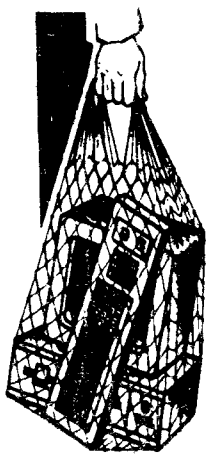
Персональный компьютер БК-0010, БК-0011м (1'96)	484-54-11 (Саша, Галя)

УВАЖАЕМЫЕ ЧИТАТЕЛИ!



Демонстрационный Центр БК при редакции журнала продолжает свою работу. Вашему вниманию предлагается новейшее аппаратное и программное обеспечение для БК-0010(.01) и БК-0011(М).

Специалисты Центра БК предоставят вам консультации по подключению периферийных устройств и использованию программного обеспечения, окажут помощь в ремонте БК и периферии.

Для заказа каталогов имеющегося программного и аппаратного обеспечения по почте достаточно прислать в редакцию письмо с заявкой и вложенным пустым конвертом с надписанным обратным адресом и наклеенными почтовыми марками

Здесь же вы можете приобрести отдельные выпуски журналов «Персональный компьютер БК-0010 — БК-0011М» и «Персональный компьютер УКНЦ», а также оформить полугодовую подписку на эти издания с получением экземпляров в редакции или по почте.

Заключаем с авторами договоры на рекламу и коммерческое распространение программных и аппаратных разработок. Приглашаем к сотрудничеству книготорговые организации, фирмы и заинтересованных лиц для реализации нашей печатной продукции.

Почтовый адрес: 125315, Москва, а/я 17

Телефон: (095) 292-53-87

E-Mail: mail@infoobr.msk.su

МАИ

Международная Академия Информатизации

ПЕРСОНАЛЬНЫЙ КОМПЬЮТЕР БК-0010, БК-0011м

Компьютика

1'96 (13)

Издается с 1993 г.

В НОМЕРЕ

Издательство **Компьютика**

Москва 1996

Авторы выпуска

Камнев С.
Канивец И. В.
Котов Ю. В.
Лашкевич С.
Неробеев С. М.
Прудковский А. Г.
Сергеев Б. Н.
Сорокин А. В.
Усенков Д. Ю.

РЕДАКТОРЫ: *ВАСИЛЬЕВ Б. М.*
УСЕНКОВ Д. Ю.

Лицензия на издательскую деятельность
ЛР №064197 от 10 августа 1995 г.

Свидетельство о регистрации средства массовой информации
№ 013550 от 26 апреля 1995 г.

Почтовый адрес: 125315, Москва, а/я 17
E-Mail: mail@infoobr.msk.su
Телефон: (095) 292-53-87

Завершаем публикацию материалов по практическому освоению ассемблера БК-0010(.01). Начало см. в №4—6 за 1995 г.

Б. Н. Сергеев,
Москва

Практикум на ассемблере

Предотвращение ошибок переполнения

При переполнении разрядной сетки процессор выставляет в «1» флаг V. Дальнейшее решение, как использовать полученную информацию, должен принять уже программист.

Для проверки состояния флага V в ассемблере имеется две команды ветвления:

- BVS MET — если переполнение ЕСТЬ, то переход на метку MET;
- BVC MET2 — если переполнения НЕТ, то переход на метку MET2.

(Имена меток выбирает программист, и в реальной программе они могут быть другими.)

Ниже предлагается два варианта использования флага V. Прогоните их в пошаговом режиме (лучше всего в Mirage) и понаблюдайте за изменением числа в R1, состоянием флага V и соответствующим изменением хода выполнения программы. (Число в первой строке MOV #77776,R1 находится вблизи критической точки, и поэтому уже через несколько шагов наступает переполнение.)

```
; использование флага V (переполнение)
        MOV #77776,R1
1:      ADD #1,R1       ; прибавить 1
        BVC 1
                ; если НЕТ переполнения
```

```
        MOV #T,R1   CLR R2   EMT 20
        HALT
T:      .ASCIZ/Ответ не верен!/
        .END
```

Флаг V устанавливается не только при переходе от положительного числа к отрицательному, но и при обратном изменении.

```
        MOV #100002,R0
1:      SUB #1,R0       ; отнять 1
        BVC 1
        MOV #T,R1   CLR R2   EMT 20
        HALT
T:      .ASCIZ/Ответ не верен!/
        .END
```

Команда BIT

Команда BIT #K,A проверяет биты, указанные значением K, в числе A и соответствующим образом выставляет флаги ССП. С ее помощью можно решать множество интересных задач. Ниже даны фрагменты, определяющие такие характеристики чисел, как четность и знак. Проверяемые числа вводятся в первой строке (A=...).

```
; определение четности числа
        A=123
;-------------
        MOV #A,R0       ; число в R0
        BIT #1,R0       ; проверить бит 0
        BEQ M           ; если бит 0 = 0
        MOV #T2,R1   CLR R2   EMT 20
        HALT
M:      MOV #T1,R1   CLR R2   EMT 20
        HALT
T1:     .ASCIZ<12>/Это число четное./
        HALT
T2:     .ASCIZ<12>/Это число нечетное./
        .END


; определение знака числа
        A=123422
;-------------
        MOV #A,R0       ; число в R0
        BIT #100000,R0  ; проверить бит 15
        BEQ M           ; если бит 15 = 0
        MOV #T2,R1   CLR R2   EMT 20
        HALT
```



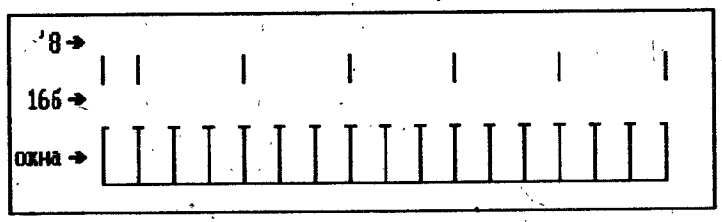
```
M:      MOV #T1,R1   CLR R2   EMT 20
        HALT
T1:     .ASCIZ<12>/Это число положительное./
        HALT
T2:     .ASCIZ<12>/Это число отрицательное./
        .END
```

Попробуйте вводить различные числа, положительные (от 0 до 77777) и отрицательные (от 100000 до 177777), в операторе присваивания A=... и понаблюдайте за работой этих программ.

В первом примере мы определяли такую характеристику числа, как четность, на основании следующего правила арифметики: признаком четности (нечетности) числа является четность (нечетность) его самого младшего разряда. Так как в двоичном числе самый младший разряд (бит) может принимать значения только «0» (для четного числа) или «1» (для нечетного), то для решения поставленной задачи достаточно узнать именно эту информацию. Но как заставить процессор «смотреть» только на один младший бит?

Для этого закроем все биты, кроме самого младшего, от «взгляда» процессора маской-заслонкой, и пусть он выставит флаги ССП для «увиденного» числа. При этом флаг Z (НУЛЬ или НЕ НУЛЬ) покажет, четное или нечетное (соответственно) взято число.

Просчитаем конкретный пример вручную. Пусть дано число 123456_8 (1010011100101110_2). Сделаем бумажную маску (идея принадлежит А. Пискунову) с шестнадцатью окошечками (по количеству бит в числе). Каждое из окошек имеет крышечку-клапан, которую можно приподнять и посмотреть, что под ней находится. На поле маски предусмотрим место для написания чисел, показывающих, какие окошечки следует открыть, в восьмеричном и 16-битовом двоичном отображении*:



На этом рисунке все окошечки закрыты.

Проверяемое число в 16-битовом виде напишем так, чтобы при наложении на него маски под каждым окошком находился один разряд (бит):

1 0 1 0 0 1 1 1 0 0 1 0 1 1 1 0

* Еще удобнее изготовить такую маску из тонкого гибкого белого пластика, чтобы можно было многократно стирать ластиком старые числа и писать карандашом новые. — *Прим. ред.*

Наложим на него маску и надпишем на ней восьмеричное число (в нашем случае #1), которое указывает, какие окошечки следует открыть для обзора. Переведем это число в 16-битовый вид и откроем те окошечки, над которыми оказались ЕДИНИЦЫ. В нашем примере это единственный бит 0. Открываем крайнее справа окошечко и видим там самый младший бит проверяемого числа — в нашем примере это 0.

По аналогичному алгоритму действует и рассмотренная ранее программа определения четности числа. В роли маски в ней выступает команда BIT #1,R0. Процессор «через открывшееся окошечко» видит содержимое нулевого бита и выставляет флаг Z=1, тем самым говоря нам: «это число четное» (переход по команде BEQ на вывод сообщения T1).

По тому же принципу работает и программа определения знака числа. Ее отличие состоит в том, что «открывается окошечко» над битом 15 (знаковым).

Команды BIC и BIS

Запустите приведенный ниже фрагмент командой RU и нажимайте буквенные клавиши, меняя регистры РУС/ЛАТ, ЗАГЛ/СТР.

```
N:   EMT 6          ; ожидание нажатия клавиши
     BIC #240,R0    ; СБРОСИТЬ биты по маске
     EMT 16         ; вывести символ на экран
     BR N           ; и идти в начало
     .END
```

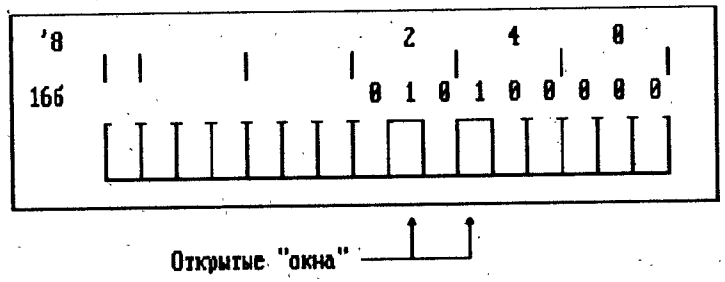
Какие бы регистры вы ни включали, на экран выводятся только латинские заглавные буквы. Такая возможность иногда бывает очень полезной, например когда программа запрашивает согласие пользователя на какое-либо действие и нужно ответить «D» или «N». Обычно приходится либо требовать от пользователя включать перед ответом режим ЛАТ ЗАГЛ, либо предусматривать в программе анализ всех возможных вариантов («N», «n», «Н», «н», «D», «d», «Д» и «д»). А показанный выше короткий программный фрагмент — «переводчик» дает гораздо более удобное решение проблемы.

Вот еще одна программа, отличающаяся от предыдущей только одной-единственной командой (BIS вместо BIC). Но взгляните на результат ее работы — он прямо противоположен предыдущему.

```
N:   EMT 6          ; ожидание нажатия клавиши
     BIS #240,R0    ; УСТАНОВИТЬ биты по маске
     EMT 16         ; вывести символ на экран
     BR N           ; и идти в начало
     .END
```

Такие эффекты достигаются применением команд BIC и BIS, которые, соответственно, стирают или устанавливают биты числа по маске (аналогичной применяемой в команде BIT).

Вновь просчитаем пример вручную. Впишем в нашу бумажную маску число 240, переведем его в 16-битовый вид и откроем окошечки под единицами:



Возьмем для начала символ «D». Посмотрим в таблицу кодов БК: символы «D», «d», «Д» и «д» расположены на одной вертикали этой таблицы. Составим из них свою мини-таблицу с указанием кодов:

```
D      104   01 000 100
d      144   01 100 100
д      304   11 000 100
Д      344   11 100 100
             ↑  ↑
Маска  240   10 100 000
```

Если наложить на каждый из этих кодов нашу маску с числом 240, то в открытых окошках будут видны только биты, отмеченные стрелками. Команда BIC вписывает в них нули (сбрасывает, или очищает, биты), а остальные разряды числа ей «не видны» и остаются без изменений. Поэтому все коды превращаются в код символа в регистре ЛАТ ЗАГЛ (в данном примере — в код символа «D»). То же самое происходит при нажатии любой другой буквенной клавиши, поэтому первая из рассмот-

ренных ранее программ всегда печатает латинские заглавные символы. Теперь, если нужно по ходу выполнения программы нажать клавишу «D», вы можете не беспокоиться о включении именно регистра ЛАТ ЗАГЛ.

Что же касается команды BIS #240,R0 , она вписывает в указанные биты «1» (устанавливает биты) и тем самым преобразует коды букв к регистру РУС ЗАГЛ (алгоритм работы программы аналогичен предыдущему).

Неперемещаемая программа

Каждая программа имеет свой адрес загрузки, который обычно является и адресом запуска. Так, в Turbo4 адрес загрузки программы всегда равен 1000 (о чем заботится сам Turbo4). Другие ассемблеры могут устанавливать для написанной в них программы иные адреса загрузки. Пример:

```
      MOV #MET,R1   CLR R2   EMT 20  ; выдать на экран
      HALT                           ; фразу "Текст."
MET:  .ASCIZ/Текст./
      .END
```

Даем команды «ВС», «ТАБ», CO, RU — на экране фраза «Текст.». Программа работает так, как и должна согласно ассемблерному листингу. Запишем на диск два файла с именами «A1T» и «A1A».

Обычно программа может работать правильно, только если она загружена в память с того же адреса, с которого она была записана на диск. Загрузим нашу программу в Mirage с адреса 1000:

```
F1000/3000/0 "ВВОД" LOAD A1A  "ВВОД" ASM1000  "ВВОД"

1000 MOV #1012,R1 ; #1012 получилось из #MET
1004 CLR R2
1006 EMT 20
1010 HALT
1012 BICB #151714,27324(R4) ; фраза "Текст.", коды
                            ; которой ошибочно опознаны
                            ; отладчиком как команды
                            ; ассемблера
1020 HALT

СТОП   G1000 "ВВОД"
```

Программа работает нормально, выдавая фразу «Текст.». Теперь загрузим ее с другого адреса, например с 1040, предварительно очистив память от прежней программы:

```
F1000/3000/0 "ВВОД" LOAD A1A/1040 "ВВОД" ASM1000 "ВВОД"
```

Мы видим тот же текст, но расположенный по другим адресам, начинающимся с 1040:


```
1040 MOV #1012,R1
1044 CLR R2
1046 EMT 20
1050 HALT
1052 BICB #151714,27324(R4)
1060 HALT
```

Попробуем его запустить: «СТОП», G1040. Программа не работает! Проанализируем причину этого. В первой строке команда MOV #1012,R1 загружает в R1 содержимое ячейки 1012 (ранее это был адрес первого символа фразы «Текст.»). А что сейчас находится в ячейке с адресом 1012? Нуль! Вот и загружается в R1 этот нуль, а не ожидаемый нами адрес начала фразы «Текст.». Отсюда и результат (вернее, его отсутствие).

Данный эксперимент показал, что наша программа должна загружаться только с адреса 1000. При загрузке с другого адреса она работать не может. Такая программа называется НЕПЕРЕМЕЩАЕМОЙ.

Перемещаемая программа

В некоторых случаях желательно иметь программу, работоспособную при загрузке с любого адреса (а не только с которого она была записана). Например, сам ассемблер-транслятор Turbo4 сделан именно так: его можно загружать с адреса 120000 (в NORD) или 140000 (в ANDOS), при этом объем свободной памяти для написания нашей программы заметно увеличивается от 1000 до 40000 байт. Такие программы называются ПЕРЕМЕЩАЕМЫМИ.

Чтобы программу из предыдущего примера сделать перемещаемой, изменим ее в Mirage, вписав вместо первой строки две другие. А чтобы не испортить остальную часть программы, сместим имеющийся текст ниже строки с адресом 1000. Для этого поставим курсор в эту строку и два-три раза нажмем на клавишу «ВС» — в начало текста окажется вставлено соответствующее количество команд NOP (лишние из них позже можно убрать клавишей «СБР»). Теперь впишем две новые строки и отредактируем весь листинг, чтобы он стал таким:

```
1000 MOV PC,R1     ; Новые строки вместо одной   ; 1
1002 ADD #12,R1    ; прежней MOV #1012,R1        ; 2
1006 CLR R2                                      ; 3
1010 EMT 20                                      ; 4
1012 HALT                                        ; 5
1014 BICB  #151713,27324(R4) ; фраза "Текст"     ; 6
1022 HALT
```

Даем команды запуска: «СТОП», G1000, «ВВОД». Программа сработала! Запишем ее: SAVE O1/1000/1022. Очистим память: F1000/2000/0 «ВВОД» и загрузим программу с другого адреса: LOAD O1/1040

«ВВОД». Снова запустим: G1040 «ВВОД» — программа работает нормально. Она стала перемещаемой!

