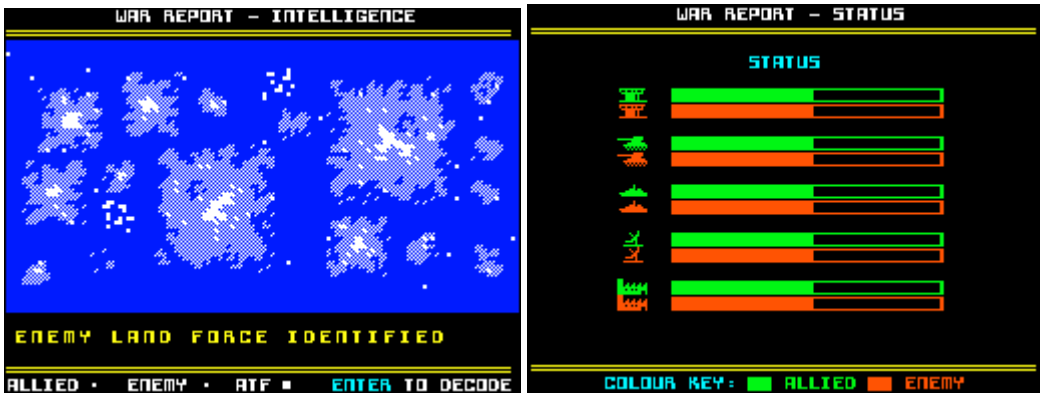
"Digital Integration" 1988

Эксперт Фокин С. А.
г. С.-Петербург

Глядя на название фирмы, сразу вспоминается такой серьезный имитатор воздушного сражения, как "TOMAHAWK". И "ATF" - это тоже имитатор воздушного боя, правда Вы будете управлять своим истребителем не из кабины, а как бы со стороны, но можете быть уверены, что игра от этого ничего не потеряла. Ваш тактический истребитель "ATF" будет летать над плавно перемещающейся поверхностью суши и океана, уничтожать вражеские перехватчики и производить пуски ракет по наземным и надводным целям, осуществлять посадки на автопилоте, пополнять боеприпасы и горючее и, в конечном счете, он должен выиграть сражение.

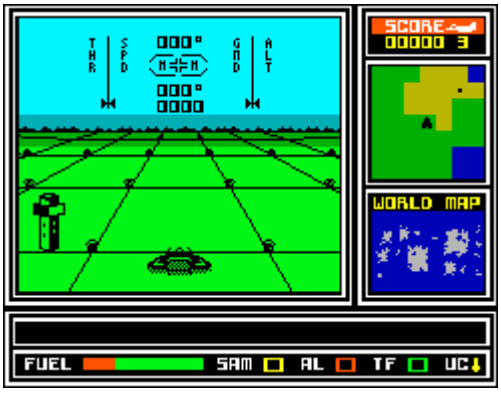
После загрузки программы на экране появится основное меню, в котором Вы можете выбрать тип управления, рейтинг пилота и звуковое сопровождение. Указатель перемещается курсорными клавишами (6 и 7), выбор пункта осуществляется клавишами "0"или "ПРОБЕЛ". После старта на экране появится карта боевых действии, на которой можно наблюдать расположение вашего истребителя, союзнических и вражеских объектов. Здесь же выводится сообщение о типе индентифицированных вражеских объектов, которое заносится в базу данных бортового компьютера самолета. Для продолжения - нажмите "ОГОНЬ". Затем на экране появятся результаты расчета соотношения сил борющихся сторон. Выход также осуществляется клавишей "ОГОНЬ". После этого Вы можете выбрать необходимое Вам оружие суммарной массой в пределах 6000 кг. Вам будут предоставлены магазины для скорострельной пушки, ракеты "воздух - воздух", управляемые ракеты "МЭЙВЕРИК" класса "воздух - земля" и, конечно, горючее. Выбор снаряжения производится клавишами "ВВЕРХ", "ВНИЗ", увеличение или уменьшение количества - клавишами "ВПРАВО" - "ВЛЕВО" в соответствии с выбранной системой управления игрой.

Оснащенный и заправленный самолет начинает выполнение полетного задания на посадочной площадке одной из Ваших баз.



На основной экране Вы видите 4 указателя: слева - обороты двигателя (THR) и датчик скорости (SPD), справа - расстояние до земли (GRD) и высота над уровнем моря (ALT). Между ними находятся индикатор прицела выбранной ракеты, над которым расположен указатель азимута курса истребителя, а под ним - указатели азимута и дистанции до выбранного объекта. Справа от основного экрана расположены два табло: на верхнем изображена окружающая обстановка, а нижнее табло мы рассмотрим более подробно. В исходном состоянии на нем изображена карта с указанием местоположения Вашего истребителя и выбранного объекта. Если Вы нажмете клавишу "C", то на табло появится выбранный бортовым компьютером объект. Переключение на объекты противника производится клавишей "D". Клавишей "E" можно пересмотреть все различные типы целей, а если их несколько, то все цели данного типа можно просмотреть клавишами "R" или "F". При дальнейшем нажатии на клавишу "C" на табло выводятся данные о запасах вооружений и боевой статус вашего истребителя. Темная полоса под экраном - это бегущая строка, на которую будут выдаваться текущие сообщения и предупреждения. А в самом низу слева находится индикатор горючего, справа от него расположен индикатор предупреждения о старте ракеты "земля - воздух" (SAM), который дублируется звуковым сигналом. Еще правее находится индикатор зоны действия базы (AL) (когда он мигает, можно предоставить исполнение посадки бортовому компьютеру).

Автоматизированная посадка задействуется клавишей "L". Следующий индикатор (TF) указывает на низковысотный режим полета, когда бортовая аппаратура ведет ваш истребитель и он плавно огибает рельеф поверхности земли. Этот режим включается клавишей "U".



Ознакомившись с органами управления самолета и с бортовыми приборами, Вы можете смело произвести взлет. Клавиши "Q" и "A" отвечают за обороты двигателя. Набрав необходимую скорость, можно произвести отрыв от земли. Вам доставит истинное удовольствие процесс полета в игре, но не увлекайтесь, ведь воздушное пространство заполнено перехватчиками врага, а с каждым попаданием двигатели теряют мощность и скорость падает.

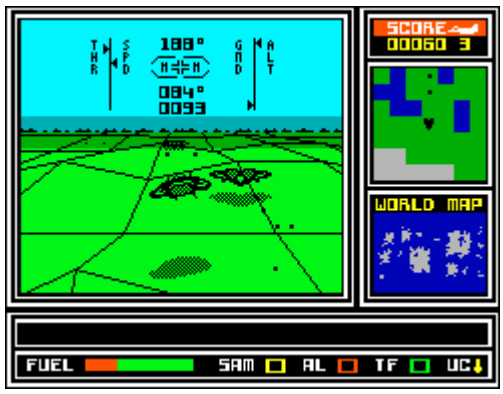
Итак, Вы выбрали наиболее близкую к Вам цель и на предельной скорости пошли на перехват. Время от времени индикатор "SAM" будет информировать Вас о наведении вражеской ракеты. Чтобы избежать катастрофы, вам нужно срочно, клавишей "J", запустить

генератор волновых помех. Когда до цели осталось меньше ста миль, Вы можете произвести пуск ракеты клавишей "M", но все же не рекомендуется стрелять ракетами с такого большого расстояния, т.к. эффективность попадания существенно снижается, и Вам придется пускать еще несколько ракет для полного уничтожения цели.

В момент пуска азимут цели и азимут курса должны совпадать, иначе попадание также будет не очень эффективно, если оно вообще будет.

