ZX - РЕВЮ
N11,12 1991

МП "ИНФОРКОМ" Москва, 107241, а/я 37

Внимание!!!

"ИНФОРКОМ" сообщает, что начиная с 01.01.92г. изымаются из обращения все бланки заказов, высылавшиеся в 1991 году. Заказы и оплата по ним не принимаются и не исполняются.

Начиная с 01.01.92 мы рассылаем бланки двухмесячного действия. Заказы и оплата по ним принимаются только в течение их срока действия.

Заказы, присланные не на наших бланках не исполняются. Рекомендуем Вам перед тем, как что-то заказывать, обратиться к нам письмом для получения бланка и уточнения текущих цен.

"ИНФОРКОМ"

"ЖИВАЯ НИТЬ"

Дорогие друзья!

"ZX-РЕВЮ" отмечает год своего существования. Можно подводить какие-то итоги.

И самый первый итог: мы живы, и собираемся жить дальше. Более того, почти все наши постоянные читатели подписались и на 1992 год. В этом мы видим признание того, что делаем нужное дело.

Мы рады, что смогли предоставить Вам возможность подписаться на 92-ой год без повышения цены, несмотря на то, наши затраты многократно возросли.

Вспомните, что мы писали Вам более года назад: "ZX-РЕВЮ" - это не журнал и не газета. Это персональное письмо." Сегодня мы к этому добавляем "Это живая нить, связывающая нас с Вами."

У "ZX-РЕВЮ" своя особенная миссия и мы рады Вам сообщить, что по-нашему мнению цель, поставленная полтора года назад, успешно достигнута: в стране сложился дружественней многотысячный коллектив поддержки "ИНФОРКОМа". Нам кажется, что мы не ошибаемся в своих оценках - мы судим по тысячам очень и очень теплых писем.

Мы рады Вам сообщить, что начиная с 1 января 1992 года мы уже не считаем Вас просто клиентами и просто читателями, мы вступаем с Вами в ПАРТНЕРСКИЕ ОТНОШЕНИЯ. Организационная, материальная, программная базы для этого созданы.

В тысячах городов, больших и малых, есть наши представители. Более того, они нас ЗНАЮТ, а мы ЗНАЕМ их. Мы не раз к Вам обращались с просьбами и Вы откликались. Мы же, как могли, выполняли Ваши просьбы и пожелания. Мы хорошо друг друга знаем. Знаем кто что может, а что нет. И теперь мы начинаем новый этап совместной работы. Надеемся, что он продлится долгие годы.

Мы решительно заявляем, что сегодня "ИНФОРКОМ" выходит в стране на лидирующие позиции в области разработки обучающих программ для IBM-совместимых компьютеров. Любой сомневающийся в этом может попробовать предложить нам что-нибудь более значительное и удобное в работе, чем наш программный комплекс "АНГРАМ" (см. последнюю страницу).

Мы полтора года готовили те проекты, к распространению которых начинаем сегодня привлекать и Вас. Пробный маркетинг этих программных продуктов уже проведен и поддерживающая реклама по стране уже прошла. Мы точно знаем что и как стоит и как и кем покупается. Но сколь бы ни были наши программы хороши и необходимы, мы не сделаем

самый решительный шаг без Вашей помощи. Каждый из Вас, кто хочет, может стать дистрибутором нашей фирмы в своем городе, поселке, на своем предприятии или в своем учебной заведении. Совместными усилиями мы не только поможем друг другу, мы дадим стране то, чего она не дождется в полной мере от Академии Педагогических наук.

Итак, приступаем к делу.

Отныне в каждом номере на последней странице мы будем давать развернутое описание какой-либо нашей системы или программы.

Зная нужды своего региона или знакомых вам предприятий Вы всегда сможете выбрать из наших продуктов что-то, что можно предложить в той или иной организаций для приобретения.

Условия предельно просты. (Подробности на стр. 263)

Вы можете ничего не знать об IBM-совместимых компьютерах, но если Вы знаете НАС и верите НАМ, а те, кому наши разработки необходимы, знают ВАС и верят ВАМ - это уже победа. Вам должно очень крупно не повезти, если Вы не сумеете десятки раз окупить тех средств, которые затратили на "ZX-РЕВЮ". Но цель не только в этом.

Наши планы и наши проекты не кончаются ни в 92-ом ни в 93-ем годах. Ведутся уже работы над продуктами 94-го года. Мы не оставим ни одного учебного предмета, не охваченного нашей программной поддержкой. Складывается мощная структура из преподавателей, программистов, методистов, менеджеров, специалистов по маркетингу и опираться она будет на Вас - региональные дистрибуторские сети.

Где бы Вы ни жили и кем бы ни работали, можете быть нашим опорным пунктом и мы на Вас рассчитываем.

Вы же можете рассчитывать на достойные комиссионные отчисления, программно-информационную поддержку, постоянное обновление ассортимента нашей программной продукции и ее самую высокую конкурентоспособность, и самое главное - наши планы работ рассчитаны на очень и очень многие годы. Наше развитие имеет гигантские перспективы, приглашаем Вас развиваться в своих регионах параллельно с нами не меньшими темпами.

Если Вы еще школьник и у Вас нет нужных контактов - не отчаивайтесь. Мы думаем и о завтрашнем дне. Придет и Ваше время. Может быть через десять лет нам еще дороже будут наши проверенные годами партнерские связи. Пока читайте "ZX-РЕВЮ", набирайтесь опыта и знаний. Мы не живем одним днем и рассчитываем свои шаги на далекое будущее.

Итак, всем желающим войти в нашу структуру - путь открыт. Читайте последние 2 страницы - там написаны условия работы по проекту "АНГРАМ". В следующем номере представляем многоцелевую информационно-поисковую систему "РЕГИСТРАТУРА", которая найдет свое место в любом учреждении, на любом заводе - где угодно, где бы ни стоял IBM-совместимый компьютер.

ИНФОРКОМ.

FORUM

От имени группы читателей к нам обратился С.Е. Троицкий из Москвы с просьбой прокомментировать явление, с которым они столкнулись в своей работе и дать рекомендации по путям его обхода.

Проблема связана с организацией работы в циклах, когда параметр цикла задан числом с плавающей точкой. Суть проблемы хорошо демонстрируется на двух следующих примерах.

```
10 FOR i=0 TO 1 STEP 0.01
20 IF i <> 0.5 THEN PRINT i
30 NEXT i
```

По этой программе компьютер печатает все значения от 0 до 1, в том числе и 0.5 (?!).

```
10 FOR i=0 TO 1 STEP 0.01
20 IF i=0.5 THEN PRINT 1
30 NEXT i
```

Здесь компьютер ничего не печатает (!?).

Примеры можно продолжать, но суть ясна - что-то не так и надо что-то делать.

Нас в этом вопросе заинтересовала прежде всего его академичность. Проблема в общем-то типовая и, что самое интересное - она имеет отношение не только к "Спектруму", а вообще к любым видам вычислительной техники и знать пути обхода ее надо каждому, кто собирается применять компьютер в инженерных вычислениях.

Итак, начнем. Вы знаете, что компьютер хранит числа в байтах, каждый из которых состоит из восьми битов. Для целого числа от 0 до 65535 достаточно и двух байтов, но для действительных чисел с плавающей точкой этого недостаточно - здесь "Спектрум" использует 5 байтов на каждое число (другие компьютеры могут использовать другое количество байтов, но суть проблемы не меняется). Запись действительного числа в 5-ти байтной форме называется интегральным представлением числа. Хотите узнать все подробно - читайте наш трехтомник по программированию в машинных кодах - там это доступно изложено.

Хоть 5 байтов это и немало - целых 40 битов, тем не менее точность, с которой число может быть занесено в память, все равно не бесконечная. Так, для "Спектрума" эта точность - 8 значащих цифр.

После всего вышесказанного Вы не будете удивляться, если число 0.01 будет храниться в памяти компьютера как 0.0099999999.

Теперь дело за процедурами. Процедуры, выполняющие сложение, после 50 проходов по Вашему циклу могут привести к тому, что вместо 0.5 параметр цикла i может иметь что угодно от 0.049999949 до 0.049999999, но никак не 0.5. И процедура сравнения в строке 20 однозначно ответит, что эти числа "НЕ РАВНЫ". Тогда в действие вступит процедура печати, которая в первом примере должна напечатать 1. Она округлит имеющийся результат и выдаст 0.5.

Другое дело имеет ли она право выполнять такое округление. На вычислительных системах, предназначенных для выполнения инженерных и физических расчетов конечно никто такого права компьютеру не даст. "Спектрум" же - бытовая машина и домохозяйке (школьнику, студенту, поэту, журналисту, рабочему и др.) гораздо приятнее видеть на экране 0.5, чем громоздкое 0.049999949 или еще того лучше 4.9999949 E-02.

Теперь другое дело - как Вам это обойти, чтобы сравнение все-таки работало и работало правильно.

Во-первых, давайте договоримся, что такие операции нужны прежде всего в инженерных расчетах, а они, как известно, выполняются всегда с ограниченной точностью. Например, вычисляя длину окружности по ее радиусу и закладывая радиус равным, например 0.25 м Вы тем самым утверждаете, что радиус вам известен с точностью равной двум значащим цифрам. Какой же Вам смысл получать длину окружности с точностью до 8 значащих цифр, если Вы знаете, что все равно цифрам от третьей до восьмой веры нет?

Во вторых, давайте введем на время Ваших вычислений понятие точности.

Предположим для приведенных выше примеров Вас устраивает точность порядка 0.00001. И теперь давайте с ней работать. Обозначим эту нормативную (необходимую Вам) точность переменной eps (обычно в инженерных расчетах погрешность обозначают греческой буквой "эпсилон").

Теперь Ваша программа будет иметь следующий вид:

```
5 LET eps = 0.00001
10 FOR i = 0 то 1 STEP 0.01
20 IF ABS(i-0.5) > eps THEN PRINT i
30 NEXT i
```

или

```
5 LET eps = 0.00001
10 FOR i = 0 TO 1 STEP 0.01
20 IF ABS(i-0.5) < eps THEN PRINT i
30 NEXT i
```

Итак, рецепт прост: "Если ваши числа представляют числа с плавающей точкой, то вместо того, чтобы сравнивать их одно с другим, надо вычислять разность сравниваемых чисел и сравнивать абсолютную величину это разности с разумно заданной Вами величиной допустимой погрешности eps."

Подобными приемами надо пользоваться всегда, когда Вы ведете инженерные расчеты и выделяете какое-то сравнение. Это не зависит от того "Спектрум" это или "Эльбрус" или "Коммодор". Это стиль инженерных вычислений и его надо выдерживать, чтобы быть гарантированно защищенным от возможных ошибок, связанных с формами представления чисел в известной или неизвестной Вам машине.

Ну и наконец последнее замечание. Использовать в качестве параметра цикла число с плавающей точкой - это "дурной тон". На большинстве компьютеров это приводит как минимум к замедлению скорости вычислений и делать это просто не принято. Параметр цикла надо делать целым числом. И тогда наши примеры будут выглядеть так:

```
5 LET eps = 0.00001
10 FOR i=0 TO 100
15 LET a=i/100
20 IF ABS(a-0.5) > eps THEN PRINT a
30 NEXT i
```

или

```
5 LET eps - 0.00001
10 FOR i=0 TO 100
15 LET a=1/100
20 IF ABS(a-0.5) < eps THEN PRINT a
30 NEXT i
```

БЕТА-БЕЙСИК 1.8

Дополнение к инструкции БЕТА-БЕЙСИК 1.0

Окончание. Начало см. стр. 134, 100, 201.

В данной статье мы заканчиваем печать материалов по БЕТА-БЕЙСИКу версий 1.0 и 1.6.

Версия 1.8 настолько похожа на версию 1.0, что ее можно дать как дополнение к версии 1.0.

Нам еще осталось рассмотреть версию 3.0. Она имеет значительные отличия от двух предыдущих версий и будет нами рассмотрена полностью от начала и до конца. Этим мы с Вами займемся в первой половине следующего года.

Версия 1.8 языка Бета-Бейсик имеет значительно больше команд и функций, чем выпуск 1.0.

Программа теперь имеет больший размер и занимает 9.5K вместо 5.6K у версии 1.0. В этой версии нет модификации для 16K компьютеров. Значение RAMTOP в этой версии установлено в 55800 и загрузчик содержит теперь две самоуничтожающиеся Бейсик-строки (вместо одной). Строка 1 содержит процедуру SAVE для выгрузки текущей Бейсик-программы с последующим блоком машинных кодов Бета-Бейсика. Если Вы хотите воспользоваться ею, то загрузите загрузчик через "MERGE", чтобы исключить автостарт, затем остановите ленту и дайте:

```
CLEAR rt: LOAD "" CODE
```

Теперь вновь запустите ленту. Когда вторая часть будет загружена, RUN вызовет автовыгрузку программы на ленту.

Чтобы выгрузить на микродрайв, дайте RANDOMIZE USR 58419 (это инициализация Бета-Бейсика), а затем измените строки 1 и 2 прежде, чем давать RUN:

```
1 LET rt = DPEEK (23730): RANDOMIZE USR 59904: SAVE *"m"; 1; "run" LINE 2: POKE DPEEK
      (23631)+2,181: SAVE * "m";1;"BB"CODE rt + 1,65367 - rt: STOP
2 CLEAR rt: LOAD * "m";1;"BB" CODE: CLS: RANDOMIZE USR 58419: DELETE 1 TO 2
```

1. Команда CLOCK.

В этой версии программы часы задействуются, хотя и не изображаются на экране, уже во время загрузки Бета-Бейсика. в это же время включается усовершенствованный режим BREAK.

2. Команды управления курсором.

CHR$ 8 (курсор влево)
CHR$ 9 (курсор вправо)
CHR$ 10 (курсор вниз)
CHR$ 11 (курсор вверх)

Бета-Бейсик позволяет использовать эти символы при печати. Например, PRINT CHR$ 10 сдвинет позицию печати на одну строку вниз. Эти же коды работают с оператором PLOT при печати на экране строк.

В стандартном Бейсике CHR$8 (курсор влево) имеет "жучок". Сдвигая курсор назад, нет возможности войти в верхнюю строку экрана из какой-либо нижележащей строки.

А если Вы уже находитесь на верхней строке то движением назад (влево) можете вообще покинуть экран и войти в программу. В этой версии Бета-Бейсика ошибка исправлена.

CHR$ 9 стандартного Бейсика также имеет "жучок", мешающий его работе. Здесь он также исправлен.

CHR$ 10 и CHR$ 11 в стандартном Бейсике печатают "?" от которого нет никакой пользы. Сейчас они работают, как положено.

Пример, приведенный ниже, показывает, как четыре управляющих символа включаются в строки, что дает возможность создания сложных форм с помощью PRINT или

PLOT.

```
10 LET a$="1234"+ CHR$ 8 + CHR$ 10 + "5" + CHR$ 8 + CHR$ 10 + "678" + CHR$ 8 + CHR$ 11 + "9"
20 PRINT AT 10,10; a$
30 PAUSE 100: CLS
40 FOR n = 32 TO 255
50 PLOT n,n/2; a$: NEXT n
```

С применением графики UDG здесь можно получать очень эффектные изображения.

3. Команда: DEF KEY

Структура: DEF KEY символ; строка

или DEF KEY символ: оператор: оператор:...

Клавиша: CAPS SHIFT + 1

Бета-Бейсик позволяет воспроизводить строковые переменные или целые строки программы при нажатии цифровой или буквенной клавиши. Причем они могут либо сразу идти в компьютер, либо оставаться в нижней части экрана до нажатия ENTER. Последний случай выполняется введением управляющего символа ":". Он должен стоять последним символом в строке или стоять после последнего выражения в строке.

Попробуйте:

```
DEF KEY "i"; "HELLO":
```

Теперь нажмите "SYMBOL SHIFT" и "SPACE" одновременно. Курсор изменится на мигающую звездочку. Если Вы нажмете "I", то в нижней части экрана появится "HELLO". Поскольку все прочие клавиши заданы не были, то при нажатии любой другой клавиши Вы получите ее нормальные значения.

```
DEF KEY "a"; "GOODBYE"
```

В этом примере строка "GOODBYE" присваивается клавише "a" (или "a", что одно и то же). пока она не выполняется, поскольку после выражения не стоит ":". Когда же Вы нажмете "SYMBOL SHIFT" + "SPACE", строка будет введена, а когда нажмете "a", она исполнится.

Вы можете с помощью нехитрого приема записывать строки в программу одним нажатием клавиши. Например, Вам надо, чтобы после нажатии клавиши "a$ в программу вводилась строка 10 REM hello.

Вы можете задать клавишу "a" следующим образом:

```
DEF KEY "a": "10 REM hello".
```

Когда Вы попробуете это сделать, Вас может постигнуть легкое разочарование, поскольку Вы не сможете получить ключевое слово REM, т.к. курсор в этот момент будет иметь вид не "K", а "L". Как с этим бороться мы уже писали раньше в разделе "Маленькие хитрости", стр. 52. Используем для этого особенность оператора THEN. Наберем начала

```
DEF KEY "a"; "10 THEN REM hello"
```

Теперь установим курсор справа от REM и помощью DELETE сотрем THEN - Вас останется то, что надо.

Таким приемом можно на любую клавишу "подвесить" часто повторяющиеся при наборе программ строки и облегчить себе программирование.

Клавише может быть присвоено новой значение в любое время, старое будет забыто. Если использовалась пустая строка или не было задано никаких операторов после определения клавиши, то клавиша не будет иметь определения.

DEF KEY ERASE сотрет все определения клавиш, которые хранятся выше RAMTOP и защищены от стирания другими путями, в том числе и NEW. Процедура SAVE в загрузчике Бета-Бейсика запишет все определения клавиш на ленту вместе с программой. (Запись идет от RAMTOP до конца Бета-Бейсика).

Для тех, кто программирует в машинных кодах, возможно, будет интересна следующая информация.

RAMTOP автоматически понижается для того, чтобы выделить место для занесения определений клавиш. Если Вы сами используете CLEAR (число) для изменения RAMTOP, то это может помешать воспользоваться заданными клавишами.

Эту проблему можно обойти, если сделать так, чтобы при переустановке RAMTOP синхронно перемещалась бы и область определения клавиш. Нижеследующая процедура сдвигает RAMTOP и определения клавиш вниз на заданное количество шагов, чтобы

избежать этой проблемы. Между концом области определения клавиш и старым RAMTOP, т.е. 55800 создается зазор, при желании его можно изменить.

space - размер памяти, на которую понижается RAMTOP;

oldrt - старый адрес RAMTOP;

newrt - новый адрес RAMTOP.

В процедуре используются функции MEMORY$() и STRING$(), которые мы еще не рассматривали и о которых мы сообщим ниже.

```
10 INPUT "Spaces?"; space
20 LET oldrt - DPEEK (23730)
30 DPOKE 23728,oldrt
40 CLEAR (oldrt - space)
50 LET oldrt = DPEEK (23726)
60 LET a$ = MEMORY$ ( ) (oldrt TO 55600)
70 LET newrt = DPEEK (23730)
80 LET space = oldrt - newrt
90 POKE newrt,a$
100 POKE (55801 - space),STRING$ (space,CHR$ 0)
```

Всегда оставляйте нулевой байт после последнего определения клавиш в качестве "маркера" конца.

4. Команда: FILL

Структура: FILL x,y или FILL // INK цвет//; x,y

или FILL //PAPER цвет //; x,y

Клавиша: F

По этой команде происходит заполнение области PAPER цветом INK, если была выбрана команда FILL или FILL INK или происходит заполнение области INK цветом PAPER, если была избрана команда FILL PAPER. Заполнение начинается с точки, имеющей координаты x,y. Если Вы попробуете заполнить область цветом INK, а точка x,y уже имеет цвет INK, то ничего не получится. В отличие от стандартного Бейсика, если Вы опустите номер цвета, то команда FILL сработает с текущим цветом INK или PAPER.

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

Возможно использование более сложных форм FILL:

```
FILL INK 2; PAPER 1; FLASH 1; x,y
```

В этом случае первое слово после FILL указывает какой цвет INK или PAPER Вам нужен, а остальные изменяют аттрибуты заполняемой области.

Поскольку количество цветов, допустимых для одного знакоместа, ограничено двумя, то необходимо тщательно продумывать условия стыковки двух областей с разным цветом INK. Необходимо, чтобы стык проходил по границам знакомест.

FILL будет работать с любыми формами. Попробуйте этот пример, который заполнит весь экран, за исключением областей внутри букв.

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

Метод работает быстрее, если имеется большой объем доступной памяти. Если Вы попробуете заполнить большую пустую область, например весь экран, Вы увидите паузу в те моменты, когда FILL производит проверку хранимых данных в поисках того, что можно без ущерба вычистить.

Количество пикселей, заполненных по FILL, можно определить с помощью функции FILLED ().

FILL можно прервать в любой момент с помощью BREAK.

5. Команда: JOIN

Структура: JOIN //номер строки//

Клавиша: SYM SHIFT + G (см. также SPLIT.)

По этой команде объединяются вместе две строки. Если номер не указан, то

объединяется строка, около которой стоит указатель, со следующей за ней строкой, если она есть.

Вторая строка теряет свой номер и присоединяется к предыдущей через разделитель ":",

6. Команда: KEYIN

Структура: KEYIN строка.

Клавиша: SYM.SHIFT + 4

По этой команде вводится строка так, как если бы она была впечатана с клавиатуры. Это позволяет программам самозаписываться.

Полный список возможных приложений этой функции выходит за пределы, рассматриваемые фирменной инструкцией по работе с Бета-Бейсиком. Очевидно, ее возможности столь широки и необычны, что требуют специального исследования. Указано только, что наиболее очевидным приложением является возможность автоматического написания в листинге программы строк DATA (очень утомительное занятие для программиста, и если у него есть массив данных в памяти машины, имеет смысл подготовить специализированный модуль, занимающийся тем, чтобы формировать строки DATA из массива.)

Пример

```
10 LET a$="100 DATA": REM применяем ключевое слово DATA. Как это делать через THEN мы писали
      выше
20 FOR N = 0 TO 9
30 LET A$=A$ + STR$(PEEK N) + ","
40 NEXT N
50 LET A$=A$(1 TO LEN(A$)-1): REM отсечение последней запятой
60 KEYIN A$
```

Вы увидите, что после RUN к программе будет добавлена еще одна строка. (Строка совершенно бесполезная, поскольку дает данные первых 10 байтов памяти ПЗУ, но зато иллюстрирует сам принцип).

Если Вы чувствуете в себе смелость, можете использовать SCREEN$ и KEYIN для того, чтобы с их помощью написать полноэкранный редактор Бейсика!!!

Примечание: Он может быть довольно медленным.

Примечание 2: KEYIN не может быть непосредственной (прямой командой), а только частью программы.

7. Команда: LIST

Структура: LIST номер строки TO номер строки

или LLIST номер строки TO номер строки

Первый или второй номер строки может быть опущен. Если опущен первый, то он принимается за 0, если второй, то предполагается номер последней строки.

LIST 20 TO 100

LIST TO 200

LLIST 100 TO 180

Если оба номера совпадают, то распечатывается только одна строка.

Очень эффективно можно эту команду использовать совместно с TRACE.

```
9000 LIST line TO line: RETURN
```

Эта строка распечатывает каждую строку по мере ее исполнения после того, как Вы вставите TRACE 9000 в программу (в данной примере "line" - это переменная, создаваемая процедурой TRACE).

К сожалению, каждая строка будет иметь при себе мигающий курсор.

8. Команда POKE

Структура: POKE адрес, строка

Это обычное ключевое слово.

Бета-Бейсик 1.8 разрешает выполнять POKE с символьными строками так же, как и с числами, что в сочетании с функцией MEMORY$ дает возможность быстрых манипуляций

большими блоками памяти. (Здесь надо отметить, что если Ваша программа может выйти из строя в результате нерасчетливого POKE, то при работе с длинными символьными строками это становится много более вероятным).

Давайте посмотрим на один эффект:

```
10 LET screen = 16384
20 POKE screen,STRING$(6144,"U").
```

Функция STRING$ была рассмотрена в инструкции к версии 1.0 под именем FN S$.

По этой программе экран заполняется "U", но поскольку вследствие своего положения это уже не "U", а код от "U",т.е. 85, а в двоичной форме "01010101", то экран заполняется полосами.

```
30 LET attributes = 22528
40 POKE attributes, MEMORY$()(1 TO 704)
```

Эта программа копирует начало области памяти ROM в файл атрибутов, где эта операция производит интересный эффект.

Теперь давайте запишем простую программу, которая создаст на экране рисунок, сохранит его в памяти как символьную строку и затем через POKE вернет его на экран.

```
10 CIRCLE 128,88,70
20 FILL 128,83
30 LET a$=MEMORY$() (16384 то 23255):
40 CLS: PRINT "HIT ANY KEY":PAUSE 0
50 POKE 16384,a$
```

Память компьютера может содержать несколько таких рисунков, что даст вам возможность заменять их и демонстрировать последовательно. Чтобы можно было разместить в памяти большее количество рисунков, удобно считывать в строковую переменную третью часть экрана, туда же Вы можете добавить и третью часть файла атрибутов. Чтобы сохранить в памяти три трети экрана в виде строк, пользуйтесь:

Первая треть экрана.

```
LET a$=MEMORY$() (16384 TO 16431)
```

Вторая треть экрана.

```
LET a$=MEMORY$() (18432 TO 20479)
```

Нижняя треть экрана.

```
LET a$=MEMORY$() (20480 TO 22527)
```

Памяти достаточно, чтобы создать удовлетворительный мультфильм с использованием последовательных POKE строк. Вы можете применять массив (например DIM a$(10,2048)) чтобы хранить эти данные.

Конечно, существует значительно больше потенциальных возможностей, кроме развлечений с экранной памятью. Можно очистить (CLEAR) большую область памяти и загнать туда на хранение целую программу:

```
CLEAR 33900,
```

затем запустите (RUN) следующую программу:

```
10 POKE 34000,MEMORY$ () (33552 TO 33800)
20 .... остальная часть программы
```

Теперь Вы можете дать NEW и стереть свою программу, а затем, когда она Вам вновь понадобится, вернуть ее назад:

```
POKE 23552,MEMORY$ () (34000 TO 44246)
```

Эту команду можно "спрятать" от NEW на клавише заданной пользователем. Мы уже говорили, что задания клавиш после NEW не уничтожаются, т.к. хранятся выше уровня, установленного RAMTOP. Клавиша задается после CLEAR и до RUN, например:

```
DEF KEY "J": POKF 23552,MEMORY$()(34000 TO 44248).
```

После того, как программа будет возвращена, она продолжит работу с того места, в котором она была "спрятана", т.к. системные переменные были сохранены вместе с Бейсиком, а в них запоминается состояние программы во время ее работы.

Немного труднее организовать хранение двух программ, нужно больше памяти, нужно также предоставить место в БЕЙСИК-области для их запуска, но это тоже возможно.

И, дополнительно, еще одна "маленькая хитрость". Чтобы выгрузить на ленту блок в машинных кодах с Бейсиковской программой, присвойте его символьной строке с использованием MEMORY$, а затем сделайте SAVE для Бейсик-программы, включив в нее и

эту строковую переменную. При этом предусмотрите, чтобы после автостарта POKE возвращал бы этот блок на его место при загрузке БЕЙСИКа.

Примечание: Если хотите избежать затирания каких-то важных данных после такого POKE, не забывайте предварительно выставлять CLEAR. Еще полезный совет; POKE машинного кода совершенно безопасно производится в область экрана (хотя это некрасиво выглядит) или в область буфера принтера (если Ваш код не более 256 байтов и если Вы не работаете со 128-ой моделью.

9. Команды: ROLL и SCROLL

Эти команды могут иметь довольно сложный синтаксис. Мы настойчиво рекомендуем изучить сначала их работу по инструкции к Бета-Бейсику 1.0.

В этой версии ROLL может иметь структуру:

ROLL код направления //,пиксели// //; x,y; ширина, длина//

Коды направления 1...4 перемещают только атрибуты, коды 5...8 - только пиксельную информацию, коды 9...12 - и то и другое вместе.

Поскольку атрибуты могут перемещаться только на величину знакоместа 8x8, то целесообразно при совместном перемещении двигать пиксельную информацию на 8 пикселей за шаг.

Параметр "пиксели" можно задать самому. По умолчанию он принимается равным 1. Его значение не должно превышать 255 для горизонтального перемещения и 176 для вертикального.

ROLL 5 или ROLL 5,1 передвинет экран влево на один пиксель, а ROLL 5,2 - на два пикселя. При вертикальном ROLL или SCROLL скорость перемещения примерно пропорциональна количеству пикселей, сдвигаемых за один раз. В горизонтальном направлении наилучшую скорость дает перемещение на 4 пикселя или 8, т.к. при этом процессор Z-80 применяет инструкции сдвига на байт или полубайт (ниббль).

10. Команда SPLIT

Это не ключевое слово. В действительности вводится символ <>.

Клавиша: SYM.SHIFT + W (не в режиме "G").

Команда служит для деления строк программы на несколько частей.

Если в строку, содержащую несколько операторов, при редактировании ввести символ "<>" в качестве первого символа какой-либо инструкции, то после ENTER в программу эта строка пойдет только до знака "<>", а оставшаяся часть строки останется в нижней части экрана в области редактирования.

Теперь знак "<>" можно убрать и он заменится тем же номером строки. Курсор стоит справа от номера строки в положении, которое позволяет вам легко изменить номер на новый прежде чем нажмете "ENTER".

Если Вы введете:

```
10 PRINT "hello": GO TO 10: <> PRINT "goodbye"
```

и нажмете ENTER, то в листинге появится:

```
10 PRINT "hello": GO TO 10
```

а в нижней части экрана останется:

```
10 (курсор) PRINT "goodbye"
```

ФУНКЦИИ

В новой версии Бета-Бейсика 1.8 по сравнению с версией 1.0 добавлено много новых функций. Ниже в таблице они приведены как бы в виде ключевых слов, но таковыми не являются, а набираются через FN, с последующей буквой и знаком $ или скобкой (.

Функция	Клавиша	Версия
AND	FN A(	1.8
BIN$	FN B$	1.8
CHAR$	FN C$	1.0
COSE	FN C	1.8
DEC	FN D(	1.0

DPEEK	FN P(	1.0
FILLED	FN F(	1.8
HEX$	FN H$	1.0
INSTRING	FN I(	1.0
MEM	FN M(	1.0
MEMORY$	FN M$	1.8
MOD	FN V(	1.8
NUMBER	FN N(	1.0
OR	FN O(	1.8
RNDM	FN R(	1.8
SCRN	FN K$	1.8
SINE	FN S(	1.8
STRING$	FN S$	1.0
TIME$	FN T$	1.0
USING$	FN U$	1.0
XOR	FN X(	1.8

Те функции, которые вошли в описание версии Бета-Бейсик 1.0, здесь не рассматриваются.

1. Функция: AND

Структура: AND (число, число)

Клавиши: FN A (число, число).

Эта функция записывается как обычное ключевой слово, но отличается от стандартного в листинге программы своим синтаксисом. Оно выдает результат побитной операции логического AND двух чисел, которые должны быть в диапазоне от 0 до 65535. каждый бит будет равен "1" только если соответствующий бит и в первом и во втором числе равнялся единице. Пониманию этого очень способствует новая функция BIN$.

```
BIN$(254)="1111 1110"
BIN$(120)="0111 1000"
BIN$(AND(254,120))="01111000"
```

Вы можете использовать AND для того, чтобы "маскировать" нежелаемые биты. Например:

```
PRINT AND (BIN 00000111,ATTR(line,column))
```

Эта строка дает цвет INK для позиции "line,column", путем маскировки остальных битов атрибутов. Вы могли воспользоваться числом "7" вместо "BIN 0000 0111".

В нижеследующем примере программа напишет слово "Bang", если была нажата клавиша F, причем возможно ее сочетание с любыми клавишами (см. т. 1 нашего трехтомника по программированию в машинных кодах, где каждая клавиша рассматривается в качестве серии внешних портов).

```
10 IF AND (BIN 0000 1000, IN 65022) 0 THEN PRINT "bang!";
20 GO TO 10
```

2. Функция BIN$

Структура: BIN$ (число)

Клавиша: FN B$ (число)

Дает двоичный эквивалент числа R качестве восьми-символьной строки. Если число меньше, чем 256 или в качестве шестнадцати-символьной строки, если число лежит между 256 и 65535.

Эта функция полезна для понимания машинных кодов и операций с битами для функций AND, OR и XOR.

Она может быть также полезной при проверке генератора символов из ПЗУ, области графики пользователя, области атрибутов, системных переменных и клавиш клавиатуры.

```
10 PRINT AT 10,10; BIN$(IN 65022):GO TO 10
```

Если Вам хочется иметь в строке какие-то символы, отличные от "0" и "1", то дайте POKE 62865 или 62869 с желаемым символом.

3. Функция: COSE

Структура: COSE (число)

Клавиши: FN C (число)

Функция выдает косинус "числа" с меньшей точностью, чем стандартная - точность до 4-х значащих цифр, но в шесть раз быстрее.

4. Функция: FILLED

Структура: FILLED( )

Клавиша: FN F( )

Функция дает количество пикселей, заполненных последней командой 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 ( )
```

Одна сторона квадрата - 10 пикселей. (Заметьте, что если бы мы вместо 9 использовали бы 1 в функции DRAW, то сторона квадрата равнялась бы двум пикселям). Это потому, что реальные линии имеют на компьютере толщину в один пиксель. Внутренняя сторона построенного квадрата имеет 8 пикселей, поэтому FILLED даст нам 64.

Если бы мы дали FILL PAPER:5,5 то квадрат удалился бы с экрана и функция FILLED дала бы 100.

Разница между 100 и 64, равная 36 - это количество пикселей, составляющих периметр.

5. Функция: MEMORY$

Структура: MEMORY$ ()

Клавиши: FN M$( )

Выдает всю память в качестве символьной строки. На самом деле сюда не включается первый байт компьютера (адрес 0), поэтому MEMORY$( ) (1)- это тоже самое, что и PEEK 1. По техническим соображениям сюда не входят также три последних байта памяти, поэтому результат работы функции на самом деле имеет длину LEN 65532.

Совместно со способностью Бета-Бейсика делать POKE для символьных строк, эта функция дает программисту возможность перемещать большие области памяти с высокой скоростью. Для более полного описания этого аспекта см. POKE.

Другое приложение MEMORY$ состоит в том, что она позволяет делать быстрый поиск в памяти, используя функцию INSTRING.

Несмотря на то, что область памяти, в которой производится поиск, можно ограничить например вырезкой MEMORY$ ()(23759 TO)).

INSTRING работает так быстро, что как правило это не имеет смысла делать.

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

Эта программа отыщет тот адрес, где в REM содержится строка "ASDFG". Вместо 1 мы могли бы поставить DPEEK(23635), где содержится переменная PROG, указывающая на адрес начала Бейсик-программы. Тогда поиск проводился бы от начала программы, а не от начала ПЗУ.

Поскольку Бета-Бейсик позволяет выполнять POKE для строк, то поиск строк и их замена выполняются очень просто, если Вы хорошо представляете, что Вы делаете. Может быть Вы не пожелаете, чтобы программа допускала замену строки другой, если та имеет большую длину.

6. Функция MOD

Структура: MOD (число,число).

Клавиши: FN V (число,число).

Функция дает остаток деления одного числа на другое.