Посмотрим, как мы этого добились. В строке 1 в регистр R1 записывается содержимое регистра PC (счетчика команд), который в этот момент равен 1002 (но не 1000, так как PC всегда показывает не на текущую команду, а на ту, которую процессор будет выполнять на следующем шаге).

В строке 2 к числу 1002 в R1 прибавляется число 12, которое рассчитывается программистом по правилу: адрес первого символа фразы («Текст.») минус адрес ячейки с командой ADD, т. е. 1014–1002=12. Это число называется смещением. Теперь в R1 записано 1002+12=1014. Это и есть адрес ячейки, с которой начинается фраза «Текст.».

Остальные строки остались без изменения, если не считать изменения их адресов из-за вставки двух строк в начале листинга. Таким образом, если в первой строке неперемещаемой программы в R1 «жестко» записывалось конкретное число — адрес начала фразы, то здесь он рассчитывается в процессе работы по значению смещения к началу фразы — «расстояние от 1002 до 1014».

Итак, заменив команду MOV #MET,R1 на две команды

```
MOV PC,R1
ADD #<адрес фразы минус адрес ADD>,R1
```

мы получили перемещаемую программу. Теперь при ее загрузке с любого адреса в R1 будет записываться правильное значение адреса начала фразы.

Таким образом, если вы хотите сделать имеющуюся неперемещаемую программу перемещаемой, в ней надо проделать такие изменения со всеми командами типа MOV #MET,A , где A — регистр или ячейка. Но сделать это даже в небольшой программе очень трудно. Ведь надо в каждом случае вручную вычислить смещение, раздвинуть текст и вписать новые команды! К тому же неизбежны ошибки, найти и устранить которые еще труднее.

Эту трудную работу можно поручить ассемблеру, заменив команду MOV #MET,A на следующие три строки:

```
MOV PC,A        Стандарт замены MOV #MET,A
ADD (PC)+,A          для получения
.WORD @MET+2         перемещаемости
```

Переписав нашу программу с учетом такой доработки, получим:

```
        MOV PC,R1      ;  установка в R1 относительного
        ADD (PC)+,R1   ;  адреса метки MET
        .WORD @MET+2   ;
        CLR R2
        EMT 20         ; выдать на экран
        HALT
MET:    .ASCIZ/Текст./
        .END
```

Запишем ее на диск по командам SA и ST, перейдем в Mirage и будем загружать с разных адресов и запускать. Программа работает нормально.

Проделайте теперь самостоятельно такой опыт. Загрузите программу с любого адреса и вставьте перед началом листинга ряд команд NOP (по клавише «ВС»), наблюдая перемещение текста программы по памяти и изменение вида команд. Передвинув программу на другое место, определите новый начальный адрес и запустите ее с этого адреса командой G. Программа должна работать — она стала ПЕРЕМЕЩАЕМОЙ*.

Работа с встроенным таймером

В вашей БКшке (в зависимости от конкретного экземпляра) может иметься «нелегальное» устройство отсчета времени — «системный» (встроенный) таймер, который после запуска может работать независимо от процессора, пока включено питание ЭВМ. «Нелегальное» — потому, что о нем в прилагаемой к БК документации ничего не сказано. Дотошные пользователи обнаружили его наличие, но не во всех экземплярах БК он имеется. Ниже дана программа, запустив которую вы сможете увидеть работу таймера (а заодно проверить его наличие). Если таймер работает правильно, то вдоль левого края экрана выводится непрерывная колонка чисел.

```
        ;  проверка работы таймера                          ; 1
        CLR   177710       ; подготовка к работе            ; 2
        MOV   #0,177706    ; установка константы (здесь 0)  ; 3
        MOV   #160,177712  ; пуск таймера                   ; 4
N:      MOV   #177777,R3   ; установка времени замедления   ; 5
        MOV   177710,R4    ; значение таймера -> в R4       ; 6
        MOV   #12,R0       ; выводить с начала строки       ; 7
        EMT   16                                            ; 8
        CALL  W8           ; вывод показания таймера        ; 9
3:      NOP                ; замедление вывода              ; 10
        SOB   R3,3                                          ; 11
        BR    N                                             ; 12
        HALT                                                ; 13
; -------- 163220 ПЗУ ------ вывод восьм. числа из R4
W8:     MOV #6,R5          CLR R0
2:      ROL R4   ROL R0    ADD #60,R0   EMT 16   DEC R5   BEQ 1
        CLR R0   ROL R4    ROL R0       ROL R4   ROL R0   BR 2
1:      MOV #40,R0         EMT 16       RET
        .END
```

Строки 4, 9 и 10 замедляют темп вывода показаний таймера, чтобы можно было проследить изменение чисел. Попробуйте убрать это замедление, вписав перед соответствующими строками знак «;», — что получится?

---

* Более подробно о перемещаемости программ и правилах их написания можно прочитать в журнале «Персональный компьютер БК-0010 — БК-0011М», №4 за 1994 г., с. 61. — *Прим. ред.*

К таймеру относятся три регистра с адресами 177706, 177710 и 177712*.

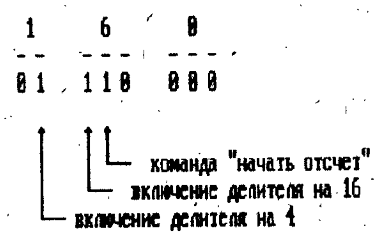
Регистр начальной установки 177706 служит для задания начального значения в регистре-генераторе, из которого он начнет вычитать по единице при каждом «тике» таймера. Тем самым можно задавать требуемый отрезок времени до появления в регистре-генераторе нуля. (В нашем примере мы записали в регистр 177706 также нуль, поэтому счет начинается с 000000, затем получается 177777, 177776 и т.д. Попробуйте в строке 3 менять записываемое в регистр-генератор число и посмотрите результат.)

Если в регистр 177712 записать определенное значение (здесь — константа 160 в строке 3), то он дает команду на запуск таймера, устанавливает определенный темп его «тикания».

Регистр-генератор 177710 переписывает в себя начальное число из регистра 177706 и начинает вычитать из него по единице, непрерывно двигаясь по замкнутому «числовому кольцу». Темп такого вычитания настолько высок, что даже если в нашей программе отключить все замедления, то в выводимых на экран числах разница между соседними значениями наверняка окажется больше единицы (т. е. наша программа работает медленнее и не может «уследить» за каждым изменением числа в регистре-генераторе).

В нашем примере мы записали в ячейку 177712 число 160. Представим его в виде двоичного байта, в котором роль каждого бита следующая:

Итак, число 160 дает команды: «начать отсчет» и «замедлить темп работы в 4 · 16 = 64 раза». В стандартной БКшке такое замедление темпа работы таймера приводит к тому, что полное «числовое кольцо» от 177777 до 0 (напомню, что отсчет идет в сторону уменьшения чисел) будет пройдено за 180 секунд. Это наибольшая длительность одного цикла.



При других числах длительность одного цикла меньше: 60 — 45 с, 120 — 11.25 с и 20 — 2.812 с. (Это в «стандартной» БКшке, работающей на частоте 3 мГц, а не в перенастроенной «самодельщиками» на другую частоту, например на 4 или 6 мГц.) Знание этих чисел может понадобиться при создании таймера, отсчитывающего время в том же темпе, что и привычные нам часы.

Начиная со строки 13 расположена подпрограмма W8, которая выводит содержащееся в R4 число на экран для нашего обозрения. Ее вызывает команда CAL W8 в строке 8, причем нужная информация копируется в Ꞩ4 из регистра-генератора в строке 5. Эта подпрограмма взята из ПЗУ БК, там она начинается с адреса 163220.

* См. также: Персональный компьютер БК-0010 — БК-0011М. 1994. №2. С. 4. — *Прим. ред.*

Если вы работаете с магнитофоном (причем именно на БК-0010.01 с встроенным БЕЙСИКом), а не с дисководом, то листинг можно несколько укоротить. Вместо команды CAL W8 впишите CAL 163220, тогда подпрограмму W8 вы можете убрать. Дело в том, что подпрограмма, размещенная в ПЗУ с адреса 163220, при работе с дисководом «перекрыта» дисковой операционной системой.

В строке 12 записана команда HALT. Ее обычное назначение — отделить расположенные ниже подпрограммы от основной программы. Но здесь перед HALT записана команда BR, которая не позволяет процессору перейти к HALT. Последняя стоит в листинге скорее для его «логической завершенности», при желании ее можно стереть и убедиться, что программа продолжает нормально работать.

Команда EMT 34

В программе, которую мы пишем, по ходу ее выполнения может потребоваться изменить характеристики вывода на экран, например перейти в цветной режим, включить инверсию символов или всего экрана и т. д. В короткой программе нетрудно запомнить состояние дисплея в любой момент времени. А вот в длинной... Да и исходное состояние (перед запуском программы на исполнение) в общем случае заранее неизвестно. Здесь вам поможет команда EMT 34 — «чтение слова состояния дисплея (ССД)», работу которой можно проследить в следующей программе. Эта команда записывает ССД в регистр R0, после чего мы пересылаем его в R4 и обращаемся к подпрограмме W8 для выдачи его на экран.

```
; -------------- EMT 34 --------------
          EMT 34
          MOV R0,R4
          CAL W8
          HALT
; ----------- 163220 ПЗУ ------ Вывод восьм. числа из R4
W8:       MOV #6,R5               CLR R0
2:        ROL R4    ROL R0        ADD #60,R0    EMT 16    DEC R5
BEQ 1
          CLR R0    ROL R4        ROL R0        ROL R4    ROL R0    BR
2
1:        MOV #40,R0              EMT 16        RET
          .END
```

Изменяйте вручную режимы работы дисплея и запускайте программу снова. На экран выводится восьмеричное число из R4, копия ССД из R0, представля-ющее собой 16-битовое слово. Каждый бит в нем несет информацию об установке одного из возможных режимов дисплея, как показано в таблице*.

* Более простой способ проверки текущих режимов дисплея — тестирование байтов-признаков из области системных ячеек (адреса 40_8—56_8), с которых функция EMT 34 и копирует биты ССД. Значение байта 377_8 (режим РУС — 200_8) означает включенный режим, 0 — выключенный или альтернативный. Подробнее о системных ячейках см. №3 за 1994 г., с. 56. — *Прим. ред.*

Номер бита ССД	Режим включен, если бит = 1; иначе отключен или включен альтернативный режим	Команда ручного включения (переключения) режима	Код включения через EMT 16
0	32 символа в строке	«АР2»+«;»	233
1	Инверсия экрана	«АР2»+«-»	235
2	Расширенная память	«АР2»+«СБР»	214
3	РУС (0 = ЛАТ)	«РУС»/«ЛАТ»	
4	Подчеркивание	«АР2»+«/»	237
5	Инверсия символа	«АР2»+«,»	234
6	ИНД СУ	«ИНД СУ»	202
7	БЛОК РЕД	«БЛОК РЕД»	204
8	ГРАФ	«АР2»+«СУ»+«E»	225
9	ЗАП	«АР2»+«СУ»+«F»	226
10	СТИР	«АР2»+«СУ»+«W»	227
11	32 символа в служебной строке	«АР2»+«.» — установка режимов отображения информации в служебной строке по текущим режимам экрана	236
12	Подчеркивание в служебной строке		
13	Инверсия в служебной строке		
14	Гашение курсора	«АР2»+«:»	232
15	Не задействован		

Примечание к таблице.

Включение режимов ГРАФ, ЗАП и СТИР возможно также при помощи комбинации клавиш «АР2»+«НР»+«5», «АР2»+«НР»+«6» и «АР2»+«НР»+«7», соответственно. (На клавиатуре БК-0010.01 клавиша «НР» обозначена широкой направленной вниз стрелкой — «⇓», расположенной в верхнем левом углу под клавишей «ПОВТ».)

Установить режим дисплея можно и программно. Вставьте перед EMT 34 две команды: MOV #K,R0 и EMT 16, где число K берется из правой графы таблицы.

Приведенную выше программу можно укоротить. Поставьте знак «;» перед командами CAL W8 и HALT. Запустите программу и убедитесь в ее работоспособности. А раз так, то смело стирайте «закрытые» строки. Память машины мы при этом сэкономили, но проиграли в наглядности (структурированности) листинга. Так что почти гамлетовский вопрос «стирать или не стирать?» решать вам самим.

Команда EMT 22

Во многих программах служебная строка используется для вывода вспомогательной информации. Вы тоже можете это сделать, если воспользуетесь следующим листингом:

```
        ; Текст в служебной строке
        EMT 6                   ; ждать нажатия любой клавиши
        CLR R0   EMT 22  EMT 6  ; очистить сл. строку и ждать
        MOV #30,R1              ; начать вывод с позиции 30
        MOV #T,R2               ; начало текста - в R2
N:      MOVB (R2)+,R0           ; вывод очередной буквы
        BEQ 1                   ; если символ 0 - закончить вывод
        EMT 22                  ; иначе - вывод
        INC R1                  ; следующая позиция в сс
        BR N                    ; идти за новым символом
1:      EMT 6                   ; ждать
        HALT
T:      .ASCIZ/Текст в служебной строке./
        .END
```

После запуска программа ждет нажатия любой клавиши, по второму нажатию выводит фразу в служебную строку, а по третьему — завершает работу. При включении данного фрагмента в собственную программу эти ожидания, конечно же, не нужны (здесь они добавлены только для обеспечения правильной демонстрации при совместной работе с ДОС), и все три команды EMT 6 можно удалить.

Вывод текста возможен с позиции 0 (крайней слева), отступ же с левого края делается путем печати требуемого количества пробелов в начале строки текста. А вот при выводе с правого края будьте внимательны: туда выдается служебная информация (сообщения «РУС»/«ЛАТ», «ИСУ», «БЛР» и т. д.), так что часть вашей строки может быть затерта.

Ячейка 40

Эта ячейка хранит информацию об установленной в данный момент длине строк экрана: 32 (цветной режим) или 64 (черно-белый) символа. Если нужно установить в программе определенную длину строки (или цветной/черно-белый режим), то вначале потребуется проверить текущее содержимое ячейки 40. В режиме 64 символа в строке оно равно 0, а при 32 символах — 377.

Алгоритм программной установки требуемого режима следующий. Сначала надо «показать» процессору содержимое ячейки 40, чтобы он соответствующим образом выставил флаги условий. Если в ячейке записан 0, то процессор установит в «1» флаг Z, иначе сбросит его в «0». В нашем случае НУЛЬ в бите Z означает, что установлен режим 64, а НЕ НУЛЬ — режим 32. И теперь, в зависимости от имеющегося и требуемого режима, используем один из показанных ниже фрагментов:

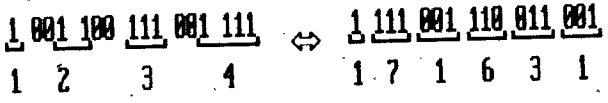
```
; ------ установка режима 32 символа в строке
     TSTB @#40              ; в ячейке 0 или НЕ 0 ?
     BNE 1                  ; если НЕ 0 (если 32) - оставить
     MOV #233,R0   EMT 16   ; если 0 - изменить на 32
1:   ....

; ------ установка режима 64 символа в строке
     TSTB @#40              ; в ячейке 0 или НЕ 0 ?
     BEQ 1                  ; если 0 (если 64) - оставить
     MOV #233,R0   EMT 16   ; если НЕ 0 - изменить на 64
1:   ....
```

Строки на экране: символьные и точечные

Напомню, что биты на экране располагаются слева направо от младшего к старшему так же, как по экрану движется электронный луч. Это направление движения луча совпадает и с общепринятым способом написания текста на бумаге. А вот написание чисел европейцы переняли у арабов, которые до сих пор пишут справа налево любой текст. Поэтому все числа и отражают типично «арабское» мышление — «справа налево». А мы, как и все европейцы, читаем их слева направо. Так, число 21 мы читаем как «двадцать один», а «по-арабски» это было бы «один и двадцать». Кстати, именно так прочтет данное число и немец.