Выбор типа ракеты выполняется клавишей "N", при этом на основном экране происходит смена формы прицела: с буквой "M" для ракеты " МЭЙВЕРИК", с буквой "A" для ракет "воздух - воздух". Для большей наглядности вы можете отключать сетку ландшафта клавишей "SYMBOL SHIFT".

Чтобы выиграть сражение, нужно как можно оперативнее уничтожать вражеские объекты, появляющиеся в базе данных компьютера. Если Вы уничтожили все объекты, не успокаивайтесь, т.к. через некоторое время будут идентифицированы новые, о чем будет выдано предупреждающее сообщение бегущей строкой. Не забывайте также, что горючее имеет тенденцию расходоваться, да и ракеты тоже. Поэтому время от времени Вам придется наводиться на ближайшую базу дружественных сил и, войдя в ее зону действия, предоставить посадку автопилоту. При посадке не следует забывать, что у истребителя есть выпускающееся шасси. После остановки самолета на экране появится информация об уничтоженных объектах, полученных в бою повреждениях истребителя, данные о его технических характеристиках и соотношении сил союзников и противника. Далее все продолжается по знакомой схеме: вы получаете информацию на карте боевых действия, проверяете соотношение сил борющихся сторон, пополняете боеприпасы и горючее и поднимаете свой самолет на задание. Вы достигнете победы, когда полностью уничтожите три каких-либо класса объектов. Если это Вам удастся сделать, Вы получите сообщение о победе.



В заключение хотелось бы отметить, что не надо полностью доверять автопилоту при полете на низкой высоте, т.к. иногда он не успевает отслеживать резкие изменения рельефа при максимальной скорости, и Вы можете задевать за вершины холмов. И еще: советуем не уклоняться от перехватчиков врага, а уничтожать их, т.к. чем больше Вы их уничтожите в начале игры, тем легче Вам будет в конце.

Желаем успеха!

Примечание ИНФОРКОМа: автор не указал, какой клавишей следует выпускать шасси самолета при посадке, а у нас, к сожалению, нет под рукой этой програмны, чтобы сделать проверку.

Начинающим пилотам мы рекомендуем проверить клавишу G (от слова GEAR). В большинстве испытанных нами имитаторов именно она выполняет эту функцию. Возможно, что после приземления следует затормозить самолет, обычно для этого служит клавиша B (BRAKES - ТОРМОЗА).

FLYER FOX

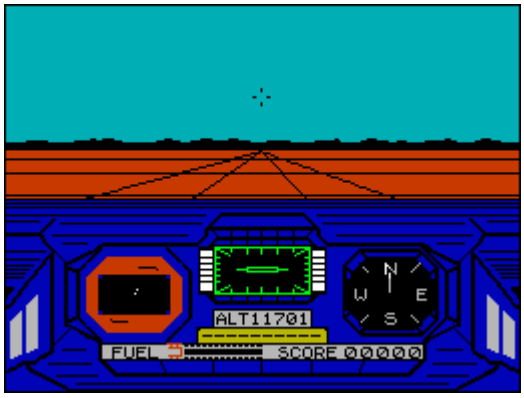
"Bug Byte", 1984

Эксперт Фокин С.А.
г. С.-Петербург

Эта игра хоть и относится к имитаторам воздушного сражения, но она настолько элементарна по своим задачам, что ее можно отнести к разряду обычных "стрелялок". Вам предстоит стать пилотом самолета сопровождения, которому поручено охранять авиалайнер от нападения истребителей врага. Все внимание вражеских летчиков приковано не к вам, а к авиалайнеру, но это не значит, что Вы можете сидеть сложа руки.

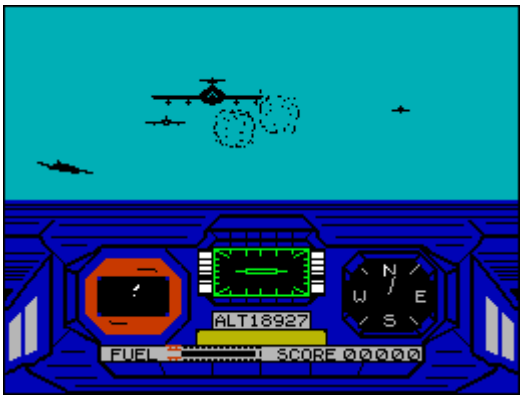
Когда программа загрузится, на экране появится заставка, в которой Вам необходимо выбрать свой вариант управления, а после этого нажать "ENTER".

Игра запускается с 1-го уровня сложности. Вы взлетаете и пристраиваетесь позади авиалайнера. Приборная панель очень проста: в центре панели вверху находится авиагоризонт, под ним - альтиметр, а еще ниже - указатель повреждений авиалайнера. Справа расположен компас с указателем курса, слева - экран радара. В самом низу находится указатель уровня топлива и счетчик очков.



Программа несложно управляется:
"Q","Z" - вверх, вниз;
"I","P" - влево, вправо;
от "V" до "M" - огонь;
от "H" до "ENTER" - пауза (снятие паузы производится любой клавишей управления);
от "A" до "G" - включение/ выключение звука
"W" - окончание игры.

Через некоторое время после начала полета компьютер выдаст звуковое предупреждение о приближающейся атаке. Ваша задача - сбить истребители как можно раньше, пока они не успели причинить повреждения авиалайнеру. Повреждения подразделяются на четыре категории: незначительные, серьезные, глобальные и окончательные. На приборной доске постоянно указывается характер текущих повреждений. Кроме того, Вам выдается звуковое сообщение о полученных повреждениях.



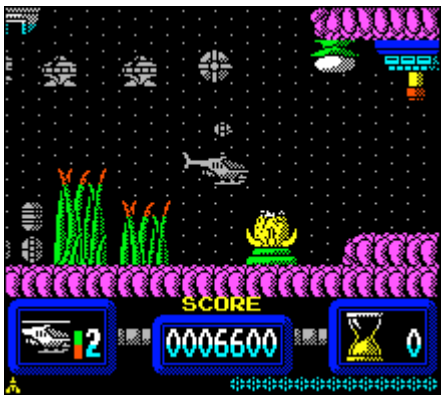
На первом уровне сложности самолетов врага не очень много и, чтобы перейти на следующий уровень, Вам надо сбить 4 самолета. На втором уровне самолетов противника уже больше и надо сбить 5 самолетов, чтобы перейти на третий уровень и т.д. За каждый сбитый самолет Вам насчитывается 100 очков. За переход на очередной уровень присваиваются дополнительные очки. На этом, собственно говоря, сюжет игры себя исчерпывает, и нам остается только порекомендовать эту программу младшим школьникам, поскольку несложная система управления не отвлекает детей от содержания игры.

COBRA FORCE

"Players", 1989

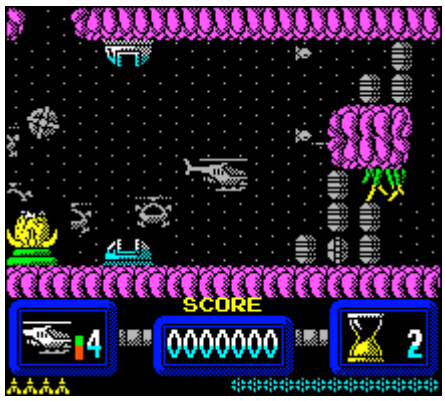
Эксперт Фокин С.А.
г. С.-Петербург

"COBRA FORCE" - типичный "боевик" с несколькими уровнями сложности. Пройдя удачно один уровень, Вы автоматически переноситесь в следующий. Так уж получилось, что самолеты и вертолеты - самые популярные боевые машины и вот в этой игре вам предстоит управлять вертолетом в чрезвычайно жесткой боевой обстановке. Мало того, что Вас встретит неимоверно плотный огонь противника, у вас будет очень мало свободного пространства. В общем, авторы игры немного перестарались и дойти до конца практически невозможно, если не... А вот что можно сделать, Вы узнаете в конце этого описания.