```
MOD(10,3) = 1
MOD(66,16) = 2
```

```
MOD(125,35,5) = 18.5
```

Ниже приведен пример программы, предотвращающей печать (PLOT) за пределами экрана.

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

7. Функция: OR

Структура: OR (число,число).

Клавиши: FN O (число, число).

Эта функция записывается как обычное ключевое слово, но отличается от стандартного OR другим синтаксисом. Она выполняет логическую побитную операцию OR для двух чисел, которые должны быть в диапазоне от 0 до 65535. Если бит равен 1 в первом числе или во втором, то соответствующий бит результата тоже будет равен 1. Чтобы в результате бит равнялся нулю, он должен быть равным нулю в обоих числах.

8. Функция: RNDM

Структура: RNDM (число).

Клавиши: FN R (число).

Если число равно "0", то RNDM дает случайное число от 0 до 1, как и RND. Однако она работает в два с половиной раза быстрее. Если число не равно нулю, то функция дает случайное число в диапазоне от 0 до этого числа. Это также выполняется в два с половиной раза быстрее, чем RND*(число).

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

RANDOMISE (число) устанавливает RNDM в определенное место в последовательности случайных чисел так же, как и для RND.

9. Функция SCRN$

Структура: SCRN$ (ряд,столбец)

Клавиши: FN K$

Работает почти как обычная функция SCREEN$, за исключением того, что распознает символы графики, как обычные символы. "Жучок", который содержится в функции SCREEN$ "Спектрума", здесь также устранен.

Сначала введите KEYWORDS 0, затем попробуйте пример, приведенный ниже. Он создает символы графики пользователя со случайным рисунком, а затем читает их с экрана.

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

"Спектрумовская" блочная графика не понимается. Если Вам это нужно, запрограммируйте символы графики пользователя так, чтобы они выглядели, как блочная графика.

10. Функция: SINE

Структура: SINE (число).

Клавиши: FN S (число).

Дает синус "числа" с меньшей точностью, чем стандартная функция, хотя 4 значащих цифры есть. Зато работает в шесть раз быстрее.

11. Функция: XOR

Структура: XOR (число,число).

Клавиши: FN X (число,число). Эта функция выполняет XOR (исключающее "или") для двух чисел. Числа должны быть в диапазоне от 0 до 65535. Если бит равен "0" или "1" в обоих числах, то он будет равен "0" в результате. Если бит равен "1" только в одном из чисел, то он будет равен "1" в результате.

КАНАЛЫ И ПОТОКИ

Сегодня в номере нет традиционного раздела "Секреты ПЗУ". Причина проста - сейчас мы рассматриваем тот раздел, который очень широко оперирует с такими образованиями, как потоки и каналы и мы уже не можем двигаться дальше, не рассказав Вам о том, что это такое, ведь раньше в своих изданиях мы на этом вопросе не останавливались.

Мы обещали Вам, что в начале следующего года дадим широкую статью на эту тему, но время идет, планы меняются и в связи с тем, что материал по ПЗУ данного выпуска обязательно требует предварительного обсуждения потоков и каналов, мы пропустили эту статью вперед и даем ее всю целиком, в одном номере, без продолжений, чтобы не ущемить интересы тех немногих читателей, кто не подписался на будущий год.

В двух словах: каналы и потоки позволяют скрыть от пользователя сложную логику работы программного обеспечения и сделать для него возможность простыми командами выполнять сложные действия.

Это не более чем еще один прием организации программ, еще один вклад в копилку Ваших знаний о компьютере "Спектрум".

Для многих начинающих пользователей "СПЕКТРУМа" такие понятия как "каналы" и "потоки" могут звучать как некоторые непонятные жаргонные обозначения, но на самом деле за ними скрывается интересная концепция, которая может позволить Вам взять от компьютера то, что другими путями сделать непросто.

Итак, как всегда, начнем с самого простого. Вы начинаете использовать каналы и потоки уже тогда, когда даете команды PPINT или INPUT.

Так, PRINT на самом деле обозначает PRINT #2, хотя #2, как правило опускают. Точно так же INPUT на самом деле INPUT #0. Кстати и LPRINT на самом деле то же самое, что и PRINT #3.

Число, которое стоит после символа # является номером потока.

Не любое число может быть использовано в качестве номера потока. Так, если Вы захотите после включения "Спектрума" напечатать что-то по шестому потоку и дадите команду PRINT №6, то получите сообщение об ошибке "0: Invalid stream" ("ошибочно задан поток"). Это происходит потому, что поток номер шесть еще не подключен ни к какому каналу (потоки 0...3 подключены стандартно, здесь Ваше участие не требуется, этим занимаются программы, "зашитые" в ПЗУ), поэтому прежде чем двигаться дальше, давайте разберемся с каналами.

Листинг 1.

```
                    ORG B000
37          CLOSE_NEW   SCF                     ;Сигнал на выгрузку дан-
                                                ;ных из буфера.
F5          CLOSE_CL    PUSH AF                 ;Сохранение номера потока.
08                      EX AF,A'F'              ;Для сохранения флага C.
CD2117                  CALL 1721,STR_DATA_A    ;Это обращение во вторую
                                                ;половину процедуры
                                                ;STR_DATA-1 (171E). На
                                                ;выходе HL содержит базо-
                                                ;вый адрес таблицы STRMS,
                                                ;а BC - адрес данных
                                                ;для данного потока.
78                      LD A,B                  ;Проверка
B1                      OR C                    ;BC на ноль.
C8                      RET Z                   ;Возврат, если поток уже
                                                ;закрыт.
E5                      PUSH HL                 ;Сохранение на стеке
                                                ;адреса таблицы STRMS.
```

```
21E2A3                  LD HL,A3E2              ;Число A3E2 взято потому,
                                                ;что если к нему прибавить
                                                ;5C1E (адрес первого
                                                ;пользовательского потока
                                                ;в STRMS, то возникнет
                                                ;переполнение и включится
                                                ;флаг переноса).
09                      ADD HL,BC               ;Проверка на переполнение.
E1                      POP HL                  ;Возврат адреса STRMS со
                                                ;стека.
D0                      RET NC                  ;Возврат, если не включен
                                                ;флаг переноса, следова-
                                                ;тельно переполнения не
                                                ;было и поток не пользова-
                                                ;тельский.
DD2A4F5C                LD IX,(CHANS)           ;Установка в IX начала об-
                                                ;ласти C.I.A.
DD09                    ADD IX,BC               ;Установка адреса блока
DD2B                    DEC IX                  ;информации о канале.
DD7E05                  LD A,(IX+05)            ;Ввод идентификатора
                                                ;пользовательского канала.
FE34                    CP 34                   ;Возврат, если канал не
C0                      RET NZ                  ;пользовательский;
DD7E06                  LD A,(IX+06)            ;Ввод идентификатора
                                                ;пользовательского канала.
FE12                    CP 12                   ;Возврат, если канал не
C0                      RET NZ                  ;пользовательский;
3600                    LD (HL),00              ;Обнуление данных
23                      INC HL                  ;по потоку
3600                    LD (HL),00              ;в таблице STRMS.
C5                      PUSH BC
DD6E07                  LD L,(IX+07)            ;В HL устанавливается ад-
DD6608                  LD H,(IX+08)            ;рес процедуры, выгружаю-
                                                ;щей буфер.
08                      EX AF,A'F''             ;Вызов флага C.
DC2C16                  CALL C,162C,CALL JUMP   ;Выгрузка буфера.
DDE5                    PUSH IX                 ;Переброска из
E1                      POP HL                  ;IX в HL.
DD4E09                  LD C,(IX+09)            ;В регистре BC устанав-
DD460A                  LD B,(IX+0A)            ;ливается длина блока.
C5                      PUSH BC
CDE819                  CALL 19E8,RECLAIM_2     ;Удаление блока из памяти.
3E10                    LD A,10                 ;10 - количество потоков.
21165C                  LD HL,5C16              ;5C16 - адрес STRMS 00.
5E          CLOSE_LOOP  LD E,(HL)               ;Теперь HL указывает на
23                      INC HL                  ;данные по только что пе-
56                      LD D,(HL)               ;ремещенному потоку, а DE
E3                      EX (SP),HL              ;- по следующему.
A7                      AND A                   ;Сброс флага переноса.
ED42                    SBC HL,DF               ;Проверка передвигался ли
19                      ADD HL,DE               ;блок после стирания.
300B                    JR NC,CLOSE NEXT        ;Если не передвигался и в
                                                ;его STRMS_n изменения не
                                                ;вносятся - переход.
EB                      EX DE,HL                ;Нет операции SRC DE,BC,
                                                ;потому переброс в HL.
A7                      AND A                   ;Сброс флага переноса.
ED42                    SBC HL,BC               ;Теперь в HL содержится
                                                ;измененное значение ука-
                                                ;зателя STRMS_n.
EB                      EX DE,HL                ;Перевод его в DE.
E3                      EX (SP),HL              ;Установка
2B                      DEC HL                  ;нового
73                      LD (HL),E               ;значения
23                      INC HL                  ;указателя
```

```
72                      LD (HL),D               ;в таблице STRMS.
E3                      EX (SP),HL
E3      CLOSE_NEXT      EX (SP),HL
23                      INC HL                  ;HL указывает на следующий
                                                ;указатель.
3D                      DEC A                   ;Уменьшение параметра
                                                ;цикла.
20E6                    JR NZ,CLOSE_LOOP        ;Возврат к началу цикла.
F1                      POP AF                  ;Балансировка стека.
C9                      RET                     ;Возврат.
```

Можете представить себе канал в качестве некоего устройства, которое используется для ввода или вывода информации. Так, например Ваш телевизор - это канал, поскольку на его экране можно печатать символы.

Надо, правда оговориться, что канал - это не всегда техническое устройство. Каналом может быть например файл на диске (или в памяти компьютера). В файл ведь тоже можно заносить информацию, можно ее оттуда и принимать. Если Вы выполните печать чего-то в файл, то ни на экране, ни на принтере Вы результатов этой печати не увидите, но впоследствии, когда будете просматривать этот файл, увидите, что информация туда вошла, то есть произошла печать (вывод).

Еще лучше представить концепцию каналов и потоков на примере морской бухты. Представьте себе побережье с многочисленными заливами и бухтами. Со стороны суши в них впадают реки, речки и ручьи. Так вот заливы и бухта - это те самые КАНАЛЫ, а ручьи и речки, впадающие в них, - это ПОТОКИ, подключенные к КАНАЛАМ. Их можно и переподключить. Если Вы построите на ручье плотину, образуется водохранилище, уровень воды поднимется и Вы сможете отвести ручей (ПОТОК) в другой залив (КАНАЛ).

Эта аналогия хороша тем, что помогает преодолеть имеющуюся в русском языке небольшую мнемоническую путаницу. Дело в том, что мы обычно под словом "канал" понимаем что-то узкое, длинное, по чему перемещаются какие-то суда, грузы и течет вода. В общем, отождествляем канал с потоком. К сожалению это неверно. "Канал" в компьютере - не средство транспортировки данных - это именно залив, бухта, (экран, принтер, файл на диске, участок в области памяти и т.п.) в которые вливаются потоки данных (ручьи).

В "Спектруме" каждый канал имеет имя, которое выражено одной буквой алфавита. Так, экран дисплея - это канал "S" потому, что по-английски Screen - экран.

Оператор OPEN служит для того, чтобы подключить поток к каналу (канал к потоку). Так, Вы можете в БЕЙСИКе дать команду OPEN#6,"S" и тем самым подключите шестой поток к экрану и тогда команда PRINT# 6 будет печатать Ваш текст на экране точно так же, как это делает обычная команда PRINT.

Листинг 2.

```
                        ORG B061
3E10        CLEAR_NEW   LD A,10
3D          CLEAR_LOOP  DEC A
F5                      PUSH AF
A7                      AND A                   ;Выключение флага C - сиг-
                                                ;нал о том, что данные в
                                                ;буферах уничтожаются.
CD01B0                  CALL B001,CLOSE_CL      ;Удаление канала, подклю-
                                                ;ченного к данному потоку.
F1                      POP AF
20F7                    JR NZ,CLEAR_LOOP        ;Возврат для повтора.
C9                      RET
```

Листинг 4.

Вектор переходов для ПЗУ "ZX-Spectrum-128".

```
                        ORG B6FC
C3AC05      V_ERROR     JP 05AC                 ;Генерация сообщений об ошибках.
```

```
C3641C      V_PAGE      JP 1C64          ;Переключение текущей страницы ОЗУ.
C3971D      V_NEWCAT    JP 1C97          ;Создание новой записи в каталоге.
C3F31C      V_SPACE     JP 1CF3          ;Проверка на достаточность
                                         ;памяти RAM-диска.
C3121D      V_FIND      JP 1D12          ;Поиск имени файла в каталоге.
C3561D      V_CATEND    JP 1D56          ;Оформление последней
                                         ;записи каталога.
```

Листинг 5.

Вектор переходов для ПЗУ "ZX-Spectrum+2"

```
                        ORG B6FC
C3CB05      V_ERROR     JP 05CB          ;Генерация сообщений об ошибках.
C3631C      V_PAGE      JP 1C83          ;Переключение текущей страницы ОЗУ.
C3B61C      V_NEWCAT    JP 1CB6          ;Создание новой записи в каталоге.
C3121D      V_SPACE     JP 1D12          ;Проверка на достаточность
                                         ;памяти RAM-диска.
C3311D      V_FIND      JP 1D31          ;Поиск имени файла в каталоге.
C3751D      V_CATEND    JP 1D75          ;Оформление последней
                                         ;записи каталога.
```

Аналогично Вы можете представить, что клавиатура - это устройство, предназначенное для ввода информации и потому это тоже канал. Он имеет имя "K" (от слова Keyboard = клавиатура). К этому каналу можно подключить поток точно так же, как мы это делали с экраном. Можете дать команду OPEN #n,"K" и подключить поток n к каналу "K".

Всего Вы можете иметь не более 16 потоков, пронумерованных от 0 до 15. Шестнадцатого потока не существует и, если Вы попробуете его использовать, то получите сообщение об ошибке.

Мы уже сказали о том, что потоки от 0 до 3 являются стандартными и организованы без нашего участия. Они стандартно подключены к стандартный каналам.

Потоки 0 и 1 подключены к каналу "K", поток 2 - к каналу "S", а поток 3 - к каналу "P". Канал "P" - принтер (Printer).

Каналы "S" и "P" предназначены только для вывода (ручей может только впадать в залив), поэтому например PRINT #2 или PRINT #3 возможны, а INPUT #2 или INPUT #3 - невозможны.

В отличие от них канал "K" может использоваться и для ввода и для вывода (из этого озера река может вытекать, как Нева из Ладоги).

INPUT #0 - это самая обычная команда INPUT.

Возможен и вывод:

PRINT #0 обеспечит печать Вашего сообщения в нижних двух строках экрана, которые выполняют роль "системного окна". Это те самые две строки, в которых появляется информация при работе команды INPUT.

В "родном" "Спектруме-48" есть еще один стандартный канал - "R", но из Бейсика он не достижим и мы пока его отставим, вернемся к нему позже.

Прочие каналы.

После подключения дополнительной периферии, располагающей своим ПЗУ, к стандартным каналам могут добавляться дополнительные. Так, например, подключение интерфейса-1 ("ZX-Interface One") создает несколько новых каналов:

- "M" - канал микродрайва;
- "N" - канал локальной сети;
- "T" - канал принтера для печати листинга программ (коды выше 165 интерпретируются как токены ключевых слов "Спектрума");
- "B" - канал принтера для печати данных (все коды интерпретируются по значению).

Вот практически и все каналы, возможные для "Спектрума" на стандартном оборудовании. Но это не значит, что у Вас нет дополнительных возможностей. Вы можете создавать свои каналы в оперативной памяти и эффективно использовать их. Этим мы с Вами и займемся.

Листинг 6.

Процедура служит для открывания нового пользовательского канала. Это выполняется путем создания блока информации о канале. При входе в эту процедуру регистр A'(альтернативный) должен содержать номер назначенного потока для данного канала, регистр A(основной) - код имени данного канала (код ASCII), BC - длину блока информации о канале, DE - адрес процедуры ввода (INPUT), HL - адрес процедуры вывода (PKINTJ, а регистр IX - адрес процедуры, используемой для выгрузки данных из буфера или адрес 0052 HEX, если работа с буфером не нужна.

```
                        ORG B06D
C5          OPEN_NEW    PUSH BC                 ;Запоминаем на стеке
DDE5                    PUSH IX                 ;входные значения
F5                      PUSH AF                 ;регистров процессора.
D5                      PUSH DE
E5                      PUSH HL
C5                      PUSH BC
08                      EX AF,A'F'              ;A-номер потока.
CD2117                  CALL 1721,STR_DATA_A    ;Загрузка в BC данных по
                                                ;этому потоку из системной
                                                ;таблицы STRMS.
78                      LD A,B
B1                      OR C                    ;Проверка адреса на ноль.
C1                      POP BC
2602                    JR Z,OPEN_NEW_2         ;Переход, если поток не
                                                ;подключен.
CF17                    RST 08                  ;Вызов процедуры обработки
                        DEFB 17                 ;ошибок с кодом перехвата
                                                ;17, что означает:
                                                ;"0: invalid stream" ("Не-
                                                ;верно задан поток").
E5          OPEN_NEW_2  PUSH HL
2A535C                  LD HL,(PROG)            ;(PROG) - начало БЕЙСИКа.
2B                      DEC HL                  ;Теперь HL указывает на
                                                ;байт, содержащий маркер
                                                ;80H и отмечающий конец
                                                ;области информации о каналах.
CD5516                  CALL 1655,MAKE_ROOM     ;Выделение места под новый
                                                ;блок информации о канале.
23                      INC HL                  ;Теперь HL указывает на
                                                ;начало нового информаци-
                                                ;онного блока.
22515C                  LD (CURCHL),HL          ;Канал делается текущим и
E5                      PUSH HL                 ;адрес начала его информа-
DDE1                    POP IX                  ;ционного блока переводит-
                                                ;ся в IX.
23                      INC HL                  ;HL - указывает на 2-ой
                                                ;байт блока.
ED4B4F5C                LD BC,(CHANS)           ;BC указывает на начало
                                                ;области CHANS.
A7                      AND A                   ;Обнуление флага C как по-
                                                ;дготовка к операции SBC.
ED42                    SBC HL, BC              ;Вычисленное значение
                                                ;является тем адресом, ко-
                                                ;торый должен быть занесен
                                                ;в таблицу данных по
                                                ;потокам STRMS.
EB                      EX DE,HL                ;Теперь оно - в DE,
E1                      POP HL                  ;HL содержит адрес требу-
                                                ;емой STRMS_n.
73                      LD (HL),E               ;Отправляем данные
23                      INC HL                  ;по потоку на свое место
72                      LD(HL),D                ;в STRMS_n.
DDE5                    PUSH IX
```

```
E1                      POP HL                  ;HL-адрес блока информации
                                                ;о канале.
D1                      POP DE                  ;DE - адрес процедуры
                                                ;вывода.
C1                      POP BC                  ;BC -адрес процедуры ввода.
CDAEB0                  CALL B0AE,OPEN_STORE    ;Сохранение этой инфор-
                                                ;мации в блоке информа-
                                                ;ции о канале.
F1                      POP AF                  ;A содержит имя канала.
77                      LD (HL),A               ;Записали его в блок.
23                      INC HL                  ;Напомним: 1234 - сигнал
3634                    LD (HL),34              ;о том, что данный канал -
23                      INC HL                  ;пользовательский, а не
3612                    LD (HL),12              ;стандартный.
23                      INC HL                  ;
D1                      POP DE                  ;DE-адрес процедуры, за-
                                                ;крывающей буфер.
C1                      POP BC                  ;BC-длина блока информации
                                                ;о канале.
73          OPEN_STORE  LD (HL),E               ;Помещение в блок информа-
23                      INC HL                  ;ции о канале адреса
72                      LD (HL),D               ;закрывающей процедуры.
23                      INC HL
71                      LD (HL),C               ;Помещение в блок информа-
23                      INC HL                  ;ции о канале длины блока
70                      LD (HL),B               ;информации.
23                      INC HL
C9                      RET                     ;Возврат в вызывающую
                                                ;процедуру.
```

Область информации о каналах

Итак, как же все это работает? Вся концепция потоков и каналов базируется на области памяти, называемой "область информации о канале" (CHANNEL INFORMATION AREA).

Эта область памяти расположена непосредственно перед БЕЙСИК-областью, чуть ниже нее. То есть лежит между системными переменными и БЕЙСИКом. Начинается она с адреса, хранимого в системной переменной CHANS (23631, 2 байта)(5C4FH) и заканчивается байтом, в котором стоит маркер 80H. Далее уже идет Ваша БЕЙСИК-программа, на которую указывает PROG (23635, 2 байта)(5C53H).

Для того, чтобы создать свой канал Вам практически надо переорганизовать данные в этой области, может быть и раздвинуть эту область, сдвинув вверх маркер и поменяв значение PROG и разместить где-либо в памяти две процедуры. Одну - для обеспечения ввода в Ваш канал (INPUT) и вторую - для обеспечения вывода (PRINT).

Каждый канал должен иметь блок информации о канале (CHANNEL INFORMATION BLOCK). Этот блок и располагается в области информации о каналах.

В этом блоке содержится вся информация, необходимая для того, чтобы канал мог функционировать. Четыре стандартных канала "K","S","P" и "R" имеют каждый по пятибайтному блоку, но это исключение. Все остальные каналы, в том числе и те, которые создадите Вы, должны иметь не менее, чем по 11 байтов в этом блоке на каждый канал.

По четырем стандартным каналам эти блоки выглядят так:

Байты 0 и 1 - адрес процедуры вывода (PRINT#).

Байты 2 и 3 - адрес процедуры ввода (INPUT#).

Байт 5 - имя канала (код одной буквы "K", "S","R" или "P").

При подключении стандартной периферии, например ИНТЕРФЕЙСа-1 этот блок имеет длину 11 байтов и адресация к ним производится помещением в регистровую пару процессора IX базового адреса начала блока.

IX+0 (2 байта) - адрес процедуры обработки ошибок 0008.

IX+2 (2 байта) - адрес процедуры обработки ошибок 0008.

Листинг 7.

В области информации о каналах данная процедура выполняет поиск блока с заданным именем.

Данная процедура используется при работе процедуры V_OPEN (B966), см. Листинг 20.

```
                        ORG B594
DD2A4F5C    SEARCH_CH_ALL   LD IX,(CHANS)       ;B IX - базовый адрес на-
                                                ;чала области информации
                                                ;о каналах.
011400                      LD BC,0014
57                          LD D,A              ;D содержит ASCII-код
                                                ;имени канала.
DD09        SCH_CH_LOOP     ADD IX,BC           ;IX указывает на адрес
                                                ;очередного блока.
DD7E00      SEARCH_CH       LD A,(IX+00)
FE80                        CP 80               ;Напомним, что 80H - мар-
                                                ;кер, отмечающий конец
                                                ;области информации о
                                                ;каналах.
37                          SCF                 ;Включение флага C.
C8                          RET Z               ;Выход с включенным C-фла-
                                                ;гом, если блок канала C
                                                ;таким именем не найден.
DD7E04                      LD A,(IX+04)        ;A содержит ASCII-код
                                                ;имени канала.
BA                          CP D                ;Сравнили с искомым.
C8                          RET Z               ;Выход, если найден.
DP4E09                      LD C,(IX+09)        ;BC - длина блока инфор-
DD460A                      LD B,(IX+0A)        ;мации о канале.
18EA                        JR SCH_CH_LOOP      ;Переход в начало цикла
                                                ;для продолжения поиска.
```

Листинг 8.

Нижеприведенная процедура служит для обслуживания такой "хитрой" конструкции как "трехбайтный" регистр. Ранее мы писали, что в 128-килобайтных моделях для хранения адреса недостаточно двух байтов и приходится привлекать еще и кодовый номер страницы. Такой "адрес" хранится в трех регистрах процессора, например, B,H,L.

Процедура выполняет операцию декремента (DEC BHL).

```
                    ORG B70E
2B          DEC_BHL DEC HL                  ;Декремент пары HL.
78                  LD A,B                  ;в A был код страницы.
FE05                CP 05                   ;Возврат, если это обычное
C8                  RET Z                   ;ОЗУ.
CB74                BIT 6,H                 ;Возврат, если не было пе-
C0                  RET NZ                  ;рехода между страницами.
CBF4                SET 6,H
05                  DEC B
C9                  RET
```

IX+4 (1 байт) - имя канала.

IX+5 (2 байта) - адрес процедуры PRINT#, которая находится в "теневом" ПЗУ интерфейса.

IX+7 (2 байта) - адрес процедуры INPUT#, которая находится в "теневом" ПЗУ интерфейса.

IX+9 (2 байта) - длина блока информации о канале (не менее 000B).

IX+0B (длина любая) - любая дополнительная информация.

Блоки информации о пользовательских каналах тоже лучше создавать и обслуживать,

используя адресацию через индексный регистр IX. Тогда такой блок имеет следующий вид:

IX+0 (2 байта) - адрес процедуры вывода (PRINT#).

IX+2 (2 байта) - адрес процедуры ввода (INPUT#).

IX+4 (1 байт) - имя канала,

IX+5 (2 байта) - число 1234, свидетельствующее о том, что этот канал не стандартный, а пользовательский.

IX+7 (2 байта) - адрес закрывающей процедуры (CLOSE#).

IX+9 (2 байта) - длина блока информации о канале (не менее 000B).

IX+0B (длина любая) - любая дополнительная информация.

Использование процедур ПЗУ

Процедура вывода, содержащаяся в ПЗУ (RST 10H) и процедура ввода INPUT_AD (15E6H) работают со стандартными каналами. Поэтому при выводе происходит сразу обращение в первые два байта блока информации о каналах для поиска адреса печатающей процедуры, а при вводе - обращение в 3-ий и 4-ый байты для поиска адреса читающей процедуры.

Если Вы используете стандартную периферию, например Интерфейс-1, то в этих байтах содержится адрес 0008 (адрес процедуры обработки ошибки). При попадании туда происходит "впечатывание" страницы "теневого" ПЗУ вместо стандартного, а уже из него выполняется выбор процедур печати и чтения (байты IX+5 ... IX+8).

И снова о потоках

Теперь, когда мы разобрались с каналами и поняли, что это не так сложно, что это всего лишь еще один способ организации памяти компьютера и взаимодействия процедур друг с другом, вернемся к потокам.

Листинг 9.

Данная процедура предназначена примерно для тех же целей, что и знаменитая команда процессора LDIR и служит для быстрого перемещения значительных блоков памяти в пределах RAM-диска.

Поскольку адресация в дополнительной памяти 128-килобайтных машин невозможна через двухбайтный регистр, эта процедура введена для обслуживания трехбайтных конструкций типа BHL, B'H'L', CDE и др.

Процедура выполняет три функции:

1) Выполняет декремент BHL и B'H'L';

2) перегружает байт из BHL в B'H'L';

3) если BHL не равен CDE, переход на пункт 1.

```
                        ORG B71A
CD0EB7    V_TRANSFER    CALL B70E,DEC_BHL       ;Декремент BHL.
D9                      EXX
CD0EB7                  CALL B70E,DEC_BHL       ;Декремент B'H'L'.
D9                      EXX
78                      LD A,B                  ;В A стал код страницы
                                                ;источника.
CDFFB6                  CALL B6FF,V_PAGE        ;Впечатывание страницы ОЗУ
7E                      LD A, (HL)              ;Прием перебрасываемого
                                                ;байта и
F5                      PUSH AF                 ;сохранение его на стеке.
D9                      EXX
7B                      LD A,B                  ;В A стал код страницы
                                                ;назначения.
CDFFB6                  CALL B6FF,V_PAGE        ;Впечатывание страницы ОЗУ
F1                      POP AF                  ;Восстановление пересылае-
77                      LD (HL),A               ;мого байта и пересылка.
D9                      EXX
78    V_TRANSFER_2      LD A,B                  ;В A стал код страницы
                                                ;источника.
```

```
B9                      CP C
20E6                    JR NZ,V_TRANSFER        ;На повтор цикла.
ED52                    SBC HL,DE               ;Включение флага Z, если
                                                ;все байты переданы.
19                      ADD HL,DE               ;Эта операция на Z флаг
                                                ;не влияет.
20E1                    JR NZ,V_TRANSFER        ;На повтор цикла, если не
                                                ;все байты переданы.
C9                      RET
```

Листинг 10.

Эта вспомогательная процедура рассчитывает адрес при страничной организации памяти, отстоящий от адреса, содержащегося в комплексе AHL на величину, содержащуюся в паре BC.

Предполагается, что величина в BC не превосходит размера одной страницы – 4000H (16 килобайт).

```
                        ORG B73A
09          ADD_AHL_BC  ADD HL,BC               ;16-разрядное сложение.
FE05                    CP 5                    ;Сравниваем код страницы с
                                                ;числом 5.
C8                      RET Z                   :Возврат, если это стандартное ОЗУ.
CB74                    BIT 6,H                 ;Возврат, если не было пе-
C0                      RET NZ                  ;рехода между страницами.
CBFC                    SET 7,H
CBF4                    SET 6,H
3C                      INC A                   ;Увеличиваем код страницы,
                                                ;если был переход.
C9                      RET
```

Среди системных переменных компьютера есть переменная STRMS. Ее адрес - 23568 (5C10H). Ее назначение - указание на адреса каналов, подключенных к потокам. Длина этой системной переменной - 38 байтов и если говорить откровенно, то никакая это не переменная, а самая настоящая указательная таблица, в которой каждому потоку отданы два байта, содержащие адрес канала, к которому данный поток подключен.

Правда, у внимательного читателя может сразу возникнуть вопрос - почему же потоков 16, да на каждый по 2 байта, итого 32, а STRMS содержит 38 байтов.

Дело в том, что есть еще три потока, которые из БЕЙСИКа вам недоступны - только из машинного кода. Эти потоки занимаются своими "внутренними" делами при работе ПЗУ. Это "минус третий" поток (FD), "минус второй" (FE) и "минус первый" (FF). Таким образом, таблицу STRMS можно представить, как 19 двухбайтных системных переменных:

STRMS_FD 5C10 (23568)
STRMS_FE 5C12 (23570)
STRMS_FF 5C14 (2357E)
STRMS_00 5C16 (23574)
STRMS_01 5C16 (83576)
.....
STRMS_0F 5C36 (23606)

Поток FD подключен к каналу "K" и не должен переподключаться. Аналогично поток FE подключен к каналу "S". Интересен поток FF, подключенный к "внутреннему" "Спектрумовскому" каналу "R", который отвечает за динамическое копирование информации из одних областей памяти компьютера в другие, производя при этом "раздвигание" информации для вставки новой в середину имеющейся. Мы об этом писали в разделе "Секреты ПЗУ", когда рассматривали процедуры БЕЙСИКовского редактора.

Итак, с помощью таблицы переменных STRMS выполняется привязка потоков к каналам. Если какая-либо переменная из набора STRMS содержит 0000, то это означает, что к данному потоку не подключен ни один канал, иначе говоря, канал закрыт. Если же там не ноль, значит канал открыт и данный поток подключен к этому каналу. Фактически же поток подключен к тому каналу, информационный блок которого начинается с адреса, на который

указывает системная переменная CHANS (23631) плюс величина, содержащаяся в STRMS для данного потока минус единица:

(CHANS) + (STRMS_n) - 1

Для тех, кто не знаком с программированием в машинных кодах, укажем, что круглые скобки в этой формуле означают, что речь не идет об адресе самой системной переменной CHANS, а об адресе, на который она указывает, т.е. который в ней хранится, то же и (STRMS_n).

Листинг 11.

Процедура предназначена для поиска в каталоге RAM-диска файла с именем, которое задано в блоке информации о канале. Если файл с таким именем в каталоге не существует, выдается системное сообщение об ошибке. При успешном окончании работы процедуры на выходе в индексной паре IX устанавливается адрес, указывающий на требуемую информацию в каталоге. При входе там должен быть выставлен адрес места расположения блока информации о канале.

```
                        ORG B747
DDE5        FIND_FILE   PUSH IX                ;Переброска адреса блока
E1                      POP HL                 ;через стек.
010E00                  LD BC,000E             ;Имя в блоке начинается с
                                               ;байта IX+0E.
09                      ADD HL,BC              ;HL указывает на имя.
0E0A                    LD C,0A                ;Длина имени - 10 знаков.
11675B                  LD DE,5B67,NSTR_1      ;В системной переменной
                                               ;NSTR_1 должно содержать-
                                               ;ся имя искомого файла.
EDB0                    LDIR                   ;Переброска имени из блока
                                               ;информации в системную
                                               ;переменную.
CD08B7                  CALL B708,V_FIND       ;Вызов процедуры ПЗУ для
                                               ;поиска по каталогу.
C0                      RET NZ                 ;Выход, если файл найден.
CDFCBP                  CALL B6FC,V_ERROR      ;Вызов системной процеду-
23                      DEFB 23                ;ры генерации сообщения об
                                               ;ошибке с кодом перехвата
                                               ;23H (h "File does not
                                               ;exist") - "Файл не су-
                                               ;ществует."
```

Листинг 12.

Данная процедура выполняет важную задачу. Мы организовали в блоке информации о канале "V" полукилобайтный буфер и эта процедура устанавливает взаимное соответствие между данным буфером и областью оперативной памяти на RAM-диске. Эта область представляет тем самым как бы последовательность полукилобайтных секторов.

На входе в эту процедуру необходимо, чтобы в IX был установлен адрес начала блока информации о канале "V".

На выходе из этой процедуры устанавливаются:

- в регистрах BHL - адрес первого байта, следующего за последним сектором, заполненным на RAM-диске (для данного файла);

- в регистрах CDE - адрес начала сектора на RAM-диске;

- в регистрах B'H'L' адрес байта, следующего за буфером канала "V" (в блоке информации о канале).