Отсюда некоторое неудобство преобразования двоичных чисел в экранное изображение: они взаимно зеркальны. Так, для числа 348 = 00 011 100 на экране получается ..###... (где точками показаны «черные», не высвеченные пикселы, а символами «#» — светящиеся).

Для подтверждения сказанного проделаем в отладчике следующие действия: 60000A «ВВОД» 171631 «ВВОД». На экране видна полоска, светящиеся точки которой расположены слева направо четырьмя группами из одной, двух, трех и четырех штук (зеркально по отношению к двоичному числу):



Для определения начальных адресов ТВ-строк на экране запустим Отладчик-16 и будем читать содержимое ячеек ОЗУ экрана по левому краю сверху вниз, начиная с адреса 40000 с шагом 1008. Справа от них будем расписывать значения байтов — младшего и старшего слева направо (как они выводятся в Отладчике-16), чтобы было удобнее делать битовую расшифровку символов. Для большей наглядности нулевые биты обозначим пробелами, а по периметру знакоместа — точками. Единичные же биты обозначим символом «#» для лучшего восприятия получаемого изображения.


```
        байты                                  номера (названия)
        мл.ст                                  символьных строк

 40000   0.0  ................ -+ СС:
   100   0.0  .          ..   .  | служебная строка
   200  34.0  . ###     ..    .  |
   300  42.0  .#   #  ..      .  | первые две буквы
   400  42.76 .#   #  ..##### .  | слова "Отладчик"
   500  42.10 .#   #  ..  #   .  |
   600  42.10 .#   #  ..  #   .  |
   700  42.10 .#   #  ..  #   .  |
 41000  34.10 . ###   ..  #   .  |
   100   0.0  ................ -+
   200   0.0  ................ -+ РП:
   300   0.0  .          ..   .  | разделительная полоса
   400 377.377 ################  | под служебной строкой
   500 201.201 #      ##      #  | высотой 6 ТВ-строк
   600   0.0  .          ..   .  |
   700   0.0  ................ -+
 42000   0.0  ................ -+ 1:
   100  74.0  . #### ..       .  |
   200  44.0  . #  # ..       .  | первая строка
   300  44.42 . #  # ..#   # .  | рабочего поля экрана:
   400  44.62 . #  # ..#  ## .  | первые две буквы слова
   500  44.52 . #  # ..# # # .  | "Директивы"
   600  44.46 . #  # ..##  # .  |
   700 176.42 .######..#   # .  |
 43000 102.0  .#    #..       .  |
   100   0.0  ................ -+
   200   0.0  ................ -+ 2:
   300   0.0  .          ..   .  | вторая строка
   400   0.0  .          ..   .  | рабочего поля экрана
   500   0.0  .          ..   .  | (пустая)
   600   0.0  .          ..   .  |
   700   0.0  .          ..   .  |
 44000   0.0  .          ..   .  |
   100   0.0  .          ..   .  |
   200   0.0  .          ..   .  |
   300   0.0  ................ -+
   400   0.0  ................ -+ 3:
   500  34.20 . ###    ..    #  . | третья строка
   600  42.30 .#   #   ..   ##  . | рабочего поля экрана:
   700  62.24 .#  ##   .. # #  . | первые две цифры
 45000  52.22 .# # #  ..#  #   . | адресов памяти БК
   100  46.76 .##  #  ..##### . |
   200  42.20 .#   #  ..    #  . |
   300  34.20 . ###   ..    #  . |
   400   0.0  .          ..   .  |
   500   0.0  ................ -+
```

(Окончание на с. 79)

СПРАВОЧНОЕ БЮРО

Ю. В. Котов,
Москва

Некоторые полезные сведения по БЕЙСИКу БК-0011М

Известно, что ПЭВМ БК-0011М может эксплуатироваться с дисководом или без него, при этом пользователь имеет дело как бы с двумя разными машинами, так как эксплуатационные возможности и доступное программное обеспечение сильно различаются. Без дисковода с машиной можно работать преимущественно на прошитом в ПЗУ БЕЙСИКе и частично в машинных кодах (конечно, можно также использовать различные готовые программы). Да и при наличии диска БЕЙСИК применяется многими пользователями. (Старые версии БЕЙСИКа ДВК, загружаемые с диска, предоставляют значительно меньшие возможности.) Ниже речь пойдет в основном о БК-0011М без дисковода. Значительно увеличенный объем памяти по сравнению с десятой моделью, казалось бы, дает надежду на существенный рост возможностей машины и при работе на БЕЙСИКе, в частности по длине программ. Однако на практике здесь возникает ряд проблем.

БЕЙСИК БК-0011М, в целом аналогичный своему предшественнику в БК-0010.01, является его модифицированной версией. Сохранена основная схема обработки (трансляция в «объектный код» и последующая его интерпретация), обеспечена совместимость «снизу вверх», но в связи с расширением памяти и ее постраничной организацией в описании к машине указаны новые модификации ряда операторов. В частности, можно использовать две страницы экранной памяти и выборочно выводить их на экран. Переключение страниц, однако, имеет ограничения, и, к сожалению, не предусмотрено использование различных страниц для размещения текста самой программы или объектного кода. Дополнительные страницы можно применять только для хранения информации, машинных подпрограмм или «виртуальных» массивов. Тем не менее объем «многоцелевой» памяти может быть по сравнению с БК-0010.01 увеличен. Вместо предельного адреса 40000 (адреса здесь и далее восьмеричные) с помощью оператора CLEAR можно установить 61000, сохраняя при этом отображение целого экрана. Однако фактически, по крайней мере на некоторых экземплярах БК-0011М, оказалось, что объявленные в описании расширенные возможности операторов PEEK, POKE, DEF USR, BSAVE, BLOAD, DIM, SCREEN, OUT и INP, касающиеся указания номеров страниц, не реализованы. Не вводятся «виртуальные массивы», не действует оператор VARPTR. Зато работает оператор DRAW, о котором в документации сказано, что он не реализован. Очевидно, в этих экземплярах ПЭВМ установлена лишь слегка подправленная версия предыдущего, «десяточного» БЕЙСИКа. Оказались неисправленными и некоторые старые ошибки в обрабатывающей системе. Так, по-прежнему не работают правильно массивы размерностью больше двух, «не проходят» такие имена, как IN или LL (начальные пары букв зарезервированных слов). Не работают в программном режиме команды BLOAD и BSAVE, хотя это иногда бывает необходимо. Остается и такой крупный недостаток: в одной строке нельзя записывать по нескольку операторов. Это, кстати, не только ухудшает читабельность программ, но и требует лишнего расхода памяти, так как на фиксацию каждого номера строки уходит 6 байт.

Преодолеть некоторые недостатки, в частности в отношении дополнительных страниц памяти, можно с помощью машинных подпрограмм и функций USR, хотя это, конечно, усложняет программирование.

Ниже приведены более конкретные сведения по обрабатывающей системе, в том числе для возможности использования блоков этой системы в машинных подпрограммах для арифметических операций и вычисления функций.

Размещение обрабатывающей системы в памяти ПЭВМ

Хотя, согласно описанию, система БЕЙСИКа хранится в ПЗУ, при загрузке первая ее часть перекачивается в оперативную память (в третью страницу, подключенную начиная с адреса 40000), а продолжение остается в ПЗУ, подключенных с адреса 100000. В целом распределение памяти таково:

Адреса	Назначение
0—777	Рабочие системные ячейки и «системный стек»
1000—1777	Стек БЕЙСИКа
2000—3050	Рабочие ячейки и массивы БЕЙСИКа
3050—(txend−1)	Исходный текст программы на БЕЙСИКе
txend—(str−1)	Объектный код программы
str—endcod	Символьные переменные и массивы
....	Остающееся свободное место
lent—(tabt−1)	Область описания переменных
tabt—(fcb−1)	Таблица номеров и адресов строк
fcb—(himem−1)	Буфер для ввода-вывода информации

Адреса	Назначение
....	Свободное место, если himem<61000
61000	Начало обрабатывающей системы
61176	Точка «BREAK»
61750	Блок чтения строки
63452	Блок вывода строки
63654	Блок вывода на печатающее устройство
63722	Блок вывода и др.
64764	Обработка функции INKEY$
65014	Блоки выхода на магнитофон, оператор CALL и др.
65524	Блок занесения текста «ключа»
65564	Блок обслуживания ИРПС
67516	Блоки и подпрограммы графических операторов
71030	Обработка команды DRAW
72330	Начало монитора БЕЙСИКа
72460	Таблица имен операторов
73134	Таблица адресов имен операторов (указанных в предыдущей таблице по адресу 72460)
73322	Таблица адресов подпрограмм для операторов
73510, 73516	Точка «STAR» — ввод команд, директив
73556	Анализ строки директивы
74334	Блоки обработки операторов и команд NEW, LLIST, LIST (см. таблицу адресов подпрограмм с учетом таблицы имен операторов)
74634	Подпрограмма загрузки строки в буфер (2422) по ее номеру
74776	Начало обработки команды RUN
75746	Блоки отработки операторов LOAD, FIND и др.
103034	Таблица дополнительных ключевых слов (встречающихся в середине различных операторов)
103166	Блоки отработки операторов IF, FOR, NEXT и др.
106714	Обработка прерываний TRAP
107212	Таблица адресов подпрограмм для прерываний
107274	Ключевые слова логических операций и имена функций, затем арифметические знаки и спецсимволы
107620	Блоки обслуживания прерываний TRAP

Адреса	Назначение
115122	Блоки обработки трассировки
115216	Подпрограммы обработки команд объектного кода END, STOP, NEXT
115672	Подпрограмма обработки оператора IF в объектном коде
116022	Подпрограммы обработки команд GOSUB, GOTO, RETURN и др. (см. ниже)
124146	Преобразование числа из символьного формата в цифровой и обратно
127016	Переход к обработке «псевдокода» («польская запись»)
127022	Функции IABS, INEG, ABS и др.
127370	Действия с нормализованными числами
131614	Математические функции
140000	Мониторная система, драйверы внешних устройств (начиная с таблицы адресов подпрограмм)
160000	Незанятая область или драйверы дисков

Таблица рабочих ячеек обрабатывающей системы (аналогично БК-0010.01)

Адрес	Условное имя	Назначение
До 1777		Стек
2000		Рабочая ячейка
2002	txend	Указатель конца текста исходной программы
2004	str	Конец объектного кода, начало области символьных переменных
2006	strsiz	Длина области символьных переменных
2010	stfre	Начало свободной части области символьных переменных
2012	stlen	Длина свободной части области символьных переменных
2014	endkod	Конец областей символьных переменных и массивов
2016	cicl	Вершина стека для циклов

Адрес	Условное имя	Назначение
2020	limit	Вершина стека для возвратов из подпрограмм
2022	lent	Начало области переменных
2024	tablop	Вершина таблицы номеров и адресов строк
2026	fcb	Начало области для чтения/записи на магнитофон
2030	himem	Указатель предельно доступной памяти
2032—2044		Рабочие ячейки
2046	halt	Адрес в объектном коде для оператора CONT
2050—2054		«Флаги» и приращение адресов для трассировки и автонумерации
2056		Текущее положение курсора
2060—2077		Рабочие ячейки
2100—2123		Таблица начальных адресов для машинных подпрограмм
2124—2132		Рабочие ячейки
2134	savj	Хранение адреса возврата из подпрограммы
2140—2411		Тексты программируемых ключей
2414—2416		Адрес драйвера записи и указатель устройства ввода-вывода
2420		Буферная ячейка для оператора INKEY$
2422—3021	buf	Буфер ввода с клавиатуры
3022—3051	bufout	Буфер вывода
3052		Начало текста исходной программы

Примечание. Условные имена даны для удобства чтения и не соответствуют именам в программах.

Схема обработки программ на БЕЙСИКе

Не считая других адресов расположения в памяти, обрабатывающая система БЕЙСИКа в основном сохранена от старого варианта (для БК-0010.01). Небольшие различия обусловлены изменением мониторно-драйверной системы, которая стала более мощной. Обращения к драйверам выполняются не через прерывания EMT, а путем вызова подпрограмм. Наиболее крупные изменения в связи с этим претерпели графические операторы. Некоторые различия в начале программы и других местах связаны с общей архитектурой машины. Назначение же рабочих ячеек практически осталось прежним (см. предыдущую таблицу).

Таким образом, основной алгоритм функционирования обрабатывающей системы заключается в следующем. После запуска система выходит в режим ожидания ввода символа с клавиатуры (обращение к драйверу из подпрограммы чтения строки). Поступающие символы обрабатываются, служебные (от клавиш редактирования и т. п.) вызывают соответствующие программные действия, печатаемые символы (в том числе буквы и цифры) накапливаются в буфере, начинающемся с адреса 2422. Клавиша «ВВОД» производит выход из подпрограммы. Далее строка анализируется. Непосредственные команды отрабатываются, строки программы (начинающиеся с номеров) добавляются к исходному тексту программы, уже имеющемуся в памяти (при этом сами операторы заменяются числовыми кодами). В таблицу адресов строк заносится номер данной строки и начальный адрес ее размещения в памяти. Эта таблица (tabt) для каждой строки содержит три двухбайтных слова: адрес соответствующей точки в объектном коде (до трансляции — нуль), адрес начала строки в исходном тексте и номер строки. Таблица упорядочена по номерам строк и «растет» от старших адресов (himem−2) к младшим.

При вводе непосредственной команды строка обрабатывается одной из подпрограмм обработки команд и операторов. В подпрограмме производится синтаксический анализ текста команды, затем некоторые из них исполняются непосредственно, а другие — с образованием фрагмента объектного кода. После исполнения программа возвращается к ожиданию ввода символа с клавиатуры (выдав надпись «OK»).

Отдельные команды выполняются особым образом. В первую очередь имеется в виду команда RUN. Она производит инициализацию переменных и ряда рабочих ячеек, а затем по исходному тексту в соответствии с таблицей адресов строк строит объектный код, записывая его в память после исходного текста. Начальный адрес фрагмента объектного кода для каждой обработанной строки записывается в таблицу адресов. Объектный код состоит в основном из адресов обрабатывающих подпрограмм («псевдокоманд»), адресов размещения переменных в памяти, непосредственных операндов (констант), адресов нужных участков самого объектного кода (например, для операторов переходов) и некоторых дополнительных сведений. На заданные в исходном тексте символьные константы из объектного кода делаются ссылки. Таким образом, объектный код неперемещаем, в общем случае не может работать без исходного текста и рассчитан на фиксированное расположение переменнных.

Построение объектного кода можно назвать компиляцией, далее программа автоматически переходит к отработке (интерпретации) объектного кода (другие его возможные названия — «псевдокоманды», «виртуальная машина»). Из кода выбираются «команды» адреса подпрограмм, и им передается управление. Подпрограммы читают из объектного кода константы, адреса переменных и др., выполняя соответствующие операции. При этом активно используется стек, начинающийся с адреса 1776 и идущий «вверх». Туда заносятся константы, значения переменных и другие данные. В принципе режим обработки исходного листинга БЕЙСИК-программы аналогичен таковому на БК-0010.01.