Итак, после загрузки на экране появится меню, в котором Вы можете сами назначить клавиши: "ОГОНЬ", "ПОДЪЕМ", "ВЛЕВО", "ВПРАВО" и "БОМБЫ". Затем, выбрав свой вариант управления, Вы переходите на первый уровень. Ваша задача (на всех уровнях) - расчистить плацдарм от наземных пусковых установок неприятеля, в то же время ваш вертолет подвергается непрерывным атакам с воздуха, со стороны летающих объектов, которые как быстро появляются, следуя друг за другом цепочкой, так же быстро и исчезают.

В левом нижнем углу экрана, рядом с изображением вертолета, указано количество "жизней" и текущий уровень энергии. В правом нижнем углу - количество бомб блокирующего действия (на некоторое время останавливаются все объекты на экране, кроме Вашего вертолета). В самом низу экрана расположены два индикатора: слева - количество ракет, которыми нужно уничтожать пусковые установки, Справа - количество оставшихся в этой зоне контейнеров, которые нужно собрать.



Пусковые установки можно уничтожить только ракетами. Чтобы их выпустить, необходимо нажать клавишу "ОГОНЬ" и задержать ее чуть дольше, чем это необходимо для простого огня из пулемета. Запасы ракет можно периодически пополнять. Для этого необходимо перехватить "звездочку", которая иногда попадается в строю нападающих на Вас объектов. После попадания, она превращается в ящик с боеприпасами, и Вам нужно успеть его подхватить.

Заканчивается расчистка плацдарма появлением двух вертолетов противника. Их можно уничтожить как ракетами, так и из пулемета. Только после этого зона считается расчищенной, и Вы переходите на очередной уровень. Уровни отличаются друг от друга только графикой, которая (надо отдать должное автору программы) становится все интереснее.

На этом можно закончить общее описание игры, но мы обещали дать одну небольшую хитрость, которая поможет Вам в этой тяжелой битве. Итак, если в самом начале, при переназначении клавиш, Вы наберете имя автора программы "SIMON", то получите неограниченную жизнь. Но, даже с такой поддержкой, не всякому хватит терпения довести игру до конца.

Желаем удачи!

THUNDER BLADE

U.S. GOLD, 1988

Эксперт Фокин С.А.
г. С.-Петербург

Многоуровневая (восемь уровней сложности) игра "THUNDER BLADE" -это типичный представитель жанра ACTION ("боевик") с прекрасной графикой и неожиданными поворотами сценария.

Небольшое, но весьма враждебное государство готовится нанести ядерные удары по нашей территории и спровоцировать третью мировую войну. Необходимо разрушить его стратегические объекты до того, как ракеты покинут пусковые шахты. Все попытки наших специальных подразделений по диверсионной работе потерпели неудачу. Командование приняло решение осуществить операцию по ликвидации вражеских объектов с помощью эскадрильи штурмовых вертолетов. Тем, кто смотрел такие фильмы, как "THUNDERBOLT-1" и "THUNDERBOLT-2", не надо объяснять, на что способен современный боевой вертолет огневой поддержки. Если же вы их не смотрели, то не отчаивайтесь, перед Вами игра, где Вы примете самое непосредственное участие в сценарии.

Вам предстоит занять место первого пилота (портрет справа). Можете взять себе напарника оператора управления оружием (портрет слева), тогда он будет нажимать клавишу "ОГОНЬ", но можете действовать и в одиночку. Ваш вертолет вооружен ракетами и шестиствольной пушкой, что вполне достаточно для успешного выполнения задания. Всего для выполнения задания у Вас имеется 6 вертолетов (5 силуэтов внизу и сам действующий вертолет на экране); совсем немного, учитывая плотность огня противника.

После загрузки нескольких первых блоков игры, ввод с магнитофона приостанавливается и выдается запрос на тип игры:

1 - CHEAT MODE.

2 - NORMAL MODE.

Это меню ввели не авторы программы, а Билл Гилберт, "сломавший" программу. В "хитром" варианте CHEAT MODE Вы имеете бесконечное число вертолетов.

Настоятельно рекомендуем пока им не пользоваться, а нажать клавишу "2" и продолжить нормальную загрузку. Для начала поиграйте в авторском варианте, чтобы

прочувствовать всю дьявольскую задумку создателей игры. Вам вряд ли удастся пройти хотя бы пару первых уровней. После этого, вероятно, Вы перезагрузите игру и отдадите должное хаккеру Биллу Гилберту, предусмотревшему упрощенный вариант. При бесконечном числе вертолетов у Вас появится шанс угробить пару сотен из них, но в конце концов все же успешно выполнить миссию.

Самое сложное в этой игре, как ни странно - освоить управление. В начале игры компьютер запросит тип управления. Если Вы выбираете клавиатуру ("1"), то управление следущее:

"O" - влево; "P" - вправо;

"Q" - вниз; "A" - вверх;

"SPACE" и "CAPS SHIFT" - огонь.

Чтобы войти в игру, достаточно нажать "ОГОНЬ". Вы окажетесь в городе, заполоненной боевой техникой неприятеля: самоходными зенитными установками, которые ведут по Вам прицельный огонь. Иногда между зданиями появляются патрульные вертолеты и они тоже стремятся вас сбить.

Вы можете набрать высоту (клавиша "ВВЕРХ") или опуститься вниз (клавиша "ВНИЗ"), а также перемещаться влево и вправо. Однако, чтобы набрать скорость, надо одновременно со снижением нажать клавишу "ОГОНЬ", естественно, перед этим набрав высоту. Чтобы снизить скорость или вовсе зависнуть на одном месте, необходимо нажать "ОГОНЬ" одновременно с набором высоты.

Если Вы выбираете в качестве органа управления джойстик, то компьютер выдаст сообщение:

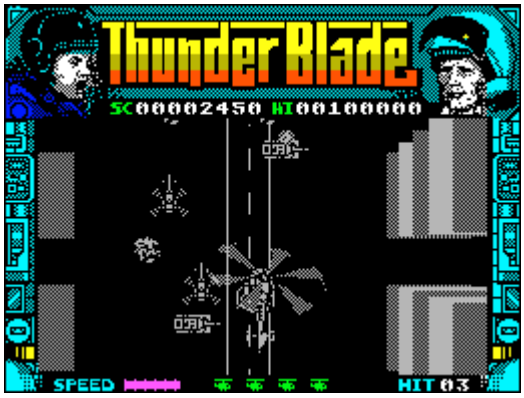
A - управление скоростью джойстиком;

B - управление скоростью от клавиатуры.

Если Вы нажимаете "A", то управление аналогично работе с клавиатурой. Во втором случае скорость вертолета можно изменять клавишами:

E - увеличение;

D - уменьшение.



В этой игре Вам предстоит уничтожить 4 стратегических объекта: атомный крейсер, пусковой комплекс, взлетающий космический корабль и командный пункт. Чтобы добраться до каждого из них, Вам предстоит преодолеть полосу обороны, которая воздвигнута перед этим объектом. Полоса перед крейсером - город, следующая - каньон, далее - каналы с боевыми катерами, и наконец, перед командным пунктом - опять город.

В начале описания мы не зря упомянули о неожиданных поворотах сценария игры. В каждой полосе обороны вид сверху неожиданно сменяется объемным изображением, хотя управление остается прежним. Преодолев полосу обороны и, оказавшись над стратегическим объектом, не спешите радоваться - он также хорошо прикрыт от налетов с воздуха. Многочисленные пусковые установки можно уничтожить, ведя непрерывный огонь, когда они показываются перед Вами. Здесь управление меняется: скорость набрать невозможно, она постоянна, а клавиши "ВВЕРХ" и "ВНИЗ" служат не для изненения высоты, а для маневра по вертикали. Впрочем, следует учесть, что вернуться назад можно недалеко.