```
                        ORG B75D
DDE5        V_MATCH     PUSH IX                ;Сохранили на стеке адрес
DD7E0D                  LD A,(V_CHREC)         ;блока информации о канале
F5                      PUSH AF                ;и номер записи в файле.
CD47B7                  CALL B747, FIND_FILE   ;Теперь IX указывает на
                                               ;запись о данном файле в
                                               ;каталоге, а
```

```
C1                      POP BC                  ;B - на номер записи в файле.
CB20                    SLA B                   ;Удвоение содержимого B.
0E01                    LD C,01                 ;В BC теперь 200H*(номер записи)+1
37                      SCF                     ;Сигнал "конец блока" и
08                      EX AF,A'F'              ;сохранение его.
DD6E0D                  LD (L,SF_LEN)           ;В AHL устанавливается
DD560E                  LD (H,SF_LEN+1)         ;длина

DD7E0F                  LD (A,SF_LEN+2)         ;файла.
A7                      AND A                   ;Обнуление флага переноса,
                                                ;чтобы не искажать следу-
                                                ;ющую операцию.
ED42                    SBC HL,BC               ;Вычитание с "займом"
DE00                    SBC A,00                ;Вычитание "займа" из A,
                                                ;теперь в AHL - размер
                                                ;"остатка" файла.
A7                      AND A                   ;проверка на ноль.
2008                    JR NZ,V_M_N_EOF         ;Переход, если старший
                                                ;байт AHL еще не обнулен.
110102                  LD DE,0201
ED52                    SBC HL,DE               ;Сравнение размера "остат-[1]
19                      ADD HL,DE               ;с длиной буфера. Если бо-
3604                    JR C,V_M_EOF            ;льше, то включится флаг
                                                ;переноса и тогда переход.
210002      V_M_N_EOF   LD HL,200               ;Длина записи max 200H.
08                      EX AF,A'F'              ;Запомнили выключенный
                                                ;флаг C , означающий "Не
                                                ;конец записи".
EB          V_M_EOF EX  DE,HL                   ;DE - длина записи.
DD6E0A                  LD L,(SF_START)         ;В AHL устанавливается
DD660B                  LD H,(SF_START+1)       ;адрес начала
DD7E0C                  LD A,(SF_START+2)       ;файла.
CD3AB7                  CALL B73A,ADD_AHL_BC    ;CM. B73A.
C5                      PUSH BC                 ;200*(номер записи)+1
D5                      PUSH DE                 ;Длина записи.
E5                      PUSH HL                 ;Адрес сегмента и код
F5                      PUSH AF                 ;его страницы.
42                      LD B,D
4B                      LD C,E
CD3AB7                  CALL B73A,ADD_AHL_BC    ;В AHL помещается адрес
                                                ;очередного сектора на
                                                ;RAM-диске.
47                      LD B,A
F1                      POP AF
4F                      LD C,A                  ;В CDE адрес сегмента
D1                      POP DE                  ;и код его страницы.
D9                      EXX                     ;Включение альтернативных
                                                ;регистров.
D1                      POP DE                  ;D'E'- длина записи.
C1                      POP BC                  ;B'C'=200"(номер записи)+1
DDE1                    POP IX                  ;В IX - адрес блока инфор-
                                                ;мации о канале.
DD7319                  LD (V_RECLRN),E         ;Заносим длину записи в
DD731A                  LD (V_RECLRN+1),D       ;V_RECLEN.
DDE5                    PUSH IX                 ;в HL - адрес блока инфор-
E1                      POP HL                  ;мации о канале.
011B00                  LD BC,001B              ;В блоке информации о ка-
                                                ;нале до буфера 1B байтов.
09                      ADD HL,BC               ;H'L' указывает на буфер
                                                ;канала "V".
19                      ADD HL,DE               ;H'L' указывает на байт,
                                                ;следующий за текущей
                                                ;записью.
```

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

```
0605                    LD B,05                 ;B'=05 - сигнал "страница
                                                ;стандартного ОЗУ".
D9                      EXX                     ;Основной набор регистров.
DDCB188E                RES 1,(V_CHFLAG)        ;Сигнал "не последний
                                                ;блок в файле."
08                      EX AF,A'F'              ;Напомним - флаг C несет
                                                ;информацию о конце файла.
D0                      RET NC                  ;Возврат, если блок не по-
                                                ;следний.
DDCB18CE                SET 1,(V_CHFLAG)        ;Сигнал "конечный блок
                                                ;файла".
C9                      RET
```

Из всего вышесказанного вытекает одна тонкость. Оказывается с программистской точки зрения проще создать и открыть новый канал, чем закрыть его. Действительно, если Вы решили создать канал, Вы в конце области информации о каналах, на которую указывает (CHANS), припишете блок информации о канале (11 байтов или более) и для желаемого Вами потока #n запишете в соответствующую переменную STRMS_n указание на этот блок.

А что же, если Вы хотите закрыть канал? Тогда в соответствующее значение STRMS_n Вы запишете нули, но надо еще уничтожить блок информации о канале. Хорошо, когда он - последний. Убрали его и все. А если он находится в середине, тогда придется все вышестоящие блоки сдвигать вниз с изменением при этом содержимого STRMS_n+1, STRMS_n+2 .....

Как закрыть канал.

Итак, если мы хотим научиться сами создавать свои каналы, подключать к ним потоки, т.е. открывать эти каналы и использовать их для своих целей, то прежде всего надо научиться закрывать свои каналы, для чего и служит предлагаемая ниже процедура CLOSE_NEW (листинг 1).

При входе в эту процедуру регистр A микропроцессора должен содержать номер подключаемого потока. Если канал уже закрыт или если канал не является каналом, созданным пользователем, то процедура выполняет немедленный возврат и ничего не делает.

С другой стороны, если канал открыт (поток подключен) и это пользовательский канал, процедура "вычищает" память, занятую блоком информации о канале, смещает вышестоящие блоки вниз ("убирает мусор") и перестраивает системные переменные в таблице STRMS.

При работе процедуры, учитывается также состояние флага переноса (флаг C флагового регистра F). Если при входе в процедуру через адрес CLOSE_CL флаг C включен, значит в буферах есть данные и перед тем, как закрыть канал надо их очистить, выдав все данные, чтобы информация не пропала. Если флаг C выключен, то наличие данных в буферах можно проигнорировать.

Если однозначно необходимо буфер отгрузить, то входить в процедуру надо через точку входа CLOSE_NEW.

Ниже представлена также вспомогательная процедура CLEAR CHANS (листинг 2), служащая для закрывания всех пользовательских каналов. При вызове этой процедуры все данные, оставшиеся в буферах, уничтожаются.

Листинг 13.

Процедура настраивает буфер канала, подготавливаясь к работе с файлом, открытым для чтения.

```
                        ORG B7C5
CD5DB7      V_ASSIGN    CALL B75D,V_MATCH
CD30B7                  CALL B730,V_TRANSFER_2  ;Копирует данные в буфер.
DD360B00    V_BUF_EXIT  LD(V_CHBYTE),00         ;обнуляет указатель дан-
DD360C00                LD(V_CHBYTE+1),00       ;ных в буфере.
C9                      RET
```

Эти программы нерелоцируемы, то есть их загружать можно только в те адреса памяти, для которых они написаны, и которые стоят в директиве АССЕМБЛЕРа ORG. Это сделано потому, что за этими процедурами пойдут другие и все они должны работать совместно, как единое целое.

Файлы последовательного доступа на RAM-диске

Теперь, когда мы научились, в принципе, закрывать созданные каналы, можно попробовать их создавать. В качестве примера мы рассмотрим идее о том, как владельцы 128-килобайтных машин могут использовать их потенциал, прибегнув к концепции потоков и каналов.

Что еще можно делать с помощью каналов и потоков? Мы назовем несколько направлений. Например, организуется нестандартная печать, скажем 45, или 55 или 60 символов в строке. С их же помощью организуется, например, работа с "окнами" так, как это сделано в "МЕГАБЕЙСИКе" или, скажем, в "ЛАЗЕР-БЕЙСИКе". Вы можете расширить БЕЙСИК своего компьютера, придумав для него несколько новых команд. И самое интересное, пожалуй, - это организация обмена между разными машинами. Мы бы с удовольствием привели пример именно из этой области, но для этого пришлось бы освещать работу и этой (другой) машины, что никак не входит в наши планы. Приходится ограничиться вопросами обмена данными между различными областями одной машины.

Итак, в качестве примера мы создадим канал, позволяющий работать с файлами последовательного доступа на RAM-диске 128-килобайтных компьютеров.

Примечание.

Это, конечно не значит, что Вам надо завтра сразу все бросить и побежать покупать 128-килобайтную машину, пока Вам кто-то не сумеет объяснить, что же хорошего, кроме лишнего десятка "игрушек" Вы получите от этих дополнительных 80 килобайтов, к которым у процессора нет и не может быть прямой адресации.

Кстати, уникальной особенностью этих машин (в импортном исполнении) является трехголосый звуковой синтезатор, ради которого все и затевалось. Пока у нас в стране нет никаких аналогов этой микросхемы, все попытки "продать" пользователю 128-килобайтную модель по цене хотя бы на 200 рублей дороже 48-килобайтного аналога - это просто эксплуатация его доверчивости и неинформированности.

Эта задачка так и осталась бы академической, но есть и факты: темп продаж 128 килобайтных машин на Западе так и не достиг и 10% темпа продаж обычных 48-килобайтных машин в период их расцвета. Даже попытка встроить дисковод в "ZX-Spectrum+3" ничего кроме очередного разочарования не дала и количество программ, поддерживаемых этой версией, можно пересчитать на пальцах одной руки.

Вообще история сэра К. Синклера это такой роман с потрясающими взлетами и глубокими падениями, изучая который можно уберечь пользователя от излишних тысячных затрат, а производителя - от многомиллионных.

Листинг 15.

Данная процедура принимает из "V"-канала единичный символ и помещает его в аккумулятор процессора.

```
                        ORG B803
CD005B      V_INKEY     CALL 5B00,SWAP          ;"Впечатывание" ПЗУ-0.
2A5A5B                  LD HL,(RETADDR)
E5                      PUSH HL
D9                      EXX
C5                      PUSH BC
D5                      PUSH DE
E5                      PUSH HL
DD2A515C                LD (IX,CURCHL)          ;IX указывает на начало
                                                ;блока информации о
                                                ;канале.
```

```
DDCB1846                BIT 0,(V_CHFLAF)        ;Если это READ-файл -
2804                    JR Z,V_INKEY_2          ;то перевод. Иначе
CDFCB6      V_ERROR_1   CALL B6FC,V_ERROR       ;генерация сообщения об
1D                      DEFB 1D                 ;ошибке с кодом перехвата
                                                ;0D ("b: Wrong file type")
                                                ;"Неверный тип файла".
DD5E0B      V_INKEY_2   LD E,(V_CHBYTE)         ;В DE - позиция следующего
DD560C                  LD D,(V_CHBYTE+1)       ;читаемого байта.
DDCB184E                BIT 1, (V_CHFLAG)       ;Проверка на содержание в
280F                    JR Z,V_INKEY_RD         ;блоке метки "конец файла"
DD6E19                  LD L,(V_RECLEN)         ;В HL - длина текущей за-
DD661A                  LD H,(V_RECLEN+1)       ;записи в файле.
A7                      AND A
ED52                    SBC HL,DE               ;Проверка на достижение
2004                    JR Z,V_INKEY_RD         ;последней записи.
CDFCB6                  CALL B6FC,V_ERROR       ;Генерация сообщения об
07                      DEFB 07                 ;ошибке с кодом перехвата
                                                ;07 ("8: End of file")
                                                ;"Конец файла".
DDE5        V_INKEY_RD  PUSH IX
E1                      POP HL                  ;HL указывает на блок информации
                                                ;по каналу "V".
011B00                  LD BC,001B
09                      ADD HL,BC               ;HL указывает на буфер.
19                      ADD HL,DE               ;HL указывает на очеред-
                                                ;ной читаемый символ.
7E                      LD A,(HL)               ;В аккумулятор поступает
                                                ;символ от INKEY$.
F5                      PUSH AF
13                      INC DE                  ;Передвинули указатель.
DD730B                  LD (V_CHBYTE),E         ;Запомнили положение
DD7E0C                  LD (V_CHBYTE+1),D       ;указателя.
15                      DEC D                   ;Если D=2 значит пора об-
15                      DEC D                   ;новлять бУФер.
2006                    JR NZ,V_INKEY_EXIT
DD340D                  INC (V_CHREC)           ;Увеличили номер записи.
CDC5B7                  CALL B705,Y_ASSIGN
F1          V_INKEY_EXIT POP AF                 ;В A - только что
                                                ;прочитанный байт.
37                      SCF
E1          V_INPUT_EXIT POP HL
D1                      POP DE
C1                      POP BC
D9                      EXX
E1          V_EXIT      POP HL                  ;В HL - адрес возврата в ПЗУ-0.
225A5B                  LD (RETADDR),HL         ;Настройка системной пере-
                                                ;менной на адрес возврата.
C3005B                  JP 5B00,SWAP            ;"Впечатывание" ПЗУ-1 и
                                                ;возврат.
```

Те, кто знаком с такими носителями информации как флоппи-диск или микродрайв, представляют, что такое файлы последовательного доступа, открытые для чтения (READ-файл) или для записи (WRITE-файл). В файл, открытый для записи, Вы можете "впечатывать" текст, затем этот файл может быть закрыт (CLOSE) и снова открыт (OPEN) для чтения. Вводить информацию (текстовую и числовую) в такой файл Вы сможете прямо из БЕЙСИКа.

Нужны ли Вам "файлы последовательного доступа" или нет - дело Ваше, но показать, каким путем идея потоков и каналов используется в приложениях мы должны, а Вы сами разберетесь, как Вам поступить.

RAM-диск

Файлы, размещенные в RAM-диске, который нередко еще называют виртуальный диском, работают точно так же, как и файлы, размещенные например на гибких дисках,

когда Вы открываете (OPEN) на RAM-диске файл для записи, то в верхних страницах памяти компьютера создается файл с заданным именем. В него можно вводить какие-то данные. Когда файл закрыт (CLOSE), в него нельзя ввести ничего, но можно открыть файл для чтения и тогда из него можно ввести в какую-либо БЕЙСИК-переменную то, что там содержится посредством INPUT.

Как и при работе с обычным диском, файл на RAM-диске должен обязательно быть закрыт после того, как закончился ввод в него или вывод из него. Если файл, открытый для записи, не будет закрыт, то есть вероятность того, что часть данных (а может быть и все) будут безвозвратно утрачены, поскольку специальный буфер не будет очищен (отгружен в файл). Последствия для незакрытого READ-файла не столь неприятные, хотя надо признать, что каждый канал, обслуживающий RAM-диск, занимает пол-килобайта памяти и освободить их, не закрыв файл, нельзя. Поэтому ВСЕГДА ЗАКРЫВАЙТЕ ФАЙЛ, КОГДА ЗАКОНЧИЛИ РАБОТУ С НИМ.

Листинг 16.

Эта крупная и важная процедура выполняет операцию вставки байтов в уже существующий на RAM-диске файл. Те файлы, которые в результате такой вставки должны быть передвинуты, перемещаются и переиндексируются.

При входе в эту процедуру комплекс регистров AHL должен содержать адрес (с кодовой страницей), в который выполняется вставка байта (байтов), а пара BC - длину вставляемого блока.

```
                        ORG B85F
C5          V_MAKEROOM  PUSH BC
E5                      PUSH HL
F5                      PUSH AF
AF                      XOR A                   ;Обнуление аккумулятора,
                                                ;выключение флага C.
67                      LD H,A
6F                      LD L,A                  ;Обнуление HL.
ED42                    SBC HL,BC               ;в результате в AHL - ми-
9F                      SBC A,A                 ;нус число вводимых байтов
CD05B7                  CALL B705,V_SPACE       ;Проверка на достаточность
                                                ;памяти для вставки.
F1                      POP AF                  ;Восстановление
E1                      POP HL                  ;исходных
C1                      POP BC                  ;данных.
C5                      PUSH BC
E5                      PUSH HL
F5                      PUSH AF
3E04                    LD A,04                 ;4-ая страница ОЗУ содер-
CDFFB6                  CALL B6FF,V_PAGE        ;жит каталог.
DD2A835B                LD IX,(SF_NEXT)         ;IX указывает на конец
                                                ;каталога.
DD6E0A                  LD L,(SF_START)         ;AHL указывает на первый
DD660B                  LD H,(SF_START+1)       ;свободный байт простран-
DD7E0C                  LD L,(SF_START+2)       ;ства на RAM-диске.
F5                      PUSH AF
E5                      PUSH HL                 ;Запомнили этот адрес.
CD3AB7                  CALL B73A,ADD_AHL_BC    ;AHL указывает на первый
                                                ;свободный байт который
                                                ;будет после операции
                                                ;вставки блока байтов.
47                      LD B,A                  ;теперь этот адрес в BHL.
D9                      EXX                     ;Теперь он в B'H'L'.
E1                      POP HL                  ;BHL - адрес первого сво-
C1                      POP BC                  ;бодного байта (старый).
F1                      POP AF                  ;ADE - адрес куда идет
D1                      POP DE                  ;вставка.
4F                      LD C,A                  ;Теперь это - CDE.
D5                      PUSH DE                 ;Запомнили
F5                      PUSH AF                 ;этот адрес.
```

```
CD30B7                  CALL B730,V_TRANSFER_2  ;Перемещение байтов.
3E04                    LD A,04                 ;4-ая страница ОЗУ содер-
CDFFB6                  CALL B6FF,V_PAGE        ;жит каталог.
C1                      POP BC                  ;BDE - адрес куда идет
D1                      POP DE                  ;вставка.
DD6E0A      V_MR_LOOP   LD L,(SF_START)         ;AHL указывает на первый
DD660B                  LD H,(SF_START+1)       ;свободный байт (старый)
DD7E0C                  LD L,(SF_START+2)       ;на RAM-диске.
B8                      CP B
382E                    JR C,V_MR_FOUND         ;Переход, если файл нахо-
                                                ;дится до точки вставки
                                                ;(проверили старший байт).
ED52                    SBC HL,DE               ;Переход, если файл нахо-
19                      ADD HL,DE               ;дится до точки вставки
3829                    JR C,V_MR_FOUND         ;(проверили младшие
                                                ;байты).
E3                      EX (SP),HL              ;B DE теперь количество
EB                      EX DE,HL                ;вставляемых байтов.
19                      ADD HL,DE
3005                    JR NC,V_MR_ADDR
CBFC                    SET 7,H
CBF4                    SET 6,H
3C                      INC A                   ;B AHL новый адрес файла
EB          V_MR_ADDR   EX DE,HL
E3                      EX (SP),HL
EB                      EX DE,HL                ;BDE - позиция вставки.
DD750A                  LD (SF_START),L         ;Откорректировали каталог
DD740B                  LD (SF_START+1),H       ;под новые адресные данные
DD770C                  LD (SF_START+2),A       ;файла.
C5                      PUSH BC
011400                  LD BC,0014
DD09                    ADD IX,BC               ;IX - указывает на следую-
                                                ;щий файл.
C1                      POP BC                  ;BDE - позиция вставки.
DD7510                  LD (SF_END),L           ;Откорректировали каталог
DD7411                  LD (SF_END+1),H         ;под новые адресные данные
DD7712                  LD (SF_END+2),A         ;следующего файла.
18C6                    JR V_MR_LOOP            ;Возврат к началу цикла
                                                ;для работы с этим файлом.
DD6E0D      V_MR_FOUND  LD (L,SF_LEN)
DD660E                  LD (H,SP_LEN+1)
DD7E0F                  LD (A,SF_LEN+2)         ;AHL-старая длина файла.
EB                      EX DE,HL
E3                      EX (SP),HL
EB                      EX DE,HL                ;DE-число вставляемых
                                                ;байтов.
19                      ADD HL,DE
CE00                    ADC A,00                ;AHL-новая длина файла.
DD750D                  LD (SF_LEN),L           ;(Откорректировали каталог
DD740E                  LD (SF_LEN+1),H         ;под длину нового
DD770F                  LD (SF_LEN+2),A         ;файла.
78                      LD A,B                  ;A-код страницы места
                                                ;вставки.
42                      LD B,D                  ;BC-число вставляемых
4B                      LD C,E                  ;байтов.
E1                      POP HL                  ;AHL - адрес, куда был
                                                ;вставлен блок.
C9                      RET
```

Как только файл на RAM-диске открыт, он будет внесен в каталог, который Вы можете вызвать из БЕЙСИКа командой CAT!

Принцип, по которому работает приведенная ниже программа, основан на концепции организации электронного диска (RAM-диска) в расширенной памяти 128-килобайтных машин. Вы уже знаете из предыдущих наших материалов, что в этой области памяти можно

хранить программы, данные или машинный код в течение того времени, пока компьютер остается включенным. RAM-диск значительно быстрее любого физического магнитного диска, но зато он "погибает" при выключении питания машины и своевременно должен быть отгружен.

Все это относится и к файлам последовательного доступа, о которых мы ведем речь.

Память на электронном диске организована с помощью каталога. Каталог является как бы указателем к тому, что есть на этой диске. Сам каталог размещается на седьмой странице дополнительной памяти и фактически организован по принципу стека, начинающегося по адресу 7EBFF и расширяющегося вниз. Каждая запись в каталоге занимает 20 байтов. Ниже приведены значения каждого из этих 20 байтов. IX указывает на начало записи. В конце каталога стоит 20-байтный маркер, обозначающий "конец каталога" (хотя на самом деле из него используются только три байта). Системная переменная, имеющаяся в 128-килобайтных машинах SFNEXT, указывает на этот маркер и служит как бы "указателем стека". Таким образом, ведя каталог в требуемом формате, мы можем управлять RAM-диском из машинного кода.

Структура каталога.

IX+00 SF_NAME Имя файла.

IX+0A SF_START Адрес начала файла (с кодом страницы).

IX+0D SF_LEN Длина файла вместе с заголовком.

IX+10 SF_END Адрес байта, следующего за окончанием файла (с кодом страницы).

IX+13 SF_FLAG Флаговый байт. Выключен, если каталог не завершен (т.е. как правило он выключен).

Файлы RAM-диска начинаются на первой странице ОЗУ и развиваются вверх, проходя через страницы ОЗУ - 3,4,6 и 7. Необходимо принимать во внимание, что они в своем развитии не должны перекрыть область, отведенную под каталог и, тем самым, погубить всю информацию.

Чтобы избежать конфуза с такой "странной" нумерацией страниц, введено понятие "кодовой страницы", о чем мы писали в статьях "128 K" (см. стр. 114,115 ИНФОРКОМ). Это означает, что пока один регистр процессора запоминает "код страницы", другой в это время хранит ее реальный физический адрес. Номера этих "кодовых страниц" - 0,1,2,3 и 4, а номер 5 относится к тем файлам RAM-диска, которые размещены в обычных первых 48 килобайтах компьютера.

Канал "V"

Мы назовем наш пользовательский канал, с помощью которого мы будем управлять файлами на RAM-диске, латинской буквой "V" (от слова "виртуальный") и создадим его так, чтобы это управление можно было делать средствами обычного БЕЙСИКа.

Для организации такого канала нам потребуется блок информации о канале размером более 500 байтов, хотя большую его часть будет занимать не сама информация, а буфер. То, что Вы будете засылать в файл на RAM-диске, на самом деле будет поступать не туда, а в этот буфер и перебрасываться в файл только по полному заполнению буфера (или при закрытии канала).

Вам надо иметь в виду, что адрес места расположения файла на RAM-диске не является величиной постоянной. Все файлы могут мигрировать по мере стирания каких-то файлов или по мере вставки новой информации в готовые файлы. Таким образом, они могут перемещаться всякий раз, когда мы с ними работаем. Использование буфера позволяет нам сделать этот процесс непостоянным и, тем самым, сэкономить машинное время.

Ниже приведена структура блока информации о канале "V".

Обратите внимание на то, что используемые в этом блоке переменные V_CHREC, V_RECLEN и первый бит V_CHFLAG используются только при работе с файлами, открытыми для чтения (READ). Прочие же используются с файлами обоего типа.

Листинг 17.

Приведенная ниже процедура V_STORE выполняет перенос содержимого буфера канала "V" из блока информации о канале в соответствующее ему место на RAM-диске.

```
                    ORG B8FE
DD4E0B      V_STORE LD C,(V_CHBYTE)
DD460C              LD B,(V_CHBYTE+1)       ;BC-число байтов в буфере.
DDES                PUSH IX                 ;сохранили адрес блока
                                            ;информации о канале.
C5                  PUSH BC                 ;Сохранили число байтов в
                                            ;буфере.
CD47B7              CALL B747,FIND-FILE     ;IX-адрес информации о
                                            ;файле в каталоге.
C1                  POP BC                  ;Восстановили BC.
DD6E10              LD L,(SF_END)           ;В AHL выставляем адрес
DD6611              LD H,(SF_END+1)         ;первого байта за данным
DD7E12              LD H,(SF_END+2)         ;файлом в каталоге.
CD5FB6              CALL B85F,V_MAKEROOM    ;Выделение места в ОЗУ.
DDE1                POP IX                  ;Восстановили BC.
CD3AB7              CALL B73A,ADD_AHL_BC    ;В AHL выставляем адрес
                                            ;первого байта за пересы-
                                            ;лаемым блоком байтов.
C5                  PUSH BC
47                  LD B,A                  ;То же в BHL,
D9                  EXX                     ;То же в B'H'L'.
DDE5                PUSH IX                 ;Переброска IX
E1                  POP HL                  ;в HL.
011B00              LD BC,001B              ;HL указывает теперь
09                  ADD HL,BC               ;на начало буфера.
C1                  POP BC                  ;Восстановили BC.
E5                  PUSH HL
09                  ADD HL,BC               ;HL указывает теперь
                                            ;на байт за буфером.
D1                  POP DE                  ;Начало буфера.
010505              LD BC,0505              ;Напомним: 05 - код
                                            ;страницы стандартного
                                            ;ОЗУ, в котором и раз-
                                            ;мещен блок информации
                                            ;о канале "V".
CD30B7              CALL B730,V_TRANSFER_2  ;Копирование буфера
76                  LD A,B                  ;A=5
CDFFB6              CALL B6FF,V_PAGE        ;"Впечатывание" стан-
                                            ;дартного ОЗУ.
C3CBB7              JP B7CB,V_BUFF_EXIT     ;Приведение указателя
                                            ;буфера в исходное со-
                                            ;стояние и "выход".
```

Листинг 18.

Процедура, отвечающая за вывод информации. Она выполняет "печать" по каналу "V". Сначала символ, содержащийся в аккумуляторе помещается в буфер и только потом переправляется на RAM диск.

```
                    ORG B92B
CD005B      V_PRINT CALL 5B00,SWAP          ;Впечатывание страницы 0.
2A5A5B              LD HL,(RETADDR)
E5                  PUSH HL                 ;Сохранили адрес возврата.
D9                  EXX
c5                  PUSH BC
D5                  PUSH DE
E5                  PUSH HL
DD2A515C            LD IX,(CURCHL)          ;IX указывает на адрес ин-
                                            ;формации о канале.
DDCB1846            BIT 0,(V_CHFLAG)
CA18B8              JP Z,B818,V_ERROR       ;Ошибка, если это файл для
                                            ;чтения.
DD5E0B              LD E,(V_CHBYTE)         ;DE - количество байтов
DD560C              LD D,(V_CHBYTE+1)       ;в буфере.
DDE5                PUSH IX
```

```
E1                      POP HL                  ;HL - указывает на блок
                                                ;информации о канале.
011B00                  LD BC,001B
09                      ADD HL,BC               ;HL-начало буфера.
19                      ADD HL,DE               ;HL-первый свободный байт.
77                      LD (HL),A               ;Байт - в буфер.
13                      INC DE                  ;Новое число 6-в в буфере.
DD730B                  LD (V_CHBYTE),E         ;Запомнили новое число
DD720C                  LD (V_CHBYTE+1),D       ;байтов в буфере.
15                      DEC D                   ;Если буфер заполнен,
15                      DEC D                   ;он опорожняется
CCF2B8                  CALL Z,B8F2,V_STORE     ;в файл на RAM-диске.
A7                      AND A                   ;Выкл. флага переноса.
C354B8                  JP B854,V_INOUT_EXIT    ;Переход на выходную
                                                ;процедуру.
```

Структура блока информации о канале "V".

IX+00 V_OUT - адрес процедуры вывода информации из файла на RAM-диске (B92B)

IX+02 V_IN - адрес процедуры ввода информации в файл на RAM-диске(B7D4).

IX+04 V_NAME - имя канала ("V")

IX+05 V_IDEN – идентификатор пользовательского канала ("1234").

IX+07 V_CLOSE - адрес процедуры, закрывающей файл (B960).

IX+09 V_LEN - Длина блока информации о канале (021B).

IX+0B V_CHBYTE- указатель буфера

IX+0D V_CHREC - номер записи в файле.

IX+0E V_CHNAME - имя файла.

IX+18 V_CHFLAG- вспомогательные флаги:

Бит 0 - выключен для файла, открытого для чтения и включен для файла, открытого для записи. Бит 1 - включен, если в конце текущей записи стоит маркер "конец файла", иначе выключен. Биты 2...7 - не используются.

IX+19 V_RECLEN- длина текущей записи в буфере.

IX+1B V_BUFFER- буфер, хранящий текущую запись. Ниже мы привели пакет процедур, необходимых для организации вышеописанной концепции. Чтобы открыть файл последовательного доступа на RAM-диске, необходимо предварительно в аккумуляторе процессора выставить номер потока, подключенного к каналу. Имя же самого файла хранится в системной переменной N_STR1 по адресу 5B67 (23399). Далее вызывается процедура V_OPEN (B988). Есть и другие точки входа. Это OPEN_4, OPEN_5, CLOSE_4 и CLOSE_5. Вызовом OPEN_4 Вы открываете на RAM-диске файл с условным именем FILE_1 и подключаете его к четвертому потоку.

Точно так же вызовом OPEN_5 открывается файл последовательного доступа с условным именем FILE_2 и подключается к пятому потоку.

Процедуры CLOSE_4 и CLOSE_5 закрывают эти файлы и отключают эти потоки.

Таким образом, вновь созданный нами канал "V может легко быть задействован из БЕЙСИКа. Нижеприведенная программа (Листинг 3) демонстрирует файлы RAM-диска в работе сначала в качестве WRITE-файлов, а затем в качестве READ-файлов.

Конечно, Вы вовсе не обязаны называть свои файлы именами FILE_1 и FILE_2, также как и не обязаны подключать к ним только потоки #4 и #5. Для этого в пакете процедур имеется более общая точка входа V_OPEN, при входе в которую, как уже было указано, регистр A процессора должен содержать номер подключаемого потока, а системная переменная N_STR1 - имя файла (если имя меньше положенных 10 символов, пробелы делаются последующими).

Листинг 19.

Данная процедура предназначена для закрывания "V" канала. Основным в ней является опорожнение буфера, если в нем что-то есть для файла, открытого для записи. Для файла, открытого для чтения, содержимое буфера может игнорироваться.

```
                        ORG B960
CD005B      V_CLOSE     CALL 5B00,SWAP          ;Впечатывание страницы 0.
2A5A5B                  LD HL,(BETADDR)
E5                      PUSH HL                 ;Запомнили адрес возврата.
DDE5                    PUSH IX
2A3D5C                  LD HL,(ERR_SP)          ;HL указывает на адрес
                                                ;процедуры обработки
                                                ;ошибки.
E5                      PUSH HL                 ;Запомнили указатель.
21FEFF                  LD HL,FFFE
39                      ADD HL,SP               ;В HL помещен указатель
                                                ;стека минус 2.
E33D5C                  LD (ERR_SP),HL          ;Новый адрес возврата.
DDCB1846                BIT 0,(V_CHFLAG)        ;Если файл является файлом
C4F2B8                  CALL NZ,B8F2,V_STORE    ;открытым для записи.
                                                ;Здесь отгружается буфер.
E1                      POP HL                  ;Напомним, что данный
                                                ;адрес при работе V_STORE
                                                ;служил адресом возврата
                                                ;при любой ошибке.
223D5C                  LD (ERR_SP),HL
DDE1                    POP IX
2158E7      V_OC_EXIT   LD HL,2758
D9                      EXX
C358BB                  JP B858,V_EXIT          ;Переход на выход.
```

Листинг 20.

Эта процедура открывается наиболее универсальной точкой входа. Процедура предназначена для открывания канала. При входе в процедуру аккумулятор процессора должен содержать номер потока, подключенного к данному каналу, а в десяти байтах системной переменной N_STR1 должно быть записано имя открываемого файла. Если имя файла менее десяти символов, то пробелы выполняются как последующие.