Использование машинных подпрограмм

Выше упоминалось, что при работе с БК-0011М бывает целесообразным использование машинных подпрограмм, вызываемых через функции USR. В частности, с их помощью можно производить обмен данными с другими страницами памяти, напрямую (через системные драйверы) обращаться к диску или магнитофону и др. Кроме того, с известными ограничениями объектный код все же можно использовать и без исходного текста или брать его за основу для построения машинных подпрограмм. Поскольку процессор непосредственно не может даже умножать и делить целые числа, а тем более нормализованные, для вычислений в машинных подпрограммах целесообразно использовать блоки обрабатывающей системы БЕЙСИКа. Частью системы можно также воспользоваться, если, например, с клавиатуры нужно вводить и редактировать строки текстов и т. п.

То, что система БЕЙСИКа частично размещена в оперативной памяти, с одной стороны прискорбно, но, с другой, имеет свои преимущества, ибо после загрузки ее можно изменить или дополнить. Например, изменяя таблицу имен операторов, можно перейти на русскую мнемонику или же сократить длину некоторых ключевых слов.

Так или иначе, могут быть полезны сведения об использовании блоков и подпрограмм системы для арифметики и вычисления функций. Напоминаем, что такие подпрограммы вызываются из объектного кода путем использования регистра R4. Текст машинной подпрограммы при этом состоит из чередующихся участков «настоящих» машинных команд и «псевдокода», содержащего адреса блоков системы БЕЙСИКа (подпрограмм), адреса переменных и константы. Переход к «псевдокоду» может быть выполнен с помощью следующей подпрограммы (в восьмеричном виде — коды 5726, 134):

```
adsl:  TST (SP)+
       JMP @(R4)+
```

Здесь важно не забывать, что подпрограммы объектного кода заканчиваются кодами 134 (а не, например, 207). Для обратного перехода к машинным командам в «псевдокод» включается адрес первой из них:

```
          ....
       машинные коды
          ....
          JSR R4,adsl   ; (4437, adsl)
          adpp1         ; псевдокод
          adpp2
          adpp3
          ...
          adrprod       ; адрес след.ячейки
adrprod:  ....          ; продолжение машинных команд
```


В системе БЕЙСИКа можно пользоваться вспомогательной подпрограммой перехода, которая находится по адресу 127016.

«Псевдокод» имитирует нуль-одноадресную машину со стеком. Различные псевдокоманды-подпрограммы служат для загрузки в стек операндов разного типа, как констант, так и переменных (адресов их размещения). Другие псевдокоманды пересылают результаты из стека по заданным адресам. Как обычно, при выполнении «двухместной» операции два операнда из стека снимаются, а на месте первого из них размещается результат. Выбор элемента массива выполняется как своеобразная функция, предварительно в стек загружается значение индекса и описатель массива — его адрес, размерность и число индексов. Присвоение результата элементу массива тоже реализуется в виде функции. Команда перехода и обращение к подпрограмме («псевдоподпрограмме») выглядит обычным образом: за самой псевдокомандой идет адрес точки перехода. Однако для обслуживания условных переходов и циклов существует обширный набор дополнительных подпрограмм, обрабатывающих разные условия переходов, разные типы переменных и др.

В целом отработка объектного кода выполняется так же, как на БК-0010.01, но адреса подпрограмм («псевдокоманды») другие. Ниже приведена таблица «псевдокоманд» для пересылки данных в стек и из стека, арифметических операций, основных математических и других функций. Для сравнения указаны «псевдокоманды» для БК-0010.01.

Таблица адресов блоков обрабатывающей системы

Операция	Усл. имя	Адрес в БК-0011М	Адрес в БК-0010.01	Параметры
Пересылка в стек целой переменной	IVAR	116526	156232	Адрес переменной
Пересылка в стек переменной одинарной точности	SVAR	116510	156214	
Пересылка в стек переменной двойной точности	DVAR	116464	156170	
Пересылка в стек целой константы	ICONS	116454	156160	Константа, 2 байта
Пересылка в стек константы одинарной точности	SCONS	116452	156156	Константа, 4 байта
Пересылка в стек константы двойной точности	DCONS	116446	156152	Константа, 8 байтов

Операция	Усл. имя	Адрес в БК-0011М	Адрес в БК-0010.01	Параметры
Пересылка в стек описателя массива	INDM	120352	160056	Адрес дескриптора и размерность массива, 4 байта
Организация массива, с указанием длины	ARR	120522	160226	Адрес дескриптора и размерность массива, а также данные в стеке
Выборка элемента массива (целого типа)	IXV	116522	156226	Нет (данные в стеке)
Выборка элемента массива (одинарной точности)	SXV	116504	156210	
Выборка элемента массива (двойной точности)	DXV	116460	156164	
Пересылка из стека в память целого числа	IMOV	116644	156350	Адрес переменной
Пересылка из стека в память числа одинарной точности	SMOV	116624	156330	
Пересылка из стека в память числа двойной точности	DMOV	116630	156334	
Пересылка в массив целого числа	IXM	116620	156324	Нет (данные загружены в стек, см. INDM)
Пересылка в массив числа одинарной точности	SXM		156276	
Пересылка в массив числа двойной точности	DXM		156304	
Преобразование целого числа в нормализованное, одинарной точности	IS	127122	166656	Нет (операнд в стеке)
Преобразование целого числа в число двойной точности	ID	127112	166646	
Преобразование числа одинарной точности в число двойной точности	SD	127206	166742	
Преобразование числа одинарной точности в целое число	SI	127244	166770	
Преобразование числа двойной точности в целое	DI	127234	166760	
Преобразование числа двойной точности в число одинарной точности	DS	122264	162020	
Сложение целых чисел	SUM	122346	162102	
Вычитание целых чисел	SKR	122356	162112	

Операция	Усл. имя	Адрес в БК-0011М	Адрес в БК-0010.01	Параметры
Умножение целочисленное	IMUL	122362	162116	Нет (два операнда в стеке)
Деление целочисленное (знак \)	IDIV	122316	162052	
Остаток от деления (операция MOD)	IMOD	122322	162066	
Сложение чисел двойной точности	ADD	127410	167144	
Вычитание чисел двойной точности	SUB	127370	167124	
Умножение чисел двойной точности	MUL	130454	170210	
Деление чисел двойной точности	DIV	131242	170776	
Сравнение целых чисел	ICMP	115752	155172	
Сравнение чисел двойной точности	DCMP	137300	177034	
Возведение нормализованного числа в целую степень	PW	137400	177134	
Функции (для аргумента двойной точности)				
Квадратный корень	SQR	131614	171350	Число в стеке
Синус	SIN	134060	173614	
Косинус	COS	134032	173566	
Тангенс	TAN	134552	174306	
Арктангенс	ATN	134700	173406	
Экспонента	EXP	132226	171762	
Логарифм	LOG	133316	173054	
Целая часть числа	FIX	136456	176212	
Наибольшее целое, не превосходящее аргумент	INT	136604	176340	
Абсолютная величина	ABS	127032	166566	
Знак числа	SGN	127060	166614	
Абсолютная величина дробной части	FRAC	136204	175740	
Смена знака числа	NEG	117244	156750	
Генерация случайного числа	RND	135442	175176	Нет
Функции целого аргумента				
Абсолютная величина	IABS	127022	166566	Число в стеке
Смена знака	INEG	127026	166562	
Знак числа	ISGN	127040	166574	
Логическая операция «нет»	NOT	122552	162306	

Операция	Усл. имя	Адрес в БК-0011М	Адрес в БК-0010.01	Параметры
Логическая операция «или»	OR	122556	162312	Два кода в стеке
Логическая операция «и»	AND	122562	162316	
Логическая операция «исключающее или»	XOR	122570	162324	
Логическая операция эквивалентности	EQV	122604	162336	
Операции с символьными переменными				
Пересылка из стека в область SIM	SQU	121130	160634	Два кода в стеке
Инициализация имени и длины новой переменной	NEWS	117166	156672	
Функция CHR$	CHR	121004	160510	
Функция STRING$	STRG	121656	161412	
Функция INKEY$	INK	64764	124134	
Функция MID$	MID	121576	161332	
Конкантенация строк	CONC	121722	161456	
Функция LEN	LEN	121774	161530	
Функция ASC	ASC	122000	161534	
Функция VAL	VAL	122014	161550	
Функция OCT$	OCT	122070	161624	Целое число в стеке
Функция BIN$	BIN	122136	161672	
Функция HEX$	HEX	122200	161734	
Перевод в символьную форму целого числа	STRI	122054	161610	
Перевод в символьную форму числа одинарной точности	STRS	122040	161576	Число в стеке
Перевод в символьную форму числа двойной точности	STRD	122046	161602	
Получение символьной переменной	SGET		157100	
Некоторые алгоритмические операторы				
Переход безусловный	JMP	116054	155560	Адрес точки перехода
Вызов подпрограммы	JSR	116022	155526	
Возврат из подпрограммы	RTS	116216	155722	Адрес возврата в стеке
Завершение работы программы и переход к точке «START»	END	115216	154722	Нет
Останов	STOP	115250	154754	

Дополнительные пояснения к таблице.

1. 4- и 8-байтовые переменные в объектном коде располагаются «вверх ногами», т. е. сначала младшие разряды, затем старшие и мантисса.

2. Операции сравнения чисел извлекают из стека оба операнда, результат сравнения возвращается в виде значения соответствующего флага ССП.

3. Пересылка текстовой константы в стек или из стека фактически заключается в пересылке дескриптора (адреса и длины, всего 4 байта) с помощью команд SVAR и SMOV.

4. Сложение или вычитание целых чисел и логические операции быстрее выполняются машинными (ассемблерными) командами, но в контексте алгоритма может быть удобнее не выходить за пределы «псевдокода» и воспользоваться упомянутыми подпрограммами.

5. Описание реализации циклов и условных переходов в объектном коде заняло бы много места. Кроме того, при составлении машинных программ для этих целей проще использовать машинные команды.

6. Перед использованием блоков системы БЕЙСИКа в машинной программе надо где-либо сохранять и затем восстанавливать содержимое ячейки 2134.

Пример составления машинной подпрограммы

В качестве примера возьмем машинную подпрограмму решения квадратного уравнения (для простоты не будем проверять знак перед извлечением корня). Величины A, B, C уравнения $A*x^2+B*x+C=0$ перед обращением к машинной программе засылаем в некоторый массив M (начиная с нулевого элемента). Элементу M(3) дается значение 1 или −1, в зависимости от того, какой знак перед радикалом мы хотим выбрать (для получения двух решений надо будет дважды обратиться к подпрограмме). При обращении к машинной подпрограмме указываем нулевой элемент массива M. Таким образом, текст на БЕЙСИКе может быть следующим:

```
10 DEFUSR=&O40000
20 DIM M(4%)
30 M(0%)=...
40 M(1%)=...
50 M(2%)=...
60 M(3%)=1
70 B1=USR(M(0%))
80 M(3%)=-1
90 B2=USR(M(0%))
100 ? B1,B2
...
```

В машинной подпрограмме сначала сохраняем значение слова 2134, затем пересылаем адрес, где надо разместить результат, из регистра R5 в нужное место «псевдокода» (у нас — адрес 40144). После этого начинаем «ловить» адреса элементов массива. К сожалению, функция USR не может передавать адрес массива или даже его дескриптора, в регистре R5 размещается лишь адрес в стеке, куда

заблаговременно передается выбранное значение аргумента. Тем не менее адрес дескриптора может быть найден вблизи окрестности объектного кода БЕЙСИК-программы около точки, откуда произошло обращение к машинной подпрограмме:

```
          ...
n-16:  120352     ; засылка в стек дескриптора массива
n-14:   kkk       ; адрес дескриптора
n-12:   1         ; размерность массива
n-10:  116460     ; выборка элемента массива
n-6:   124116     ; вызов подпрограммы USR
n-4:   0
n-2:   0
n:     116630     ; пересылка значения функции
n+2:   mmm        ; адрес пересылки (у нас - адрес переменной B1 или B2)
```

При входе в машинную подпрограмму в регистре R4 сохраняется адрес возврата n, адрес дескриптора можно выбрать по адресу n-14. Адрес дескриптора указывает на слово — длину массива, нулевой элемент размещается на два байта ниже, следующий — еще через восемь байтов и т. д. Для удобства перешлем нужное количество элементов в область с фиксированным адресом (начиная с ячейки 1000), тогда начало машинной подпрограммы будет таким:

```
40000: 13727 2134 авт          ; сохранение SAVJ
40006: 10537 40144             ; пересылка адреса результата
40012: 10401 162701 14 11101   ; в R1 адрес дескриптора
40022: 11101 5021              ; в R1 адрес нулевого элемента
                               ;  массива
40026: 12702 1000
40032: 12122 22702 1042 1374   ; пересылка в цикле элементов
                               ;  массива
40042: 4437 127016             ; переход к "псевдокоду". Далее
                               ;  размещается "псевдокод"
40046: 116464 1010             ; засылка в стек коэффициента B
40052: 116464 1010             ; еще раз
40056: 130454                  ; перемножение (в стеке - B^2)
40060: 116464 1000 116464 1020
40070: 130454                  ; A*C
40072: 116446 0 0 0 40000      ; константа 4
40104: 130454 127370           ; B^2-4*A*C
40110: 131614                  ; извлечение корня
40112: 116464 1030 130454      ; умножение на 1 или -1
40120: 116464 1010 127370      ; вычесть B
40126: 116464 1000 116464 1010 127410 ; знаменатель 2*A
40140: 131242                  ; деление на знаменатель
40142: 116630  авт             ; пересылка результата
                               ;  по адресу, ранее
                               ;  засланному программой
40146: 40150                   ; выход из "псевдокода"
```


Продолжение программы в машинных кодах:

```
40150: 13737 40004 2134 ; восстановление SAVJ
40156: 207              ; завершение подпрограммы USR,
                        ;  возврат в БЕЙСИК
```

Аналогичную подпрограмму можно создать на БК-0010.01, но адреса блоков («псевдокоманды») в объектном коде будут другие.

Если в БЕЙСИКе не обнаружатся собственные возможности использования «скрытой памяти», можно воспользоваться короткими машинными подпрограммами, обращающимися к драйверам управления памятью. Например, записываем с адреса 40000:

```
40000: 11500 104013 207 0          ; для назначения
                                   ;  "рабочей страницы"
40010: 11500 13701 1000 104016 207 ; для записи слова
                                   ;  в "рабочую страницу"
40022: 11501 104014 100015 207     ; для чтения слова
                                   ;  из "рабочей страницы"
40032: 11500 104020 207            ; для пересылки массива
```

Тогда для записи, а затем чтения слова из «рабочей страницы» на БЕЙСИКе следует написать:

```
DEFUSR=&040000
DEFUSR1=&040010
DEFUSR2=&040022
  ...
N%=USR(ном)       ' ном - номер страницы
POKE &01000,адр   ' адр - адрес в рабочей странице,
N%=USR1(слово)    ' слово - записываеый код (целое число)
  ...
N%=USR2(адр)
  ...
```

Для пересылки массива в «скрытую память», а затем обратно записываем:

```
DEFUSR3=&040032
  ...
POKE &01000,надр      ' надр - начальный адрес массива в памяти
POKE &01002,надр2     ' надр2 - начальный адрес
                      '  в "скрытой памяти"
POKE &01004,nstr      ' nstr=nom1+nom2*256 - номера страниц,
                      ' подключаемых к адресам 40000 и 100000;
                      '  обычно или nom1 или nom2 нулевые
POKE &01006,-(длина)  ' длина - число пересылаемых слов
I%=USR3(&01000)       ' запись в назначенную страницу
  ...
POKE &01006, длина
POKE &01000,новадр    ' если надо, назначается новый адрес
                      '  в активной памяти
I%=USR3(&01000        ' чтение из "скрытой памяти"
```

Специальными приемами можно сделать обращения к блокам пересылки информации более компактными (скажем, используя функцию типа FN, где можно списком задавать несколько параметров).