Когда вы достигнете своей цели - командного пункта противника, то вертолет вообще не движется вперед, а может лишь уворачиваться от шквального заградительного огня. Не стоит пугаться - смело расстреливайте пусковые башни из всех видов оружия и командный пункт вскоре будет уничтожен.

Хочется надеяться, что вы долго будете вспоминать эту прекрасную игру.

В заключении приведем некоторые полезные советы.

1. Столкновение со зданиями в городе смертельно опасно, но некоторые из них можно пролететь, набрав максимальную высоту.

2. Не пытайтесь поразить все цели, это приведет к неоправданным потерям, помните, что ваша главная задача - преодолеть полосу обороны.

3. Т.к. огонь зенитных средств сосредотачивается на вашем курсе, то старайтесь лететь не по прямой, а меняя направление полета.

4. После потери очередного вертолета, скорость приходится набирать заново.

5. Появляющиеся над равнинной местностью эскадрильи самолетов почти не причиняют вреда, чего не скажешь о патрульных вертолетах.

6. При полете над стратегическими объектами также не старайтесь поразить все пусковые установки, помните, что Ваше главное оружие маневр.

Желаем удачи!

SANXION

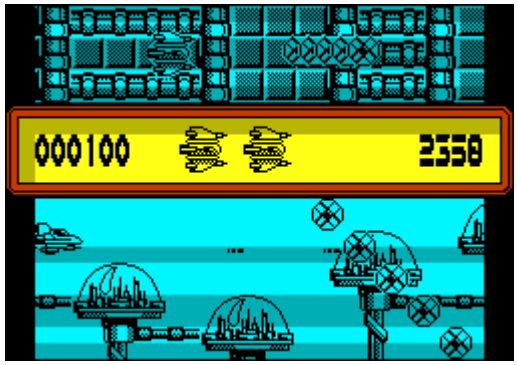
"Chalamus" 1989

Эксперт Фокин С.А.
г. С.-Петербург

Игра "SANXION" - типичная "стрелялка" с неплохой графикой. Космический крейсер "Темное пламя", находящийся на станционарной орбите над неизвестной планетой, доставил на нее три хорошо вооруженных атмосферных разведчика типа "Москит", управляемых самыми опытными пилотами. Им предстоит совершить разведывательный полет над поверхностью планеты, населенной враждебно настроенной цивилизацией негуманоидного типа.

Технические устройства злобных аборигенов атакуют все, что появится в пределах их видимости. Беспилотный разведчик землян, перед тем как был сбит, успел передать на крейсер изображения странных поселений под прозрачными куполами и механизмов, летающих между ними. Дальнейшие попытки вступить в контакт с инопланетянами успеха не имели. Тогда капитан и отдал приказ послать на планету эскадрилью "Москитов". Одним из пилотов взяли Вас.

При запуске игра представляет широкий выбор типа управления. Если нажать клавишу "2" (клавиатура), то дается возможность самому назначить все клавиши управления - вверх, вниз, влево, вправо и огонь. Игра имеет очень простую систему управления. Для старта достаточно нажать "1".



Вверху экрана есть полезное приспособление - радар. Он дает вид сверху и показывает врагов вокруг вашего "Москита" еще тогда, когда их нет на экране. Советы играющему просты: в игре приходится надеяться не столько на свою огневую мощь, сколько на высокую маневренность. Ведь некоторые враги ведут по вам такой интенсивный огонь, от которого можно только уйти, а уничтожаются они не с одного выстрела. Следует стараться держаться середины экрана, т.к. нередко атаки следуют сзади или сверху.

От строя черных шаров, летящих зигзагом, можно просто уйти, поднырнув под них (они не ведут огня, а уничтожить их можно только при многократном попадании), некоторых врагов не обязательно полностью истреблять: покружив вокруг Вас и постреляв, они полетят дальше. Других приходится полностью уничтожать, иначе они от Вас не отстанут.

Следует помнить, что смертельно опасно не только попадание вражеского снаряда, но и само столкновение с врагом. Есть интересная особенность: если уничтожить в третьей волне врагов пару верхних или нижних и стать на их место, примерно на одной вертикали с оставшимися, то можно лететь так довольно долго. Вы будете недоступны для снарядов,

которые летят под углом 45 градусов. Однако надо не прозевать тот момент, когда они все-таки от Вас отстанут, метко выпустив в Вас пару снарядов на прощание.

Нам кажется, что эта игра доставит Вам много удовольствия. С каждой новой попыткой Вы будете прорываться все дальше и дальше, изучая повадки врагов. Правда, возникает мысль, что неплохо бы увеличить число самолетов раз эдак в пять.

AIR WOLF

"Elite" 1986

Эксперт Фокин С.А.
г. С.-Петербург

Игра "AIR WOLF" ("Воздушный Волк") по жанру относится к типу "ACTION", требует молниеносной реакции и фантастической точности.

В далеком Тибете, среди недоступных гор, была обнаружена база инопланетян, построенная ими при посещении Земли много тысячелетий тому назад. По всей базе то тут, то там спрятаны информационные карты, содержащие технические знания, неизвестные землянам. Каждая уважающая себя держава посылает экспедицию за экспедицией, чтобы завладеть этими бесценными сокровищами. Однако любому здравомыслящему человеку понятно, что ставший обладателем этих карт получит опасное преимущество перед остальным человечеством и сможет использовать его в злых целях. Узнав об этом, свихнувшийся миллиардер, помешанный на идее облагодетельствования всех на свете, решает нанять опытного пилота, который бы на маленьком, но хорошо вооруженном вертолете проник бы на базу и уничтожил бы все карты с опасными знаниями. Пусть уж лучше они не достанутся никому, чем поставят все человечество на край гибели. Вот этим пилотом Вы и станете.[1]

Сложность заключается в том, что хотя самих инопланетян на базе и нет, они оборудовали ее всевозможными ловушками и устройствами, открывающими огонь по любой приближающейся цели. Во время игры Вы встретитесь с силовыми полями и радарами, излучающими неизвестную смертоносную энергию, с электрическими разрядниками, с лазерными установками и еще со многим, что неизвестно землянам, но несет гибель Вашему вертолету.

Управление игрой простое:
клавиши от "Q" до "T" - вверх,
от "A" до "G" - вниз,
"X" - вправо,
"Z" влево,

[1] AIRWOLF™ Scenario: As Stringfellow Hawke, a former Vietnam chopper pilot, and the only man in the free world trained to fly the billion-dollar helicopter, 'AIRWOLF' you have been assigned a dangerous rescue mission by the FIRM. Five important U.S. scientists are being held hostage deep in a subterranean base beneath the scorching Arizona desert. As Hawke, you must guide AIRWOLF using full stealth capabilities, on a series of perilous night-time missions and bring about the release of each scientist in turn. Only destruction of the defense control boxes strategically positioned within the cavern will allow AIRWOLF to descend to the heart of the base where the scientists are held. (Прим. OCR)

"C" - огонь.

Миссия начинается с маленького аэродрома в горах. Посадочная площадка на этой картинке - единственное место, где Вы можете беспрепятственно посадить вертолет. Во всех остальных местах, при попытке сесть он разбивается о скалы. Поэтому надо постоянно поддерживать его на лету. В пещерах он вообще не может касаться никаких стен и предметов.