```
                        ORG B988
CD005B      V_OPEN      CALL 5B00,SWAP          ;Впечатывание страницы 0.
2A545B                  LD H,(RETADDX)          ;Установка адреса возвра-
E5                      PUSH HL                 ;та в ПЗУ-0.
F5                      PUSH AF                 ;Запомнили номер потока.
3E56                    LD A,56                 ;CHR$ 55 = "V"-имя канала.
CD94B5                  CALL B594               ;Проверка на существование
                                                ;такого канала.
381F                    JR C,V_OP_OK            ;Переход, если его нет.
DDES        V_OP_LOOP   PUSH IX
E1                      POP HL                  ;HL указывает на область
                                                ;информации по уже сущес-
                                                ;твующему каналу.
010E00                  LD BC,000E              ;HL указывает
09                      ADD HL,BC               ;на имя файла.
11675B                  LD DE,N_STR1            ;DE-адрес имени файла.
060A                    LD B,0A                 ;0AH=10 DEC-длина имени.
1A          V_OP_NAME   LD A,(DE)
13                      INC DE
2006                    JR NZ,V_OP_RETRY        ;Переход, если имя не
                                                ;совпадает.
10F8                    DJNZ V_OP_NAME          ;Переход на вершину цик-
                                                ;ла для проверки всех
                                                ;десяти символов.
CDFCB6      V_OP_ERROR  CALL B6FC,V_ERROR       ;Генерация сообщения
20                      DEFB 20                 ;"e: File already exists"
                                                ;"Файл уже существует".
1656        V_OP_RETRY  LD D,56                 ;CHR$ 56 = "V"-имя канала.
CDAAB5                  CALL B5AA               ;Поиск еще одного канала
                                                ;с тем же именем "V".
```

```
30E1                    JR NC V_OP_LOOP         ;Возврат, если найден.
CD08B7      V_OP_OK     CALL B708,V_FIND        ;Поиск имени файла на
                                                ;RAM-диске.
F5                      PUSH AF                 ;Сохранение флага Z.
2811                    JR Z,V_OP_CONT          ;Переход, если файл не
                                                ;найден, т.е. Вы откры-
                                                ;ете файл "для записи".
DD6E0A                  LD L,(SF_START)         ;AHL указывает на начало
DD660B                  LD H,(SF_START+1)       ;файла с данным именем
DD7E0C                  LD L,(SF_START+2)       ;на RAM-диске.
CDFFB6                  CALL B6FF,V_PAGE        ;Выбор страницы, содержа-
                                                ;щей первый байт файла.
7E                      LD A,(HL)               ;A - код типа файла.
FE04                    CP 04
20DE                    JR NZ, V_OP_ERROR       ;На обработку ошибки, если
                                                ;это не READ-Файл.
3E05        V_OP_CONT   LD A,05
CDFFB6                  CALL B6FF,V_PAGE        ;"Впечатывание" стандарт-
                                                ;ного ОЗУ.
F1                      POP AF                  ;Возврат флага Z.
08                      EX AF,A'F'
F1                      POP AF                  ;A-номер потока.
08                      EX AF,A'F'              ;A'-номер потока.
F5                      PUSH AF                 ;Сохранение флага Z, кото-
                                                ;рый указывает тип Файла.
3E56                    LD A, 56                ;CHR$ 56 = "V"-имя канала,
011B02                  LD BC,021B              ;длина блока информации о
                                                ;канале "V".
11D4B7                  LD DE,B7D4,V_INPUT      ;DE-адрес процедуры ввода.
212BB9                  LD HL,B92B,V_PRINT      ;HL-адрес процедуры вывода
DD2160B9                LD IX,B9650,V_CLOSE     ;IX-адрес закрывающей про-
                                                ;цедуры.
EF6DB0                  RST 28                  ;Вызов калькулятора.
DEFB 6D                                         ;Это уже не команды ма-
DEFB B0                                         ;шинного кода Z80 - это
                                                ;команды кода калькулятора
                                                ;о котором мы писали в
                                                ;нашем трехтомнике по про-
                                                ;граммированию в машинных
                                                ;кодах.
                                                ;Здесь на стек калькулято-
                                                ;ра помещается адрес про-
                                                ;цедуры OPEN_NEW (B06D).
DDE5                    PUSH IX                 ;Адрес блока информации о
E1                      POP HL                  ;канале переводится из IX
                                                ;в HL.
010B00                  LD BC,000B
09                      ADD HL, BC              ;В HL адрес V_CHBYTE.
70                      LD (HL),B
23                      INC HL
70                      LD (HL),B               ;Обнулили V_CHBYTE.
23                      INC HL
70                      LD (HL),B               ;Обнулили V_CHREC.
23                      INC HL
EB                      EX DE, HL               ;В DE адрес V_CHNAME.
21675B                  LD HL,5B67,N_STR1       ;В HL адрес имени файла.
0E0A                    LD C,0A                 ;0AH=10 DEC-длина имени
EDB0                    LDIR                    ;копирование имени файла
                                                ;в блок информации о канале.
F1                      POP AF                  ;Восстановление флага Z.
2809                    JR Z,V_OP_WRITE         ;Переход, если файл -
                                                ;"для записи".
DDCB1686                RES 0,(V_CHFLAG)        ;Сигнал: "Файл для чтения"
CDC5B7                  CALL B7C5,V_ASSIGN      ;Привязка буфера к RAM
                                                ;диску.
```

```
1840                    JR V_OP_EXIT            ;Переход на выходную
                                                ;процедуру.
DDCB18C6    V_OP_WRITE  SET 0,(V_CHFLAG)        ;Сигнал: "Файл для записи"
CD02B7                  CALL B703,V_NEWCAT      ;Создание новой записи в
                                                ;каталоге.
21FFFF                  LD HL,FFFF
7C                      LD A,H                  ;В аккумуляторе минус 1.
CD05B7                  CALL B705,V_SPACE       ;Проверка на достаточность
                                                ;места для вставки байта.
3E04                    LD A,04                 ;Напомним: каталог - на
                                                ;странице 4.
CDFFB6                  CALL B6FF,V_PAGE        ;Выбор страницы 4.
DD6E0A                  LD L,(SF_START)         ;AHL указывает на первый
DD660B                  LD H,(SF_START+1)       ;свободный байт (старый)
DD7E0C                  LD L,(SF_START+2)       ;на RAM-диске.
F5                      PUSH AF                 ;Запомнили код страницы.
CDFFB6                  CALL B6FF,V_PAGE        ;Выбор страницы, содержа-
                                                ;щей первый свободный байт
F3                      POP AF                  ;AHL указывает на первый
                                                ;свободный байт.
3604                    LD (HL),04              ;Записали 04 в качестве
                                                ;байта "тип файла".
010100                  LD BC,0001
CD3AB7                  CALL B73A,ADD_HL_BC     ;AHL указывает на первый
                                                ;свободный байт.
5F                      LD E,A                  ;То же и EHL.
3E04                    LD A,04
CDFFB6                  CALL B6FF,V_PAGE        ;Выбор страницы 4.
DD7510                  LD (SF_END),L           ;Откорректировали каталог
DD7411                  LD (SF_END+1),H         ;под конец адресных данных
DD7312                  LD (SF_END+2),E         ;данного файла.
CD0BB7                  CALL B70B,V_CATEND      ;Оформили конец каталога.
3E05                    LD A,05
CDFFB6                  CALL B6FF,V_PAGE        ;Выбрали стандартное ОЗУ.
C381B9      V_OP_EXIT   JP B981,V_OC_EXIT       ;Переход на выходную
                                                ;процедуру.
```

Листинг 3

```
1000 REM WRITE FILE DEMO
1010 RANDOMIZE USR 47713: REM открываем OPEN#4, Файл "FILE_1"
1020 RANDOMIZE USR 47720: REM открываем OPEN#5, Файл "FILE_2"
1030 FOR i=1 TO 512
1040 INPUT "": PRINT i
1050 FRINT #4; 2*i
1060 PRINT #5; i*i
1070 NEXT i
1080 RANDOMIZE USR 47736: REM закрываем CLOSE#4, Файл "FILE_1"
1090 RANDOMIZE USR 47740: REM закрываем CLOSE#5, Файл "FILE_2"
1100 REM READ FILE DEMO
1110 RANDOMIZE USR 47713: REM OPEN#4
1120 RANDOMIZE USR 47720: REM OPEN#5
1130 FOR i=1 TO 512
1140 INPUT #4; a
1150 INPUT #5; b
1160 PRINT a,b
1170 NEXT i
1180 RANDOMIZE USR 47736: REM CLOSE#4
1190 RANDOHIZE USR 47740: REM CLOSE#5
1200 STOP
```

Ниже приведены листинги процедур, предназначенные для манипуляции с файлами на RAM-диске 128-килобайтных моделей.

Надо сразу оговориться, что существуют две основные отличающиеся друг от друга версии ПЗУ 128-килобайтных машин. Отличия касаются нулевой страницы ПЗУ. Первая

версия - это более ранняя модель "ZX-Spectrum+128", а вторая - "ZX-Spectrum+2". Поскольку в ПЗУ этих машин адреса некоторых процедур отличаются, то чтобы не писать два отдельных пакета для той машины и для другой, вводится таблица адресов переходов (вектор переходов). Вы можете использовать либо ту, либо другую. Если ПЗУ Вашей модели отличается и от той и от другой версии, то Вы имеете в принципе возможность (если у Вас хватает информации о точках входа своего ПЗУ) подкорректировать эти таблицы (Листинги 4,5) так, как Вам угодно. Во всем остальном приведенный пакет процедур идентичен и для той и для другой модели.

Листинг 21.

Последняя процедура служит для интеграции всего вышеприведенного пакета процедур с БЕЙСИКом компьютера. Приведенные в ней строки являются образцами и Вы можете переделать их под свои конкретные требования. OPEN_4 открывает файл FILE_1 и подключает к нему поток номер 4. OPEN_5 подключает поток 5 к открываемому файлу FILE_2. CLOSE_4 и CLOSE_5 соответственно закрывают четвертый и пятый потоки.

```
                        ORG BA4D
46494C4531  FILE_1      DEFM "FILE1"
2020202020              DEFM "     "            ;Имя файла+5 пробелов.
46494c4532  FILE_2      DEFM "FILE2"
2020202020              DEFM "     "            ;Имя файла+5 пробелов.
3E04        OPEN_4      LD A,04                 ;A - номер потока.
214DBA                  LD HL,FILE_1            ;HL-адрес имени файла.
1805                    JR OPEN_4_5             ;Переход на открывание
                                                ;потока.
3E05        OPEN_5      LD A,05                 ;A - номер потока.
2157BA                  LD HL,FILE_2            ;HL-адрес имени файла.
11675B      OPEN_4_5    LD DE,NSTR1
010A00                  LD BC,000A
EDB0                    LDIR                    ;Кодирование имени файла
                                                ;в системную переменную.
C388B9                  JP B988,V_OPEN          ;Переход на открывание
                                                ;канала.
3E04                    LD A,04
1802                    JR CLOSE_4_5            ;Переход на открывание
                                                ;канала.
3E05                    LD A,05
C300B0                  JP B0000                ;Закрываем каналы.
```

MACHINE CODE

Сегодня своими приемами создания игровых программ в машинных кодах делится наш постоянный читатель Антон Александрович Яицкий.

Нам особенно приятно напечатать эту статью потому, что это первая ласточка от наших отечественных программистов. Ну не все же время читать Стива Тернера. Ведь есть же и у нас прекрасные профессионалы, имеющие свой опыт, свои секреты и приемы и, главное, готовые поделиться с теми, кто делает первые шаги в освоении глубин компьютера.

При создании программ в машинных кодах (например игровых) с использованием АССЕМБЛЕРа Z-80 очень трудно бывает сосредоточиться на самой программе, т.к. приходится рассеивать внимание на создании различных подпрограмм.

Есть весьма простой способ обойти это препятствие. И его алгоритм легко применим.

1) Сначала пишутся независимые подпрограммки - модули, которые являются законченными, т.е. оканчиваются кодом 201 (RET -возврат).

2) Потом они компилируются в машинный код с помощью компилятора (СИ, Ассемблер и т.п.) и размещаются в памяти таким образом, чтобы их координаты были известны.

3) На БЕЙСИКе пишется программа, которая управляет их работой. Это главная программа и ее сразу трудно написать в машинных кодах, поскольку ее приходится постоянно изменять, переделывать, корректировать и т.п.

4) Когда увязка процедур с помощью головной БЕЙСИК-программы закончена, переносим ее на бумагу со всеми точными данными - паузами, переходами, определениями. Это и будет алгоритм основного блока программы, другими словами "телом" будущей игры.

5) Далее переводим эту программу в язык низкого уровня, пользуясь ею как алгоритмом. Т.к. процедуры, вызываемые ею, уже отлажены, то ее отлаживать не так сложно. Во всяком случае нетрудно обнаружить место происхождения той или иной ошибки, раз алгоритм (сценарий игры) перед Вами и процедуры нижних этажей проверены и должны работать. Будут, конечно, ошибки, допущенные при переводе Бейсика в машинный код, но Вы с ними справитесь.

Располагать в памяти компьютера "тело" главного управляющего блока следует так, чтобы вместе с подготовленными ранее подпрограммами она представляла единое целое. После старта она будет вызывать в нужной последовательности те же процедура, что и БЕЙСИК-программа, повторяя ее алгоритм.

Какие при этой должны быть учтены нюансы?

1) Во-первых, есть разница в быстродействии. БЕЙСИК-программа работает гораздо медленнее, чем машинный код. Вследствие этого откомпилированная программа будет работать несколько не так, как ранее отлаженная в БЕЙСИКе ее предшественница. Это может повлиять на динамику управления программой и т.п. Поэтому заранее такое явление необходимо учитывать.

Надо предварительно продумать и определить, для каких частей Вашей программы зависимость от скорости БЕЙСИКа будет наиболее критической. Их лучше сразу написать на Ассемблере или СИ и откомпилировать, а из БЕЙСИКа вызывать через RANDOMIZE USR addr, где addr - адрес расположения данной подпрограммы.

2) Могут возникать ошибки при исполнении переходов. В Ассемблере адрес относительного перехода до инструкции JR n не должен отстоять более, чем на 128 байтов от точки исполнения перехода. Поэтому, если в независимых процедурах использовать эту команду желательно, то при написании головного блока ее использовать нельзя. Следует пользоваться абсолютным переходом JP nn или вызовом CALL.

Какие же модули в первую очередь нуждаются в быстродействии? Это прежде всего все выводы на экран (например спрайтов). Это смена экрана по команде LDIR (LDDR). Это процедуры компрессии/декомпрессии экранов и процедуры, управляющие

вводом/выводом информации.

Когда все будет сделано, Вам останется только дать команду: RANDOMIZE USR begin.

По всей видимости надо дать и несколько примеров перевода операторов БЕЙСИКа в стандартные конструкции в машинных кодах.

1) Часто встречающаяся в БЕЙСИКе команда PAUSE (пауза). Рассмотрим например PAUSE 200 (пауза на 4 секунды).

Для этой цели предлагается процедура WAIT. Она требует, чтобы заранее в регистр A (аккумулятор) было заслано число, определяющее длину этой паузы.

```
            LD A,200
            CALL WAIT
```

Процедура WAIT будет ждать 4 секунды и вернется туда, откуда был сделан вызов. А вот как она выглядит:

```
WAIT        LD HL,23672
            LD (HL),0
WAIT1       CP (HL)
            RET Z
            JR WAIT1
```

Здесь использован младший байт системной переменной FRAMES (23672), которая как известно служит как бы внутренними часами компьютера.

2) Так же полезной может быть и процедура TXT, обеспечивающая вывод текстового сообщения на экран.

В вызывающей процедуре нужно сделать:

```
.....................
            CALL TXT
            DEFB 22
            DEFB y
            DEFB x
            DEFM "ZX SPECTRUM"
```

Вместо y и x Вы подставите те значения, которые обычно в БЕЙСИКе Вы пишете после PRINT AT (x,y). Сама же печатающая процедура может выглядеть например так:

```
TXT         POP HL
TXLOOP      LD A,(HL)
            CP 0
            JR Z,TXEND
            RST 16
            INC HL
            JR TXLOOP
TXEND       INC HL
            PUSH HL
            RET
```

В данном случае процедура использует для печати процедуру ПЗУ RST 16. Использование процедур ПЗУ следует применять шире - это позволяет значительно экономить память и упрощает программирование.

Такой способ написания программ значительно облегчает трудоемкое занятие отслеживания правильности программирования и отладки.

* * *

Наш комментарий

В заключение работы Антона Александровича нам хотелось бы высказать ряд и своих соображений, а также внести некоторые дополнения, полезные тем, кто сейчас делает первые шаги в машинном коде или собирается их делать.

Конечно, приемов организации программирования в машинном коде может быть столько же, сколько и программистов, но как нам кажется, у предлагаемой системы есть ряд очень весомых преимуществ, которые должны заинтересовать наших читателей.

1. Доступность.

Пожалуй, самое приятное в ней для тех, кто освоил БЕЙСИК, и никак не может решиться на освоение АССЕМБЛЕРа - то, что прямо сейчас он скажет сам себе "Все, я уже программирую на Ассемблере", после чего вставит вместо оператора PAUSE в своих программах команду RANDOMIZE USR pause. Затем по адресу pause запишет несколько

байтов из приведенного выше образца и встанет из-за компьютера с гордо поднятой головой. Да, действительно он уже программирует с использованием машинного кода и не заметил, как преодолел сложный психологический барьер.

Постепенно подготавливая одну за другой процедуры, можно незаметно перейти к качественно новому уровню работы с машиной.

2. Наглядность.

Это обстоятельство даже не нуждается в комментариях.

3. Структурность.

Важная особенность состоит в том, что Вы можете готовить программы коллективно.

Предположим, что вместе с группой единомышленников Вы решили подготовить интересную игровую программу. В одиночку на это уйдет год и Вы решили сократить время, распараллелив работу между людьми и поручили каждому подготовку процедур и данных по одному направлению:

- один пишет все процедуры, связанные с выдачей на экран текстовых сообщений;
- другой пишет все процедуры, связанные с управлением программой от клавиатуры или от джойстика;
- третий - управление работой звукового динамика;
- четвертый - процедуры, связанные с экраном - построение экрана, помещение спрайтов, перемещение спрайтов, компрессия и декомпрессия экранов; пятый вообще ничего писать не умеет (он - художник) и рисует в редакторе ARTSTUDIO экран за экраном; а шестой - пишет музыку.

Последний - руководитель проекта. Он собирает все то, что сделано остальными, объединяет разнородные куски в единое целое с помощью своего главного БЕЙСИКовского блока. Он координирует работу остальных, назначает адреса, через которые передаются параметры из одной процедуры в другую, отводит те или иные размеры памяти участникам проекта по их запросу.

Вот Вы уже и создали свою фирму. Теперь пару лет кропотливой работы и можете выходить на международный уровень. В Вашей фирме найдется место и художнику и композитору и специалисту по иностранным языкам и менеджеру и даже специалисту по проведению маркетинговых исследований. Добавьте отдел распространения, рекламный отдел, бухгалтерию и отдел кадров и назначьте себя Генеральным Директором или хотя бы президентом.

4. Из того факта, что программа сделана из многочисленных мало зависящих друг от друга модулей вытекает и возможность создания частных и коллективных библиотек.

Когда Ваша группа закончит работу над своим проектом и решит создать новый, то окажется, что значительный пакет процедур может быть перенесен из предыдущего проекта без каких-либо изменений.

Поэтому сразу надо начинать работу не над одной программой, а над сериалом и, прорабатывая процедуры, учитывать возможность их использования и в других программах тоже.

Так у Вас будет создана библиотека. Придет время и Вы напишете на БЕЙСИКе нехитрую программу, которая прокрутит на магнитофоне одну-две кассеты с Вашими библиотеками, выберет из них те процедуры, которые Вам нужны в сегодняшнем проекте, загрузит их в память компьютера и объединит в единый модуль.

И в заключение несколько практических советов "ИНФОРКОМа".

I

Мы рекомендуем всем, кто пишет в машинных кодах, завести для себя единый стандартный блок, разместив его в раз и навсегда отведенной для него области памяти. Мы называем этот блок - керналь (KERNAL). Вот для чего он нужен.

Вы, конечно, понимаете, что абсолютное большинство созданных Вами стандартных процедур должны обмениваться друг с другом и с головной программой какими то данными. Например, процедура, которая занимается переброской экрана должна как минимум получить два параметра - адрес, начиная с которого хранится блок (2 байта) и длину этого блока (2 байта). Более того, если экран хранится в компрессированном виде, то его еще

надо декомпрессировать. Поэтому мало принять эти два параметра - их еще надо передать в процедуру декомпрессии. Так отведите раз и навсегда 4 байта памяти для хранения параметров, используемых процедурой переброски экрана. Пусть это будут скажем адреса с 60000 по 60003. далее отведите 4 байта для параметров, передаваемых в процедуру, выполняющую музыкальное сопровождение - с 60004 по 60007. и т.д. У Вас образуется некий блок размером 1...2 K, который Вы всегда сможете вставлять в любые свои программы. Сегодня Вы не знаете, в каких адресах у Вас завтра будет находиться звуковоспроизводящая процедура, но Вам об этом и не надо думать. Размещайте ее где угодно. Ей это все равно, если и сегодня и завтра и всегда она будет получать свои данные из одних и тех же адресов и отправлять результат работы в одни и те же ячейки. Вот данные в этих ячейках могут меняться, но сами адреса - никогда.

Более того, если Вы делаете программу, в которой нет ни стрельбы ни бомбежки, а в Вашем кернале есть ячейки, отвечающие за передачу параметров в процедуры бомбометания и стрельбы, пусть они там и будут. Оставьте их, чтобы не сдвигать раз и навсегда сделанный керналь. Невелика потеря памяти, зато в конечном итоге экономятся годы труда.

И, наконец, в кернале можно хранить не только данные. В нем же можно хранить и адреса каких-то процедур. Посмотрите, как организован вектор переходов в статье "Потоки и каналы". Одним словом, принцип керналя - тот же, по которому организован раздел системных переменных в "Спектруме", только там он сделан К. Синклером в поддержку его ПЗУ, а Вы сделаете его в поддержку всех своих программ и он будет постепенно год от года наращиваться по мере накопления вами библиотек и личного опыта.

II

И последний практический совет. Он касается хранения текстовых сообщений (впрочем то же относится и к хранению в памяти экранов и прочих данных).

В целях упрощения подачи материала наш автор в примере процедуры TXT смешал в процедуре и машинный код и данные. То есть текстовое сообщение "ZX SPECTRUM" им размещено там же в процедуре, где и сам код. Адрес начала сообщения, как Вы видите из примера, передается через стек командой POP HL (а значит кто-то должен был заранее позаботиться и заготовить его там). Длина сообщения не определена, вместо этого маркером конца сообщения служит CHR$ 0 (операция проверки на конец - CP 0).

В качестве примера этот вариант годится, да видимо именно так и надо представлять идею, чтобы не усложнять ее излишними деталями, но теперь, в качестве комментария, мы должны сказать, что при написании реальных программ делают не так.

Размещать текстовые сообщения в процедурах вообще- то нельзя. Дело в том, что текст сообщения, как и любые данные, Вам потом может захотеться изменить, и тогда если изменится его длина, то после его вставки в процедуру переместятся команды, и в результате "поплывут" все команды относительных переходов JR, что очень неприятно.

Поэтому все сообщения хранят в едином блоке (таблице), отведя ей какой-то фиксированный участок памяти, например с 53000 и выше (только последите, чтобы он не перекрыл керналь или другую какую-либо таблицу). Более того, сам адрес начала этой таблицы текстовых сообщений можно хранить в кернале, отведя ему раз и навсегда 2 байта.

Теперь Вы можете разместить в таблице сколько хотите сообщений.

Например:

```
58000 - Сообщение1
58007 - Сообщение2
..............
59567 - Сообщение250
```

Вызывается процедурой TXT то или иное сообщение по его номеру, который должен быть установлен предварительно в аккумуляторе. Например вызов 15-го сообщения пройдет так:

```
        LD A,15
        CALL TXT
```

Теперь осталось выяснить каким образом процедура TXT найдет пятнадцатое сообщение, если адрес его ей не известен. Во-первых, ей известно начало таблицы

сообщений (его можно взять из керналя). Во вторых, вводится еще одна таблица - указательная таблица, в которой хранится величина смещения начала каждого сообщения от начала таблицы сообщений. На каждое смещение надо дать 2 байта и тогда если в Вашей программе 250 сообщений, то указательная таблица займет 500 байтов. Предположим, что мы ее разместим, начиная с адреса 57500.

Дальше логика такова. Номер сообщения Вам известен - 15. Перед ним стоят 14 сообщений. Поскольку в указательной таблице на каждое сообщение уделено 2 байта, то его надо удвоить - получим 28. Начало указательной таблицы Вам известно - 57500 (кстати его тоже надо брать из керналя, ведь оно может меняться от программы к программе). Теперь определяем сумму 57500+28=57528. Значит в адресах 57539 и 57530 хранится величина смещения в таблице сообщений. Определим ее. Пусть мы получили 1560. Теперь из керналя вводим базовый адрес таблицы сообщений. У нас это 58000. " Прибавляем к нему величину смещения 1560 и получаем 59560 - адрес, начиная с которого содержится ваше искомое сообщение.

А вот как это выглядит на практике см. Листинг-1.

И ряд дополнительных практических замечаний:

1. Хранение и вызов экранов по их номеру организуется точно так же.

2. Атрибуты, с которыми печатается текст (цвет фона, символа, признаки мигания, яркости, координаты позиции печати и т.п. хранятся вместе с текстом).

3. Управляющие коды, такие как CHR$ 13, CHR$ 6 и т.п. хранятся вместе с текстом.

4. В приведенных примерах не задана длина текстового сообщения. Компьютер определяет, что оно закончилось по появлению кода CHR$ 0 (NOP). Это не вполне экономично, т.к. сколько у Вас в программе сообщений, столько байтов Вы потратите на хранение этих маркеров. Обычно применяют другой способ - такой, какой применен в ПЗУ компьютера. Последний символ каждого сообщения увеличивают на 128, т.е. включают старший (седьмой бит). При проверке же на конец сообщения проверяют седьмой бит и если он включен, то печатают последний символ и выполняют возврат.

5. При переброске экранов удобнее не маркировать конец экрана, а хранить в указательной таблице длину каждого закомпрессированного экрана или вычислять ее через разность соседних значений в указательной таблице.

6. При переброске спрайтов указательная таблица вообще не нужна. Спрайты имеют как правило стандартный размер, скажем 32 байта каждый. Тогда по номеру спрайта всегда в таблице спрайтов можно найти его начало и конец. Адрес начала таблицы спрайтов хранится в кернале. Там же может храниться и максимальное число спрайтов в Вашей программе.

7. После всего вышесказанного Вы можете представить структуру своей будущей программы примерно в таком виде: (см. рис. 1).

.....
Головной блок.
Рабочие процедуры.
Область Ваших программных переменных.
Указательная таблица экранов.
Таблица закомпрессированных экранов.
Таблица графики спрайтов.
Таблицы шрифтов (если они нужны)
Указательная таблица сообщений
Таблица текстовых сообщений
К Е Р Н А Л Ь
.....

Рис. 1

Обратите внимание на то, что еще до начала написания программы Вы должны примерно прикинуть распределение памяти и решить сколько байтов и на что Вы отведете. При программировании же должны тщательно следить за тем, как Вы укладываетесь в намеченные пределы. Впрочем, это было лучше описано у Стива Тернера в статье "Профессиональный подход".

Обратите внимание также на то, что адреса границ этих программных областей - это тоже вещь, которую можно хранить в кернале.

Мы надеемся, что материал этой статьи не вызовет больших сложностей в понимании у начинающих, а профессионалам напомним, что мы ориентируемся в первую очередь не на них.

Зато спешим порадовать всех. В развитие данной темы в будущем году (причем в первом полугодии) мы напечатаем замечательную книгу Стюарта Николса "Применение АССЕМБЛЕРа для создания игровых и других скоростных программ на компьютере "ZX SPECTRUM".

Перевод подготовлен по нашему заказу специально для "ZX-РЕВЮ" Пашориным Валерием Ивановичем (г. Балашов, Саратовской обл.)

В значительной степени книга и посвящена вопросам конверсии БЕЙСИКа в машинный код.

Листинг 1.

Предположим, что в кернале базовый адрес указательной таблицы хранится, скажем, в 61245, а базовый адрес таблицы сообщений - в 61247.

```
TXT         AND A                   ;Этой безобидной операцией мы сбрасываем
                                    ;флаг переноса без изменения аккумулятора
                                    ;(это делается на всякий случай).
            LD HL,(61245)           ;В пару HL загружается начальный адрес
                                    ;указательной таблицы
            ADD A,A                 ;Удвоение содержимого аккумулятора. Обра-
                                    ;тим внимание на то, что если в нем было
                                    ;установлено число большее, чем 127, то
                                    ;после удвоения аккумулятор переполнится
                                    ;и включится флаг переноса.
            LD E,A                  ;Удвоенное содержимое аккумулятора пере-
            LD D,00                 ;сылается в регистровую пару DE.
            JR NC,BYPASS            ;Если аккумулятор не переполнялся, то
                                    ;следующую операцию надо обойти.
            INC D                   ;В противном случае надо увеличить регистр
                                    ;D на единицу.
BYPASS      ADD HL,DE               ;Теперь в DE хранится удвоенный номер на-
                                    ;шего сообщения. Прибавим его к начальному
                                    ;адресу указательной таблицы, хранящемуся
                                    ;в HL.
            LD E,(HL)               ;Ищем в нем смещение. Младший байт смеще-
                                    ;ния отправляем в регистр E.
            INC HL                  ;Переходим к старшему байту.
            LD E,(HL)               ;Отправляем его в регистр D.
            LD HL,(61247)           ;В пару HL загружается начальный адрес
                                    ;таблицы Ваших текстовых сообщений.
            ADD HL,DE               ;Прибавив его к взятой из таблицы величине
                                    ;смещения, находим адрес начала сообщения
TXLOOP      LD A,(HL)               ;Загружаем в аккумулятор первый символ
                                    ;сообщения.
                                    ;Далее все также, как у автора статьи.
            CP 0
            JR Z,TXEND
            RST 16
            INC HL
            JR TXLOOP
TXEND       INC HL
            RET
```

ПРОФЕССИОНАЛЬНЫЙ ПОДХОД

Окончание. Начало на стр. 49,70,117,143,167.

В прошлый раз мы с Вами рассмотрели вопрос как избегать ошибок при программировании, как их находить и устранять. Сегодня мы заканчиваем печать статей Стива Тернера под общим заголовком "Профессиональный подход". Правда это не значит, что мы навсегда прощаемся с этим выдающимся программистом. Может быть ему не раз еще удастся поделиться с нами маленькими хитростями и своими профессиональными секретами.

Итак, вернемся к плану тестировании готовой программы с целью поиска в ней возможных ошибок "мин замедленного действия".

Ниже приведена программа "Монитор реального времени" (MONITR) и на примере тестирования ее самой же и показано, как можно составить план тестирования.

В отличие от большинства других программ-мониторов эта очень мала и позволяет запускать Вам Вашу программу под ее управлением. Во время работы Вашей программы Вы сможете вести контроль за своими переменными и отслеживать их изменение. Ее применение не исключает возможность применения внешнего более общего монитора. Наоборот, монитор реального времени служит как бы дополнением к нему. Чтобы в дальнейшем не путаться между двумя мониторами, давайте будем называть монитор реального времени резидентным монитором, а главный монитор - просто монитором.

Когда Ваша игровая программа работает, MONITR непрерывно ее сопровождает, но может ее прервать и передать управление главному монитору для глубокого анализа состояния программы в момент прерывания.

Вы можете значительно ускорить процесс тестирования программы благодаря тому, что Вам предоставлена возможность выставлять значения своих переменных перед тестовым запуском. Это неоценимо важно при настройке игровых программ. Вы сможете вносить изменение в какие-либо программные константы, от которых зависит скорость передвижения персонажей, интенсивность стрельбы, позиционирование объектов на экране и многое тому подобное. Все это позволяет достичь максимальной играбельности готового программного продукта.