При отсутствии функции VARPTR, возвращающей адрес размещения переменной в памяти, можно воспользоваться подпрограммами:

```
40000: 10401 162701 10 11115 207     ; для целой переменной
40012: 10401 162704 10 11165  6
       11601 62706  6 10116          ; для нормализованного
                                     ;  числа двойной точности
       62737 2 2134 207
40044: 10401 162701 10 12702 1000    ; для текстовой переменной
       11122 11101 12122 11112 207
```

Для целой переменной подпрограмма должна вызываться в виде: **I%=USR(P%)**, где **P%** — имя переменной, а в переменной **I%** возвращается адрес размещения P%. Для 8-байтовой переменной в виде исключения подпрограмма вызывается как **I%=USR1(P)**, т. е. аргумент действительного типа, а результат — целого. Для текстовой переменной подпрограмму приходится вызывать с фиктивной переменной (можно указать то же имя, что для аргумента): **I%=USR2(P$)** или **P$=USR2(P$)**. Подпрограмма заносит сведения в фиксированные ячейки: по адресу 1000 — адрес дескриптора, 1002 — длину переменной, 1004 — адрес начала самой переменной. Далее можно использовать функции PEEK.

Функция KEYLIST для вильнюсского БЕЙСИКа

Во всех компьютерах БК-0010.01 версия вильнюсского БЕЙСИКа, к сожалению, имеет в отношении использования программируемых ключей крупный недостаток: оператор KEYLIST, предназначенный для просмотра их содержимого, в ней не реализован. Хотя первые символы ключей выводятся в служебной строке, угадать, что они означают, порой весьма трудно. (К тому же русские буквы и полуграфика, входящие в состав ключа, в служебную строку не выводятся вообще.)

Чтобы не выучивать ключи наизусть и не держать постоянно под рукой список их значений, можно использовать несложную программу в машинных кодах. Для просмотра содержимого ключей с ее помощью нужно считать программу с магнитофона командой **BLOAD "KEYLST",R** (не забудьте предварительно ввести команду **CLEAR ,&O37440**). При успешной загрузке программа сразу же начнет выполняться и выведет на экран содержимое всех десяти ключей. Для продолжения работы нужно нажать любую клавишу. После выхода в БЕЙСИК можно ввести в непосредственном режиме команду **DEF USR=PEEK(&O264)** (или **DEF USR=&O37440**) и затем, при необходимости, просматривать содержимое ключей командой **A=USR(A)**.

Ввести коды программы можно, например, в режиме ТС или в отладчике, после чего необходимо записать ее на магнитофон под именем KEYLST.BIN (программа неперемещаема, ее начальный адрес должен быть равен 37440_8).


```
Начальный адрес: 037440      Длина: 000340
010046 010146 010246 010346 010446 010546 012701 037730
012702 000020 104020 012700 000204 104016 000240 012703
000001 012705 000005 012701 000003 010502 104024 012701
037724 012702 000004 104020 010300 020327 000012 001403
062700 000060 000405 012700 000061 104016 012700 000060
104016 012700 000040 104016 010304 006304 062704 037676
011401 112102 104020 005203 005205 022703 000013 001333
012700 000204 104016 000240 012701 037750 012702 000027
104020 104006 012700 000233 104016 104016 012700 000014
104016 012605 012604 012603 012602 012601 012600 000207
002140 002161 002202 002223 002244 002265 002306 002327
002350 002371 042513 020131 111414 111235 017036 045436
054505 044514 052123 035040 017012 017036 117436 050040
042522 051523 040440 054516 045440 054505 020056 000237
Контрольная сумма: 041457
```

Программа рассчитана на работу с цветным монитором (вывод зеленого текста на синем фоне). Если же требуется установить «стандартные» монохроматические цвета (белый текст на черном фоне), нужно заменить коды по адресам 37730_8 и 37732_8 на 14_8 и 0 соответственно. Можно также уже после считывания с магнитофона использовать команды **POKE &O37730,&O14** и **POKE &O37732,0%** .

Для частого применения можно предложить следующий порядок работы с программой KEYLIST:

- введите команду **CLEAR ,&O37440** в начале работы с БЕЙСИКом:
- загрузите программу с магнитофона: **BLOAD "KEYLST",R** ;
- введите команду **DEF USR=&O37440** ;
- с помощью оператора KEY запрограммируйте какой-либо ключ (например «АР2»+«5») на последовательность символов «A=USR(A)»: **KEY 5,"A=USR(A)"+CHR$(10)** ;
- вызов оператора KEYLIST будет производиться нажатием «АР2»+«5».

В заключение приведем для интересующихся таблицу адресов строк ключей БЕЙСИК-системы:

Номер ключа	Адрес (восьм.)
1 («АР2»+«1»)	2140
2 («АР2»+«2»)	2161
3 («АР2»+«3»)	2202
4 («АР2»+«4»)	2223
5 («АР2»+«5»)	2244
6 («АР2»+«6»)	2265
7 («АР2»+«7»)	2306
8 («АР2»+«8»)	2327
9 («АР2»+«9»)	2350
10 («АР2»+«0»)	2371

Первый байт каждой строки, адрес которого и указан в таблице, содержит количество символов ключа, до 20_8.

СПРАВОЧНЫЙ ЛИСТОК

Принципиальная схема БК-0011М

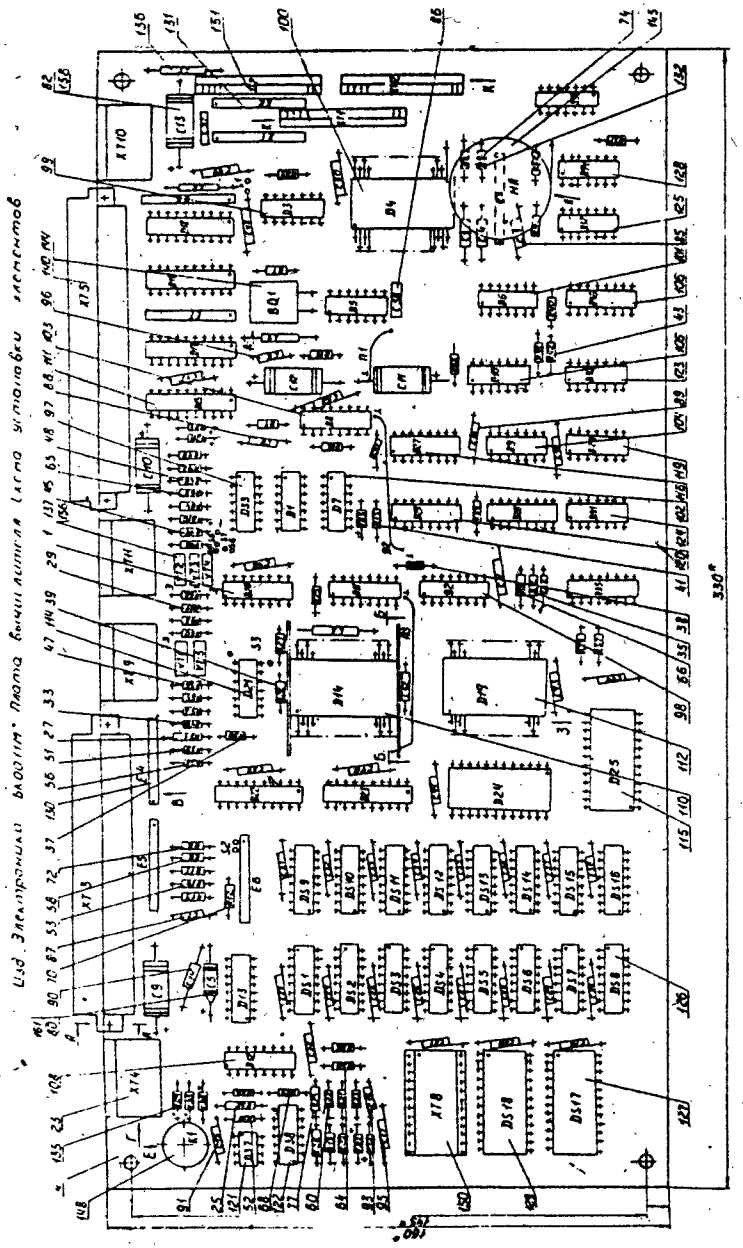
На этой и последующих страницах, а также на двойном листе вкладки дана заводская принципиальная схема компьютера БК-0011М, перечень радиоэлементов и схема их установки на печатной плате, которая взята без каких бы то ни было изменений из заводской документации. (К сожалению, предоставленный редакции оригинал имеет невысокое качество, поэтому возможна плохая пропечатка мелких надписей на схемах.)

Перечень радиоэлементов принципиальной схемы БК-0011М

Поз. обозначение	Наименование и номинал	Кол-во
BQ1	Резонатор РК 15 9МВ-6ВТ-12000 кГц	1
	Конденсаторы	
C1—C4, C19—C41, C43—C50, C52	0.068 мкФ	36
C5	10 мкФ	1
C6	82 пФ	1
C7	47 пФ	1
C8, C51	300 пФ	2
C9—C13	47 мкФ	5
C14	0.47 мкФ	1
C15	0.022 мкФ	1
C16	4700 пФ	1
C17	1000 пФ	1
C18	180 пФ	1
C42, C56, C57	0.33 мкФ	3
	Микросхемы	
D1, D5, D33	К555ЛН1	3
D2	К561ПУ4	1

D3	К555ТМ2	1
D4	КР1801ВП1-014	1
D6	К555ЛЕ4	1
D7	К555ЛА3	1
D8	КР531ТВ9	1
D9, D39	К155ТМ2	2
D10	К155ЛА13	1
D11, D13, D26, D35, D36	К555ТМ9	5
D12	КР531АП2	1
D14	КР1801ВМ1А	1
D15—D18, D22, D23	К555ИР23	6
D19	КР1801ВП1-037	1
D20	КР556РТ4А	1
D21	К155ЛП9	1
D24, D25	К155ИР13	2
D27	К555ТМ7	1
D28	К555ИЕ5	1
D29	К555КП2	1
D30	К555ЛЕ1	1
D31	К555ЛЛ1	1
D32	К555ЛИ6	1
D34	К555ЛН2	1
D37	К155ЛА18	1
D38	К554СА3А	1
DS1—DS16	КР565РУ5Г	16
DS17	КМ1801РР1-01	1
DS18	КМ1801РР1-02	1
	Наборы резисторов	
E1, E4—E6	НР1-4-9-0.125-3.3 кОм	4
E2, E7, E8	НР1-4-9-0.125-22 кОм	3
E3	НР1-4-9-0.125-180 кОм	1
HA1	Звонок пьезокерамический ЗП19	1
K1	Реле РЭС-15	1
L1—L4	Дроссель ДМ2.4-6 мкГ	4

Резисторы		
R1, R7, R41, R42	750 Ом	4
R2, R50	200 Ом	2
R3—R5	180 кОм	3
R6, R21	10 кОм	2
R8, R35	43 кОм	2
R9	5.1 кОм	1
R11, R19, R20, R33, R37, R44, R45, R47, R48, R53, R54, R56, R58	1 кОм	13
R12, R23	20 кОм	2
R13	620 Ом	1
R14—R17, R25—R27, R36, R38—R40, R57, R59	2.2 кОм	13
R22	4.7 кОм	1
R24	470 кОм	1
R28	75 Ом	1
R29	18 кОм	1
R30, R31	6.2 кОм	2
R32	30 Ом	1
R46	1.2 кОм	1
R49	150 Ом	1
R51	51 Ом	1
R52	2.7 кОм	1
R55, R64	1.5 кОм	2
R60—R63	68 Ом	4
R65	2 кОм	1
R66	100 Ом	1
VD1—VD8, VD10, VD11	Диод КД521Д	10
VT1—VT5	Транзистор КТ315Г	5
Разъемы		
XT1, XT2, XT12	ОНП-СГ-83-12/37х6.5-Р55	3
XT3, XT5	Вилка СНП56-64/94х9В-23-2-В	2
XT4, XT9—XT11	Соединитель ОНЦ-КГ-5/16Р	4
XT8	Розетка РС24-7	1



Удобной возможностью, реализованной в файловой оболочке Disk Master из комплекта ANDOS 3.1, является механизм дополнительных подгружаемых модулей, называемых М-файлами. После загрузки они могут оставаться резидентными, доступными в любой момент при работе с Disk Master, и обеспечивают доступ пользователя к каталогу дискеты, служебным ячейкам и подпрограммам операционной системы. Но в прилагаемой к ANDOS технической документации отсутствует какая-либо информация о том, как самостоятельно разрабатывать М-файлы.

По просьбе редакции автор оболочки Disk Master предоставил публикуемое ниже руководство программиста. Надеемся, что оно поможет читателям реализовать многие интересные идеи.

С. Камнев,
Москва

Краткое руководство по написанию М-файлов для Disk Master 3.1

Сервисная оболочка Disk MASTER 3.1 предусматривает возможность использования дополнительных подгружаемых модулей (М-файлов), расширяющих ее возможности. Эти модули, имеющие расширение .M, могут быть запущены на исполнение путем их выбора на файловой панели или из командной строки. При этом выбранный М-файл загружается с указанного в каталоге диска начального адреса, а затем выполняется проверка его первого двухбайтного машинного слова. Оно должно быть равно 104130_8 (код команды EMT 130), что является признаком дополнительного модуля оболочки. Запуск программы производится со второго слова, причем на экране сохраняются панели оболочки и остаются доступными ее внутренние системные запросы и ячейки.

Адрес загрузки М-файла может быть любым, однако нежелательно использовать адреса меньше 24000_8, так как это может привести к сбоям при загрузке и запуске такого модуля. Целесообразнее всего располагать модули с адреса 137000_8, поскольку в этом случае они постоянно сохраняются в памяти (до загрузки другого модуля или перезапуска системы) и могут быть вызваны в любой момент командой «СУ»+«F» оболочки (эта команда выполняет переход на адрес 137002_8, если в ячейке 137000_8 находится слово 104130_8). Правда, длина модуля в этом случае ограничена (не более 1000_8).

Особое внимание нужно обратить на системные запросы с кодами 107200_8 и больше (кроме 107204_8). Обрабатывающие их модули с целью

обеспечения работоспособности Disk Master как на БК-0010(.01), так и на БК-0011(М) являются оверлейными, хотя и хранятся постоянно в ОЗУ (на БК-0010(.01) — ДОЗУ с адресами 140000_8—157777_8, на БК-0011(М) — седьмая страница памяти) без подгрузки с диска. При вызове они вначале копируются в основное ОЗУ по адресам менее 30000. Поэтому при использовании данных запросов М-файл надо располагать по адресам выше 30000.

В этой статье приведены некоторые значения адресов ОЗУ, используемых оболочкой, а также список ее системных ячеек и запросов. Системный запрос (оформленный по стандарту подпрограммы) вызывается по резервному коду из диапазона 107002_8—107377_8, причем после кода вызова могут быть записаны входные параметры запроса. Резервные же коды из интервалов 7000_8—7377_8 и 70000_8—70377_8 выводят на экран символы, коды которых совпадают с младшими байтами команд (например, код 70101_8 выдает на экран букву «А»). Вывод символов производится аналогично выполнению запроса SYMB (см. далее). Для вывода символа на экран можно также использовать команду IOT, при этом корректно обрабатываются непечатаемые коды: для кодов от 200_8 до 237_8 выдаются строчные латинские буквы, некоторые коды из интервала 0—37_8 выглядят как в режиме БЛОК РЕД, а остальные, меньшие 40_8, — в виде своеобразных «иероглифов».

Прерывание по клавише «СТОП» (или команде HALT) приводит к перерисовыванию текущей панели и переходу в режим обычной работы с ней (аналогично запросу PAN).

И наконец, многих заинтересует следующий очень интересный факт, обнаруженный и проверенный автором данной статьи, но нигде до сих пор не упоминавшийся. Любая программа в машинных кодах, имеющая автозапуск, после переименования ее в «⟨имя⟩.M» загружается и запускается на выполнение существенно быстрее, чем до «превращения» в М-файл. Правда, в этом случае надо учесть ограничения, накладываемые оболочкой Disk Master, при выходе из программы: завершающей ее работу командой должна быть не просто HALT, а пара EMT 14 и HALT, переход на начало монитора JMP @#100000 или означающая то же самое М-файловская команда EMT 130.