На втором экране вас поджидает силовое поле, появляющееся прямо перед вертолетом. В нем можно пробить проход, непрерывно стреляя из пушки. Но следует спешить: через некоторое время генератор полностью его восстанавливает.

На следующем экране такое же поле перекрывает путь вашему вертолету в пещеру, где и находится база. Самый простой способ преодолеть его - это снизиться в предыдущей картинке почти до самой поверхности и лететь в экран с силовым полем. Оно появляется быстро, но не мгновенно (подвела инопланетян их техника). Учитывая это, необходимо, оказавшись над провалом, немедленно начать снижение, тогда, едва не коснувшись поля, Вы попадете в очередной экран.

Здесь установлены два пульсара, время от времени излучающие потоки энергии неизвестной природы. Летите вправо, и Вы наконец найдете первую из информационных карт (по виду она напоминает квадрат с крестом). Смело расстреляйте ее, и Вам добавят 75 очков, а в картинке с пульсарами появится проход в скале. Миновав его, Вы попадете в экран с двумя электрическими разрядниками. Чтобы преодолеть их, Вам понадобится незаурядная реакция и расчет. В следующем экране выстрелами проложите себе путь в полу пещеры и, избегнув снарядов автоматической пушки и электрическую дугу (опыт прохождения уже есть), попадете в следующий экран с инопланетной картой. Она защищена двумя излучателями антиматерии, но мы думаем, что Вы без труда уничтожите ее и вернетесь назад.

Теперь направо и, избегнув снарядов, проскочите в следующее помещение. Здесь вас уже поджидают старые знакомые: автоматическая пушка, электрическая дуга, а также - очередная карта.

Преодолейте заслон и Вы в следующем помещении. Но что это? Здесь злокозненные инопланетяне установили башню с чудовищным лазером, который уничтожает все вокруг себя! Но хочется надеяться, что если Вы добрались до этого помещения, то сумеете преодолеть эту преграду.

Больше нет смысла утомлять Вас описанием всех картинок, придуманных изощренным умом программиста, а лучше привести несколько полезных советов. Вертолет может стрелять не только прямо, но и под углом. Для стрельбы наискось вверх необходимо, чтобы вертолет находился в свободном падении (ни одна из клавиш не нажата), тогда он задирает нос, и можно вести огонь по диагонали. Кстати, только таким способом можно попасть в первую карту. Наискось вниз он стреляет, если лететь вперед и вниз - тогда нос опускается.

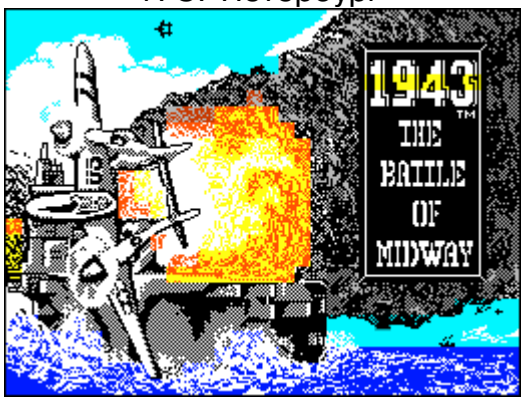
Вам не пройти далеко, если вы не научитесь стрелять вниз. Для этого необходимо повернуть вертолет в противоположную сторону, но вовремя отпустить клавишу, тогда вертолет станет "в фас". Кстати, такое положение полезно для проникновения в узкие щели. Хочется надеяться, что Вы приятно проведете время за этой игрой, если не повредитесь рассудком от неудач, ведь для прохождения всей игры у Вас всего 5 вертолетов.

Успеха Вам!

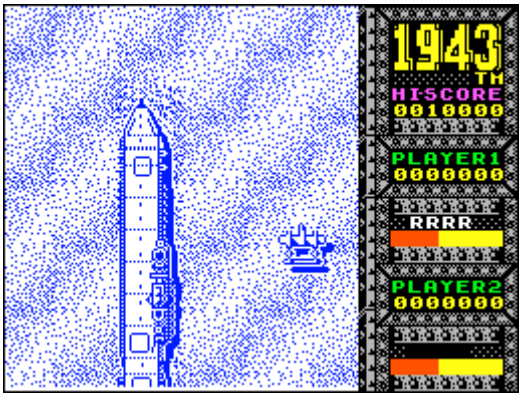
1943 THE BATTLE OF MIDWAY

"Capcom", 1986

Эксперт Фокин С.А.
г. С.-Петербург

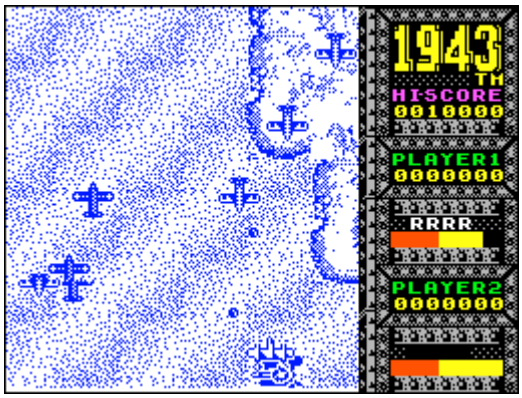


В этой игре, которую можно отнести к разряду обычных "стрелялок", Вам предстоит перенестись в годы Второй Мировой войны и вступить в бой с самолетами противника, а также с его военно-морскими силами. Вы будете управлять небольшим, но грозным истребителем, который подвергается нападениям вражеских истребителей, бомбардировщиков и обстреливается кораблями противника. Вся графика игры соответствует стилю рассматриваемой эпохи.



После загрузки программы нажмите любую клавишу, и внизу экрана появится бегущая строка. Вы можете выбрать свой тип джойстика или переназначить клавиатуру, а затем, в зависимости от того, сколько будет играющих, нажмите "1" или "2". Появится надпись, предупреждающая о готовности и Вы начнете свою миссию.

Игра разбита на несколько этапов, после прохождения которых она начинается сначала, но количество атакующих и интенсивность огня возрастают. Чтобы пройти этап и перейти к следующему, нужно пролететь определенный участок суши. Впрочем, это может быть и поверхность океана. Иногда на экране появляется большой четырехмоторный бомбардировщик, который будет стрелять в Вас до тех пор, пока Вы не повредите ему все четыре двигателя. Иногда, после удачного истребления нескольких самолетов, Вам будут попадаться контейнеры с надписью "POW". Если его перехватить, то Ваша энергия увеличится (индикатор энергии находится справа в средней части экрана), вы можете не брать этот контейнер, а расстрелять его. В этом случае вы получите новый, более мощный тип вооружения. Каждое попадание в контейнер "POW" приводит к переключению на другой тип оружия. Поражение вражеским снарядом вызывает утрату того, что вы приобрели таким путем.



Несмотря на то, что сюжет игры очень прост, в ней есть та изюминка, которая заставит Вас начинать все сначала и стрелять, стрелять, стрелять...

Желаем успеха!

* * *

Военными имитаторами и стратегическими играми интересуются многие наши читатели, но в то же время мы хорошо знаем, что среди них добрую половину составляют военнослужащие. Более того, "Синклер" в армии имеет популярность по-видимому большую, чем в других социальных группах. И для военных такие программы представляют не праздный, а профессиональный интерес. Сегодня у нас есть для них приятное сообщение. Редакционная коллегия журнала для военных профессионалов "Военный вестник" приняла решение о широкой поддержке пользователей компьютеров типа "ZX-Spectrum" в Вооруженных Силах. Первые публикации уже появились. Если Вам есть что сказать о возможностях использования Вашего компьютера в системе боевой подготовки, страницы журнала открыты для Вас. Ваши идеи, решения, разработки и программы будут опубликованы.

Это могут быть тренажеры, системы автоматического контроля уровня знаний, экспертные системы и многое-многое другое.