```
            MONITR      DL                  ;Определение метки.
3A27B1                  LD A,(MONON)        ;Включение.
A7                      AND A               ;Проверка "Включен/выключен?"
200F                    JR NZ,ISON          ;Переход, если включен.
01FEF7                  LD BC,F7FE          ;Задание адреса порта
                                            ;клавиатуры. Выставлен верхний
                                            ;ряд, левый полуряд.
ED78                    IN A,(C)            ;Прием информации от клавиатуры.
F6FE                    OR FE               ;Маскирование 7-ми младших
                                            ;битов.
FEFF                    CP FF               ;Проверка не была ли нажата
                                            ;клавиша "1".
C8                      RET Z               ;Поскольку MONON=0 и клавиша "1"
                                            ;не нажата - возврат.
3E01                    LD A,01             ;Если клавиша "1" была нажата,
3227B1                  LD (MONON),A        ;изменяем значение MONON.
210040      ISON        LD HL,4000          ;организация замедляющего цикла.
2D          DELAY       DEC L
20FD                    JR NZ,DELAY
25                      DEC H
20FA                    JR NZ,DELAY
            KEYSCAN     DEFL                ;Определение метки.
01FEEF                  LD BC,EFFE          ;Установка адреса порта
                                            ;клавиатуры. Выставлен верхний
                                            ;ряд, правый полуряд.
```

```
ED78                    IN A,(C)        ;Чтение данных с порта. Поступил
                                        ;байт, имеющий вид:
                                        ;111????? - здесь и далее:
                                        ;1 - бит включен;
                                        ;0 - выключен;
                                        ;? - не определен.
F6E2                    OR E2           ;Включение тех битов, которые
                                        ;нас не интересуют.
                                        ;E2 HEX = 1110 0010 BIN, имеем
                                        ;в итоге: 111? ??1?
6F                      LD L,A          ;Сохранили значение.
01FEF7                  LD BC,F7FE      ;Установка адреса порта
                                        ;клавиатуры. Выставлен верхний
                                        ;ряд, левый полуряд.
ED76                    IN A,(C)        ;Чтение данных с порта. Поступил
                                        ;байт, имеющий вид 111? ????
17                      RLA             ;Теперь байт имеет раскладку:
                                        ;11?? ???1. Мы гарантированно
                                        ;включили младший нулевой бит,
                                        ;взяв его из 7-го бита.
F6DF                    OR DF           ;DF HEX = 1101 1111, имеем в
                                        ;итоге: 11?1 1111.
A5                      AND L
FEFE                    CP FE           ;В аккумуляторе будет FE тогда и
                                        ;только тогда, когда нажата
                                        ;клавиша 0 и никакие другие
                                        ;клавиши верхнего ряда не
                                        ;нажаты.
2003                    JR NZ,NOQUIT    ;Переход для продолжения работы.
C30FF7                  JP EXITAD       ;Если была нажата клавиша "0",
                                        ;здесь выполняется переход в
                                        ;главный монитор по адресу 0FF7,
                                        ;заданному константой EXITAD.
                                        ;Если Вы не пользуетесь
                                        ;монитором фирмы PICTURESQUE,
                                        ;задайте здесь свой адрес.
FEFF        NOQUIT      CP FF
CA36B0                  JP Z,OPTYPE     ;Если никакая клавиша не нажата,
                                        ;остаемся в режиме 1.
1F                      RRA             ;Биты, выключаемые при нажатии
1F                      RRA             ;клавиш "0" и "9" нас не интере-
                                        ;суют и мы их "прокручиваем".
0E0                     LD C,0          ;C - выполняет роль счетчика
                                        ;оборотов байта.
1F          FINDK       RRA             ;"Крутим" байт, пока не найден
                                        ;нулевой бит, т.е. нажатую
                                        ;клавишу.
3003                    JR NC,KEYNO     ;Как только найдется нулевой
                                        ;бит, выполняется переход в
                                        ;KEYNO.
18FA                    JR FINDK        ;Переход на очередное
                                        ;вращение байта.
79          KEYNO       LD A,C          ;Сколько оборотов сделали?
                                        ;Отсюда и будем вычислять какая
                                        ;клавиша была нажата.
2128B1                  LD HL,MONDIG    ;позиция курсора MONDIG.
1129B1                  LD DE,MONADD
FE03                    CP 3            ;3 оборота - клавиша "5".
2811                    JR Z,LEFTCR     ;Переход на "курсор влево".
FE02                    CP 2            ;2 оборота - клавиша "6".
2316                    JR Z,DOWN       ;Переход на "вниз".
FE01                    CP 1            ;1 оборот клавиша "7".
2816                    JR Z,UP         ;Переход на "вверх".
                                        ;В противном случае начинаем
                                        ;обрабатывать режим "курсор
```

```
                                        ;вправо".
7E                      LD A,(HL)       ;Ввели позицию курсора.
3C                      INC A           ;Увеличили ее.
FE07                    CP 07           ;Позиция не может быть более 6.
3028                    JR NC,OPTYPE    ;Продолжение работы в режиме 1.
77                      LD (HL),A       ;Запомнили позицию курсора в
                                        ;MONDIG.
1825                    JR OPTYPE       ;Продолжение работы в режиме 1.
7E          LEFTCR      LD A,(HL)
FE00                    CP 0            ;0 - минимальная позиция
                                        ;курсора.
2820                    JR Z,OPTYPE
3D                      DEC A
77                      LD (HL),A
1B1C                    JR OPTYPE       ;Продолжение работы в режиме 1.
0EFF        DOWN        LD C,FF         ;Ввод начального значения.
1802                    JR TURNBT       ;Переход на изменение.
0E01        UP          LD C,1          ;Ввод начального значения
            TURNBT      DEFL            ;Определение метки.
7E                      LD A,(HL)       ;Ввод позиции курсора.
47                      LD B,A          ;Запомнили ее в B.
2600                    LD H,00
CB2F                    SRA A           ;Деление на 2, поскольку инди-
                                        ;кация каждого байта на экране
                                        ;занимает 2 позиции, то поделив
                                        ;номер позиции пополам, получим
                                        ;номер байта.
6F                      LD L,A
19                      ADD HL,DE       ;HL указывает на младший или
                                        ;старший байт адреса, на который
                                        ;настроен монитор.
78                      LD A,B          ;Старший или младший
E601                    AND 01          ;полубайт?
79                      LD A,C
2006                    JR NZ,TOMONB
17                      RLA             ;Если младший, то вращаем его
17                      RLA             ;четыре раза.
17                      RLA
17                      RLA
E6F0                    AND F0
86          TOMONB      ADD A,(HL)      ;Установленное значение отправ-
77                      LD (HL),A       ;ляется на хранение в
                                        ;программную переменную.
112E31      OPTYPE      LD DE,(MONLIN)  ;MONLIN указывает на начало
                                        ;выходного буфера.
3A29B1                  LD A,(MONADD)   ;
67                      LD H,A          ;Теперь в HL -
3A2AB1                  LD A,(MONADD+1) ;анализируемый
6F                      LD L,A          ;адрес.
3A2CB1                  LD A,(MONOP)    ;A содержит код операции.
A7                      AND A           ;Проверка на 0.
2806                    JR Z,PEEK       ;Переход на процедуру ис-
                                        ;полнения режима PEEK.
FE20                    CP 20H          ;Если там "пробел" -
2808                    JR Z,POKE       ;Переход на процедуру ис-
                                        ;полнения режима POKE.
180A                    JR NULLOP       ;В противном случае обраба-
                                        ;тывается режим 0.
7E          PEEK        LD A,(HL)       ;Снимается значение из ана-
                                        ;лизируемого адреса
322BB1                  LD (MONVAL),A   ;и запоминается в программ-
                                        ;ной переменной MONVAL.
1804                    JR NULLOP       ;Продолжение работы.
3A2BB1      POKE        LD A,(MONVAL)
77                      LD (HL),A       ;Засылка байта в адрес.
```

```
2129B1      NULLOP      LD HL,MONADD            ;В HL - адрес выходного буфера.
                                                ;В нем хранится образ экранной
                                                ;строки.
CD07B1                  CALL HEX                ;Для первой пары позиций вызывается
                                                ;конвертер перевода 16-
                                                ;тиричного кода в ASCII.
23                      INC HL                  ;Пропуск второй позиции 1-ой
                                                ;пары.
CD07B1                  CALL HEX                ;Для второй пары позиций.
13                      INC DE                  ;Вводится пробельная позиция.
23                      INC HL                  ;
CD07B1                  CALL HEX                ;3-я пара.
13                      INC DE                  ;Пробел.
23                      INC HL
CD07B1                  CALL HEX                ;Обрабатывается последняя
                                                ;позиция.
3E20                    LD A,20H                ;20H - это код пробела.
3237B1                  LD (BLANK),A            ;Подготовка к "стиранию"
3A28B1                  LD A,(MONDIG)           ;Номер позиции.
210058                  LD HL,58000             ;Адрес в дисплейном файле.
0620                    LD B,20H                ;Счетчик цикла.
360F        COLMON      LD (HL),0F              ;Выставление
23                      INC HL                  ;экранных
10FB                    DJNZ COLMON             ;атрибутов.
210056                  LD HL,58000             ;Адрес в дисплейном файле.
FE04                    CP 04                   ;4-ая позиция пробельная и
3806                    JR C,YE                 ;не обрабатывается.
23                      INC HL
FE04                    CP 06                   ;6-ая позиция пробельная и
3801                    JR C,YE                 ;не обрабатывается.
23                      INC HL
1600        YE          LD D,0
5F                      LD E,A
19                      ADD HL,DE
3630                    LD (HL),30H             ;Изменение цвета в позиции
                                                ;курсора.
3A29B1      PUTMON      LD A,(MONADD)           ;Подготовка
67                      LD H,A                  ;к печати
3A2AB1                  LD A,(MONADD+1)         ;содержимого
6F                      LD L,A                  ;ближайших
1138B1                  LD DE,MONDMP            ;восьми байтов.
13                      INC DE
0604                    LD B,04                 ;Сначала 4 байта.
CD07B1      HEXDMP      CALL HEX                ;Вызов 16-ричного конвертера.
23                      INC HL
10FA                    DJNZ HEXDMP             ;Основание цикла.
0604                    LD B,04                 ;еще 4 байта.
13                      INC DE                  ;Пробельная позиция.
CD07B1      NEXTWO      CALL HEX                ;Вызов 16-ричного конвертера.
23                      INC HL
10FA                    DJNZ NEXTWO             ;Основание цикла.
11000                   LD DE,0000H             ;Установка начальной позиции
                                                ;печати строки на экране.
212DB1                  LD HL,MONOUT
CD4AB1                  CALL PRINT              ;Вызов процедуры, печатающей
                                                ;результаты работы программы на
                                                ;экране. Можете здесь поставить
                                                ;свой адрес.
C9                      RET                     ;Конец работы и возврат в
                                                ;исполняемую программу.
```

Конвертер 16-ричного кода в код символа ASCII.

```
AF          HEX         XOR A                   ;Обнуление аккумулятора.
ED6F                    RLD                     ;Вращение полубайта, подхватывая
                                                ;его значение из адреса (HL)
12                      LD (DE),A
13                      INC DE
ED6F                    BLD                     ;Второй полубайт.
12                      LD (DE),A
ED6F                    RLD                     ;Восстановление значения в
                                                ;адресе, на который указывает
                                                ;HL, в исходное состояние.
1B                      DEC DE
1B                      DEC DE
0E02                    LD C,2                  ;Ввод параметра цикла.
1A          DIGIT       LD A,(DE)
FE0A                    CP 0AH
3004                    JR NC,DOLET             ;Если число больше 9 и должно
                                                ;изображаться буквой - переход
                                                ;на DOLET.
C630                    ADD 30H                 ;Перевод цифры в код ASCII.
1802                    JR DONE                 ;Конец.
C637        DOLET       ADD 37H                 ;Перевод буквы в код ASCII.
12           DONE       LD (DE),A
13                      INC DE
0D                      DEC C
20F0                    JR NZ,DIGIT             ;Возврат к началу цикла.
C9                      RET                     ;Возврат в вызывающую программу.
```

Таблица программных переменных.

```
00          MONON       DB 0                    ;0- выкл. ; 1 - вкл.
00          MONDIG      DB 0                    ;Позиция курсора.
00AF        MONADD      DW AF00                 ;Адрес вызова внешнего монитора.
00          MONVAL      DB 0                    ;Значение анализируемого адреса
00          MONOP       DB 0                    ;Код режима работы.
1C          MONOUT      DB 1C                   ;Количество символов в выходном буфере.
00000000    MONLIN      DEFM                    ;Содержимое выходного буфера для
20000020                                        ;адреса и содержимого байта.
00
20          BLANK       DB 20H                  ;Символ "пробел".
20000000    MONDMP      DEFM                    ;Содержимое выходного буфера
00000000                                        ;для восьми ближайших байтов
00
20000000
00000000 00
```

Чтобы резидентный монитор сопровождал Вашу игру, Вам надо в самой игре сделать периодический его вызов. Лучше всего это делать где-то в главном управляющем цикле. Я, например, этот вызов (CALL) вставляю в ту процедуру, которая занимается сканированием клавиатуры в поисках нажатой пользователем клавиши, так что у меня он работает даже когда игра стоит в ожидании чего-то.

Первое, что делает резидентный монитор - он обеспечивает выход из вашей игры, если нажата клавиша "0" и вход в главный монитор. Этот вход делается по методу установки точки прерывания, чтобы отработав в главной мониторе, можно было бы вновь вернуться и продолжить игру.

Я не знаю каким монитором в качестве главного пользуетесь Вы и поэтому не знаю какие команды он вставляет, когда устанавливает точку прерывания, но Вы можете легко это сами установить.

Работая со своим монитором, (например MONS 3) установите где-либо точку

прерывания, а потом посмотрите, что он туда подставил. Это будет либо команда CALL, либо JUMP. В зависимости от прочитанного адреса перевода замените в нижеприлагаемой программе монитора реального времени переменную EXITAD на то, что Вы там нашли. В приведенной распечатке там стоит адрес F70F (63247) - адрес вызова программы MONITOR 48 фирмы PICTURESQUE, которым я и пользуюсь.

Начинать процесс мониторинга игры надо из главного монитора. Стартуйте его, чтобы правильно прошла настройка его собственных переменных, а потом из него пойдите в тестируемую программу.

Поместить вызов резидентного монитора Вы можете в любом цикле Вашей программы, в зависимости от того, что Вы тестируете в данный момент. Он может вам даже дать возможность прервать исполнение зацикленной программы.

Что позволяет Вам делать резидентный монитор? Во время работы Вашей программы он изображает в верхней части экрана:

- адрес контролируемой Вами ячейки;
- содержимое данной ячейки;
- код режима работы;
- содержимое ближайших восьми байтов.

Печать на экране выполняется с помощью процедуры PRINT, которую мы рассматривали ранее (стр. 119).

На экране информация выглядит так:

```
6000  00  1  00010203  04050607
|     |   |   содержимое 8-ми
адрес |   |   соседних байтов
содержимое|
          код операции
```

Управляющие клавиши:

0 - прерывание работы резидентного монитора и переход в главный монитор.

1 - включение резидентного монитора.

5 - курсор влево (выбор цифры, подлежащей изменению);

8 - курсор вправо (выбор цифры, подлежащей изменению);

6 - изменение цифры вниз;

7 - изменение вверх.

Так, перемещаясь между цифрами, изображаемыми на экране, Вы можете их изменять. Например, Вы можете изменить адрес, содержимое которого изображается на дисплее. Вы можете изменить режим работы резидентного монитора.

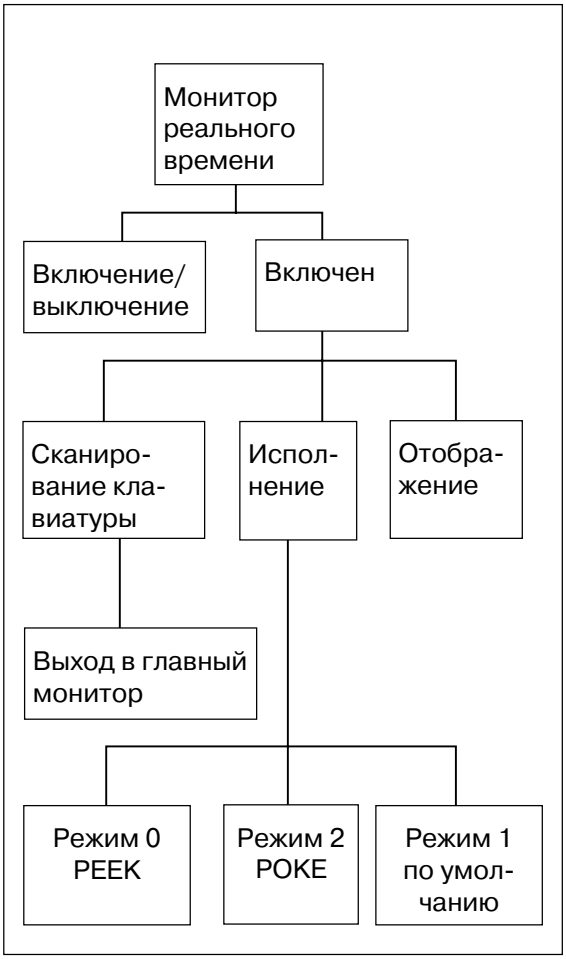
Всего он имеет три режима.

Режим - 0. В этом режиме происходит считывание (PEEK) содержимого заданного адреса и изображение его на экране.

Если Вы не успеваете отслеживать происходящие изменения, то "замедлить" работу программы можно перейдя в режим 1. Это основной режим, принимаемый по умолчанию. В этом режиме можно задать байт и задать адрес для исполнения POKE.

Режим 2 служит для засылки выставленного байта в выставленный адрес. После выполнения этого вернитесь в режим 0. Оставлять резидентный монитор в этом режиме опасно.

Для справки я привожу структурную схему монитора реального времени.



Теперь на примере программы MONITR рассмотрим план тестирования реальной программы. Ниже перечислены точки, в которых можно выполнить тестирование и намечено, что в этих точках подвергается проверке.

1. MONITR.

Никаких действий не должно выполняться, если переменная MONON равна нулю, т.е. если не нажата клавиша "1".

2. MONITR.

При нажатии "1" MONON должна принимать значение = 1.

3. MONITR.

Если MONON = 1 процедура должна запускаться и работать при нажатии любой клавиши, кроме "0".

4. DELAY.

Проверяется не слишком ли быстро работает программа.

5. KEYSCAN.

Должна прерываться работа при нажатии клавиши "0".

6. KEYNO.

Не должно происходить никаких изменений при нажатии любых клавиш, кроме "5", "6", "7", "8".

7. KEYNO.

При нажатии клавиши "8" (курсор вправо) должно возрастать значение переменной MONDIG, но не более 6.

8. LEFTCR.

При нажатии клавиши "5" должно уменьшаться значение MONDIG, но не ниже, чем до 0.

9. DOWN.

При нажатии клавиши "5" значения цифры на экране должны убывать.

10. UP.

При нажатии клавиши "6" значения цифры на экране должны возрастать.

11. TURNBT.

Проверяется изменение второй цифры в изображаемом на экране содержимом байта.

12. TURNBT.

Проверяется изменение второй пары цифр в изображаемом адресе.

13. OPTYPE.

Проверяется влияние переменной MONOP на режим работы программы. Выполняется ли PEEK, когда MONOP равно 0? Выполняется ли POKE, когда MONOP = 2? При любых других значениях ничего не должно выполняться.

14. PEEK.

Читается ли содержимое памяти?

15. POKE.

Вводится ли в память установленное значение байта?

16. HEX.

Вывод на экран соответствует шестнадцатеричному представлению чисел?

17. NULLOP.

Формат печати информации на экране соответствует требуемому?

18. COLMON.

Есть ли курсор на экране? он передвигается? Он перепрыгивает через промежутки между числами? После прохождения курсора по числу его цвет восстанавливается исходный?

19. PUTMON.

Проверяется правильность изображения на экране данных по 8-ми соседним байтам. Также проверяется их занесение в буфер MONDMP.

20. HEXTWO.

Окончательно проверяется качество печати.

И, в заключение, пару интересных наблюдений.

Может быть Вы заметили, что план тестирования программы в значительной степени отражает структурную диаграмму, и это не случайно. Если бы было желание дробить диаграмму до более мелких деталей, Вы бы увидели, что они вообще идентичны. Каждому блоку на диаграмме соответствовал бы один тест. Отсюда важный вывод - еще и не приступив к работе над программой, занимаясь только ее проектированием и исполнением структурных диаграмм, Вы уже закладываете основы и возможность будущей отладки и задаете надежность будущей программы.

Как говорится, что заложите, то и получите и остается только еще раз повторить: "Сначала тщательное проектирование - программирование потом".

И второе. Несмотря на то, что список возможных тестов оказался достаточно длинным проверка всего того, что в нем понаписано занимает у меня 30 секунд! (Конечно если все работает, но ведь и в этом тоже надо убедиться).

ADVENTURE PROJECT

Окончание.
Начало см. на стр. 122, 193.

Взаимодействие между играющим и программой - это один из важнейших вопросов, оказывающих влияние на качество программы. Между тем в программах адвентюрного жанра это взаимодействие организуется посредством текстового ввода (довольно архаичная, хотя и любимая многими манера).

Большинство адвентюрных программ используют технологию интерпретаторов для выяснения того, какую же команду хотел дать пользователь. В минимальной виде он включает конструкции типа Глагол + Существительное. В этом случае глагол выполняет функции команды, а существительное указывает на объект, к которому эта команда относится и, тем самым, служит как бы параметром при команде. Например программа поймет команду GO NORTH (Идти на север), но вполне может не понять команду PLEASE GO NORTH (Пожалуйста иди на север) или, еще того хуже, GO DOWN THE ROAD TO THE NORTH (Идти по дороге на север).

Таким образом, словарь который понимает программа, конечно ограничен. Раздвинуть пределы этих ограничений - это основная цель каждой фирмы, занимающейся выпуском адвентюрных программ. В то время как лучшие экземпляры достигают нескольких тысяч слов, типовым размером словаря минимальной (учебной) программы являются примерно 200 слов, которые организуются в программе в виде базы данных.

Вместе с тем, организация базы данных и работы с ней, это еще не все те проблемы, которые требуется решить. Прежде всего, если Вы хотите, чтобы Ваши персонажи могли взаимодействовать друг с другом и обмениваться сообщениями, необходимо подумать над специальными приемами. Они ведь не смогут общаться по системе глагол + существительное.

Мы в своем проекте пока обходились без общения между персонажами и вернемся к Оре и Инфу, которые победили монстров в глубине океана, проникли на затонувшую яхту и оказались в подземном мире короля Кельроса.

Под охраной подземных жителей их провели в столицу страны и теперь они ждут допроса у короля.

Теперь необходимо взводить программные средства для общения с персонажами. Дело-то ведь не в том, что Вы хотите сообщить им, куда им следует пойти, им еще надо провести переговоры с королем.

Чтобы не изобретать велосипед, мы вернемся к технике, которая уже была опробована в адвентюрных играх.

Когда в конце 60-х на гигантских вычислительных машинах, имевших огромные размеры памяти, но совершенно негибкие и неэффективные операционные системы, появились первые текстовые игры, они были по сегодняшним меркам невероятно примитивными и имели словарь порядка 100 слов. Как же они работали?

Приняв команду, например GO NORTH, они делили ее на две части и в каждой оставляли только два первых символа, отрезая остальные. Так что у Вас оставалось GO NO, после чего начиналась трансляция.

Трансляция проводилась одним из двух возможных способов. Либо была одна центральная база данных, в которую были засланы все возможные слова, которые программа могла интерпретировать, либо в соответствии со сценарием программы она разбивалась на сцены и для каждой сцены хранился свой допустимый набор слов. Второй способ требовал больше памяти, но зато был гибче первого, поскольку в первом случае приходилось в одной базе хранить слова самых разных типов, участвующих в программе: команды перемещения, боевых действий, названия предметов и др.

Второй способ хоть и требует большего расхода памяти, имеет предпочтение в большем просторе алгоритмизации и кодирования, особенно на БЕЙСИКе и ограничения, накладываемые на программу здесь не столь жестки.

Например, в какой-то локации Вам встретилась кошка. Вполне может быть, что Вы захотите ее погладить и программа может это предусмотреть, но если Вы захотите ее накормить, а вокруг нет и не было никакой еды, то и предусматривать здесь нечего.

Если же Вы окажетесь в сцене, где будет еда, то связанные с ней слова будут предусмотрены, но зато не надо обрабатывать слова, имеющие отношение к контактам с кошкой, а при наличии рядом и кошки, можете ее и накормить, но это уже как бы третья сцена: "кошка + еда".

Преимущество такого подхода еще и в том, что Вы подгоняете словарь под контекст реальной ситуации и он может быть хорошо продуман. При образовании же центральной базы данных Вам необходимо иметь в голове столько возможных вариантов, что все равно что-то не предусмотрите и пользователь будет разочарован.

С тех пор прошли годы, было изобретено множество новых приемов. Техника программирования становится все изощреннее и все ближе подходит к искусству, по так или иначе но новые приемы зачастую не снижают естественных ограничений, а нередко и добавляют их.

Например, такими технологиями управления адвентюрными играми является управление через меню, пиктограммы или с помощью ключевых слов. Например Вы хотите куда-то пройти: - нажимаете клавишу G. На экране появляется запрос "Куда?". Вы нажимаете клавишу N (На север) и т.д. а если Вам надо что-то взять (TAKE), то первым нажатием будет "T". Такая техника обычно используется в программах, имеющих центральную базу данных, а не систему, опирающуюся на сценарии.

Обе системы имеют свои за и против и нам надо определиться с двумя вопросами - "Что мы хотим иметь?" и "Как мы это будем делать?" и именно в таком порядке, а не наоборот. Нередки еще программисты, которые садятся писать что-то, полагая, что по ходу дела все прояснится и пути откроются туда, куда надо сами собой.

Нам нужен метод коммуникации достаточно удобный, чтобы не раздражать пользователя и достаточно простой, чтобы не раздражать программиста (об элегантности мы здесь не говорим, поскольку мы обсуждаем идеи, а не пишем реальную программу).

Мы вам предложим две системы обеспечения коммуникации. Первая простая, но ограниченная, а вторая - позволяет вводить команды от пользователя полными предложениями, но нуждается в корректировке при переходе от сцены к сцене, т.е. одни действия возможны в одной сцене, но невозможны в другой.

Система ввода через меню или ключевые слова может быть использована в простейшей подпрограмме ввода и будет работать таким образом, чтобы введенные слова одинаково интерпретировались во всех сценах Вашей игры. Пример для ключевых слов GO ("G") и TAKE ("T") приведен на листинге 1.

Пример открыт для расширения и Вы можете добавить к нему столько вариаций, сколько захотите. Все, что Вам требуется - это строка, перехватывающая ввод после строки 30.

В качестве примера объект введен в переменную в строке 5. В реальной программе в каждой сцене здесь должны вводиться соответствующие сцене объекты. Например, в сцене могут быть ведро и лопата на берегу моря. Это может быть кинжал, лежащий на палубе яхты и т.п.

Строка 10 принимает от пользователя команду в виде одного символа и помещает его в переменную a$. Строки 20 и 30 проверяют что это за команда - GO или TAKE. Если принятый символ не является ни "g" ни "t", тогда компьютер печатает в нижней части экрана "Не понимаю!" и ждет другой команды.

Если была нажата клавиша "g", то это команда GO и выполняется переход на строку 60 для анализа этой команды.

Строка 60 - начало блока управления сценой. Переменные n,s,e,w,u,d - соответствуют тем шести направлениям, в которые может захотеть переместиться играющий. Содержатся же в этих переменных номера строк, к которым следует перейти, чтобы отработать поданную пользователем команду. Ноль указывает на то, что в данном направлении перемещение невозможно.

Строка 70 вызывает подпрограмму, выполняющую ввод направления перемещения от пользователя и проверку на допустимость этого направления. Если оно недопустимо, в соответствующую переменную вводится ноль.

После возврата в главный блок проверяется значение переменной B. Если это ноль, пользователю выдается сообщение о невозможности перемещения в данном направлении и ему предлагается изменить команду.

Если же все прошло нормально, Вы переходите к следующей сцене по номеру строки, содержавшемуся в B.

Подпрограмма, занимающаяся командой TAKE расположена начиная со строки 100. Играющего спрашивают о том, какой предмет он хочет взять и вводят его в a$.

Переменная o$ хранит информацию о том объекте, который может быть взят в данной сцене. Если o$ ="" (пустая строка), программа сообщит о том, что нет объектов в данной комнате.

Если a$ не совпадает с o$, пользователя предупреждают о том, что данный предмет взять нельзя, он не здесь. Если же взять предмет удалось, вам об этом сообщают и данный предмет заносится в Ваш инвентарный список (переменная i$, которая отслеживает пополнение Вашей коллекции). Символ "*" служит для отделения объектов друг от друга и список может быть просмотрен, например с помощью программы, о которой мы уже говорили (см. стр. 194).

Листинг 1.

```
5 LET o$="SWORD":LET i$=""
10 PAUSE 0: LET a$=INKEY$: IF a$="" THEN GO TO 10
20 IF a$="g" THEN GO TO 60
30 IF a$="t" THEN GO TO 100
40 PRINT #0;"Я Вас не понимаю"
50 GO TO 10
55 REM Начало процедуры перемещения.
60 LET n=0: LET d=0: LET s=4000: LET w=5000: LET e=5050: LET u=4050
70 GO SUB 1000
80 IF B=0 THEN FRINT "Вы не можете идти в этом направлении": GO TO 10
90 GO TO B
100 INPUT "Что взять?"; a$
110 IF a$="" THEN FRINT "Здесь нет предметов, которые можно было бы взять.": GO TO 10
115 IF a$<>o$ THEN PRINT "Этот предмет находится не здесь.": GO TO 10
120 PRINT "вы берете ";o$
130 LET i$=i$+"*"+o$
140 GO TO 10
1000 INPUT "Куда идти? ";b$: LET b$=b$(1)
1010 IF b$="n" THEN LET b=n: RETURN
1020 IF b$="s" THEN LET b=s:RETURN
1030 IF b$="e" THEN LET b=e:RETURN
1040 IF b$="w" THEN LET b=w:RETURN
1050 IF b$="u" THEN LET b=u:RETURN
1060 IF b$="d" THEN LET b=d:RETURN
1080 RETURN
4000 CLS: PRINT "Вы находитесь в тронном зале короля Кельроса. Сверкающий трон поражает
      своим великолепием. Во взгляде властелина читается жестокая решимость. "
4010 GO TO 10
4050 CLS: PRINT "Оттолкнув стражу, Вы бросаетесь вверх по лестнице и, преодолев ее,
      попадаете в пустой коридор. Все боковые двери закрыты. Дальше дороги нет. Погоня
      настигает, сопротивляться бесполезно. "
4060 GO TO 10
5000 CLS: PRINT "Воспользовавшись секундным замешательством стражи с криком 'Бежим!' Вы
      хватаете Ору за руку и бросаетесь к выходу. "
5010 GO TO 10
5050 CLS: PRINT "Собрав все силы, Вы вырываетесь из рук стражи и бежите к ближайшей двери,
      но она сказывается закрытой. Погоня настигает Вас, под тяжелыми ударами Вы падаете и
      теряете сознание."
5060 GO TO 10
```

Это то, что касается первой технологии коммуникации.

Теперь рассмотрим второй прием, позволяющий вводить длинные предложения.

Итак, наши герои предстали перед повелителем подземного царства, который восседает на троне, а рядом с ним на столике лежит цель Вашего приключения - похищенные часы, управляющие ходом развития Вселенной. Вы должны вернуть бесценный прибор любыми усилиями.

Фактически перед ними три пути. Либо попытаться убежать, либо схватить часы и попытаться убежать, либо попробовать договориться мирным путем.

Переведя все это на язык компьютера, мы получаем: GO (идти) или TAKE AND GO (взять и идти), или SAY TO (сказать).

Рассмотрим эти варианты и прикинем как их можно реализовать в нашей экспериментальной программе. Итак, если мы хотим, чтобы пользователь мог вводить свои команды длинными предложениями нормальным человеческим языком, нам необходимы программные средства, чтобы выделить из его заявления ту ключевую информацию, которая действительно играет роль.

В нашем примере мы рассмотрим именно эти три функции GO, TAKE и SAY TO. Давайте сразу отметим, что в любом языке одно и то же действие можно выразить по-разному, но тем не менее возможный набор синонимов все-таки ограничен. Так, например вместо "взять", можно употребить "схватить", "украсть", "похитить" и даже что-либо типа "умыкнуть", но надо все-таки признаться, что последнее очень маловероятно. Скорее всего пользователь будет применять TAKE или GET и их и надо искать в его команде. Если же программа пишется на русском языке, то имеет смысл настраивать ее на "взять" и на "хват" (от слов схватить, захватить). Если же он употребит иные слова, то ему можно сообщить, что мысль его в общем-то понятна, но не мог ли бы он попробовать ее выразить другими словами. Итак, мы остановимся на TAKE и GET. Точно также и с перемещением GO. Способов перемещения не так много - GO, RUN, MOVE (идти, бежать, передвигаться) и пожалуй из общеупотребимых это все. Направления же возможного движения вообще известны однозначно: СЕВЕР, ЗАПАД, ЮГ, ВОСТОК, ВВЕРХ, ВНИЗ (NORTH, SOUTH, WEST, EAST, UP, DOWN). Только уж если Вы захотите создать шедевр искусственного интеллекта, Вы сделаете так, что программа будет понимать вперед и назад (FORWARD, BACK).

Отсюда начинается самая трудная часть работы. Для процедуры, описывающей перемещение, Вам необходимо, чтобы она просмотрела введенную фразу и нашла есть ли в ней GO, затем определила есть ли в ней указание на направление и еще раз проверила нет ли других указаний.

При таком подходе пользователь сможет ввести команду: "Весело идти по дороге вымощенной желтым кирпичом на север и потом повернуть на запад" и будет правильно понят.

Если выделить в такой команде ключевые слова, то она будет выглядеть так:

(...ИДТИ) (...СЕВЕР...) (...И...) (...ЗАПАД...). GO...NORTH...AND...WEST

GO - основной носитель команды в этой фразе и искать этот глагол надо в первую очередь. Если он не найден, то значит надо проверить прочие глаголы-команды, которыми может воспользоваться играющий.

Если же он найден, то ищется направление, в данном случае NORTH, затем проверяется нет ли других команд в этой фразе (ищется AND) и если и это слово найдено, тогда дешифрируется и вторая часть команды.

В смысле алгоритма это означает, что сначала ищется подстрока "GO" в строке введенной пользователем. Подпрограмму поиска подстроки в строке может написать наверное всякий, владеющий хоть каким-нибудь языком программирования, но тем не менее в прилагаемом примере (Листинг 2) есть один вариант.

Если она находит GO, то выполнит переход в подпрограмму, которая занимается вопросами перемещения и проанализирует остальную часть фразы.

Аналогично может строиться и строка команды TAKE (GET).

Например:

GET...CLOCK...AND...TORCH...

ВЗЯТЬ...ЧАСЫ...И...ФАКЕЛ...

Теперь рассмотрим команду SAY TO (сказать). На первый взгляд она может

представлять проблему. Нет, особых проблем нет, решается все той же техникой, хотя немного посложнее. Именно ее-то мы и рассмотрим в конкретном примере (Листинг 2), не отвлекаясь на TAKE, GO и т.п.

Во-первых, установим, что команда должна состоять из трех основных разделов и программа должна с ними соответственно и разобраться. На первом этапе компьютер выделяет ключевое слово SAY. Делается это точно также, как для GET и GO. Далее управление передается подпрограмме, обслуживавшей SAY, которая анализирует оставшуюся часть команды.

Следом за командой SAY TO естественно должно стоять имя того, к кому речь обращена. на втором этапе программа проверяет присутствует ли данный персонаж в данной сцене. Если его нет, выдается соответствующее сообщение.

Если он есть, проверяется намерен ли он разговаривать. Это не значит, конечно, что программа должна анализировать его настроение (хотя и это возможно, раз у нас есть критерии оценки его состояния), все гораздо проще. Вы ведь не можете готовя программу предусмотреть диалоги с абсолютно всеми персонажами и поэтому отделываетесь от диалога с тем, для кого он Вами не предусмотрен. сообщениями типа "Он сейчас не в настроении для беседы и не отвечает на Ваше обращение" или, скажем, "Вам лучше обратиться к кому-нибудь другому по этому вопросу."

Если диалог для этого персонажа Вами предусмотрен, тогда декодируется остальная часть Вашей команды. Это скорее всего либо просьба (указание), либо вопрос. Кстати, отличить их легко по наличию или отсутствию вопросительного знака в ходе прямой речи. Но сейчас речь не об этом. То, что один персонаж говорит другому, стоит в кавычках и, выделив строку, стоящую в кавычках, Вы получаете как бы независимую команду, с которыми мы уже умеем работать.

Конструкция получается типа:

СКАЗАТЬ КОРОЛЮ "...ДАТЬ...ЧАСЫ..

" SAY TO KING "...GIVE....CLOCK..."

Возможно, конечно, что Вы захотите дать пользователю возможность строить очень головоломные конструкции типа: "Идти на север и потом повернуть на восток. Взять веревку и ведро и спросить у старика с палкой: "Где колодец?"."

Все это вполне реализуется серией условных переходов с логическим оператором IF. Компьютер воспринимает точку как разделитель команд. Выполняет все, что положено до точки. Потом то, что осталось до следующей точки и так до конца команды. Союз "И" (AND) выполняет, правда, двойную функцию. Он может объединять в одной команде два объекта, на которые направлено действие, а может и объединять два действия. Различить то и другое несложно, если посмотреть какого типа ключевые слова стоят по обе стороны от "И". Если это существительные (объекты), то это первый случай, а если одно из них глагол (команда из словаря команд), то это второй случай - тут-то и действует серия IF. Тогда компьютер разобьет Вашу длинную команду на серию коротких и, пока будет исполнять первую, остальные подержит временно в каком-то буфере (в какой-либо переменной).

Тогда Ваша длинная последовательность упростится до:

ИДТИ...СЕВЕР...И...ВОСТОК...

ВЗЯТЬ...ВЕРЕВКА...И...ВЕДРО...

СПРОСИТЬ СТАРИКА...

...КОЛОДЕЦ ?

Когда Вы программируете на английском языке, мало проблем с приставками, окончаниями слов и т.п., но на русском это проблема, которую программа, ищущая подстроку в строке, должна решить за счет сокращения словарных слов до 3-х 4-х коренных букв и их-то и искать.

ИДТИ...СЕВЕ...И...ВОСТ...

ВЗЯТ...ВЕРЕ...И...ВЕДР...

СПРО...СТАР...

...КОЛО?

Практика показывает, что если Вы делаете словарь игры примерно на 200 слов, то сокращение до трех букв проходит нормально. Если же словарь подходит к тысяче слов, то безусловно четыре буквы необходимы.

В нижеприведенном образце программы рассмотрен вариант обработки SAY TO. Программа ждет, что Вы обратитесь к королю. Он правда не очень разговорчив и подарив ему свой меч SAY TO KING 'TAKE MY SWORD', Вы уловите момент, когда он отвлечется рассматриванием подарка, схватите часы и броситесь бежать. А уж как вам удастся вывести героев на поверхность - это дело Ваше.

Листинг 2