Распределение адресного пространства БК при работе Disk Master

Ниже перечислены восьмеричные адреса ОЗУ БК, начиная с которых расположены подгружаемые с диска блоки информации о файлах, а также фрагменты памяти, зарезервированные в качестве буфера:

- 500 — буфер строчного редактора;
- 1000 — FAT диска, каталог которого отображен на левой панели;
- 3000 — каталог диска для левой панели;
- 12040 — FAT диска, каталог которого отображен на правой панели;
- 14040 — каталог диска для правой панели.

Дополнительные байты записей о файлах в каталоге диска, используемые оболочкой Disk Master

Смещение относительно начала записи (восьм.)	Назначение
16	Признак выделенного файла (по нажатию клавиши «ВС»): байт не равен нулю — файл или подкаталог выделены
24	Номер данного подкаталога (для файлов этот байт равен нулю)
25	Номер «родительского» подкаталога, в котором содержится данный подкаталог или файл (нулевое значение — принадлежность корневому каталогу диска)

Резидентные флаги и параметры

Имя	Адрес (восьм.)	Длина	Назначение
COLORB	126504	1	Наличие режима оцветнения: 0 — оцветнение выключено (черно/белый дисплей)
PRIB	126505	1	Номер драйвера принтера
ILLP	126506	—	Строка инициализации (первый байт — длина строки), выводимая на принтер подпрограммой ANDOS INIPRN (нач. адр. 120314)
NLIN	126524	1	Количество строк при печати из режима VIEW
OTSTUP	126525	1	Отступ слева при печати
TSS	126531	1	Задержка включения «хранителя экрана» (зарезервировано для будущих версий Disk Master)
HSTEP	126532	2	Шаг горизонтального скроллинга во VIEW

Параметры активной файловой панели

Имя	Адрес (восьм.)	Длина	Назначение
DAB	126542	1	Номер устройства (A: — 0, B: — 1,...)
DIRB	126543	1	Номер текущего подкаталога

AFAT	126544	2	Адрес FAT для активной панели
ADR	126546	2	Текущий адрес ячейки FAT
Y	126550	2	Текущая вертикальная координата указателя (курсора в файловой панели)
CWO	126552	2	Контрольное слово (=@ADR)
ILIB	126554	1	Индикатор режима прозрачности (0 — режим отключен)
XB	126555	1	Горизонтальная координата панели (1 или 41)
DATEB	126556	1	Режим вывода в правой колонке панели «ДАТА/ДЛИНА» (0 — «ДЛИНА», 377_8 — «ДАТА»)

Параметры пассивной панели

Имя	Адрес (восьм.)	Длина	Назначение
DAPB	126560	1	Номер устройства (A: — 0, B: — 1,...)
DIRPB	126561	1	Номер текущего подкаталога
AFATP	126562	2	Адрес FAT
ADRP	126564	2	Текущий адрес
YP	126566	2	Текущая вертикальная координата указателя
CWOP	126570	2	Контрольное слово (=@ADRP)
ILIPB	126572	1	Индикатор режима прозрачности (0 — режим отключен)
XPB	126573	1	Горизонтальная координата панели (1 или 41)

Флаги и параметры, сбрасываемые при входе в оболочку

Активная панель

Имя	Адрес (восьм.)	Длина	Назначение
FILT	126370	11	Фильтр (текстовая строка)
NFIL	126404	2	Количество помеченных файлов и занимаемых ими кластеров
NCL	126406	2	

Имя	Адрес (восьм.)	Длина	Назначение
TDA	126410	2	Таблица параметров привода (берется из соответствующей ячейки ОС ANDOS)
FRECL	126412	2	Количество свободных кластеров
ADR0	126414	2	Базовый адрес начала каталога
ADR1	126416	2	Адрес начала каталога (с учетом «..»)*
ITEGB	126420	1	Индикатор режима вывода помеченных файлов
ITTB	126421	1	Текущий индикатор режима вывода помеченных файлов (0 — режим неактивен)
IMDOS	126422	2	Индикатор DOS (0 — ANDOS, иначе MicroDOS)

Примечание к таблице. * — если на активной панели было выведено содержимое подкаталога, в ОЗУ, где хранится копия каталога дискеты, временно помещается запись о «псевдофайле» с именем «..», по которому производится выход из данного подкаталога. Длина этой записи — 32. (40_8) байт.

Пассивная панель

Имя	Адрес (восьм.)	Длина	Назначение
FILTP	126424	11	Фильтр
NFILP	126440	2	Количество помеченных файлов и занимаемых ими кластеров
NCLP	126442	2	
TDAP	126444	2	Таблица параметров привода (берется из соответствующей ячейки ОС ANDOS)
FRECLP	126446	2	Количество свободных кластеров
ADR0P	126450	2	Базовый адрес начала каталога
ADR1P	126452	2	Адрес начала каталога (с учетом «..»)
ITEGBP	126454	1	Индикатор режима вывода помеченных файлов
ITTBP	126455	1	Текущий индикатор режима вывода помеченных файлов (0 — режим неактивен)
IMDOSP	126456	2	Индикатор DOS (0 — ANDOS, иначе MicroDOS)

Специальные ячейки

Имя	Адрес (восьм.)	Длина	Назначение
FLAG	126460	2	Флаг запоминания фрагмента экрана

Примечание. Если в данной ячейке содержится ненулевое число, то при выполнении операции вывода рамки (окна) часть экранного ОЗУ под ней сохраняется в динамическом буфере, «растущем» (аналогично стеку) от адреса 40000$_8$ вниз. Стирание рамки тогда производится быстрым переписыванием участка экрана, но нужно учитывать, что часть программы в старших адресах пользовательского ОЗУ может оказаться затертой. При нулевом же значении ячейки сохранения экрана не происходит и при стирании рамки выполняется более долгая полная перерисовка панели. Очень важно, что в пользовательском М-файле допускается только обнулять данную ячейку (если это требуется), записывать же туда другое ненулевое число (отличное от уже содержащегося в ней) НЕЛЬЗЯ — это может нарушить работу оболочки в целом.

Системные запросы

Знак «*» рядом с кодом вызова указывает, что все регистры сохраняются (за исключением тех, куда записываются результаты работы подпрограммы).

Поддержка клавиатуры

- INPUT (107002*) — прием символа с клавиатуры в R0.
- INPUT1 (107107*) — прием символа с клавиатуры в R0 без задержки при автоповторе.
- INPC (107025*) — ввод с клавиатуры кода клавиши (с ожиданием ее нажатия) в R0 с включением курсора и его гашением после окончания ввода. (Данная функция предназначена в основном для работы строчного редактора.)
- ASC2 (107104) — ввод @R3 двухзначного десятичного числа.
- BLS (107021*) — блокирование прерывания по клавише «СТОП».
- UNBLS (107052*) — отмена блокировки «СТОП».

Вывод на экран

- CUR (107024*) — вкл./выкл. курсора.
- SYMB (107003*) — вывод символа из R0 на экран.
- LINE (107004*) — вывод текста на экран. Текст в виде строки ASCII-кодов должен располагаться вслед за кодом вызова. Признак конца текста — нулевой байт, если количество символов нечетное, или нулевое слово, если четное.
- TYP10 (107007*) — вывод содержимого R2 (R2<1000.) в десятичной форме.

Интерфейсные функции

- DRIV (107200) — меню «выбор привода».
- YN (107073*) — меню «YES»/«NO». Если выбрано «YES», то работа программы продолжается, иначе происходит прерывание по HALT. После кода вызова записывается байт, равный 300+x (где x — левый

отступ для текста, выводимого в заголовке рамки редактора), а затем сам текст заголовка, завершаемый нулевым байтом или словом (в зависимости от четности количества занимаемых байтов). Пример:

```
EMT 130
CLR @#177130
.WORD 107073    ; вызов окна меню "Yes"/"No"
                ;  с заголовком "Сигнал"
.BYTE 302       ; отступ заголовка равен 2
.ASCII/Сигнал/<0>
                ; текст заголовка окна
.WORD 107034    ; если выбрано "Yes", звуковой сигнал
RET             ; если "No"; выход без сигнала
```

- RAMKA (107064*) — вывод текста в рамке. После кода вызова записываются четыре байта параметров рамки:

B1 — горизонтальная координата левого верхнего угла (X0);
B2 — вертикальная координата (B1 и B2 задаются по стандарту EMT 24);
B3 — высота рамки (в точках);
B4 — длина (в символах).

Затем следует сам текст (аналогично LINE). Текст может быть разбит кодами 12 на отдельные строки, каждая из которых будет иметь горизонтальное смещение X0. Фрагмент экрана под рамкой сохраняется в специальном буфере, если FLAG не равен 0. Пример:

```
EMT 130
CLR @#177130
.WORD 107064
.BYTE 2,3       ; левый верхний угол рамки - во второй
                ;  позиции третьей строки
.BYTE 40.       ; высота рамки равна 40 (дес.) графических
                ;  строк (т. е. на три строки текста)
.BYTE 10        ; ширина рамки равна 8 символам,
                ;  т. е. максимальная длина выводимых
                ;  в нем строк не более 6 символов)
.ASCII/TEKCT1/<12>
.ASCII/TEKCT2/<12>
.ASCII/TEKCT3/<0>
EMT 6           ; ожидание нажатия клавиши
.WORD 107216    ; использование запроса WPAN для выхода
                ;  из М-файла с перерисовкой оболочек
```

- EDIT (107065) — строковый редактор. После кода вызова записывается байт, равный 300+x (где x — левый отступ для текста, выводимого в заголовке рамки редактора), а затем сам текст заголовка, завершаемый нулевым байтом или словом (в зависимости от четности количества занимаемых байтов). Редактируемая строка должна располагаться начиная с адреса 500 и состоять не более чем из 1210 символов (признак конца текста — нулевой байт). Пример:

```
EMT 130         ; признак начала М-файла
CLR @#177130    ; останов мотора дисковода
```



```
        MOV #TXT,R1     ; перенос редактируемой строки в буфер
        MOV #500,R2
1:      MOVB (R1)+,(R2)+
        BNE 1
        CLRB (R2)       ; обнулить последний байт
                        ;  (признак конца буфера)
        .WORD 107065    ; вызов системного запроса EDIT
        .ASCII/a//EDIT/<0>
                        ; вывод окна с заголовком "EDIT :"
                        ;  (код 301 в виде русской строчной
                        ;  буквы "а" задает отступ заголовка,
                        ;  равный 1)
        RET             ; возврат из М-файла (измененный вариант
                        ;  строки находится в буфере @#500)
TXT:    .ASCII/TEXTED/<0>
                        ; исходный вариант редактируемого текста
```

- GOR (107047) — горизонтальное меню. После кода вызова располагаются несколько байтов, содержащих буквенные символы («горячие клавиши»), по которым должны вызываться пункты меню (их количество определяет число пунктов), признак конца списка — 0. В ячейке 160 должен находиться базовый адрес экрана, в R2 — адрес таблицы параметров, в R3 — номер текущего пункта меню (т. е. можно уже при вызове меню поместить указатель на требуемый «приоритетный» пункт). Таблица параметров (для каждого пункта) имеет следующую структуру:

смещение относительно базового адреса (слово);
высота маркера в точках (байт);
длина маркера в символах (байт).

Подпрограмма помещает в R3 номер выбранного пункта меню. Пример см. ниже для запроса VERT.

- VERT (107050) — вертикальное меню (работает аналогично GOR). Пример:

```
        EMT 130
        CLR @#177130
        .WORD 107064    ; вывод окна с текстом пунктов меню
        .BYTE 5,5,40.,15
        .ASCII/TEKST1/<12>
        .ASCII/TEKST2/<12>
        .ASCII/TEKST3/<0>
        MOV #BL,R2      ; адрес блока параметров меню
        CLR R3          ; 0 - номер первого пункта меню
        .WORD 107050    ; вызов вертикального меню (VERT)
        .ASCII/ABC/<0>
                        ; список "горячих клавиш"
        ; сейчас в R3 находится номер выбранного пункта
        TST R3          ; 0 - выбран первый пункт меню
        BNE 1
        .WORD 107034    ; первый пункт - один звуковой сигнал
        RET             ;  и выход
1:      CMP R3,#1       ; 1 - выбран второй пункт меню
```

```
        BNE 2
        .WORD 107034    ; второй пункт - два
        .WORD 107034    ;  звуковых сигнала
        RET             ;  и выход
2:      CMP R3,#2       ; 2 - выбран третий пункт меню
        BNE 3           ; переход для R3 = 0 или 1
        .WORD 107034    ; третий пункт -
        .WORD 107034    ;  три звуковых
        .WORD 107034    ;  сигнала
3:      RET             ;  и выход
; блок параметров для меню
BL:     .WORD 175372    ; (-2406) смещение к первому пункту
        .BYTE 12,10     ; размер инвертируемой области для
                        ;  первого пункта
        .WORD 176572    ; (-1206) смещение ко второму пункту
        .BYTE 12,10     ; размер инвертируемой области
        .WORD 177772    ; (-6) смещение к третьему пункту
        .BYTE 12,10     ; размер инвертируемой области
```

От редакции. Данные, помещаемые в ячейки блока параметров BL, рассчитываются следующим образом. Ячейка 160 в момент вызова меню содержит адрес верхнего левого угла курсора, который, в свою очередь, находится в конце строки третьего пункта (после цифры 3). В первом слове блока параметров нужно записать смещение относительно содержимого ячейки 160 к верхнему левому байту инвертируемой области для первого пункта (имитация наведения курсора-подсветки на первую строку). Очевидно, что данное смещение равно −2406, т. е. соответствует перемещению адреса текущего байта на 6 символов (длина пунктов меню) влево и на две текстовые строки вверх. Следовательно, содержимое первой ячейки блока параметров равно 175372 («беззнаковая» запись числа −2406). Аналогично рассчитываются смещения ко второму и третьему пунктам (в обоих случаях перемещение на 6 байт влево и для второго пункта — еще на одну текстовую строку вверх).

Что же касается размеров инвертируемой области, ее высота (12_8 графических точек) соответствует высоте одной текстовой строки, а ширина выбрана произвольно (не меньше длины пунктов меню и не больше ширины рамки минус 2). Для всех трех пунктов меню размеры инвертируемой области выбраны одинаковыми (что чаще всего и делается), но реально они (как и смещения к текстам пунктов) могут различаться. А это, в свою очередь, позволяет задавать (и правильно «подсвечивать»!) меню с пунктами разной длины и высоты и даже расположенными в произвольных местах рамки с текстом (реализуя таким образом полноценный аналог гипертекстовой подсказки).

- SIGNAL (107034) — звуковой сигнал. Пример см. для запроса YN.
- UNFILD (107072*) — выводит фрагмент, сохраненный подпрограммой RAMKA, из памяти на экран (восстановление экрана и стирание рамки).
- UFILD (107103*) — операция UNFILD для всех запомненных фрагментов экрана.