Кроме того, в редакционный портфель журнала входят интересные статьи, посвященные общетактическим и стратегическим проблемам. Любители компьютерных игр найдут здесь неисчерпаемый источник идей для создания собственных военно-стратегических программ.

Так, например, анализ операций типа "Буря в пустыне" - готовый план будущей игры. Разбор тактических целей наступающей и оборонявшейся сторон и анализ действий инженерных средств поддержки послужит для этого отличной базой.

Те, кто любят стратегические игры, знают, что большой интерес в них представляют вопросы тылового обеспечения. Ведь именно здесь военная стратегия сливается со стратегией деловой (STRATEGIC MANAGEMENT). Практически в каждом номере журнала вы найдете готовые алгоритмы для расчетов проблем, связанных со снабжением, пополнением, боепитанием, расходом сил и средств на прорыв обороны. Эти алгоритмы только ждут, чтобы их перевели на язык программирования и оформили в виде увлекательной игры.

Любители военной истории, военной техники, стрелкового оружия тоже найдут для себя много интересного. И, наконец, в последние годы лицо журнала представляют интересные и откровенные интервью с профессионалами. Такие интервью нечасто встретишь на страницах прочих газет и журналов даже в нашу эпоху всеобщей гласности.

Если у вас есть собственные идеи и разработки, редакция ждет ваших писем по адресу:

103175, Москва, Мясницкая ул., д.41, редакция журнала "Военный Вестник" Лушников Александр Васильевич, тел. (095) 896-79-49.

Желающие подписаться на "Военный Вестник", могут это сделать на почте. Индекс издания на 1993 год - 70140.

THE DARK WHEEL

ГЛАВА 3.

Если Вы хотите посетить кладбище на Тионисле, то лучше всего подлетать к нему со стороны солнца. Это и удобно и безопасно, поскольку политическая система Тионислы - Демократия и пираты здесь встречаются крайне редко. Тионисла выглядит из космоса ярко желтой планетой, а кладбище всегда расположено между планетой и звездой.

Первое, что Вы видите, когда подлетаете к нему - это серебряный диск и два спиральных рукава маленьких сверкающих точек. Это галактика в миниатюре. То же медленное вращение, тот же яркий блеск центральной части (именно здесь сосредоточены наиболее внушительные мемориалы).

Подлетев ближе, Вы увидите, что звезды в этой галактике вовсе и не звезды, а маркеры - внушительные металлические блоки, расписанные надписями на тысячах языков и символами тысяч религий. Это причудливое и волнующее зрелище. Маркеры редко бывают меньше, чем по тысяче футов в поперечнике. Среди них есть и хромированные кресты и титановые звезды Давида, а также всевозможные символы многочисленных миров и различных религий - порождения разума, которому довелось почить в этом необычном для космических путешественников месте.

У нижней границы этого необычного мавзолея расположена додекаэдрическая конструкция космической станции. Эта станция класса "Додо" - место жительства администрации и охраны некрополиса. Отсюда начинается путешествие по кладбищу, здесь служба безопасности проверяет Ваши документы и гостевую визу. Пока движется очередь прибывших на регистрацию, Вы можете любоваться замечательным зрелищем сквозь прозрачные стены и потолки станции. Здесь и там разбросаны разбитые и помятые корабли всех времен и народов. Ко многим пришвартована усыпальница, ставшая последним приютом для космического путешественника.

Существует масса причин для посещения кладбища на Тионисле. Многих притягивают те умопомрачительные сокровища, которые скрываются в недрах кораблей и саркофагов. Может быть и в этом черном кубе, порождении неземного разума, сделанном из непонятного металла скрываются вековые сокровища внеземных цивилизаций. Эх, если бы знать, как и на какую панель этого парящего саркофага нужно надавить, чтобы получить доступ к этим богатствам.

Но чаще включаются системы защиты. Это может быть скрытый лазер, а может быть робот-охранник с острыми, как бритва лезвиями вместо рук. Вас может всосать гиперпространственный вакуум и выбросить в другом измерении или в другой эпохе. И Вы, как и все прочие посетители, очень осторожно пробираетесь среди этих орбитальных руин, чтобы ничего не задеть и ни к чему не прикоснуться. Те, кто здесь похоронен, люди они или инопланетяне, так или иначе были очень и очень богаты. У них хватило средств приобрести здесь место последней стоянки для избранных и, конечно же, у них более, чем достало средств сделать свое убежище неприкосновенным.

Формальности завершены, новенькая, только что полученная пилотская лицензия проверена, туристский корабль довольно необычной формы выдан и Алекс готов к посещению кладбища.

Он поспешно отлетел от станции и, сверяясь с планом, начал разыскивать могилу Флейшера.

Он отыскал ее довольно быстро. Кем бы Флейшер ни был при жизни, он был чудовищно эгоцентричен. Его надгробием стала огромная кристаллическая структура, похожая на одуванчик. Каждая игла в этой конструкции сверкала, как чистый бриллиант и имела буквально сотни футов в длину. Его тело, облаченное в красную униформу бойца класса "Элита" парило в стасисе в самом центре этой конструкции, освещенное сфокусированными лучами солнца.

Рядом, у простого монумента, покоился пришвартованный корабль "Кобра". Все

жизненно необходимое оборудование: топливозаборники, боевые ракеты, грузовые отсеки, лазерные надстройки - было демонтировано.

Алекс внимательно пригляделся к кораблю. Он не имел ничего общего с той "Коброй", которая уничтожила отцовский корабль. Та имела все мыслимое вооружение, которое только можно купить за деньги.

Алексу показалось, что на корабле что-то мигнуло. Приглядевшись, он убедился, что это не обман зрения. Действительно, красный фонарик корабля сигналил кодом: "LAND ON DOR PL".

"Посадка на платформе" - легко расшифровал код Алекс. Сманеврировав легким челноком, он быстро подошел к "Кобре", пришвартовался на выгоревшей надстройке и виновато огляделся. Правила, действующие здесь, были очень суровы. Не то что швартоваться, но и прикасаться к любому сооружению на кладбище было смертельно опасно.

Пространство патрулировалось "Крейтами" службы безопасности, имевшими приказ расстреливать любого, независимо от пола и возраста, кто будет застигнут при попытке проникновения в гробницу.

К счастью, кладбище было слишком большим и тень многочисленных монументов создавала в этом городе мертвых достаточно безопасных мест для того, чтобы скрыться живым.

Входной люк открылся и зеленая лампочка призывно промигала: "Заходи". Алекс ввел свой туристический челнок в трюмное пространство и, когда загорелся сигнал "давление в норме", вышел из него и направился в рубку управления. Он открыл раздвижную дверь и на миг прищурился от яркого света приборов и панелей. Перед ним на широком экране сверкало изображение памятника Флейшеру.

Темным пятном на фоне яркого экрана вырисовывался силуэт мужчины, облаченного в космический костюм. Одна рука лежала на навигационной консоли, а другая - на кнопке боевого лазера.

- Я на борту, - сказал Алекс и подошел к молчаливому пилоту. Тот не сделал ни движения, не произнес ни слова.

На какое-то мгновение Алекс застыл, уставившись в экран, глядя на медленно перемещающиеся монументы и на сверкающие в черной пустоте звезды, а затем повернулся к хозяину корабля.

То, что он испытал, было похоже на шок и заставило отшатнуться. Перед ним было высохшее, мумифицированное лицо трупа. Оно смотрело на Алекса из под шлема и казалось, что высохшие губы растянулись в широкой улыбке.

- Как ты думаешь, стоит нам брать его с собой? - раздался голос откуда-то из-за спины.

Алекс удивленно обернулся и увидел фигуру, выходящую из тени.