```
5 DIM m(10):DIM c$(10,32)
10 CLS
11 REM Ввод команды играющего.
20 INPUT a$
21 REM Первый этап анализа команды.
30 LET nend=2
31 LET c$(1)="G"
32 LET c$(2)="S"
33 LET m(1)=100
34 LET m(2)=200
35 LET e=20
36 REM Правильнее конечно было бы вводить эти данные через READ/DATA, но так для примера
     нагляднее.
37 GO SUB 1000
38 GO TO gt
100 LET nend=1
110 LET c$(1)="GO"
120 LET m(1)=300
130 LET e=200
140 GO SUB 1000
150 GO TO gt
200 LET nend=1
210 LET c$(1)="SAY TO"
220 LET m(1)=400
230 LET e=20
240 GO SUB 1000
250 GO TO gt
300 REM Здесь расположен блок анализа перемещения. Мы не будем его рассматривать в данном
     примере.
310 REM ......................
398 REM ......................
399 STOP
400 REM Второй этап анализа команды SAY TO.
410 LET nend=1
420 LET c$(1) = "TO KING"
430 LET m(1)=500
440 LET e = 900
450 GO SUB 1000
460 GO TO gt
500 REM Поиск прямой речи. Предположим, что она должна быть ограничена слева и справа не
     кавычками, а апострофами.
510 LET nend=1
520 LET c$(1)="'"
530 LET m(1)=600
540 LET e = 910
550 GO SUB 1000
560 GO TO gt
600 REM Найден первый апостроф! Теперь можно найти и второй.
605 LET a$=a$(k+1 TO)
610 LET nend=1
620 LET c$(1)="'"
630 LET a(1)=700
640 LET e=920
650 GO SUB 1000
```

```
660 GO TO gt
700 REM Высказывание определено и теперь можно его анализировать.
705 LET a$=a$( TO K)
710 LET nend=1
720 LET c$(1)="TAKE"
730 LET m(1)=800
740 LET e=930
750 GO SUB 1000
760 GO TO gt
800 REM Король понял, что он может принять от Вас что-то в подарок, осталось только понять
      что именно.
805 LET a$=a$( TO K)
810 LET nend=2
820 LET c$(1)="SWORD"
830 LET m(1)=890
840 LET e=940
850 GO SUB 1000
860 GO TO gt
890 PRINT "король благожелательно улыбнулся, беря Ваш меч в свои руки и внимательно
      рассматривая его. Внимание охраны несколько рассеялось, все внимательно смотрят на
      короля."
899 STOP: REM Наступает момент, когда Вы можете схватить часы и попытаться убежать.
900 PRINT "Попробуйте обратиться к другому персонажу"
905 GO TO 30
910 PRINT "Ваше обращение к персонажу - это прямая речь и ее надо ограничить апострофами
      слева и справа."
915 GO TO 20
920 PRINT "Вы забыли закрыть прямую речь справа."
925 GO TO 20.
930 PRINT "король грозно смотрит на Вас и о чем-то шепчется со своими воинами. Пока Вы не
      нашли нужных слов для спасения собственной жизни. "'
935 GO TO 20.
940 PRINT "'Я не понимаю, что можешь предложить мне ты, несчастный чужестранец?'- грозно
      спрашивает король. Стража напряглась, в зале воцарило напряженное ожидание. "
945 GO то 20
1000 REM Анализатор подстроки в строке.
1005 LET kend=LEN (a$)
1006 LET ktest=LEN(c$)
1010 FOR k=1 TO kend
1020 FOR n=1 TO nend
1030 IF a$(k TO k+ktest-1)=c$(n) THEN LET gt=m(n):GO TO 1070
1040 LET gt=e
1050 NEXT n
1060 NEXT k
1070 RETURN
```

Еще раз обращаем Ваше внимание на то, что приведенный пример лишен элегантности именно потому, что это учебный пример. В Вашей реальной программе Вы конечно же должны использовать тот факт, что строки 100....200....300... и т.д. имеют одинаковую структуру. Введя подпрограмму, которая будет универсальной для всех этих блоков и вводя данные не через LET, а через READ-DATA, Вы во много раз сократите объем Вашей программы.

Заканчивая этот материал, мы еще раз хотим вам напомнить, что мы не ставили целью научить Вас составлять игровые программы. Это дело очень и очень сложное, к которому надо идти кропотливым трудом не один год. Мы только напомним, что основное направление деятельности "ИНФОРКОМа" - обучающие программы, а хорошая обучающая программа строится по тем же самым принципам, что и игровая.

Сейчас очень много классов в наших школах оборудуются "Синклер"-совместимыми машинами. С ними работают тысячи учителей. Им надо дать в руки инструмент быстрого реагирования, чтобы они уже завтра могли хоть что-то давать своим ученикам, обучая их не "информатике", как она понимается в наших нелепых учебниках, а именно "компьютерной грамотности". Умению ставить перед собой задачи и находить пути их решения. Вот для них-то идеи, изложенные в эти статье, и должны пригодиться в первую очередь.

Эти идеи и концепции - канва, на которую каждый нанесет то, что ему сегодня нужнее и важнее, завтра он сам определится, что для него самое важное и подберет другую канву или создаст свою, оснастит ее программно-инструментальными средствами, наполнит своими библиотеками и обогатит личным опытом, но все делается постепенно и начинать надо с простого.

Обучающие программы

Очень и очень многие пишут о том, как нужны им обучающие программы. Сегодня для них радостное сообщение пришло из Бухары. Наш читатель Подкорытов Л.К. выполнил русификацию пакета TUTOR. Для тех, кто не знает, сообщаем - это 40 уроков АССЕМБЛЕРа и этот пакет был бы замечательным дополнением к нашему трехтомнику по программированию в машинных кодах. Мы, к сожалению, перегружены сейчас собственными проектами и не сможем принять маркетинг этого пакета на себя, но полагаем, что те, кому он нужен, смогут обратиться к т. Подкорытову напрямую по адресу:

705022, г. Бухара-22, ул. Ульянова, д.82. кв.115.

Для тех, кто тоже работает в этом направлении мы всегда дадим "Зеленую улицу". И вот Вам адреса для потенциального партнерства:

ИЩЕМ ОБУЧАЮЩЕ ПРОГРАММЫ ДЛЯ ПК "SPECTRUM" ПО СПЕЦИАЛЬНОСТЯМ:

автодело, медицина, радиомонтаж, кулинария, вычислительная техника и программирование, психология, строительство, торговля, швейное дело.

Нужны программы по "Правилам дорожного движения".

Покупаем хорошие программы по всем предметам за курс начальной, восьмилетней и средней школы.

Обработаем поступающую информацию и вышлем всем желающим.

Наш адрес: 659700, г. Горно-Алтайск, ул. Маресьева 11/1. Межшкольный учебно-производственный комбинат. Сарычевой Т.А.

ДЛЯ ЭКСПЛУАТАЦИИ В ВУЗЕ

требуются обучающие программы для ПК "Спектрум".

453135, Стерлитамак-25, а/я 75 Веревкину Игорю Дмитриевичу.

И еще одна нужная вам информация:

Малое предприятие "РИТМ" просит отозваться всех программистов, пишущих игровые программы для "Спектрума".

Контактный адрес: 400009, Волгоград, пр. Ленина, д.129,кв.45. Харионовскому А.В.

FORUM

Очень интересный вопрос, имеющий некоторое историческое содержание задает наш читатель из г. С.-Петербурга Тасев А.Д.

"Меня и многих моих знакомых - почитателей "Спектрума" интересует вопрос, связанный с полем экрана компьютера. Если взять такие известные компьютерные системы, как MSX, IBM и многие другие, так там исходный экран - черный, а текст - белый. Это естественно, ведь это меньше нагружает глаза.

Почему же К.Синклер в своем компьютере ввел белый фон с черным текстом? Что за этим стоит? Или это только рекламный трюк? Да и вообще какое сочетание является более эргономичным для глаз, если работа идет с цветным дисплеем?"

При кажущейся простоте вопрос читателя не праздный - за ним стоит идеология архитектуры всей системы.

Во-первых давайте признаем, что действительно белый текст на черном фоне гораздо более эргономичен и это важно. Одно дело - если пользователь работает с компьютером 2 часа в день и совсем другое, если 14.

Теперь давайте вспомним, что в MSX, IBM и многих других системах, о которых говорит наш читатель, четко разделены режимы работы. Есть текстовый режим и соответственно текстовый экран и графический режим и несколько графических экранов (с разным разрешением).

И там и там основным является текстовый режим - все сделано для него, а графические режимы - более позднее добавление, не вполне рационально организованные и не столь удобные, как хотелось бы.

Попробуйте-ка включив IBM или "Ямаху" (MSX) и не изменяя режимы и не загружая никакие программы, нарисовать хотя бы окружность. Не тут-то было. Ничего не выйдет.

Кстати, программистам и фирмам, выпускающим программное обеспечение, не очень удобно такое разделение режимов.

Итак, каким должен быть экран у компьютера, изначально предполагавшегося для работы в текстовом режиме? - Конечно, белым по черному.

А для графики? Наверное наоборот, ведь не очень привычно рисовать на черном экране.

Весь успех Синклера на том и был основан, что он дал людям компьютер с достаточно удобным и экономичным экраном, работающий одновременно и в текстовом и в графическом режиме, да еще и с палитрой цветов. При этом на все уходит менее 7 килобайт. Если все мы сейчас и наслаждаемся этой машиной, то скорее всего именно поэтому. Где сейчас стандарт MSX, который в зарубежной прессе в 1986 году называли "бурно развивающимся", а в 1987 году - "загнивающим". Если бы не эта "хитрость" с экраном, был бы сейчас "Спектрум" там же, где и сотни его конкурентов 1982-84 годов.

Итак, совмещение алфавитно-символьного экрана и многоцветного графического экрана с очень малым расходом памяти - вот то гениальное решение, которое и привело эту машину в Ваш дом, и вот почему у нее белый экран.

Ну, а что касается наиболее эргономичного сочетания цветов? Вопрос мнений. Авторитеты пишут, что желтым по синему. Может быть еще рассмотрен вариант белым по синему. Это если дело касается "Спектрума".

К вопросу о совместимости.

Рассматривая проблемы совместимости отечественных модификаций "Спектрума", мы уже говорили о том, что, как установлено, определенное влияние имеет формирование сигнала INT в Вашем компьютере. Сегодня наш корреспондент из г. Йошкар-Ола Сергей Викторович Полубарьев делится некоторыми рекомендациями по обеспечению параметров этого сигнала.

Вот что он пишет:

В результате анализа работы фирменного "ZX Spectrum+128" установлено следующее:

1. Длительность сигнала /INT (вывод 16 Z80A) равна 8 мксек и не зависит от текущего состояния процессора в том числе и сигналов /M1 и /MREQ.

2. Некоторое несовпадение линии на бордюре с линиями в экранном поле может наблюдаться и на этом компьютере (обнаружено на игре "Monte Carlo Casino"). При этом бордюрные линии смещаются на 1 пиксел вниз относительно экранных.

В наиболее распространенных версиях самодельных "Синклеров" нормализацией сигнала INT занимается RC-цепочка, дифференцирующая длинный кадровый синхроимпульс. Ниже дается описание этого узла для версий "Москва" (с раздельным полем памяти), Ленинград-1 и "Пентагон 48К" (дисковый вариант).

В рассмотренной нами версии "Москва" места для RC-цепи не предусмотрено. Если на Вашей компьютере она отсутствует, придется собрать ее навесным монтажом на свободных элементах DD54 (К555ЛН1, на плате располагается между рядами ОЗУ 565РУ6 и РУ5). Схема приведена на рис.1. Фрагмент, к которому относятся доработки, выделен пунктиром.

При указанных номиналах деталей обеспечивается длительность сигнала около 9 мксек, что достаточно для нормальной работы компьютера. В других перечисленных моделях RC-цепочка уже есть и достаточно уточнить номиналы ее деталей.

Необходимо отметить также, что встречаются варианты RC-цепочки, выполненные на резисторе с большим номиналом (30...5О кОм) и с очень малой емкостью (30...300 пФ). Такая схема в принципе работоспособна, но может стать причиной частых сбоев из-за высокой чувствительности к импульсным помехам (в практике автора такое бывало). Более надежно применять низкоомный резистор (0.5...3 кОм).

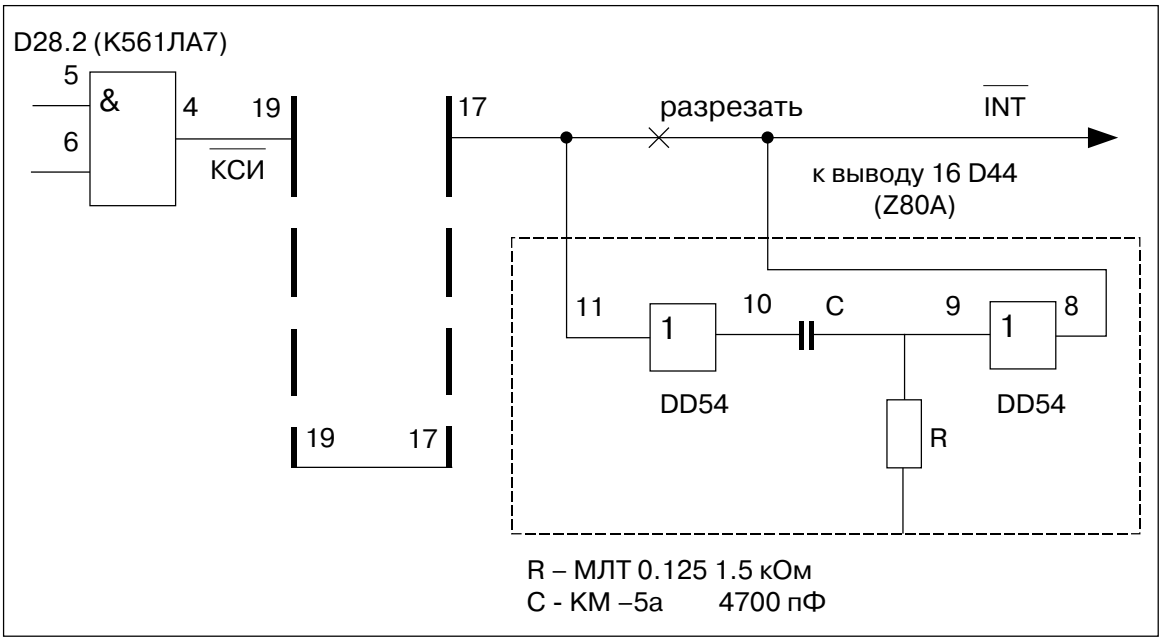


Рис.1

Если у Вас нет хорошего широкополосного осциллографа, то оценить необходимую емкость конденсатора можно прикидочным путем. Это делается так:

1) Постепенно снижая емкость конденсатора, определите порог, за которым прекращается опрос клавиатуры. Включенный компьютер выводит на экран "(C) 1982...", но не реагирует на нажатие клавиш.

2) Увеличьте полученное пороговое значение примерно в 1,5 ... 2 раза и впаяйте в схему конденсатор необходимой емкости.

Обычно этот метод неплохо срабатывает, а получаемая при этом длительность сигнала достаточно оптимальна для Вашего экземпляра компьютера.

В заключение для менее опытных пользователей небольшая справка по местонахождению дифференцирующей цепочки в других упомянутых моделях компьютера.

1) "Пентагон 48К" (дисковый вариант) - элементы C5,R13,R15, исходный сигнал

снимается с вывода 8 микросхемы D5.2 (К555ЛИ3).

2) "Ленинград-1" - элементы С3, R2, далее сигнал инвертируется на D1.6 и поступает на вывод 16 Z80A. Исходный сигнал снимается с инверсного выхода триггера D8.2 (K555TM2).

Если обозначения элементов на Вашей схеме не совпадают с указанными, то их легко найти, проследив, откуда приходит сигнал на вход /INT процессора Z80A.

Сведениями по другим моделям автор пока не располагает.

Описанная методика проверена при наладке нескольких компьютеров каждой из версий. Компьютер версии "Москва", отлаженный, как описано выше, работает в течение полугода нормально. Без проблем запускается ARKANOID и другие игры. Единственная игра, не работающая на этой машине - "MOVIE" (причины пока не выяснены, но предполагается, что дело в отсутствии "подхвата" атрибутов экрана, т.к. резисторы 470 Ом на поле памяти 16K не установлены). - см. ZX-РЕВЮ N 3, стр.52 (FORUM).

Проблемы ELITE.

Нашему читателю из г.Челябинска Кислову Л.Н. кажется удалось в открытом космосе столкнуться с космической платформой. Причем произошло это не где-нибудь в глубинах отдаленных галактик, а у самой первой планеты - LAVE. Вот выдержки из его бортжурнала.

Однажды, решив начать свою карьеру с нуля, я, имея рейтинг DANGEROUS, перезагрузил программу и начал игру с HARMLESS. При перелете с планеты LAVE на планету REORTE я столкнулся с летающим объектом необычной формы (бесформенный многогранник). Объект появился на переднем экране и шел мне навстречу. Приняв его за астероид, я начал обстрел издалека, надеясь на небольшую премию.

Каково же было мое удивление, когда "астероид" встретил меня ответным огнем. Несмотря на то, что боевой опыт у меня есть, поразить его пульсирующим лазером не было никакой возможности. Объект имел прекрасную защиту и, несмотря на многократные, длительные и точные попадания, не уничтожался. Во-вторых, он имел огромную скорость и постоянно маневрировал.

Самое интересное, когда от объекта начали "отслаиваться" корабли по 5-6 шт. за раз. (Наш читатель приводит их рисунок - это "Сайдуиндеры". - "ИНФОРКОМ").

Не обращая на них внимание, я продолжал преследовать объект. Примерно через минуту, поняв, что прекращать преследование я не собираюсь, объект выпустил новую партию кораблей - еще около 5 штук.

Общими усилиями они довели мою энергию почти до нуля, и я решил воспользоваться ракетами (обычно я ими не пользуюсь - считаю дорогими и малоэффективными). Первая ракета пропала неизвестно куда. Остальные тоже. Вылетев, они просто исчезали, не взрываясь. Последняя же ракета и не стартовала - появилась надпись о ее повреждении Missile Crashed.

Вскоре меня добили, и я так и не знаю, что же это было.

Версия программы характерна тем, что в стартовой экране выдается сообщение PHESS SPACE COMMANDER, а не LOAD NEW COMMANDER (Y/N).

* * *

Дорогие читатели. Вас, конечно не удивит, если "ИНФОРКОМ" признается, что получает сотни писем ежемесячно с описанием межзвездных баталий. Мы могли бы открыть серию романов, но должны честно сказать, что такого явного и яркого свидетельства, наличия космических платформ, как это, к нам еще не поступало.

* * *

Интересный эффект в этой программе обнаружил другой наш читатель, тоже из города Челябинска - Прыкин А. В (стаж полетов более года).

Он умеет перелетать со станции на станцию без затрат топлива и без боевых действий, воспользовавшись "жучком", имеющимся в программе, причем, как ему кажется, это не ошибка программы, а хитрая уловка фирмы.

Вот как это делается.

Перед стартом со станции выберите любую планету в данной галактике. Теперь стартуйте. Выйдя в космос, развернитесь на 180 градусов и вновь идите на стыковку со станцией. Когда сближение произойдет, перед самым входом на станцию, нажмите "H". Если выбранная Вами планета находилась в пределах круга, то Вы окажетесь на ней. Если же Вы выбирали планету вне пределов круга, ограничивающего гиперпереход, то нажимать надо "GH" - результат - тот же.

У "ИНФОРКОМа" сразу возникает вопрос: "А нельзя ли аналогичным способом перелетать и между галактиками?" Чем не предмет для исследования?

* * *

Интересное направление исследований начали вести сразу несколько наших корреспондентов. Они повели атаку на тот блок состояния, который выгружается на ленту при отгрузке программы.

Вот что пишет пилот Иванищенко (г. Орск).

Этот блок имеет всегда длину, равную 102 байтам. Методом проб удалось узнать назначение некоторых байтов в этом блоке. Прочие - пока не установлены, исследование продолжается.

В таблице приведены:

I - номер байта,

II - первоначальная установка,

III - назначение.

I	II	III
1	-	Имя командира
…	-	(10 символов в
10	-	кодах ASCII)
11	0	?
…	0	?
20	0	?
21	232	Сумма денег
22	3	в
23	0	кредах

Далее следует информация о количестве загруженных товаров

24	0	Food
25	0	Textiles
26	0	Radioactives
27	0	Slaves
28	0	Liquour/wines
29	0	Luxurious
30	0	Narcotics
31	0	Computers
32	0	Machinery
33	0	Alloys
34	0	Firearms
35	0	Furs
36	0	Minerals
37	0	Gold
38	0	Platinum
39	0	Gem-stones
40	0	Alien Items

41 – 46 – не установлены

Далее следует информация о снаряжении и бортовом вооружении

47	70	Топливо
48	3	Ракеты
49	0	Система E.C.M.
50	0	Extra Pulse Laser
51	0	Beam Laser
52	0	Fuel Scoops
53	0	Escape Pod
54	0	Energy Bomb
55	0	Energy Unit
56	0	Docking Comp.
57	0	Galactic Hyperdrive
58	0	Military Laser
59	0	Mining Laser

60 – 102 не установлены

Дополнительную информацию дает А.И. Ивлев из Симферополя. Он исследовал байты 42,13,44,45. Там находится информации о вооружении четырех боевых надстроек корабля (носовая, кормовая и две бортовых). При этом:

0 - отсутствие оружия;
1 - Pulse Laser;
2 - Beam Laser;
3 - Military Laser;
4 - Mining Laser.

Если заслать туда число, большее 4-х, можно получить интереснейшие эффекты. Например такую чепуху, как "Front Energy Bomb", причем со свойствами и боевого и технологического лазера одновременно. Очень удобно - стреляет как боевой, а обломки от астероидов летят, как от технологического.

* * *

И, наконец, вопрос к читателям, который задает наш давний партнер из г.Батуми - Гогуа А.В.: "Нет ли у кого версии программы ELITE, работающей в дисковой операционной системе". (Очевидно "самоделки", сделанные с помощью Magic-Button здесь не имеются в виду - "ИНФОРКОМ").

Советы и секреты

Просматривая код хорошо всем известной программы фирмы MIKROGEN "Three Weeks in Paradise" наш читатель Сукач К.М. (г.Херсон) нашел послание, предназначенное любителям взламывать чужие программы. Содержание его интересно, поэтому любопытным предлагается небольшая программа для его прочтения.

Так как основной игровой блок не имеет "хэдера", то сначала надо его сделать командой

```
SAVE "PARADISE!" CODE 26490, 38582
```

Затем надо считать этот заголовок и после него загрузить игровой блок.

После загрузки надо набрать небольшую программу-просмотрщик.

```
5 CLS
10 FOR n=63923 TO 65072
20 PRINT CHR$ (PEEK N);
30 NEXT n: PAUSE 0: GO TO 5
```

И запустить ее <RUN>.

На экране появится текст послания.

Фильков Андрей Петрович (Москва) и другие сообщают о том, как в программе EQUINOX, экспертную проработку которой мы недавно давали, обессмертить своего робота. Суть такова:

На первом уровне нужно найти карту PETE, вернуться с ней к месту старта и находясь в левом верхнем углу, нажать одновременно клавиши R,N,C.

А в программе EXOLON можно получить бессмертие, если в режиме переопределения управляющих клавиш (DEFINE KEYS) ввести буквы Z,O,R,B,A.

* * *

Юрий Евгеньевич Горяинов из г.Славутич Киевской обл. сообщает о том, что в программе EQUINOX оказывается есть еще и встроенный МОНИТОР. Мы, правда, не знаем встроила ли его фирма или просто в спешке забыла убрать (и такое бывает) или его встроил кто-то из тех бойцов невидимого фронта, снимавших с программы фирменную защиту, но он там есть. Впрочем, откуда берутся в программах вмонтированные мониторы Вы можете узнать, если прочтете в данном номере статью Стива Тернера "Профессиональный подход".

Чтобы его вызвать, надо убрать автостарт программы и после загрузки дать команду RUN. МОНИТОР отображает на экране содержимое всех регистров процессора и ячеек памяти, позволяет записывать в них данные как в 16-ричном виде, так и в десятиричном и в символьной виде. Вот его команды:

1 - переключение DEC и HEX систем счисления;
2 - переключатель "символьное"/"числовое" представление;
3 - запись числа в регистр (через ENTER);
4 - запись числа по адресу (первому на странице);
5 - листание вниз на 64 адреса;
6 - листание вниз на 1 адрес;
7 - листание вверх на 1 адрес;
8 - листание вверх на 64 адреса;
9 - выбор регистра процессора (вперед);
0 - выбор регистра процессора (назад);
SPACE - занесение начального адреса страницы.

Непробиваемая защита

"Очень многие сейчас занимаются "хэккерством", упражняясь во взломе и изменении программ" - пишет наш читатель из г.Таганрога Василенко В.И. - И дело это не столь сложное, как могло бы показаться. Имея "библию" Яна Логана и Ф. О'Хары "The Complete Spectrum ROM Disassembly" это сделать можно за полчаса, но увы не со всякой программой. Как подойти к защитам, в которых циклически изменяется информация, например как показано ниже?"

"ИНФОРКОМ" напоминает о том, что Вы можете заказать у нас качественно сделанную копию Мельбурнского издания книги Логана и О'Хары. Должны также добавить, что книга конечно хороша, но для неподготовленного читателя может быть сложноватой и наш комментарий к ней, который мы ведем уже год в виде сериала "Секреты ПЗУ" будет более чем полезен.

Пример такой защиты с циклической перекодировкой блока:

```
LD HL,nn          (адрес начала блока)
LD BC,nn          (длина блока)
LD A,n            (ввод ключа-декодера)
XOR (HL)          (декодирование)
LD (HL),A         (замена закодированного значения на раскодированное)
INC HL            (переход к очередному байту)
DEC BC            (уменьшение счетчика)
LD A,C            (проверка счетчика на ноль)
OR C
JR NZ,n           (повтор декодирования)
DEFB ..............
DEFB ..............
закодированный блок
```

```
DEFB ..............
```

В данном примере байт за байтом декодируется блок, размещенный сразу за декодирующей процедурой. Декодирующей функцией служит XOR. После окончания процесса декодирования исполнение переходит к первому байту из декодируемого блока. Вроде бы все просто, но вдруг! О ужас! Оказывается, что раскодированный блок содержит в самом начале примерно такую же процедуру, а дальше за ней опять "абракадабра". Раскрыв и запустив декодирующую процедуру, мы декодируем и ее, но дальше опять все то же и так до 100 - 200 раз. Что же делать, нельзя ли как-то это преодолеть?

Мы должны откровенно признаться, что бессильны чем-то помочь нашему читателю и привели здесь логику работы защиты во-первых для тех, кто не знает этого приема, а во-вторых в надежде на то, что отзовется кто-нибудь, кто решил такую проблему, хотя мы признаться мало в это верим. Дело вот в чем.

Несколько лет назад на Западе было объявлено о создании интегрального нераскалываемого загрузчика ALKATRAZ LOADER. Он работал именно по этому принципу. Была объявлена громадная премия тому, кто сумеет его "расколоть" и, насколько нам известно, эту премию никто не востребовал. Конечно "расколка" аппаратными средствами не признавалась.

Кодирующая программа продавалась фирме, желавшей защитить свою продукцию. После работы кодировщика к программе приписывался загрузчик и теперь никто, в том числе и фирма, защищавшая свою продукцию, не могла снять защиту.

Самое суровое, что этой защитой защищалась даже не сама программа, а другие защищающие процедуры. Возьмем, к примеру, шедевр адвентюрных игр - программу KAYLETH, написанную по повести Айзека Азимова. Там "Алькатрасом" защищался блок, обеспечивающий ускоренную нестандартную загрузку и, к тому же, между блоками была встроена пауза с замеряемой продолжительностью и с замеряемым уровнем шумов. Копировщики эту программу естественно не берут, а прямое переписывание с магнитофона на магнитофон удается только на очень дорогих высококачественных аппаратах, да и то работоспособной остается максимум вторая копия.

Вопрос поднятая нашим читателем достаточно интересен. Мы бы с радостью опубликовали исследования наших авторов, работающих в этом направлении. Наш собственный опыт увы исчерпывается одной-двумя безуспешными попытками. Так, например, раскрывая программу Pinball Wizard, мы прошли тяжелым трудом 20-25 этапов защиты, но наткнулись на то, что сравнительно простые декодирующие функции типа XOR (HL) или DEC A сменились гораздо более крутыми:

```
LD B, (IY+n)
LD A, (IX+n)
XOR B
```

Если Вы с самого начала не отслеживали содержимое регистров IX и IY, то после труднейшего дня кропотливой работы очень приятно столкнуться с такой функцией.

По всей видимости для снятия защит такого типа необходимо специальное программно-инструментальное средство, возможно что-то типа резидентного монитора.

Для тех читателей, кто готов осудить нас за подстрекательство к "хаккерству" скажем, что мы со всем уважением относимся к авторским правам, (сами ведь тоже пишем программы для IBM-совместимых машин) но считаем допустимым и даже полезным для самообразования "влезть" в код чужой программы, если при этом:

- программу не "портят" в угоду личным амбициям;
- не меняют данных об авторстве программы;
- внесенные изменения оставляют для себя и не отравляют жизнь будущим пользователям, распространяя их.

Одним словом, мы относимся к этому как к спорту, который как любой вид спорта может быть чистым, а может быть и грязным. И зависит это не от спорта, а от спортсмена.

По всей видимости в условиях отсутствия нормального авторского законодательства и нормального легального распространения программного обеспечения основным критерием, которым стоит руководствоваться, должно быть УВАЖИТЕЛЬНОЕ И КОРРЕКТНОЕ ОТНОШЕНИЕ К АВТОРУ ПРОГРАММЫ.

PYJAMARAMA

Фирма: MIKRO-GEN
Год выпуска: 1984

Многие любят игры аркадно-адвентюрного жанра. В них можно играть неделями. К сожалению, сложность многих программ не позволяет начинающим получить все то удовольствие от игры, на которое они вправе рассчитывать.

В последнее время наши эксперты обратили пристальное внимание на игры этого жанра. Вместе с тем, нам бы хотелось дать им какое-то направление в работе. Дело в том, что подробно расписанный порядок действий может отбить интерес к игре и здесь нужен взвешенный подход.

Сегодня в качестве одного из возможных образцов инструкции к аркадно-адвентюрной программе мы предлагаем описание программы PYJAMARAMA.

Должны сразу оговориться, что это описание достаточно хорошо распространено по стране. Мы его сделали в те далекие годы, когда никакого "ИНФОРКОМа" еще не было и в проекте. Вполне может быть, что оно вам знакомо и мы просим Вас не воспринимать его как несвежий материал. Его цель - дать определенное направление всем экспертам, которые работают по нашему заданию в этом жанре.

Цель игры: Вам надо разбудить Уолли, главного героя многих игр фирмы MIKRO-GEN. Для того, чтобы его разбудить, надо завести будильник.

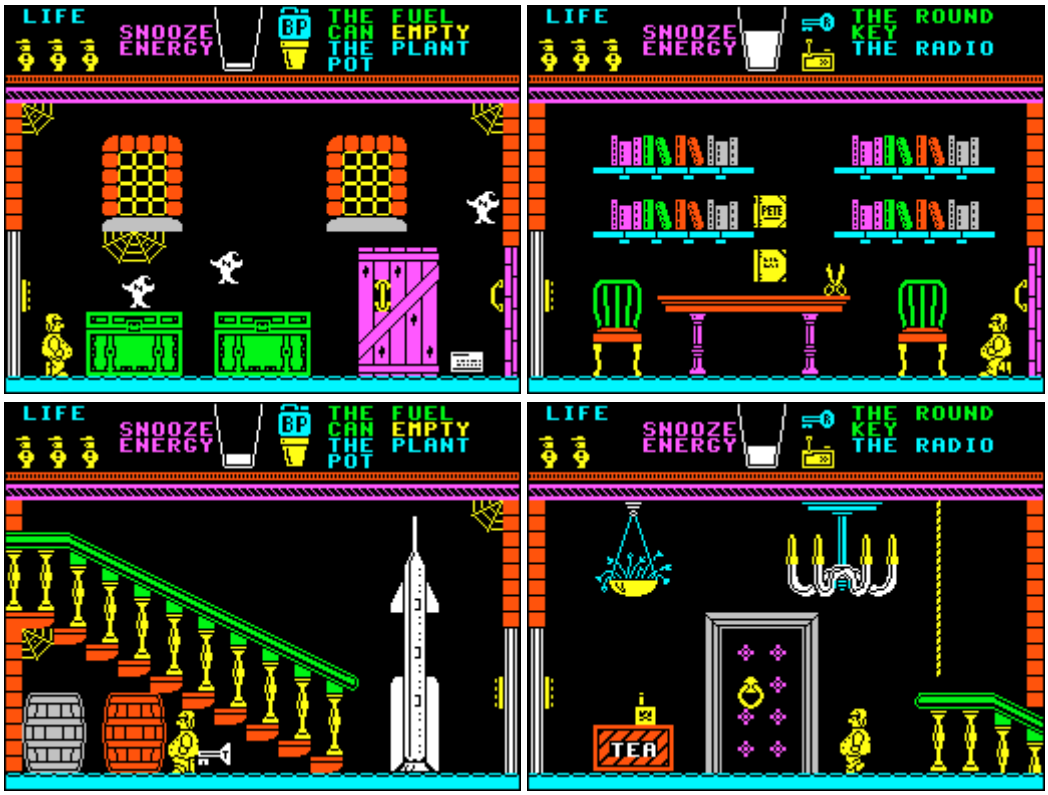
Данная инструкция построена таким образом, чтобы читатель не потерял интерес к игре. Для этого информация, необходимая для полного завершения игры сообщается постепенно, небольшими разделами. Ознакомившись с очередным разделом, попробуйте свои силы и только если у Вас все равно ничего не получается, читайте очередной раздел.

1.

Для того, чтобы найти ключ от будильника, надо слетать на Луну. Ключ находится за решеткой с магнитным замком, поэтому надо иметь при себе магнит. Проход к ключу охраняют инопланетяне, которых можно пройти только, если доставить на Луну заряженный лазерный пистолет. Ракета запускается только если у Вас есть при себе полная канистра топлива. Так как более двух предметов иметь при себе одновременно нельзя, то надо в первый рейс на Луну доставить туда заряженный лазерный пистолет, а только во втором рейсе с магнитом идти за ключом…

Итак, три основные проблемы - заправить ракету топливом, найти и зарядить лазерный пистолет, взять магнит (он на кухне под столом, закрыт на ключ).

Кроме обычного передвижения по дому, можно пользоваться лифтом. Вход в него из библиотеки. По дороге, проходя через комнату, в которой лежит пляжный мяч, надо подпрыгнуть и переключить кнопку вызова лифта так, чтобы стрелка смотрела вниз. В лифте есть четыре бокса, прыжком в которые лифт отправляется на соответствующий этаж (лампочка наверху у включенного бокса зажигается). Первый этаж - подвал. Второй этаж - исходный, с него начинается игра. На третьем этаже находится ракета. На четвертом этаже спальная, где спит Уолли, ванная комната, комната видеоигр и прочие комнаты. Кстати, если Вы войдете в комнату видеоигр и пройдете все уровни игры, Вы получите дополнительную жизнь.



Как заправить ракету топливом? Для этого надо в комнате с щелкающими капканами взять канистру. Затем на первом этаже найти топливную помпу. Если подойти к ней с пустой канистрой, она наполнится.

Где взять лазерный пистолет? Лазерный пистолет лежит в биллиардной. Туда можно войти только если у Вас есть специальный квадратный ключ. Дверь в биллиардную на третьем этаже. (С четвертого этажа спуститься по лестнице вниз), но без ключа она не открывается. Проход к ключу следующий: на первый этаж, затем вторая комната с привидениями направо, затем по ящикам выпрыгнуть в зарешеченное окно, далее направо и вниз, там горит огонь. Чтобы его пройти, надо иметь огнетушитель. Далее вниз и налево.

Как взять магнит? Чтобы взять магнит, надо иметь специальный ключ от ящика. Ключ висит под потолком в той комнате, где есть воздушный шарик. Первое - шарик Вас

поднимет наверх, если Вы перережете веревочку ножницами, второе - поднявшись наверх, Вы сможете прыжками подойти к ключу, но до него не допрыгнете. Для этого надо под ключ подставить ящик. Это делается переключением кнопки "HELP" в комнате на четвертом этаже рядом с ванной так, чтобы стрелка смотрела вниз. Помните, что эта клавиша может самопроизвольно выключаться, так что может быть Вам придется сбегать наверх еще раз.

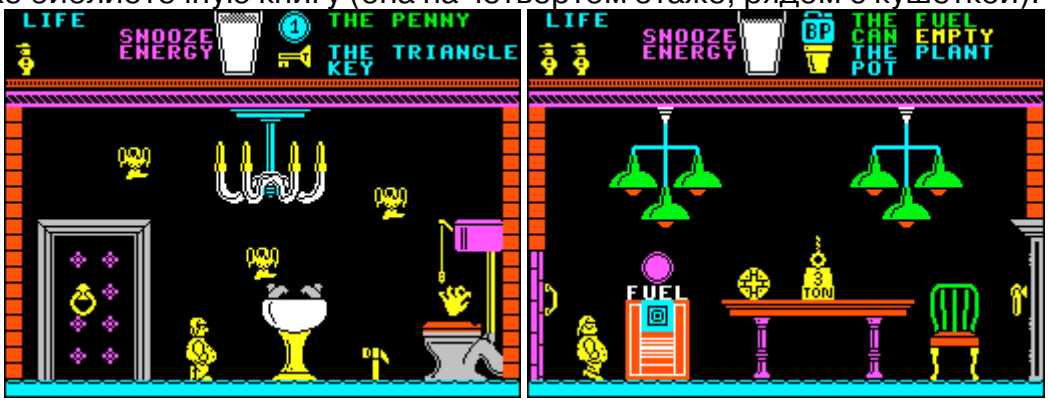
3.

Как отключить капканы, чтобы взять канистру? для этого надо иметь с собой ведро воды. Внеся его в комнату, можно там его и оставить вместо канистры, тогда капканы перестанут работать.

Ведро берется по дороге в библиотеку, прыжком со стула. Заполнить его водой можно в ванной комнате на четвертом этаже, пройдя под краном. Надпись "EMPTY" (пустое) сменится надписью "FULL" (полное). Помните, что если Вы где-либо выпустите ведро из рук (поменяете на какой-либо предмет), вода прольется и оно опять станет пустым.

Где взять огнетушитель? Огнетушитель находится в лифте. Над ним надпись "SMASH GLASS" (разбей стекло). Разбить стекло нужно молотком в прыжке.

Как взять ножницы? Ножницы находятся на столе в библиотеке, но справа со стула до них не допрыгнуть, а слева не пройти - мешают движущиеся книги. Чтобы взять ножницы, надо иметь с собой мотоциклетный шлем для защиты (он на третьей этаже лежит на сундуке), а также библиотечную книгу (она на четвертом этаже, рядом с кушеткой).



4.

Где взять молоток? Молоток лежит в туалете. Вход в туалет рядом со входом в биллиардную, но эта дверь закрыта. Вход платный, надо при себе иметь мелкую монету - пенс.

Как взять шлем? Шлем берется, если скатиться по перилам с четвертого этажа на третий, при этом надо иметь при себе ключи зажигания. Ключи зажигания находятся на пути к квадратному ключу (с первого этажа через зарешеченное окно), но берутся только в обмен на водительские права, они лежат в подвале.

Как взять библиотечную книгу? Для этого надо поменять ее на читательский билет, который находится на третьем этаже на стене в комнате по соседству со шлемом.

Где взять пенс? Надо сначала в столовой взять фунт стерлингов и идти с ним на четвертый этаж. Там надо найти разменный автомат с надписью "CHARGE" (размен). Прыжком поменять фунт на пенсы.

Как зарядить лазерный пистолет? Пистолет заряжается от батарейки, которая лежит в коннате на первом этаже. Проход в эту комнату из той комнаты, в которой стоит топливная помпа. Но дверь отпирается только трехгранным ключом. Он на третьем этаже рядом с ракетой. (При взятии этого ключа желательно, чтобы капканы уже были обездвижены, т.к. путь в лифт может быть закрыт, а путь наверх, через комнату с капканами, будет непроходим при работающих капканах). Отправляясь за батарейкой, на этаж N1, перед входом в лифт убедитесь, что и пистолет и ключ с Вами.

Войдя в комнату с батарейкой, немедленно (!) прыгайте на ящик и на стол, но не проскочите мимо батарейки.

Как запрыгнуть на перила, чтобы взять шлем? Проход четвертого этажа представляет

собой движущийся конвейер. Передвигаться по нему в противоположном направлении можно только прыжками. С движущегося конвейера запрыгнуть на перила можно, изъяв привод. Он лежит на сундуке в углу. Для этого, приехав на лифте на четвертый этаж и выйдя из двери, сразу прыгайте на ящик и берите привод.

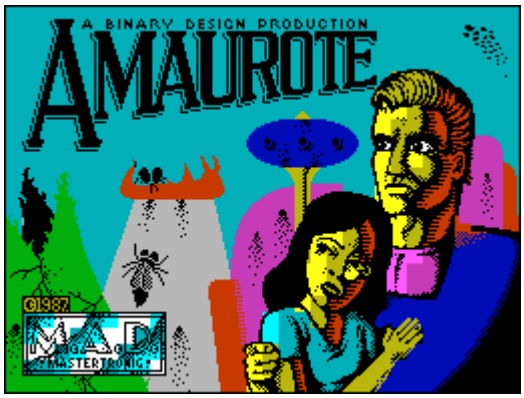
В процессе игры Вы можете подкреплять силы Уолли пищей, которая иногда появляется в некоторые местах (шоколад, джем, яблоко и пр.). Желательно пользоваться этим рационально. Чтобы не расходовать еду, надо просто через нее перепрыгнуть.

В комнатах есть и другие предметы, которые также могут быть взяты, хотя непосредственного участия в игре не принимают.

Слово экспертам

AMAUROTE

M. A. D. 1987 Г.



Эксперт Борисов П.М.
г. Норильск

Программа относится к аркадно-адвентюрным играм с элементами ACTION.

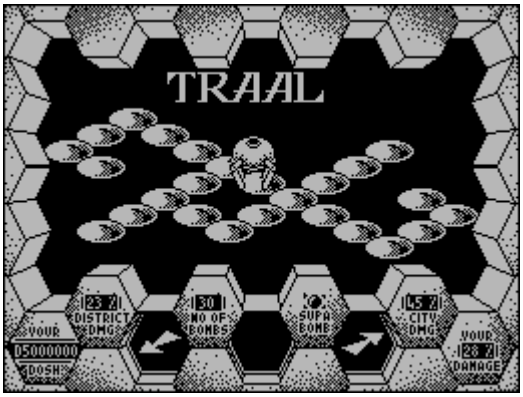
Ваша задача - очистить город от вредных инопланетных насекомых. Для этого Вам выдали в полное распоряжение боевую машину "Арахнус-4", оснащенную совершенным радарным оборудованием.

У Вас есть двусторонняя радиосвязь с базой и 30 инсектицоидных бомб. Мы надеемся, что с их помощью Вы не разрушите город до основания. Вам также ассигнована сумма в 5 млн. долларов.

Население покинуло город, армия также бессильна, теперь вся надежда на Вас.

Город.

Город разделен на 25 районов, в каждом их которых насекомые организовали свои колонии. У Вас есть карта и Вы можете входить в эти районы в любом порядке. Каждый район представлен кружком и, перемещая "Арахнус-4", Вы производите выбор района, при этом на экране зажигается название. Когда Вам удастся уничтожить всех насекомых в районе, Вы вернетесь к карте.



Насекомые.

Существует три типа насекомых:

МАТКА. Это Ваша основная цель, но к ней очень нелегко пробиться, т.к. она находится под постоянной охраной. Она выполняет две функции. Во-первых, она отдает приказания трутням, которые работают на основе той информация, которую доставляют разведчики и, во-вторых, выводит новых насекомых.

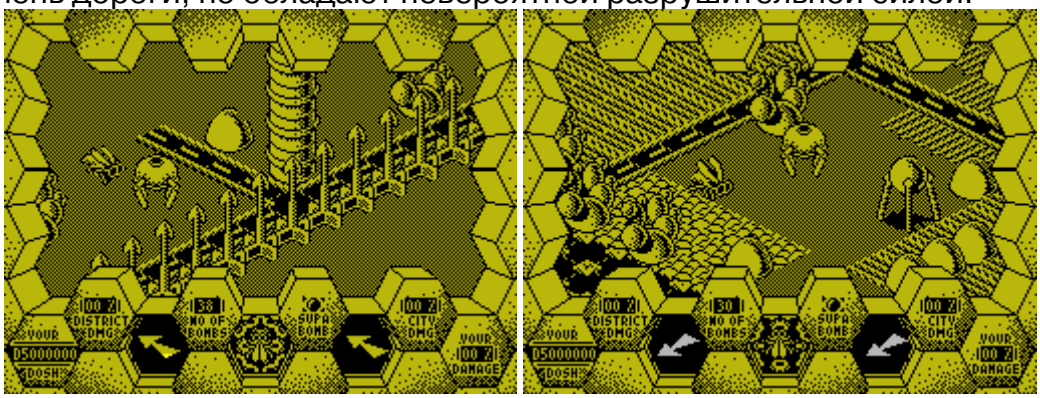
Каждый раз, когда Вы убиваете очередное насекомое, она выводит новое, но не сразу. Время, которое для этого необходимо, зависит от того, когда в последний раз ее кормили.

РАЗВЕДЧИКИ. Они летают по городу в поисках потенциальной пищи и непрошеных гостей (это касается и Вас). Если разведчики увидят Вас, они немедленно помчатся с донесением к матке и тогда у Вас будут большие осложнения.

ТРУТНИ. Их задача доставлять пищу и защищаться. Они довольно тупы, строго подчиняются данным им приказам, но не сдаются в бою и не отступают.

Бомбы.

Бомба запускается нажатием кнопки "огонь". Направление стрельбы - это направление Вашего последнего движения. Бомба летит до столкновения с насекомым или со зданием. Помните, что от Вас требуют очистить город от насекомых, а не разрушить его. Не поддаются уничтожению только периметр города и сама матка. Для ее уничтожения Вам потребуется супербомба. С собой ее у Вас нет, но Вы можете сделать заказ в любое время. Эти бомбы очень дороги, но обладают невероятной разрушительной силой.



После того, как бомба запущена, Вы не можете запустить вторую, пока первая не взорвется.

Число, которое показывает оставшееся количество бомб, в это время имеет красный цвет, даже если бомба находится за пределами экрана.

Радио

Задействование радио открывает перед Вами следующее меню:

1. Заказать бомбы.
2. Заказать супербомбу.
3. Ремонт.
4. Выход из города.

Бомбы доставляются в город по воздуху и Вы можете разыскать их с помощью своего радара. Если Ваш "Арахнус-4" претерпел большие повреждения, можете вызвать новый, но это стоит денег.

Радар.

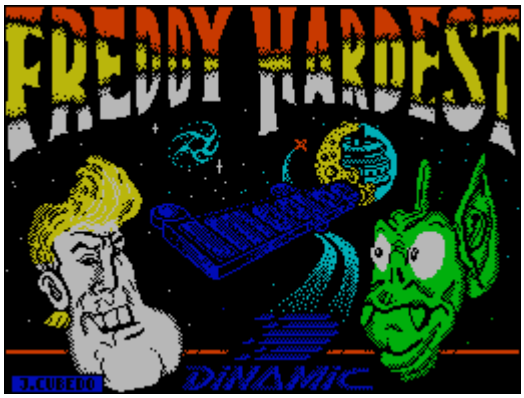
Вы можете запрограммировать свой радар на отыскание ближайшего насекомого, матки или доставленных бомб. Для этого нажмите необходимую клавишу и следуйте за указателями.

Управление.

Вправо-вверх - Y...P
Вправо-вниз - H...ENTER
Влево-вверх - Q...T
Влево-вниз - A...G
"Огонь" - B...SPACE
Радио - CAPS SHIFT
Изменение цвета - V
Для радара:
Поиск насекомых - Z
Поиск матки - X
Поиск бомб - C

FREDDY HARDEST

DINAMIC 1987



Эксперт Смирнов В. Б.
г. С. -Петербург.

Далекая галактика, незнакомая звезда... Одна из планет излучает загадочные радиосигналы.

Космический путешественник Фредди Хардест получает специальное задание Всегалактического Центра Космических Исследований узнать причину появления этих сигналов. И вот он, вдалеке от родной звезды, подлетает на своем космическом челноке к чужой планете. Бортовой компьютер сигнализирует о выходе на круговую орбиту, а затем беспристрастная техника фиксирует приближение инопланетного космического корабля. Фредди обеспокоен: корабль не отвечает на запросы, чужак открыл огонь по челноку Фредди. Завязался космический бой, который, впрочем, длился недолго. Фредди сумел уничтожить врага, но и сам еле дотянул до враждебной, как оказалось, планеты. Тщательно перевязав раны, отважный путешественник отправился в путь не зная, что его ждет впереди.

С этого момента Вы вступаете в игру. Вы должны помочь Фредди избежать мести инопланетян и улететь с чужой планеты.

Незадолго до гибели корабля бортовой компьютер выдал информацию о климате и населении планеты, а также о расположении городов инопланетян. Проведя расчеты, Фредди определил, что в пяти милях от места падения находится крупное поселение с космической базой. Тут же возник план: добраться до города и, если представятся возможность, угнать космический корабль. Планета населена рептилиями негуманоидного типа. Они имеют явное намерение уничтожить Фредди любой ценой. Ваша задача - помочь ему.

Оружие.

Итак, цель игры - добраться до города инопланетян. Выполнение задания осложняют снующие повсюду рептилии, которые, однако, не вооружены, в то время как у Фредди имеется лазерный пистолет новейшей конструкции. Для того, чтобы им воспользоваться, надо нажать клавишу "вниз" и, удерживая ее, клавишу "огонь". Пистолет обладает бесконечным, т. е. не уменьшающимся зарядом лазера.

Город.

Дойдя до лестницы, ведущей под землю, Вы должны высоко подпрыгнуть над ней и … можете загружать FREDDY HARDEST II.

Полезный совет.

Если у Вас нет желания сражаться с летающими (но не прыгающими) рептилиями, Вы можете при их подлете присесть. В этом случае они пролетают над Вами, не задевая.

Экран.

В верхней части экрана отображается игровая ситуация, в нижней расположено информационное табло. Рассмотрим его структуру.

Слева на табло изображен улыбающийся Фредди. Правее указано количество оставшихся кредитов (в начале их дается пять). Далее расположен символ пистолета. Ниже выводится количество очков.

Три квадрата, находящиеся в правой части информационного табло, не используются в первой части игры FREDDY HARDEST.

Очки.

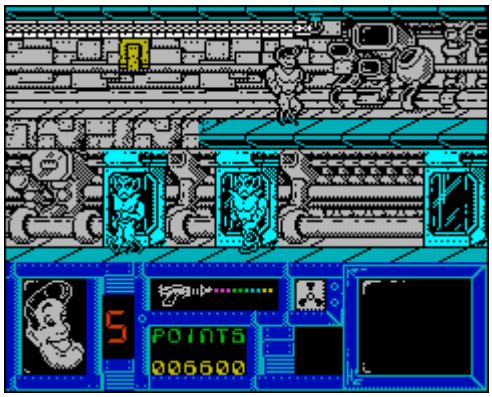
За каждую уничтоженную рептилию Вы получаете 200 очков.

FREDDY HARDEST II

Эта игра является продолжением игры FREDDY HARDEST. После того как Фредди добрался до города инопланетян, ему надо улететь с чужой планеты. Корабль космического путешественника разбит и у Фредди есть один выход: угнать один из боевых кораблей инопланетян.

Игра.

Ангары космических кораблей находятся в нижнем ярусе города. Корабли различаются по цвету. Существует четыре вида кораблей: зеленый (green), красный (red), синий (blue) и белый (white). Для побега Вы можете воспользоваться любым из них.



Для того, чтобы стартовать с планеты, необходимо подготовить корабль к полету. Подготовка заключается в проведении трех операции:

1. Заправить корабль ядерным топливом.
2. Открыть ангар заправленного корабля.
3. Определить код доступа в корабль.

Для выполнения этих заданий Вам необходимо научиться пользоваться служебными

компьютерами инопланетян. Они расположены на разных этажах города. Для получения информации от компьютера, надо встать рядом с ним и нажать клавишу "вверх". При этом на информационном табло в среднем квадрате начнет мерцать изображение дискеты, а в большом появится сообщение. Если Вы уже получаете информацию от компьютера, но передумали это делать (например, если к Вам приближается инопланетянин), нажмите клавишу "вниз".

Надо заметить, что не все компьютеры исправны. Некоторые выводят сообщение "Out of order" - неисправен.

Теперь подробнее о выполнении заданий.

Заправка.

Для начала вам надо найти контейнер с топливом. Он выглядит, как мерцающий квадрат с нарисованным на нем знаком радиации (т.к. топливо ядерное). С контейнером идите в комнату, где на полу написана буква N (от Nuclear - ядерный). Таких комнат в городе две. Встаньте на участок пола с изображением буквы. При этом контейнер окажется стоящим на полу. После этого подойдите к стоящему здесь же компьютеру и включите его. Контейнер опустится вниз. Далее по автоматическим линиям он будет доставлен к кораблю и загружен в него. На табло появится сообщение о цвете заправленного корабля. Например: Green ship full - зеленый корабль заправлен.

Открывание ангара

Для того, чтобы открыть ангар корабля, который Вы только что заправили, надо найти компьютер, управляющий открыванием данного ангара. Тут Вам придется побегать в его поисках.

При открывании ангара Вам выдается сообщение вида: Red on to hyperspace - Ангар красного корабля открыт.

Код доступа

Определение кода доступа в корабль производится также с помощью компьютера, который вам надо найти. При этом выводится такое сообщение: White captain, Code AUDAX - капитану белого корабля: код AUDAX.

Взлет

Надо заметить, что не обязательно выполнять все задания в приведенной порядке.

Итак, Вы заправили корабль, открыли ангар и знаете код доступа. Теперь спускайтесь в самый низ, к ангарам. Отыскать нужный ангар Вам помогут две буквы, написанные у каждого корабля, это сокращение от названия цвета, например: GR - GREEN (зеленый). После того, как Фредди зайдет в ангар, бортовой компьютер проверит наличие на корабле топлива и открыт ли ангар для вылета. Если все в порядке, у Вас запросят код доступа в корабль. Если Вы ответите правильно, то появится сообщение "Все системы готовы. Приготовиться к старту." и Вы можете наблюдать, как Фредди выруливает на космический фарватер.

Полезные советы

1. Для ускорения передвижения по городу можно пользоваться "темными" комнатами. Снаружи они выглядят, как большие черные раковины. Зайдя в такую раковину и выйдя с другой стороны, Вы окажетесь в другой части города.

2. Кроме того, можно пользоваться лифтами. К сожалению, они перемещаются лишь в пределах двух этажей.

Инопланетяне

С ними можно бороться двумя способами. Первый - бить их руками и ногами, а второй - стрелять в них из лазерного пистолета. Несмотря на кажущееся преимущество пистолета, первый способ предпочтительнее, т. к. им можно уничтожать все виды инопланетян, в то время как вторым только один вид - больших рептилий. К тому же, заряд пистолета во время

стрельбы уменьшается.

Экран

В отличие от первой части игры, во Freddy Hardest II используются три квадрата, находящиеся в правой части информационного табло. В маленьком квадрате индицируется наличие у Фредди контейнера с топливом. В среднем квадрате в момент получения информации от компьютера мерцает изображение дискеты. И, наконец, в большой квадрат компьютер выводит сообщения.

Длина линии, расположенной рядом со стволом пистолета, указывает оставшийся заряд. Когда Вы не пользуетесь оружием, оно автоматически подзаряжается.

Управление

Клавиши управления Вы выбираете сами перед началом игры. Для захода в ангар, включения компьютера, захода в "темные" комнаты, запрыгивания на горизонтальные и вертикальные веревки, используйте клавишу "вверх".

Если Вы нажмете клавишу "огонь" во время движения, то Фредди сделает удар рукой, а если Вы просто стоите, то - ногой. Когда Фредди висит на горизонтальном канате, удар производите клавишей "вверх".

Для использования лифта встаньте на него и нажмите клавишу "вверх" или "вниз".

Очки

Подсчет очков такой же, как во Freddy Hardest I, т.е. 200 очков за рептилию.

SILENT SERVICE.

MICROPROSE 1989

Эксперт Морозов С. Ф. г. Хабаровск

Эта игра относится к имитаторам и представляет имитатор подводной лодки.

Вы можете либо просто потренироваться в уничтожении вражеских кораблей, либо вести настоящую подводную войну. Для этих целей Вам предоставлена подводная лодка по номером 104. Это подлодка типа "К" - крейсерская, она обладает довольно мощным вооружением:

- носовая пушка с 80 снарядами;
- 6 носовых торпедных аппаратов с 14 запасными торпедами;
- 4 кормовых торпедных аппарата с 10 запасными торпедами.

Мощный дизель позволяет развивать скорость до 20 узлов в час (надводную) и до 10 узлов (подводную). Двойные переборки позволяют опускаться до 500 м.

Вам предлагаются 3 варианта ведения боевых действий.

1. Практика

Вам следует начать с этой миссии. Перед Вами будет находиться неохраняемый вражеский караван. Расстрелять его из пушки или торпедами не составит никакого труда, но именно здесь и осваиваются навыки. Начиная с мичмана, Вы вполне сможете дослужиться и до адмирала (присваивается и такое звание), став грозой морей.

2. Разгром вражеского конвоя.

В этой миссии Вам уже будут противостоять вражеские "охотники", которые будут атаковать Вас глубинными бомбами. Вот, где Вы вспомните, что Ваш корабль - подводный. Военные действия будут вестись и в ночное время, что заметно осложнит их.

3. Для самых отважных есть третья миссия - военное патрулирование.

Здесь Ваше мастерство как капитана субмарины проявится в полной мере. Вам придется не только топить корабли, но еще и искать их. Поможет Вам чутье, но и, конечно, смекалка. В погоне за транспортами будут меняться день и ночь. Вас будут пытаться протаранить и подбить глубинными бомбами, повредить лодку огнем пушек. Вы же, топя корабли и зарабатывая тоннаж, будете двигаться к главной цели - уничтожить противника и стать адмиралом.

Все действия происходят в районе Центральной Америки и Вашими главными врагами будут японские корабли. Что же, мы знаем, что американцы сумели одержать победу в этой войне, того же желаем и Вам. А чтобы победить, надо хорошо разобраться с игрой, что мы сейчас и сделаем.

Настройка программы

В этой игре нельзя использовать джойстик или что-либо другое, вся игра идет под управлением от клавиатуры. После загрузки перед Вами основное меню, где перечислены данные миссии. Нажав клавишу "1", "2" или "3", Вы выберете ту или иную миссию. Начинать, конечно, следует с первой. Итак, по порядку.

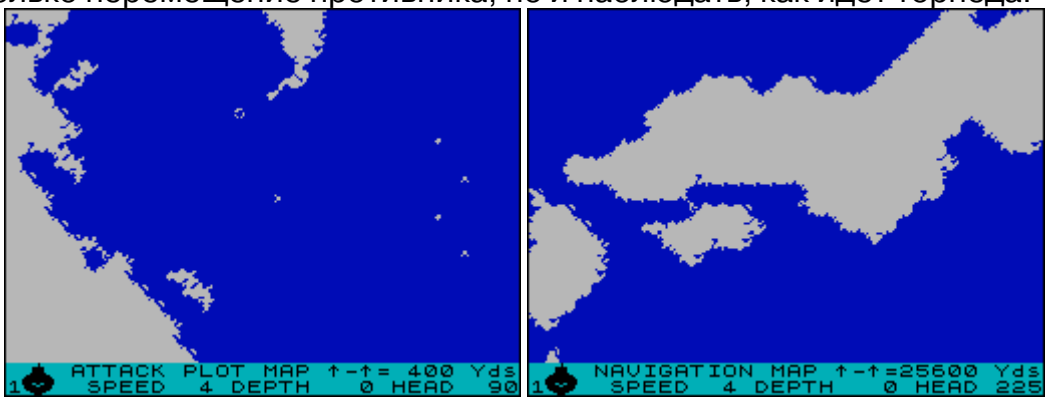
Если Вы выберете торпедно-пушечную практику (клавиша "1"), то окажетесь в дополнительном меню. Это меню, по сути дела - выбор уровня. Вначале Вы можете задать свой рейтинг от мичмана до капитана (клавиши "1" - "4"), затем можете еще более осложнить (или облегчить) жизнь, выбирая: имеются ли ограничения видимости, может ли конвой идти зигзагами, "опытный" ли охотник и (или) конвой и т.д. Управление происходит клавишами "Y" - вверх, "N" - вниз, "M" - выбор. Но для начала советуем сразу нажать "SPACE" и перейти к игре. Потом, по мере накопления опыта, Вы всегда сможете выбрать нужный Вам уровень.

Но если Вы в основном меню выберете конвойную акцию (клавиша "2"), то перед Вами станет задача выбора одного из шести мест прохождения конвоя и времени суток. За этим последует дополнительное меню, действия в котором аналогичны. И, наконец, решившимся на военное патрулирование, предстоит выбирать начальное положение подлодки (одно из пяти) и год войны. После этого Вы должны будете выбрать тип вражеского противолодочного корабля (клавиши "1" -"4"). Если Вы неудачно выберете тип японского "охотника" (то есть уровень не будет соответствовать типу корабля), то появится надпись "Потренируйтесь". Но какая бы надпись ни появилась, все равно Вы попадете в дополнительное меню, где следует действовать аналогично первым двум случаям.

Теперь нам надо освоить технику управления подлодкой. Самое сложное - освоить клавиши, сама же игра происходит в нескольких экранах, с ними разобраться легко.

1. Экраны карт

Сразу после выбора игры перед Вами на секунду появится карта побережья Центральной Америки, с ограничением того района, где Вам предстоит топить корабли. (Это относится к торпедной практике и конвойной миссии, режим же боевого патрулирования отличается, его мы рассмотрим ниже). После этого Вы окажетесь в экране-1 - навигационной карте, она имеет очень крупный масштаб. Поэтому лучше сразу переключиться на экран-2 - (карту патрулирования), у нее масштаб мельче и все цели довольно хорошо просматриваются. Переключение экранов производится клавишей "Z". Вы можете войти также в экране-3 (карта торпедной атаки). В этом экране Вы сможете наблюдать не только перемещение противника, но и наблюдать, как идет торпеда.



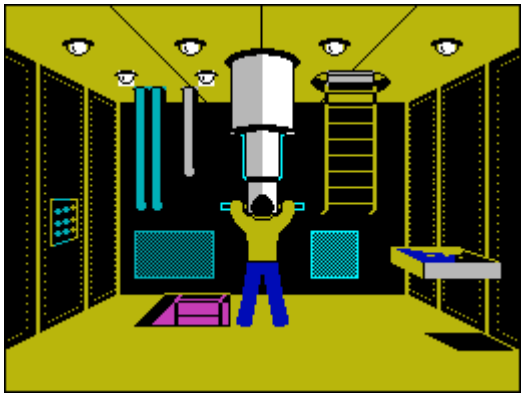
Эти экраны незаменимы при подводном положении субмарины, когда нет возможности наблюдать за вражескими кораблями через перископ и маневрировать. Обратное переключение карт осуществляется клавишей "X". Под картами - две информационные строчки, которые информируют Вас о скорости судна, глубине погружения, об угле движения подлодки, а также о том, какая карта на экране.

2. Боевая рубка

Определив местонахождение японских судов, можно переключиться на командный пост. Это делается клавишами "M" или "SPACE". Перед Вами будет внутреннее помещение корабля и капитан, находящийся у перископа. В боевой рубке Вы можете:

1. Посмотреть в перископ (если он поднят и глубина не превышает 44 метра[1]). Осуществляется нажатием клавиши "H".
2. Подняться на палубу (если субмарина в надводном положении). Клавиши "Y"+"M".
3. Обратиться к картам. Клавиши "O"+"M".
4. Осмотреть приборы, показывающие состояния всей лодки. Клавиши "9"+"M".
5. Осмотреть повреждения лодки, клавиши "H"+"M".
6. Вывести список потопленных судов и их тоннаж, а также количество оставшихся торпед. Клавиши "H"+"O"+"M".
7. Возврат в меню (кроме режима боевого патрулирования). Клавиши "H"+"9"+"M".

[1]Здесь и далее глубины указаны в футах. (Прим. NUK)



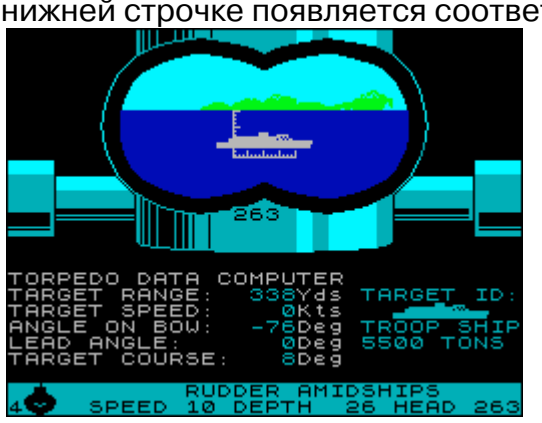
Это все, что касается действий в боевой рубке. Теперь рассмотрим эти действия по порядку.

3. Перископ

Выбрав этот режим в боевой рубке, Вы увидите то, что находится в поле зрения перископа, а также данные от системы наведения торпед. Они загораются лишь в случае захвата цели и содержат:

1. Расстояние до цели.
2. Скорость цели.
3. Угол упреждения торпеды.
4. Угол подлодки к цели.
5. Курс цели.

И опять же две последние строчки занимает информация о скорости, глубине погружения, курсе, а также дополнительная информация, которая будет рассмотрена ниже. Вначале Вам следует найти цели, что осуществляется поворотом субмарины. Он может быть выполнен при помощи руля или двигателя (для этого лодка должна иметь ход). Поворот влево - клавиша "C": один раз - поворот рулем, два раза - двигателем. Так же происходит поворот вправо - клавиша "V". В нижней строчке появляется соответствующая надпись.



Остановка осуществляется клавишей "ENTER". Можно поднять или опустить перископ - клавиша "P". Появляется соответствующее сообщение. Скорость лодки задается клавишами "2"-"5".

Погружение осуществляется клавишей "D", при этом раздается предупредительный звуковой сигнал и выдается сообщение. Если в этот момент перископ был поднят, то Вы будете продолжать наблюдение до глубины 44 м. Затем, как если бы перископ был бы опущен, Вы переходите в боевую рубку к карте.

Для остановки погружения Вам надо, нажав "9"+"M", перейти в экран приборов, и нажатием "ENTER", остановить погружение.

Дальнейшие действия могут быть например такими:

- поднять перископ (если он был опущен);
- осуществить всплытие (клавиша "S") до разрешенной глубины, после чего вновь можно производить наблюдение.

Скорость при подводном положении, конечно, меньше. Полная остановка с выключением механизмов осуществляется клавишей "1".

Можно дать задний ход. Для этого, дав одну из четырех скоростей, надо нажать