Дисковые и файловые функции

- INIT (107203) — инициализация диска (с установкой индикатора IMDOS). Если в М-файле предполагается смена дискеты, данная операция должна быть первой в алгоритме последующих действий.
- INIT1 (107077*) — инициализация диска. Если диск не в формате ANDOS, то устанавливается бит С.
- LCAT (107223) — инициализация диска и чтение каталога для активной панели. Автоматическое преобразование каталога из формата MicroDOS.
- DOS (107054*) — проверка DOS (если ANDOS, то бит Z устанавливается в «1»). Пример:

```
        .WORD DOS
        BEQ AND
MKD:    ; это не ANDOS
        .........
AND:    ; это ANDOS
```

- CMDOS (107027*) — контроль MicroDOS (если MicroDOS, то HALT).
- CMPD (107042*) — сравнение приводов (если бит Z установлен в «1», то приводы одинаковые). Пример:

```
        .WORD CMPD
        BEQ EQU
        ..........
EQU:    ; приводы одинаковые
```

- NAMER (107030) — преобразование имени файла из формата каталога ANDOS (адрес записи в R5) в формат блока параметров EMT 36 (с точкой, разделяющей имя и расширение, и т. д.). Преобразованное имя записывается начиная с адреса, указанного в R0.
- FMET (107011*) — поиск помеченного файла начиная с адреса в R1. Если файл найден, то адрес его записи в каталоге помещается в R1, иначе бит Z ССП устанавливается в единицу.
- DIR (107031) — поиск подкаталога с номером, находящимся в DIRB, адрес записи помещается в R1. Если подкаталог не найден, то R1=0.
- DIR0 (107032*) — то же с номером в R0.
- FIMDI (107033*) — поиск помеченного подкаталога (аналогично FMET).
- FIL (107015*) — проверка совпадения имени файла или подкаталога (адрес записи в R1) с фильтром. Если имя не соответствует фильтру, то устанавливается бит С; если обнаружен признак конца каталога (@R1=0), то устанавливается бит Z. Пример:

```
        ; поиск файлов на букву "А" среди выведенных на панель
        MOV ADR0, R1    ;загрузка адреса начала каталога
        BR 1¤
2¤:     ADD #40, R1     ;переход к следующей записи
1¤:     .WORD FIL
```

```
        BEQ 3д          ;проверка конца каталога
        BCS 2д          ;проверка на соответствие фильтру;
                        ; если не соответствует, то переход
                        ; к следующей записи
        CMPB @R1, #'A   ;проверка первого символа записи
        BNE 2д
        ; файл найден
        ...
3д:     ; конец каталога - файл не найден
        ...
```

Сравнение производится по нескольким признакам:

1) при включенном режиме вывода помеченных файлов — ТОЛЬКО по признаку пометки;

2) по принадлежности текущему подкаталогу (в «прозрачном» режиме игнорируется);

3) сравнение одиннадцати символов имени с фильтром (адрес FILT), при этом нулевые байты фильтра соответствуют ЛЮБЫМ символам имени.

- EOF (107026*) — контроль конца каталога (адрес записи в R5). Если @R5=0, то происходит HALT-прерывание.
- FREE (107037) — расчет количества свободных кластеров (результат в R2).
- INF (107222) — вывод на экран параметров диска (files, clust., free) в нижней строке панели.
- INDDIR (107231) — вывод в верхней части панели пути к текущему подкаталогу.
- MARK1 (107023*) — гашение указателя текущего файла (в R2 должна находиться вертикальная координата — содержимое ячейки Y).
- ENDER (107070) — запись текущего каталога на диск и перерисовка панелей.
- SAVE (107214) — запись каталога активной панели на диск.
- SPA (107014*) — смена активной панели.
- PAG (107202) — вывод файловой панели на экран. Требуется после перечитывания каталога дискеты из М-файла и в других случаях, например при порче изображения панели.
- PAN (107204) переход в режим работы с панелью. Можно использовать как выход в оболочку (аналогично HALT).
- WPAN (107216) — перерисовка обеих панелей (используется для выхода из М-файла, но в отличие от PAN перерисовывает обе панели). Пример см. для запроса RAMKA.
- WDP (107217) — дублирование панелей (используется для выхода из М-файла в случае, когда изменение в каталоге обрабатываемой панели обязательно должно быть отражено на соседней, например при удалении или переименовании подкаталога).
- MFAT (107040) — копирование FAT с адреса AFAT в буфер ANDOS.
- FATM (107041) — операция, обратная MFAT.
- TEST (107207) — проверка совпадения FAT диска с FAT текущей панели. Необходимо для обнаружения замены диска перед записью каталога.

- AES (107043*) — установка режима кеширования по чтению/записи каталога и FAT (соответствующие блоки будут обрабатываться не на диске, а в каталоге оболочки).
- AER (107044*) — сброс режима кеширования.
- DICL (107036*) — кластерная операция (чтение/запись отдельных кластеров): R3 — номер кластера, R1 — длина (положительное число — чтение, отрицательное — запись; собственно длина — абсолютное значение R1), R2 — адрес (после операции к R2 прибавляется абсолютная величина R1).
- DISK1 (107100) — дисковая операция (чтение/запись секторов). После кода вызова записываются три машинных слова: номер блока (если меньше 0, то номер блока берется из R0), длина (положительное число — чтение, отрицательное — запись; собственно длина задается абсолютным значением числа), адрес (если 0, то адрес берется из R2). При дисковой ошибке устанавливается бит С.
- DISK (107010*) — вызов ANDOSовской подпрограммы RWBLOK (см. документацию программиста из комплекта ОС ANDOS).
- DONAM (107116) — вызов ANDOSовской подпрограммы DONAM.
- FIFA (107117) — вызов ANDOSовской подпрограммы DIRFI2.
- SR1 (107020*) — запись в регистр R1 значения ADR0−40.

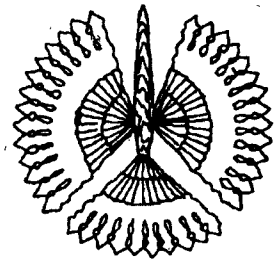
Обмен с принтером

- IPR (107074) — инициализация принтера. Если принтер отключен выводится соответствующий текст.
- PRINT (107075*) — печать содержимого R0 на принтере.
- LF (107006) — перевод строки на печатающем устройстве.
- OTS (107102) — генерация отступа при печати.
- OTSL (107111) — отступ с переводом строки на принтере.

Дополнение к статье М. И. Кондратовича «Малая издательская система БК-PageMaker»

(Персональный компьютер БК-0010 — БК-0011М. 1995. №6.)

В процессе эксплуатации издательской системы ВРМ московскими пользователями был обнаружен ряд недоработок этой программы: не работает драйвер принтера D100M (однако печать на МС-6313 проверена в редакции и признана удовлетворительной), возможны сложности при конвертации текста из формата EDASP и VorteX, иногда происходит сдвиг изображения при выполнении операции Preview и не совсем корректно осуществляется обмен графикой с программой TecnoArt. В настоящее время автору издательской системы ВРМ отправлен запрос на разработку ее исправленного варианта, которая будет разослана всем пользователям прежней версии бесплатно (с оплатой только чистой дискеты и пересылки).



Написать программу, позволяющую выводить на экран двумерные графики функций или рисовать разнообразные плоские геометрические фигуры, достаточно просто. Создать «графический редактор» для рисования аксонометрических проекций по заданным координатам опорных точек (концов отрезков) тоже не очень сложно. Но полноценная трехмерная графика подразумевает и более сложные изобразительные возможности, например произвольный поворот объекта относительно наблюдателя, масштабирование и т. д. Разработать подобную программу довольно трудно, и надеемся, что подробно изложенные в данной статье алгоритмы графических преобразований и листинги подпрограмм на БЕЙСИКе помогут читателям освоить секреты пространственной графики.

Ю. В. Котов,
Москва

Подпрограммы для отображения пространственных образов на БК-0010

Построение аксонометрических (параллельных) и перспективных (центральных) проекций пространственных объектов, в том числе с контролем видимости, было хорошо отработано многими авторами на «больших» ЭВМ еще в 1970—1975 гг. (рис. 1а, 1б, 1в). На малых машинах, таких, как БК, сложно получить реалистические изображения с раскраской или устранением невидимых линий, но в схематическом («прозрачном» или, как еще говорят, «проволочном») виде построение проекций трехмерных объектов возможно и на БК-0010.01 (рис. 2). В этой статье рассматриваются основные подпрограммы на БЕЙСИКе, служащие для этой цели. (Ограниченность оперативной памяти БК-0010.01 делает предпочтительным использование машинных подпрограмм, вызываемых операторами USR, но их основные алгоритмы будут теми же.) Однако предварительно следует обсудить геометрические схемы построения проекций разных типов.

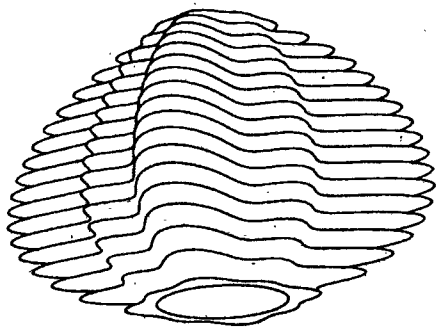


Рис. 1а

Аксонометрические и перспективные проекции

На рис. 3 показаны схемы построения параллельной (аксонометрической) и перспективной (центральной) проекций, а также центральная панорамная проекция на цилиндрическую картинную поверхность и вариант так называемой «квазисферической» перспективы. Эти схемы не идеальны, и достоверность отображения с их помощью пространственного мира ограничена. В частности, при увеличении угла зрения в обычной (линейной) проекции появляются хорошо заметные искажения пропорций и величин углов, недостоверно передается глубина пространства. В этих случаях некоторые специалисты советуют переходить к цилиндрической перспективе или к другим специальным проекциям. Последние же, увы, имеют свои недостатки (в частности, искривление прямых линий). Однако и для линейной перспективы, и для специальных проекций существенно, в каких условиях они рассматриваются. Искажения восприятия значительно уменьшаются при «расчетных» условиях рассматривания, когда на чертеж мы смотрим с определенного расстояния, а наши глаза находятся напротив главной точки картины. При этом должны восстанавливать-

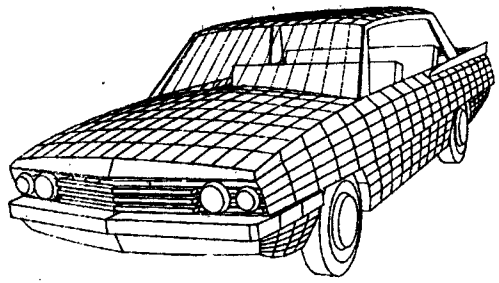


Рис. 1б

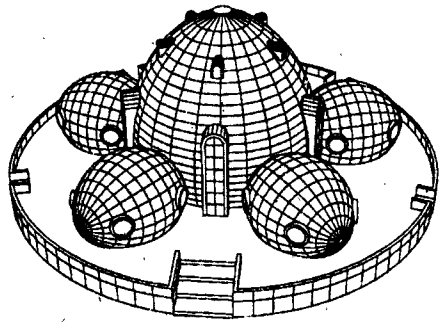


Рис. 1в

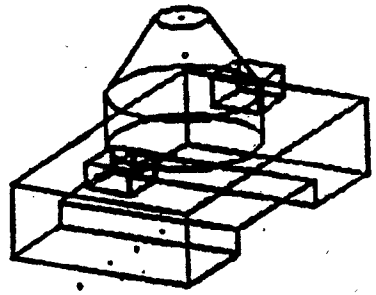


Рис. 2

ся условия проецирования, при которых было получено изображение. Цилиндрическую проекцию для «расчетного» рассматривания надо свернуть в соответствующую цилиндрическую поверхность и также рассматривать с определенной точки зрения — искривленные линии при этом должны выглядеть прямыми. Таким образом, кажущиеся искажения на обычной плоскостной или цилиндрической перспективе нужны как раз, чтобы компенсировать другие искажения при рассматривании картины, которые появляются вследствие того, что периферийные ее части мы видим «вкось», в определенном ракурсе (рис. 4). На практике обычны ситуации, когда изображения, построенные под достаточно большими углами зрения (из-за желания «показать побольше»), имеют небольшие размеры и рассматриваются с гораздо большего расстояния, чем следует. Но все же в распространенных случаях, когда угол зрения не превышает 30—40°, обычная линейная перспектива вполне удовлетворительно передает форму и пропорции предметов. По этой схеме работают фотоаппараты, кино- и телекамеры. Аксонометрическая же проекция достоверно передает изображения, пожалуй, лишь для мелких предметов, крупные объекты выглядят при ее использовании как бы «вывернутыми». (Если рассматривать проблему достоверности более глубоко, можно выявить также искажения восприятия второго порядка, связанные с бинокулярностью и аккомодацией зрения, с появлением бликов на картине и т. д. Эти соображения, однако, практически не влияют на методику построения изображений. К тому же адаптационные возможности мозга, привы-

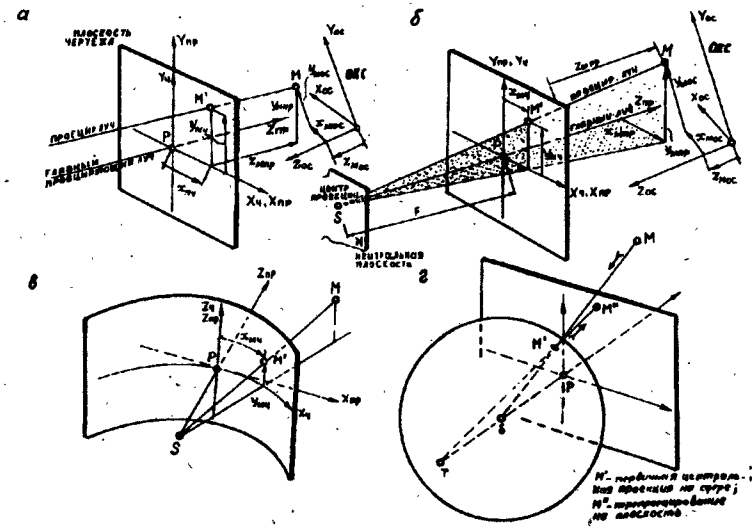


Рис. 3

кание позволяют правильно воспринимать пространство и форму даже по искаженным проекционным изображениям.)

При построении аксонометрической проекции точки пространственного объекта как бы переносятся на поверхность картины лучами, параллельными некоторому заранее заданному главному лучу. Для ортогональной проекции все эти лучи перпендикулярны картинной плоскости, что дает сравнительно достоверное изображение (ведь смотрим мы на него обычно тоже «отвесно»). Лишь в некоторых случаях («кабинетная» проекция, «военная» перспектива) для упрощения построений и измерений используется косоугольное проецирование. При построении аксонометрии надо также иметь в виду возможность дополнительного изменения масштаба (т. е. подобного преобразования изображения).

При построении центральной (перспективной) проекции точки объекта проецируются лучами, проходящими через точку-фокус (центр проецирования), совпадающую, конечно же, с позицией, с которой мы смотрим на объект. Собственно, так же происходит и построение цилиндрической перспективы, но в ней лучи пересекают не плоскую, а изогнутую картинную поверхность. Чертится же на бумаге или отображается на экране ее развертка.

Известны также специальные «квазисферические» проекции, построение которых связано с центральным проецированием на сферическую поверхность с дальнейшим переносом тем или иным способом на плоскость. Например, конформная стереографическая перспектива получается перепроецированием с помощью пучка лучей, проходящих через точку, диаметрально противоположную точке касания сферы и плоскости. Подобные изображения дают фотоаппараты с объективами «рыбий глаз».

Ручное построение проекций рассматривается во многих учебниках и пособиях. Мы же должны составить вычислительные алгоритмы, работающие с координатами точек объекта и их образами. И если в ручной работе оправданы упрощенные приемы построения, в том числе аксонометрия (как более простая по сравнению с центральной перспективой), то машинная графика позволяет активнее использовать более сложные методы, в том числе специальные.

На схемах (рис. 3) показаны трехмерные основные координатные системы (ОКС), в которых могут быть описаны объекты (т. е. можно вычислить координаты точек объекта в этих системах). На картинной плоскости, которая ассоциируется с поверхностью экрана ПЭВМ или листом бумаги, имеется двумерная координатная система чертежа (КСЧ). Она, в свою очередь, может быть различным образом расположена относительно экрана с его физической («приборной») координатной системой (ФКС) или планшета графопостроителя. Для полноты картины введем еще систему ПРКС — пространственную (трехмерную) систему, две оси которой совпадают с осями КСЧ, а третья проходит перпендикулярно картине. Для аксонометрии эта ось представляет собой главный проецирующий луч, для перспективы — главный луч зрения. В перспективной проекции главный луч всегда перпендикулярен плоскости картины, а для косоугольных аксонометрических — может быть и не перпендикулярным. Можно считать, что либо ОКС, либо ПРКС косоугольна.