- Как талисман на счастье.

Алекс попытался улыбнуться, но расслабиться ему не удалось. Слишком все это было быстро и неожиданно. Он как будто прирос к полу и смотрел на приближающуюся женскую фигуру.

Она была невелика ростом. Ее глаза были темными, а кожа имела оливковый оттенок. Одетая в светло-зеленые одеяния, она, кажется, утонула в них. Прикосновение ее руки было холодным и уверенным. На некоторое время она задержала свою руку в руке Алекса, глядя ему прямо в глаза и обезоруживающе улыбаясь.

- Итак, Рейф выбрал тебя? Хорошо, Алекс, во всяком случае путешествие с тобой будет по-крайней мере тихим. А у тебя вообще-то есть речевые функции? - с этими словами она шутливо развернула Алекса и похлопала по спине в поисках выключателя. - Или ты из старых моделей, которые умеют только жестикулировать?

- Простите, - сказал Алекс, - все это было так неожиданно для меня.

- О, боже, - сказала женщина - Где ты выключаешься? Мне кажется, молчащим ты был лучше.

- Кто ты? - спросил Алекс, слегка раздосадованный этим легкомыслием. Сейчас ему

больше всего хотелось бы знать, зачем Рейф Зеттер послал его сюда и куда он сам подевался?

- Я принадлежу к торговому союзу "Филдс"! - ответила она и в салюте приложила ребро правой ладони к левому плечу. - А зовут меня Элиссия Филдс. Имя несколько необычное, но ему я обязана своей приемной матери, которая в девятилетнем возрасте, когда инкубировала мой клон, увлеклась древнегреческой мифологией.

"Древнегреческая мифология. Инкубация клонов." - для Алекса это означало, что Элиссия происходит родом с планеты Теорг. Он напряг память, вспоминая все, что ему известно об этой планете.

Теорг. Обитаемая планета. Первое поселение основано двумя колонистскими кораблями. Экипажи кораблей приняли от местных жителей систему размножения посредством клонирования избранных особей. Все прочие - уничтожаются. На много столетий Теорг оказался отрезанным от общих путей развития цивилизации, от коммерции и торговли. По всей видимости, Элиссия Филдс - изгнанница.

- Меня зовут Алекс Райдер, - сказал юноша.

Я знаю, - ответила женщина и бросила на него взгляд, которым, казалось, можно было припечатать. Затем она похлопала по плечу тело, сидящее в кресле. - А это есть... или, вернее, был Генри Белл, космический торговец. Нам придется позаимствовать его гроб, хотя он, кажется, очень и очень этим недоволен. Это ржавое ведро было набито его трехмерными голограммами, которые рассказывали о том, как плохо будет тому, кто войдет сюда без разрешения. Большинство из них я повыключала, но возможно, что где-то что-то и осталось.

- Мы похитим этот корабль? - мягко спросил Алекс, изучая приборную панель. Топлива, судя по приборам, на борту было всего лишь на переход в 0,1 св. года. Этого явно недостаточно для того, чтобы покинуть систему Тионислы.

- Если ты предпочитаешь остаться здесь, то пожалуйста. Мы сможем ухаживать за могилкой, вырастим здесь цветочки и будем много-много разговаривать.

- Я имел в виду... как ты собираешься выбраться отсюда? - он уставился на инопланетянку. Кажется чувство постоянной душевной боли и горечи последних дней немного поутихло. Его явно интересовала эта девушка. - И почему, в конце концов, ты мне помогаешь? Где Рейф?

С отрывистым смехом Элиссия ответила.

- С Рейфом никогда ничего не поймешь. Ты можешь улететь на другой конец галактики, а он там уже тебя поджидает. Вернее, не он сам, а его трехмерная голограмма. А вот где он на самом деле, это нам еще предстоит выяснить. Что же касается помощи, то кто здесь сказал, что я тебе помогаю? Может быть, это ты помогаешь мне? А скорее мы помогаем друг другу. Ты собираешься отомстить за своего отца, мне тоже надо кое с кем, кое за что рассчитаться. Может быть, когда-нибудь я и расскажу тебе об этом. Во всяком случае, ты мне нужен, без тебя я не смогу вести этот корабль.

Алекс удивленно возразил - Но ведь "Кобра" управляется одним пилотом?"

- Да знаю я. И вообще я могу вести ее с закрытыми глазами. Не для этого ты мне нужен. Дело в том, что я с Теорга и потому мне здесь нечего делать. Мое лицо мгновенно вызовет массу подозрений. Ты нужен мне в качестве прикрытия, нам ведь придется много общаться с торговыми базами и официальными властями. Эта посудина совершенно безоружна и любой пират собьет ее с помощью палки для добычи бананов. Нам нужны защитные поля, ракеты, грузовые отсеки. Все это надо доставать, они ведь не растут на деревьях.

- Предлагаешь заняться торговлей? - спросил Алекс и мысль о длинной череде поколений, отдавших жизнь этому делу промелькнула перед ним.

Элиссия была права. Нечего и думать об охоте на "Кобру" без совершенного снаряжения, а сколько еще пройдет времени, пока будут закончены все юридические формальности с введением в права наследования, особенно если учесть, при каких странных обстоятельствах погиб отец.

Он чувствовал себя разрывающимся на части. Одна его часть хотела убить врага

прямо сейчас. Ему не терпелось вырваться на межзвездные трассы и броситься в погоню за убийцей. Другая часть подсказывала ему, что настоящий охотник должен иметь терпение. Поспешные шаги ничего не дадут, кроме провала дела. Нужен тщательно продуманный и спланированный подход, а хорошо вооруженный корабль - необходимый элемент всего предприятия.

- Все, что у меня есть - сотня кредов, - сказал Алекс, имея в виду ту сумму, которую ему выдали из страховки для возвращения домой.

- Это для начала. С этими деньгами мы начнем нашу торговую карьеру. - Ее лицо помрачнело, а в глазах засверкали отсверки огоньков приборной панели. - А затем мы полетим в такое место, о котором знает только Рейф Зеттер и устроим хорошую стрельбу. Мы разделаемся с кораблем, который убил твоего отца. Есть еще многое, за что он должен ответить....

Больше она ничего не сказала.

Для каждого, кто собрался заниматься космической торговлей, первой и самой трудной задачей является приобретение корабля. Конечно, вокруг каждой планетной системы есть свалки устаревших кораблей, проводятся и аукционы подержанных судов. Многие крупные компании нанимают вторых пилотов с гарантией расплаты через четыре года кораблем, если, конечно, новоиспеченный пилот останется к этому времени еще жив.

Но корабли очень и очень дороги, даже если приобретать их на свалке.

Алекса несколько смущала необходимость кражи этой посудины, но он не мог не оценить тех трудов, которые затратила смуглая беглянка, приводя в порядок корабль и запасая по каплям топливо и еду в количестве, достаточном для небольшого гиперпрыжка. В принципе все было готово, ей только недоставало партнера, который смог бы вести торговые операции, не вызывая подозрении в любом космопорту.

Они перетащили тело Генри Белла в туристический челнок и отправили его дрейфовать в пространстве.

- Так, отныне ты имеешь правовой статус "в розыске", - сказала Элиссия, как только они заняли места за приборной панелью. Но Рейф предполагает, что при уважительном отношении к телу этот статус не распространится за пределы Тионислы. Если бы мы уничтожили тело, будь уверен, нас начали бы искать во всех цивилизованных системах, а этого мы позволить себе не можем.

На экране видно было, как маленький челнок дрейфовал между огромными монументами. Алекс внимательно изучал показания сканеров и мониторов. Скромные запасы энергии позволяли включать только носовой и кормовой экраны. Да и для лазера этой энергии хватило бы только на один-два выстрела, а ракет, конечно, не было. Корабль по-прежнему находился в непосредственной близости от станции "Додо", положение которой хорошо просматривалось на трехмерной сетке навигационной карты.