клавишу "R". Теперь, поймав цель в перекрестие прицела, можно получить информацию от системы целеуказания.

Можно также идентифицировать тип судна, нажав "I". Можно поворачивать лишь перископ клавишами "9","0", торпеда пойдет по заданному таким образом курсу.

Дальность действия торпеды - 3000 ярдов, поэтому, когда до цели осталось меньше 3000 ярдов и она поймана в прицел, Вы можете смело атаковать торпедой (клавиша "T").

В надводном положении можно использовать и пушку.

Прицеливание осуществляется клавишами "J" (увеличение угла прицеливания) и "K" (уменьшение), стрельба - клавиша "G". Если Вы догоняете цель, но она приближается слишком медленно, обладая хорошим ходом, то можно использовать 2-ю, 3-ю или 4-ю временную шкалу (клавиша "F"). Переход на нормальную шкалу происходит при выхождении цели из прицела или по нажатию клавиши "N".

Это практически все клавиши, используемые а данном экране. Есть правда еще клавиша "A". Она начинает работать в случае, если в дополнительном меню Вы выбрали режим ручной установки угла упреждения торпеды. Тогда при нажатии на "A", появляется угол упреждения, который можно увеличивать и, уменьшать клавишами "9","0". Устанавливается он так:

1) От полученного значения отнять градусную меру курса Вашей субмарины.

Полученное значение и следует вводить. Выход осуществляется нажатием на "M". Пауза – клавиша "W".

Все команды данного экрана рассмотрены, но есть еще несколько моментов. При попадании торпеды в цель появляется сообщение о том, что сонар зафиксировал взрыв. Такая же надпись появляется при попадании снарядом из пушки.

Не все корабли тонут от одной торпеды, поэтому Вам придется стрелять несколькими.

Зарядка торпед происходит автоматически через некоторый промежуток времени. При опустошении торпедных аппаратов появляется сообщение.

При повороте перископа от оси субмарины более чем на 90 градусов, в ход вступают кормовые торпедные аппараты.

Скорость заднего хода значительно меньше скорости движения вперед.

Возврат в боевую рубку осуществляется нажатием на "SPACE".

Когда вражеский корабль тонет, то появляется информация.

Это самый главный экран, и на его освоение стоит потратить время.

4. Палуба

В этот экран Вы можете войти только в надводном положении. Вы оказываетесь над боевой рубкой. Открывающийся вид гораздо шире, чем тот, который возможен при работе с перископом. При этом появляется сообщение об условиях видимости, а также об угле наблюдения. Конечно, этот режим ничего практически не дает в военном отношении, но зато делает игру несколько более разнообразной.

В этом режиме Вы можете проводить практически все те же действия, что и в боевой рубке.

Внизу также присутствуют информационные строки. Если Вы попытаетесь стрелять торпедами, то попадете в боевую рубку. Стрельба из пушки разрешается. Возврат в боевую рубку - клавиша "SPACE".

5. Приборная панель

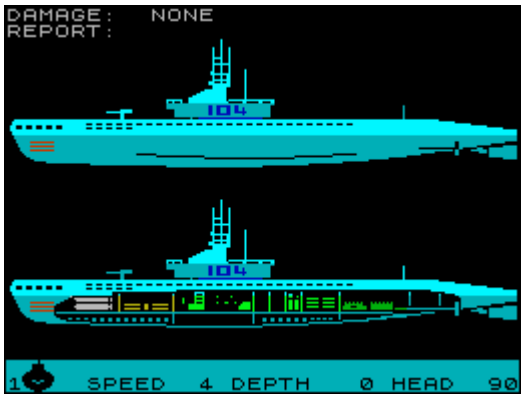
На этом экране присутствуют все приборы, необходимые для плавания: глубиномер, датчик скорости хода, компас, часы, указатель положения перископа, а также счетчик торпед в торпедных аппаратах и запасных и счетчик снарядов для пушки.

Внизу информационные строки.

Возврат в рубку - "SPACE".

6. Экран повреждений

Если Вам все-таки не удалось избежать повреждений, то Вы сможете осмотреть их в этом экране. Самое опасное повреждение - это нарушение герметичности лодки. Лодка набирает воду и начинает погружаться. Когда глубина достигает 460 м, то не выдерживают переборки и Вы погибаете.



Плохо, когда повреждаются:

1) носовые и кормовые торпедные аппараты, так как Вы не сможете стрелять из них;

2) перископ, потому что под водой Вы уже не сможете атаковать противника.

Постарайтесь избежать повреждения вообще. Конечно они со временем устраняются но заметно осложняют ведение боевых действий. Вода откачивается автоматически, а иные повреждения могут восстанавливаться через несколько минут - в зависимости от их сложности. Возврат в рубку - "SPACE".

7. Экран побед

Здесь показывается потопленные Вами корабли, их общий тоннаж, а также количество оставшихся торпед. По окончании игры здесь же выводится Ваш рейтинг в зависимости от тоннажа потопленных кораблей. Возврат в рубку или меню - "M".

Теперь, после того, как рассмотрены все экраны, можно более конкретно поговорить о трех миссиях.

I. Торпедно-пушечная практика

В этой миссии вам надо потопить четыре судна противника: один танкер, два торговых судна и одно пассажирское. Цели не двигаются. Поэтому особых сложностей на этом этапе Вы не встретите.

II. Нападение на караван

В этой миссии Вам будет поручено уничтожения вражеского конвоя, в который входит от одного до шести судов. В этом режиме появляются японские "охотники" и сторожевики.

Вам придется немало потрудиться, чтобы разгромить конвой.

В этих двух миссиях после потопления всех кораблей, Вы попадете в экран побед, а затем в меню. Вы также можете прервать игру и возвратиться в меню, как это описано в режиме боевой рубки.

III Боевое патрулирование

В этом режиме сложности возникают после первых же действии. После того, как Вы выберете тип военного патрулирования, Вам будет предложено идентифицировать вражеского "охотника". Это первые буквы A, B, C, D. Вам же надо нажать на клавиши ("1"-"4"), чтобы показать этот тип. Если Вы неправильно это сделаете, то Вам посоветуют потренироваться. Чтобы Вы не тратили время зря, ниже мы приводим характерные особенности "охотников".

A - мачта на корме, одна труба;

B - нет кормовой мачты, одна труба;

C - мачта, две трубы;

D - нет мачты и носовой пушки.

После того, как Вы правильно идентифицировали "охотника", Вы окажетесь в дополнительном меню.

Игра начинается с сообщения: "Топлива на 50 дней: нажмите клавишу <огонь> для начала патрулирования". Нажмите на "M", перед Вами возникнет карта побережья Центральной Америки с мигающими тремя точками и одной неподвижной, эти три точки - Ваши базы, а неподвижная точка - Ваша субмарина. Теперь, передвигая эту точку клавишами "Y", "H", "9" и "0", ищите вражеский караван. Как только Вы его найдете, все точки на экране остановят свое мигание. Нажмите на "M", и Вы окажитесь в экране карт, дальше действуйте аналогично первый двум миссиям - топите корабли. После того, как все японские корабли уничтожены, Вы вновь можете попасть в режим патрулирования - для этого нажмите клавиши "H"+"9"+"M". Вам сообщается количество оставшегося топлива. Когда его будет совсем мало, Вы сможете возвратиться на базу. Для этого подведя субмарину к мигающей точке (в режиме патрулирования) и, сделав накладку точек, нажмите "M". Вы окажетесь в режиме экрана побед, там отобразится количество Ваших побед, а также присвоенное Вам звание. После этого Вы окажетесь в главном меню. Как видите, в этом режиме, чтобы возвратиться в меню, недостаточно нажать "H"+"9"+"M", - сначала надо вернуться на базу.

Полезные советы

Когда на Вас идет "охотник", и Вы точно знаете, что он Вас заметил, то надо подпустить его на расстояние около 1000 ярдов и пускать торпеду или стрелять из пушки. Если Вы пустите торпеду раньше, то есть шанс, что он увернется.

Догоняя вражеские корабли, дайте полную скорость, можно включить ускоренную временную шкалу. Но следите за расстоянием, чтобы избежать столкновения.

Старайтесь, чтобы перископ был поднят, так как тогда Вы при срочном погружении все-таки сможете наблюдать картину боя и при погружении не потеряются координаты угла наблюдения.

Если Вы все же не смогли поразить "охотник", и он начинает кружить над Вами, то по карте наблюдайте за ним (подняв перископ и опустившись на глубину около 100 м). Когда Вы увидите, что он направляется к Вам, резко всплывайте на 40 метров (чтобы видеть в перископ) и атакуйте.

Если в режиме боевого патрулирования Вы обнаружили цель, но это один-два корабля, и Вы не хотите тратить на них время, то сразу же переключайтесь на патрулирование. Если Вы промедлите, то Вам придется топить и эти корабли.

Чтобы остановить погружение лодки при ее затоплении водой, используйте всплытие, это может замедлить погружение.

Некоторую сложность представляет прицеливание пушкой, а также поиск больших конвоев. Но это останется маленькой тайной. Опытные капитаны смогут дойти и до этого.

Дерзайте!

Любителям имитаторов подводных лодок может быть интересна программа HUNTER-KILLER. В данном случае это не проработка наших экспертов - это наш перевод фирменной инструкции.

HUNTER-KILLER

PROTEK 1983 г.

Вы командир британской подводной лодки типа "S" и выполняете ответственное задание вблизи берегов Германии и Дании во время Второй мировой войны. В зоне Вашего поиска где-то находится вражеская подводная лодка, ее надо отыскать и потопить. Выходить за пределы зоны нельзя, или Вы будете потоплены вражескими миноносцами. Побережье заминировано, так что если Вы не сядете на мель, то подорветесь на мине.

Подводные лодки типа "S"

Лодки этой серии были наиболее совершенными в британском флоте в годы второй мировой войны. Их длина - 217 футов, водоизмещение до 1000 тонн, численность экипажа - 44 человека, мощность дизельных двигателей - 1900 л.с., что давало скорость в надводном плавании до 16 узлов. Мощность электромоторов - 1300 л.с., скорость в подводном положении - 9 узлов. Максимальная глубина погружения - 300 футов.

Управление подводной лодкой

Подводная лодка управляется, как и обычный корабль. Перекладывание руля вправо вызывает поворот вправо, аналогично и влево.

Можно управлять и джойстиком. Максимальный угол поворота руля - 70 град. Чувствительность руля зависит от скорости, чем быстрее Вы плывете, тем быстрее выполняется поворот.

Лодка имеет два двигателя - дизельный и электрический, но поскольку дизель нуждается в воздухе, его нельзя использовать в подводном плавании. Здесь надо переключаться на электрические двигатели. При погружении с включенными дизелями зажигается контрольная лампочка на панели и, если не принять меры, двигатели необратимо выходят из строя. Работа электрических двигателей зависит от заряда батарей, которые подзаряжаются от дизелей во время надводного плавания. Следите за зарядом батарей, своевременно их подзаряжайте.

Дизельные двигатели - значительно более мощные, чем электрические. Переключение двигателей выполняется клавишей "E". Индикатор в боевой рубке показывает, какой двигатель включен.

Для погружения лодки необходимо выполнить две операции. Во-первых, надо забрать балласт "N", создав нулевую плавучесть. По мере заполнения цистерн водой изменяется состояние индикатора в левом верхнем углу боевой рубки. После этого лодка начнет медленно погружаться. Ускорить погружение можно переложив горизонтальные рули клавишей "6" или джойстиком "назад".

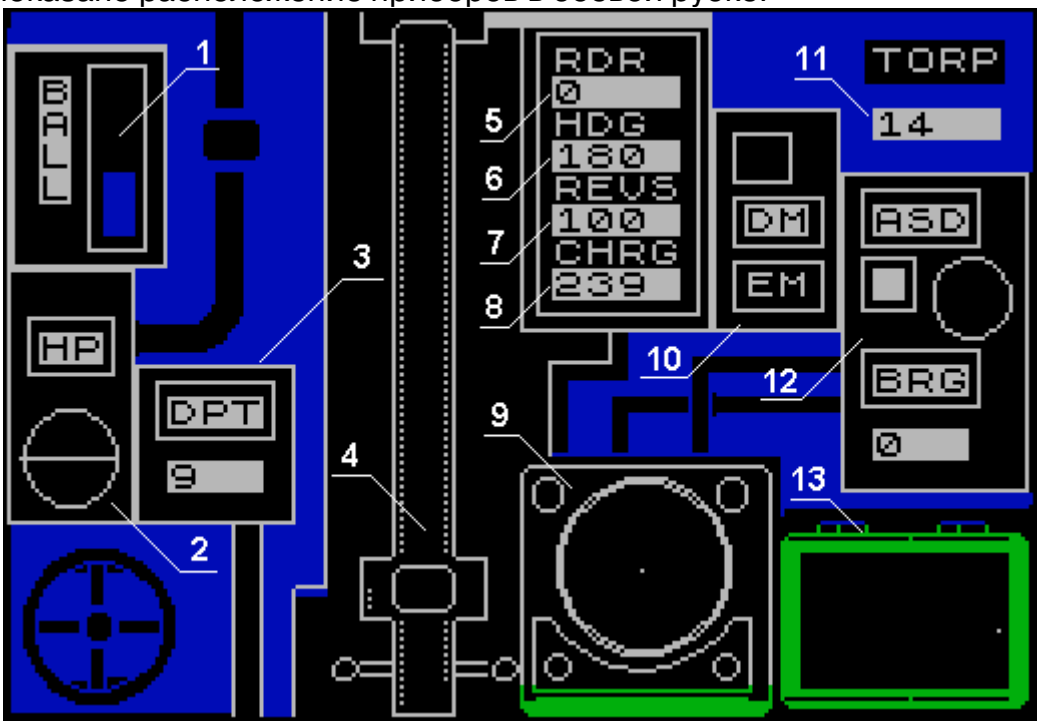
Внимание! Без выполнения обоих этих действии, Вам не удастся погрузиться быстро.

Останавливается погружение клавишей "7" или джойстиком "вперед". Необходимо также немного продуть балласт, пока индикатор уровня балласта не установится в центре.

Для всплытия надо выполнить противоположные действия. Продуть балласт - "B" и

переложить горизонтальные рули - "7"

На рис.1 показано расположение приборов в боевой рубке.



1. Уровень воды в балластных цистернах.
2. Угол отклонения горизонтальных рулей.
3. Глубиномер.
4. Перископ.
5. Угол поворота руля.
6. Направление носа лодки.
7. Обороты двигателя.
8. Заряд батарей.
9. Экран радара.
10. Индикатор двигателя.
11. Счетчик оставшихся торпед.
12. Дисплей гидролокатора.
13. Эхолот.

Работа с перископом

Перископ находится в боевой рубке. Его подъем или опускание выполняется клавишей "P", а клавиша "V" позволяет осмотреть поверхность моря через перископ. Если в зоне видимости имеется цель, ее можно увидеть. Чем глубже находится лодка, тем выше уровень воды на экране перископа. Глубже 38 футов перископом пользоваться нельзя вообще. Оптимальная глубина для работы с перископом - 34 фута.

Вращается перископ клавишей "O" (по часовой стрелке с шагом 36 град.) или клавишей "I" (против часовой стрелки с шагом 6 град.).

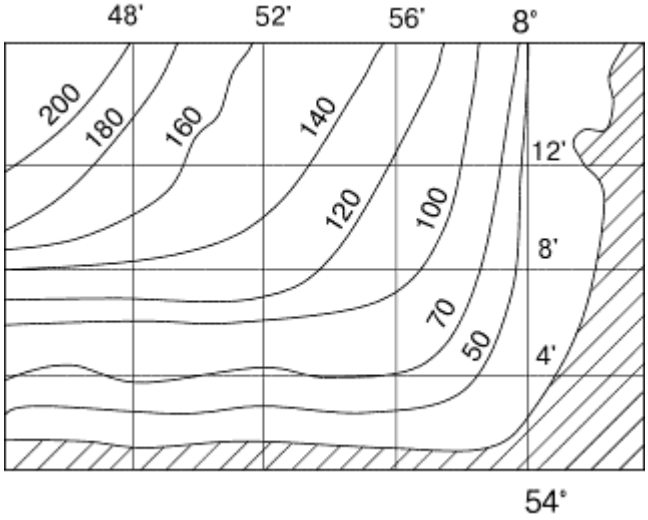
Когда Вы работаете с перископом, в нижней части изображены четыре отсчета. Первый показывает угол поворота перископа. Второй - угол, в направлении которого установлен нос корабля, оно не обязательно должно совпадать с направлением его движения. Третий - расстояние до цели, оно изображается только если противник находится в поле зрения. Четвертый - счетчик торпед.

Для возврата в боевую рубку нажмите клавишу "V.

Радар, гидролокатор, эхолот

Гидролокатор показывает направление на цель в градусах (0 гр. - север), если в радиусе 4 мили есть цель. Он включается автоматически при погружении на глубину более 10 футов. На поверхности автоматически включается радар, его радиус действия 22 мили.

Эхолот доказывает глубину под килем. На рис.2 показана карта глубин в диапазоне от 0 до 200 футов.



Штурманская рубка

Штурманская рубка находится справа от боевой рубки и вызывается клавишей "C". Боевая рубка сдвигается влево, при этом эхолот и гидролокатор остаются на экране и продолжают работать.

На экране изображается также карта, на ней нанесено очертание побережья, показаны границы минных полей (на рис.3), направления сторон горизонта, координатная сетка в градусах и минутах. В левой верхней части карты изображен кружок с маленькой линией, она указывает направление течения. Ваша позиция изображается миниатюрной подводной лодкой, под картой имеются отсчеты скорости течения, показания лага, т.е. Ваша скорость относительно воды, а также координаты последнего положения противника. Координаты даются только в минутах, показания в градусах очевидны. Эти координаты изображаются только в том случае, если противник находится в надводном положении и начинает игру в восточной части зоны. Его курс случаен, но с преимущественным движением на запад. Выход из штурманской рубки и переход в боевую рубку - клавишей "X".

1. Эхолот.
2. Дисплей гидролокатора.
3. Направление течения.
4. Позиция лодки.

5. Мины.
6. Побережье.
7. Скорость течения.
8. Скорость лодки относительно воды.
9. Координаты противника (в угловых минутах).

Стрельба торпедами

Торпедами можно стрелять только вперед, т.к. у Вас есть только носовые торпедные аппараты. Клавишей "T" запускается первая торпеда. Вторая и последующие в этом залпе запускаются клавишей "F". Счетчик торпед отсчитывает количество торпед в залпе. Залпы можно растягивать по времени, но не по углу.

Торпеды идут со скоростью 45 узлов и имеют дальность действия - 3 мили. Приблизительно следить за торпедой можно по пузырьковому следу. Если очевидно, что залп проходит мимо, его надо прервать клавишей "A".

Торпеды не могут быть выпущены, если руль не установлен точно прямо или если глубина погружения больше, чем возможная для работы с перископом. Если в момент стрельбы угол поворота перископа отличен от нуля, он автоматически устанавливается на нуль.

Прочие опасности

За Вами постоянно охотятся самолеты. Если один из них будет отмечен поблизости, Вы услышите звук. У Вас есть около двух минут, чтобы погрузиться на глубину ниже 30 футов. Если Вы опоздаете, Вас потопят.

После первой загрузки игры Вам будет предложена практика в стрельбе по неподвижной цели.

Кроме этой игры для одного компьютера существует вариант "HUNTER 2" для игры с двумя компьютерами, соединенными вместе в сеть. В этом случае подводной лодкой противника управляет второй человек с другой клавиатуры. Основные отличия:

- противник может быть как в подводном, так и в надводном положении;
- в этой игре нет воздушных атак;
- значительно снижена емкость батарей обоих противников, чтобы вызвать более частую необходимость всплытия для подзарядки.

Управляющие клавиши

5 - руль влево
6 - горизонтальный руль вниз (погружение)
7 - горизонтальный руль вверх (всплытие)
8 - руль вправо
A - прекратить залп торпедами
B - продуть балласт
C - пройти в штурманскую рубку
E - переключить двигатели
F - пуск второй и последующих торпед
I - вращение перископа против часовой стрелки на 6 град
J - уменьшить обороты двигателей
K - увеличить обороты двигателей
N - заполнение балластных цистерн
O - поворот перископа по часовой стрелке на 36 град.
P - подъем/опускание перископа
T - нацеливание торпеды и пуск первой торпеды
V - осмотр поверхности моря через перископ
X - выход из штурманской рубки.

Уважаемый читатель

Мы предлагаем вам лично подключиться к распространению обучающих программ для IBM-совместимых компьютеров, а также информационно- поисковых систем, разработанных и выпущенных нашей фирмой.

Основой Вашей деятельности является тот факт, что не все организации, предприятия, учебные заведения и пр. знают о наших программных разработках, а они достойны самого широкого распространения и внедрения.

С другой стороны, мы не знаем всех тех, кто в них нуждается и готовы ВАМ ЛИЧНО или ВАШЕЙ ОРГАНИЗАЦИИ заплатить за точную информацию о потенциальном покупателе, если она приведет к заключению сделки.

Размер оплаты:

- для частных лиц - 8 процентов от суммы сделки (оформляются трудовым соглашением и переводятся по почте или в Сбербанк).

- для организаций - 20% (оформляются договором на распространении научно-технической продукции).

Разумеется, мы рассчитываем на то, что Вы достойно представите нашу фирму потенциальному заказчику и со своей стороны готовы сделать все, чтобы Вы могли представлять нас с гордостью.

Начиная с данного номера "ZX-РЕВЮ" мы начинаем печатать аннотации на наши разработки и будем в каждом номере освещать по одной программе (пакету, комплексу). Сегодня мы начали с самой значительной работы, завершенной на сегодняшний день - комплексу из 22-ух программ "АНГРАМ" (см. следующую страницу).

Вы всегда сможете выбрать для себя, распространением какого продукта вам наиболее интересно заниматься. Следите за нашей информацией.

"ЗЕЛЕНЫЙ ПАКЕТ"

Мы можем значительно поднять Ваш уровень представительства, предложив Вам специально подготовленный комплект материалов, который мы назвали "зеленый пакет".

Вот что в него входит:

1. Инструкция коммерческого представителя.

2. Заверенное письмо-поручение на ведение переговоров от имени фирмы.

3. Два экземпляра проекта "Трудового Договора", предусматривающего отчисление вам 8% от объема продаж, произведенных по результатам Вашего исследования местного рынка. эти экземпляры должны быть вами заполнены и возвращены нам на утверждение. После утверждении Вам будет возвращен один экземпляр.

4. три полных комплекта коммерческих предложений "ИНФОРКОМа" на текущий период (обновляются один раз в два месяца). По мере их израсходования, вышлем дополнительно по запросу необходимое количество комплектов.

5. По мере наращивания Вашей деятельности и аналогичной деятельности ваших коллег из других регионов, мы будем обеспечивать Вас небольшим бюллетенем по обмену опытом. Наверняка в работе откроются какие-либо нюансы, которые сейчас нельзя предвидеть, но по мере накопления общего опыта можно будет использовать для блага всех.

6. Рабочие или демонстрационные образцы программного обеспечения.

Мы надеемся на Вашу успешную деятельность, но, к сожалению, не в состоянии обеспечить всех желающих данными образцами бесплатно. Они могут быть переданы Вам на ответственное хранение.

Получить желаемые образцы Вы можете по цене "зеленого пакета", которая складывается из:

10% текущей цены программного средства, если в качестве образца представлена полноценная рабочая программы или бесплатно, если представлена демо-версия;

+ стоимость дискет;

+ стоимость почтовых операций.

+ полная стоимость дополнительной документации, если она есть.

При любом - успешном или неуспешном характере Вашей деятельности, Вы можете вернуть нам "зеленый пакет" в любое удобное для Вас время вместе с полученными от нас образцами программ на тех же дискетах. Этот возврат Вы должны сделать посредством наложенного платежа в наш адрес, оценив почтовое отправление в ту же сумму, за которую Вы приобретали этот пакет, в данном случае Ваш риск ограничивается только почтовыми расходами.

Мы гарантируем выкуп данного почтового отправления на почте.

У активно работающих дистрибуторов мы выкупим все высланные образцы и далее будем их высылать бесплатно.

Итак, для представленного в данном номере "ZX-РЕВЮ" программного комплекса "АНГРАМ" в составе "зеленого пакета" поставляется рабочая программа "анграм-1" (урок первый) и цена составляет:

1300 руб. * 10%	=	120 руб.
Дискета	=	25 руб.
План-проспект	=	350 руб.
Почтовые расходы	=	3 руб.
Итого:		498 рублей

(Нас часто спрашивают, чем вызвана несоразмерно высокая цена плана-проспекта. Упреждая недоуменные вопросы укажем, что он является по сути готовым ТЗ и, если Вам понятно, что такое любопытство возможных конкурентов, то Вы поймете, почему мы не даем его бесплатно. Для Вас он послужит подтверждением уровня Вашего представительства при переговорах с потенциальными клиентами, мы уже начали маркетинг "АНГРАМа" и знаем, как он впечатляет клиентов.)

Образец будет Вам выслан на дискете фирмы "КОДАК" с защитным тефлоновым покрытием. Их внешний вид действует на клиентов завораживающе и будет содействовать успеху Вашего представительства.

Бланк для заказа "зеленого пакета" Вы получите вместе с данным номером "ZX-РЕВЮ". Там же указано как и куда производить оплату.

Если Вы не являетесь зарегистрированным у нас подписчиком "ZX-РЕВЮ", значит Вы не получите этот бланк и лишены возможности получить "зеленый пакет" и включиться в нормальную представительскую деятельность. Рекомендуем Вам для начала обратиться к нам с письмом для регистрации.

1. Вы знаете, что мы очень перегружены письмами и не можем реагировать на все обращения и вопросы, но уведомляем вас, что штаты для работы с дистрибуторами уже выделены, и ваши письма будут обработаны мгновенно. Выделен также телефонный номер для Вас.

2. Наша политика по выпуску обучающего программного обеспечения ориентирована на многие годы, ее основы уже заложены и работы давно ведутся. Если Вы готовы на ближайший десяток лет включиться в нашу структуру - мы даем вам "зеленый свет" и снабжаем Вас "зеленым пакетом".

3. Теперь все зависит только от Вас - от Вашей мобильности, контактности, оперативности, от умения общаться с людьми и от знания психологии.

Мы очень на Вас рассчитываем. Нам нужна прямая реклама, доходящая до конечного потребителя, стоимость рекламы в газетах достигла сотен тысяч рублей и пользоваться их услугами мы больше не можем - нам пришлось бы еще в два раза поднять цену своих программ за счет покупателей.

"АНГРАМ" - наша гордость.

"АНГРАМ" - это полный курс английского языка для IBM-совместимых компьютеров.

"АНГРАМ" - это 22 урока английского языка продолжительностью по 1б часов каждый.

"АНГРАМ" - это 22 дискеты 5,25 дюйма емкостью по 360 килобайт.

"АНГРАМ" - это полный курс грамматики английского языка. Это десять тысяч упражнений, десятки тысяч примеров.

"АНГРАМ" - это мощная лексическая поддержка - это десятки тысяч слов, осваиваемых во время занятий с компьютером. В комплексе нет ни одного слова, точный перевод которого Вы не смогли бы получить по ходу занятия.

"АНГРАМ" - это мощная поддержка фонетики. В комплексе нет ни одного слова, к КОТОРОМУ не было бы подробного описания правил произношения.

"АНГРАМ" - это полная гарантия того, что за 500 часов занятия каждый может изучить язык в достаточном объеме для чтения книжно-журнальной и специальной литературы на понимание.

"АНГРАМ" - это более 30 тысяч часов работы профессоров, доцентов, квалифицированных преподавателей и программистов.

"АНГРАМ" - не имеет аналогов ни у нас в стране, ни за рубежом - это комплекс, который дешевле приобрести за любую цену, чем пытаться превзойти.

"АНГРАМ" - ЭТО ВЫЗОВ ВСЕЙ СИСТЕМЕ ОБРАЗОВАНИЯ. МЫ ДОРОГО ЗАПЛАТИМ ТОЛЬКО ЗА ВОЗМОЖНОСТЬ УВИДЕТЬ ЧТО-ЛИБО ПОДОБНОЕ

Один только план-проспект комплекса, содержащий лишь перечень его разделов занимает 120 машинописных страниц.

И ЭТО ДАЛЕКО НЕ ВСЕ !!!

Невероятно - но факт! вся основная работа с комплексом выполняется пятью клавишами. Нет никаких проблем для самостоятельного освоения программ теми, кто не имеет профессиональных навыков работы с компьютером.

Программы комплекса могут быть использованы в учебных заведениях любого типа. Пользователь сам выбирает желаемую глубину проработки материала. Каждая программа гибко настраивается на работу от двух часов (беглый просмотр) до 16 часов. Все учебные материалы построены на реалиях жизни в США. Как припарковать машину, как вести себя в универсаме, как снять квартиру, материалы по истории США и многое другое.

Условия, в которых оказывается обучаемый, работая с нашим комплексом, напоминают те, в которых оказывается человек, впервые попавший за рубеж Обучение в бою. Методической основой комплекса "АНГРАМ" является контекстная подача материала.

Во время работы учащийся может несколькими нажатиями клавиш получить контекстный перевод любого слова из имеющихся на экране, узнать, как оно произносится.

В течение всего занятия в любую минуту учащийся может получить объективную оценку своих успехов.

Стоимость комплекса из двадцати двух программ на период январь-февраль 1992 года - 17 500 рублей.

ЭТОТ КОМПЛЕКС ДЕШЕВЛЕ ПРИОБРЕСТИ, ЧЕМ ПЫТАТЬСЯ ПРЕВЗОЙТИ
ЕГО ВНЕДРЕНИЕ ЗАНИМАЕТ МИНУТЫ. СЛУЖИТЬ ОН БУДЕТ ГОДЫ

Стоимость одной ОТДЕЛЬНОЙ ПРОГРАММЫ на период январь-февраль 1992 года - 1 200 рублей.

(Покупателям всего комплекса план-проспект прилагается бесплатно!.

ВНИМАНИЕ !

СТОИМОСТЬ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ может непрерывно изменяться в связи с общей экономической ситуацией. Указанные значения действительны только для заказов, оплаченных в период января - февраля 1992 года, во всех прочих случаях прежде чем оформлять заказ и производить оплату необходимо обратиться к нам для уточнения текущего уровня цен.

ГРАФИК ПОСТАВКИ.

Программы отправляются заказчикам, оплатившим заказ, по следующему графику:

январь -92	- 4 программы.
февраль -92	- 3 программы.
март -92	- 3 программы.
апрель -92	- 3 программы.
май -92	- 3 программы.
июнь -92	- 4 программы.
июль -92	- 2 программы.

Полное окончание поставок программ комплекса - не позднее июля 1992 года.

СПОСОБ ПОСТАВКИ.

Все программы доставляется ценной бандеролью в надежной упаковке, исключающей повреждение при почтово-транспортных операциях.

ГАРАНТИЙНЫЕ ОБЯЗАТЕЛЬСТВА

Гарантийный свидетельством при поставке программного продукта является картонный альбом, в который вложены дискеты, с указанной на нем датой продажи. При его отсутствии поставка выполняется с заверениям гарантийным талоном. Гарантиями обеспечивается:

- бесплатная замена поставочных дисков, неработоспособных в состоянии поставки (в течение месяца после поставки);

- замена с минимальной оплатой при выходе программ из строя по вине пользователя (механическое или электромагнитное повреждение, поражение вирусом на машине пользователя и т.п.) или по истечении месяца после поставки. Минимальная оплата не превышает стоимость дисков + 5% текущей стоимости программого обеспечения + стоимость почтово-транспортных расходов и согласовывается с потребителем.

- соблюдение сроков поставки (полная поставка - до 31.07.92).

ТЕХНИЧЕСКИЕ ТРЕБОВАНИЯ К АППАРАТНО-ПРОГРАММНОМУ ОКРУЖЕНИЮ

1. Полная аппаратно-программная совместимость с IBM PC XT/AT. Надежность функционирования на отечественных модификациях не гарантируется и не обсуждается.

2. Наличие "жесткого" диска ("Винчестера") емкостью не менее 2 МБ.

3. Наличие не менее одного дисковода гибких дисков 5,25 дюйма. На дискетах 3,5 дюйма программы не поставляются. Перенос с дискет 5.25" на 3,25" не обеспечивается (программы защищены от копирования).

4. Операционная система - MSDOS 3.20 и выше.

5. Русификация компьютера в стандарте гост (кодировка альтернативная).

6. Требования к монитору - не специфицируются.

7. Принтер - не требуется.

ПОРЯДОК ОФОРМЛЕНИЯ ЗАКАЗА.

а) направить в наш адрес письмо-заказ с указанием необходимого программного продукта; приложить копию платежного поручения;

Наш адрес: 107241, Москва, Б-241, а/я 37, "ИНФОРКОМ" б) произвести предварительную оплату платежным поручением на наш р/с:

Наш р/с N 500461778 во Фрунзенском коммерческом банке г.Москвы. МФО 201412.

Содержание