Медленно "Кобра" развернулась и плавно двинулась вперед, в направлении границы гравитационного поля кладбища. Алекс внимательно следил за зеленоватым свечением сканера. Медленно, крадучись, проплывали на экране монументы и станционные суда.

- Я должна тебе рассказать кое-что об особенностях неуправляемого гиперперехода.

Алекса на мгновение передернуло.

- Спасибо, я уже об этом знаю. К тому же мы перелетим не более, чем на десятую светового года и, возможно, это не очень опасно.

Элиссия усмехнулась.

- В какого бога ты веришь?

- В Фактор Случайности.

- Я тоже.

Они посмотрели друг другу в глаза. Мимо проплывали монументы и монолиты. Звездное поле перед ними расширялось.

- Почти вышли, - вздохнула Элиссия. - Готовься к переходу.

Алекс бросил взгляд на сканер. Две ярких точки внезапно вспыхнули на экране и быстро устремились к ним.

- Эскорт!, - сказал он и Элиссия громко чертыхнулась.

- У нас мала энергия лазера, - сказал Алекс.

- Только попробуй его использовать и мы потеряем все шансы для торговли. Это же полиция! Корабли, может быть и не "Вайперы", но это все же полиция!

Пространство впереди уже почти очистилось. Корабли службы безопасности разделились и начали маневр охвата. Элиссия начала отсчет готовности к гиперпереходу

- Десять секунд.

"Кобра", отвыкшая за много лет покоя от активной работы, задрожала всем телом.

- Они приближаются, открыли огонь.

- Пять секунд.

"Кобра" заскрежетала, получив первый удар лазерного луча. Последние остатки энергии защитных полей исчезли. У атакующего корабля перегрелся лазер. Его партнер замешкался на мгновение, обходя монумент внушительных размеров, выстрелил неудачно и промахнулся.

- Три…

Они на огневой позиции… Залп приближается…

Оба корабля опять сошлись. Огонь их лазеров полыхал вокруг "Кобры".

- Два…

Еще удар, крик боли, корабль почти потерял управление и вдруг… ТУННЕЛЬ!

Элиссия откинулась в кресле. Алекс криво ухмыльнулся. Когда он глянул на свою спутницу, то увидел крупные капли пота на ее лице. Он протянул руку, пальцы мелко дрожали и ничего поделать с этим было невозможно.

ГЛАВА 4

- Так, теперь у тебя есть корабль и немного денег, - сказал Рейф Зеттер. - У тебя есть второй пилот и она прекрасный стрелок, лучший, чем ты сам, впрочем я надеюсь, что это не надолго. Теперь все зависит только от тебя, Алекс. И еще один совет. Если бы Джейсон был жив, он тебе сказал бы об этой сам. В минуту опасности отбрось разум, забудь о силе, забудь о правилах. Действуй так, как подсказывают тебе чувства. Если и это не поможет, то по крайней мере тебя не будет среди тех, кто будет сожалеть о том, как все закончилось.

Сидя перед навигационной панелью, Алекс смотрел на корабль Рейфа. Это была повидавшая виды "Анаконда". Грузовой отсек пробит, топливоприемники распахнуты, бортовые огни мигают и вовсе не потому, что это сигнал, а просто поскольку давно нуждаются в ремонте.

Рейф не пригласил Алекса на борт. В десятой доли светового года от Тионислы он оборудовал себе тихое убежище, где потихоньку потрошил разбитые корабли и собирал все, что может пригодиться для своего судна - механизмы, приборы, топливо и еду. Три небольших истребителя типа "Мамба" были пришвартованы к телу "Анаконды". Над ними возились роботы.

Вскоре после прибытия "Кобры" в частную систему Зеттера, его изображение появилось в рубке Алекса.

- Непростое дело оснастить и заправить корабль для такой миссии, которая тебе предстоит. Я заправлю вас достаточно для того, чтобы добраться до Айзинора, а дальше действуйте сами. Вам нужны ракеты, лазеры, энергетическая бомба, энергозаборники и многое другое. И не показывайся мне на глаза, пока не снимешь скальп с той гадины, которая убила Джейсона.

- Почему ты делаешь все это ради меня?

- Все это я делаю не ради тебя, а ради Джейсона, - ответил Рейф. - И ради многих других. И вот еще что, Алекс. Забудь о Раксле. Не думай о ней пока. Время для этого еще придет…

- Но зачем же тогда отец назвал мне это имя?

- Чтобы ты передал его мне, чтобы я понял, что он в тебя верил, чтобы я знал, что он видел в тебе будущего бойца класса Элита. И его послание дошло.

Второй раз за сутки у Алекса закружилась голова. Да о чем говорит этот старик? Сначала отец оказался бойцом высшего класса, а теперь оказывается еще, что он видел

такой же потенциал в сыне. На имитаторах Алекс всегда имел высший балл и однажды даже получили большой приз - туристическую поездку в Фантастический город, копию легендарного Лондона. Но ему и в голову не могло прийти, что он когда-либо сможет подняться выше ранга "Опасный".

Но быть Элитой…

Головокружительные перспективы. Ему предстоит провести жизнь в битвах с пиратами, в полетах по самым опасным системам. Ему предстоит искать опасность, а не бежать от нее. Он должен широко оповещать о своем статусе самых опасных преступников и побеждать их в бою.

- Одно я могу сказать тебе точно, - продолжал Рейф. - Если ты не пройдешь через все испытания и не станешь Элитой, можешь и не думать о Раксле, забудь о ней. И ты никогда не узнаешь, что искал твой отец.

- Не понимаю.

- Ты знал о его связях с лигой "Темного Колеса"?

Час от часу не легче. "Темное Колесо" было полулегендарным союзом звездопроходцев, которые объединились для того, чтобы исследовать, что скрывается за многими мифами и романтическими преданиями, которые время от времени будоражили умы многих пилотов: города-призраки, параллельные миры, путешествия во времени. Ходили даже легенды о том, что сушествует планета, о которой на древней Земле знали, как о Рае. "Темное Колесо" для современных звездолетчиков было такой же легендой, какой миф о Короле Артуре был для первых космонавтов.

- Но этого не может быть, - вздохнул Алекс, - он бы рассказал нам…

- Черта-с-два рассказал бы он. Его за то и убили, что он слишком много знал и он не стал бы подвергать опасности своих близких. Тот корабль не был пиратским. Просто Джейсон что-то нашел. И это что-то было настолько важным, что встревожило сильные круги.

- Что именно он нашел?

Рейф рассмеялся.

- Посмотри на меня, парень. Ты думаешь, что я это я? Нет, одна нога, кусочек печени и несколько мозговых клеток - вот все, что от меня осталось. Все остальное - это бионика и чудеса хирургии. Да, я тоже когда-то был Элитой, но сейчас мне чтобы плюнуть надо сосредотачиваться десять секунд.

Я больше не принадлежу его кругу. Он не сказал мне ничего, ведь я уже не вхожу в "Темное колесо". Но я не слепой и не глухой и делаю так, как мне говорят. И будь я проклят, если не сам Джейсон незадолго до смерти просил меня приглядеть за тобой, парень и подготовить к тому, чтобы ты смог пойти по его стопам.

Это было слишком велико для Алекса. Он молчаливо сидел у приборной панели, отрешенно перебирая пальцами приборы управления.

Наконец он собрался, улыбнулся и отбросил печаль и грусть.

- Хорошо, раз мой отец хотел этого, я не разочарую его.

(Продолжение следует)

Содержание