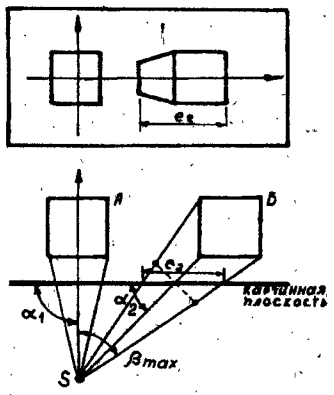


Рис. 4

Итак, зная координаты некоторой точки объекта в ОКС, мы должны вычислить ее координаты в ПРКС, затем перейти к двумерной КСЧ и, если нужно, еще к физической системе устройства. На одной из схем также изображена некоторая часть объекта, описанная в частной координатной системе (ЧКС), которая сама как-то повернута относительно основной системы. В отдельных случаях эта система может быть не декартовой, а, например, цилиндрической или сферической. Тогда в общий процесс пересчета координат надо включить и преобразование из разных типов ЧКС в ОКС. Далее этой первой ступени преобразования мы почти не будем касаться, но на практике она весьма существенна.

Вычисление координат точек изображения

Рассматривая схему аксонометрической проекции, можно видеть, что две первые координаты точки объекта в пространственной системе (ПРКС) равны координатам ее образа в координатной системе (КСЧ). Определение же координат в ПРКС по известным координатам в произвольной системе (когда обе — декартовы, в крайнем случае косоугольные) — типовая для аналитической геометрии задача трехмерного аффинного преобразования, которое описывается известными формулами:

$$x_2=a_{11}*x_1+a_{12}*y_1+a_{13}*z_1+a_1;$$
$$y_2=a_{21}*x_1+a_{22}*y_1+a_{23}*z_1+a_2;$$
$$z_2=a_{31}*x_1+a_{32}*y_1+a_{33}*z_1+a_3.$$

Двенадцать коэффициентов a_{ij} и a_i составляют так называемую расширенную матрицу преобразования. Нередко отдельно рассматривают квадратную матрицу из девяти коэффициентов и вектор переноса a_1, a_2, a_3. Собственно, эти свободные члены (вектор переноса) являются координатами в ПРКС начала основной системы ОКС. В зависимости от величин остальных коэффициентов преобразование может представлять собой перенос, поворот, подобие, симметрию или комбинацию этих вариантов. В общем случае аффинное преобразование позволяет пересчитывать координаты при наличии косоугольных систем с различными масштабами по разным осям (рис. 5). Выделим два распространенных на практике случая: когда обе системы — декартовы прямоугольные с одинаковыми

масштабами по осям (преобразование движения, причем обычно предполагают, что ориентация ортов обеих систем одинакова) и когда к движению добавляется равномерное изменение масштаба (подобие).

Величины a_{ij} при преобразовании движения являются косинусами углов наклона осей первичной системы к разным осям вторичной. При этом выполняются шесть соотношений ортогональности-изометричности. Три из них имеют вид:

$$a_{i1}*x^2+a_{i2}*y^2+a_{i3}*z^2=1 \quad (i=1,2,3),$$

а три других:

$$a_{i1}*a_{k1}+a_{i2}*a_{k2}+a_{i3}*a_{k3}=0 \quad (i\neq k).$$

Кроме того, определитель матрицы (точнее, ее квадратной части) равен единице (для «несобственного» движения — минус единице).

Если преобразование сопровождается масштабированием, вместо единицы в трех первых соотношениях записываются одинаковые величины M^2 (значение масштабного множителя в квадрате), соответственно изменяется и определитель. Для частного случая — тождественного преобразования (когда ОКС и ПРКС совпадают) матрица имеет вид:

1 0 0 0
0 1 0 0
0 0 1 0

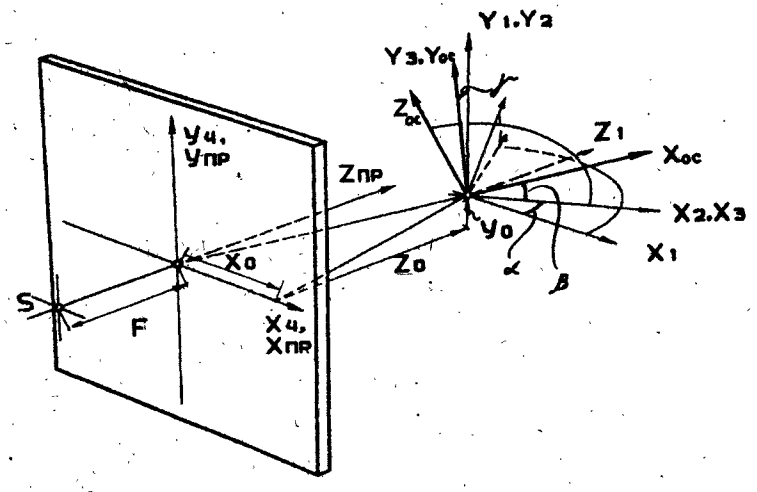


Рис. 5

В простейших вариантах (перенос, подобие, поворот вокруг одной из координатных осей и др.) элементы матрицы определить нетрудно. В других случаях задача усложняется, требуя решения нелинейной системы уравнений. Предпочтительнее использовать специальные алгоритмы формирования матриц, основанные на методике последовательных операций и матричном умножении. Например, преобразование движения в сочетании с подобием можно получить, последовательно применяя перенос, три вращения вокруг разных осей и, наконец, масштабирование. Готовая подпрограмма использует, конечно же, заранее подготовленные уравнения для разных коэффициентов, полученные по правилам умножения матриц. Конкретные формулы изменяются в зависимости от того, в каком порядке и вокруг каких осей предполагаются отдельные повороты. (На рис. 5 показана схема для одного из вариантов.) Введем текущую систему, которая в начальный момент совпадает со вторичной системой (ПРКС). Сначала выполним ее перенос на вектор (X_0,Y_0,Z_0). Его компоненты — координаты начала ОКС в ПРКС. Далее поворачиваем текущую систему на угол α вокруг вертикальной оси y_1, затем на угол γ вокруг уже повернутой оси текущей системы x_2 и наконец на угол β вокруг наклонившейся оси y_2. Такими тремя вращениями можно придать осям текущей системы любое направление (при сохранении, конечно, их взаимной ортогональности). После поворотов все элементы матрицы (включая свободные члены) умножим на масштабный множитель AM.

Согласно специфическому принципу двойственности, тот же результат можно получить, вращая текущую систему вокруг неподвижных осей, параллельных осям ПРКС, но в обратном порядке (на углы β, γ, α). Конкретные формулы можно увидеть в тексте программы на БЕЙСИКе, пока же заметим, что угол α на практике почти всегда равен нулю. При этом вертикальные линии объекта отобразятся как вертикальные и на проекции. Тогда угол β позволяет поворачивать объект направо-налево (или «обходить вокруг него»), а угол γ — наклонять объект «на наблюдателя» или «от него». Вектор переноса позволяет изображать объект в нужном месте чертежа. (Значение переменной Z_0 на аксонометрическую проекцию не влияет. Более того, для построения проекции можно не подсчитывать ее значение в ПРКС, хотя в сложных задачах оно и может потребоваться, например для поиска очерковых линий или контроля видимости.)

Для получения «вида сверху» угол γ задают равным 90°, другие углы нулевые. Для вида сбоку, наоборот, угол β равен +90° или −90° и γ=0.

Выбранное нами расположение координатных осей ПРКС и ОКС несколько необычно, обе системы «левые» (репер образует «левую» тройку). Это сделано при построении универсальной (двух- и трехмерной) графической системы, чтобы сохранить общепринятые направления осей двумерных КСЧ и понятие тождественных преобразований. («Тождественная» аксонометрическая проекция должна строить «главный вид» объекта.) При этом двумерные фигуры по умолчанию будут располагаться во фронтальной плоскости и вычерчиваться так же, как при работе в двумерном режиме.

При необходимости подпрограммы могут быть изменены, например для других исходных направлений координатных систем или иных последовательностей вращения вокруг двух или трех разноименных осей.

Другой способ задания проекции показан на рис. 6. Здесь мы считаем неподвижной «большую» систему ОКС с объектами, а наблюдателя с картиной и аппаратом проецирования — «ходящими» вокруг объекта. Исходные параметры: (X_p,Y_p,Z_p) — координаты начала ПРКС и главной точки картины, заданные в ОКС, (X_S,Y_S,Z_S) — координаты какой-либо точки на отрицательном направлении главного проецирующего луча. Задается также масштабный множитель AM. Этот алгоритм предполагает, что ось $X_{пр}$ проводится в «горизонтальной» плоскости, что соответствует условию α=0 для первого способа задания (вертикальные прямые отображаются вертикальными).

Обратимся теперь к перспективным проекциям. Как видно из схемы, координаты некоторой точки в ПРКС, вычисленные аналогично предыдущему примеру, можно преобразовать в координаты ее образа на картинной плоскости $(X_ч,Y_ч)$, учитывая подобие помеченных на схеме треугольников:

$$X_ч=X_{пр}*F/Z_{пр1};$$
$$Y_ч=Y_{пр}*F/Z_{пр1},$$

где Zпр1= Zпр+F.

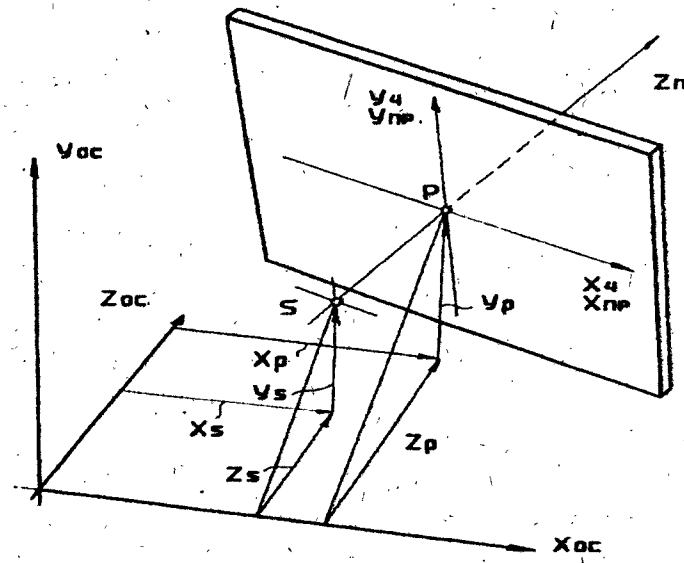


Рис. 6

Здесь $Z_{пр1}$ — значение третьей координаты точки в модифицированной пространственной системе, начало которой перенесено в центр проецирования, F — главное (фокусное) расстояние от главной точки картины до центра проецирования.

Все преобразование, таким образом, разбивается на две подступени: на аффинное и на каноническое дробно-линейное преобразование, единственным параметром которого является фокусное расстояние F.

Задать перспективную проекцию можно теми же параметрами, что и аксонометрическую, но точка S здесь является не просто направляющей, а центром проецирования (точкой зрения), т. е. F равно расстоянию между S и P.

В отличие от аксонометрической проекции, которую можно беспрепятственно «двигать» по экрану или листу путем изменения параметров (X_0,Y_0), при построении перспективной проекции эти величины нужно выбирать так, чтобы главная точка картины (она совпадает с началом ПРКС и КСЧ) размещалась где-то в средней части изображения и экрана — иначе проекция будет «перекошенной», появятся искажения, особенно заметные при увеличении углов зрения. А для того чтобы КСЧ размещалась на экране соответствующим образом, используется дополнительная ступень преобразования, переводящая координаты из КСЧ в физическую систему устройства (в данном случае экрана). Уравнения этого преобразования, обеспечивающие параллельный перенос и масштабирование, просты:

$$X_ф = X_ч * W_m + W_x;$$
$$Y_ф = Y_ч * W_n + W_y.$$

Не будем углубляться в детали построения оптимальных проекций и сопутствующие им проблемы. Отметим только, что для перспективных проекций невозможно отобразить точки, попавшие в нейтральную плоскость (N-плоскость), проходящую через центр проецирования параллельно картинной. А точки объекта, расположенные сзади нее (за спиной наблюдателя), могли бы быть отображены искаженно, «вверх ногами». Поэтому обычно в программах предусматривается их отсечение, а также исключение точек, выходящих за пределы рабочего поля экрана или установленного формата чертежа. При этом программа может останавливаться с выдачей соответствующего сообщения.

Стоит также сказать, что первый из описанных способов задания перспективы при изменении F меняет интенсивность перспективных сокращений. При этом, если уменьшать F, части объекта, расположенные перед картинной плоскостью (в пространственной модели), на чертеже будут увеличиваться, а расположенные дальше — уменьшаться. Параметрами (X_0,Y_0,Z_0) можно удалять или приближать объект, при этом изменяются размеры его изображения и, соответственно, изменяется интенсивность перспективных сокращений.

Дополнительные построения

Вот типичные сопутствующие вопросы, возникающие при построении проекций.

- Отсечение элементов изображения, попавших за нейтральную или специально указанную плоскость, а также выходящих за пределы заданной области изображения.
- Частичный или полный контроль видимости элементов изображения.
- Поиск очерковых (контурных) линий, возникающих на кривых поверхностях при их рассматривании.
- Тонирование (раскраска) элементов изображения.
- Построение собственных и падающих теней, рефлексов, бликов, отражений.
- Дополнительные преобразования координат, позволяющие использовать разные координатные системы (косоугольные, цилиндрические, сферические и др.) для описания отдельных частей объекта, размещать различным образом в пространстве типовые фигуры — как пространственные (параллелепипеды, призмы, конусы), так и плоские (окружности, прямоугольники и др.).
- Больший или меньший ассортимент подпрограмм отображения типовых фигур, пространственных и плоских.

Некоторые из этих вопросов требуют отдельного рассмотрения, поэтому займемся только простейшими случаями.

Первый из них — это преобразования координат, которые выполняются перед построением образа каждой точки (заданной в программе пользователя или сгенерированной подпрограммами отображения типовых фигур) в проекции. Как показано на рис. 7, по предлагаемой методике пользователь задает объекты в различных частных координат-

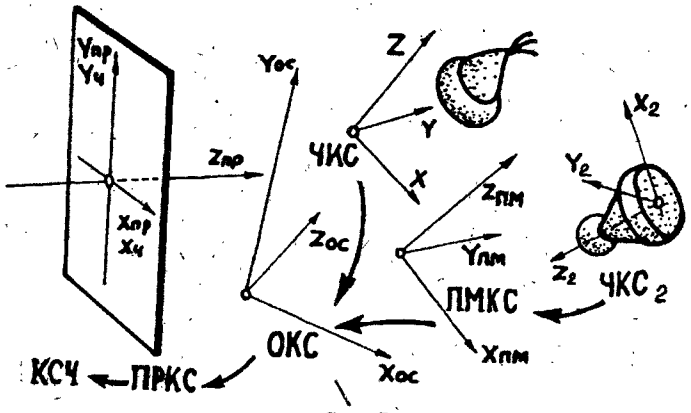


Рис. 7

ных системах (ЧКС), с помощью одного или двух (и даже трех) последовательных преобразований из ЧКС координаты переводятся в ОКС, а затем уже в КСЧ и физическую систему устройства. Преобразования разных ступеней в программе могут «подключаться» и «отключаться» условными операторами или переменными—индексами. Значения индексов одновременно задают и типы преобразований (из числа допустимых в системе или выполняемых по алгоритмам пользователя). В упрощенном варианте предусмотрены так называемая предварительная ступень преобразования с ограниченными функциями (смена координатных осей) и трехмерные аффинные преобразования либо преобразования по подпрограмме пользователя. На рис. 8 показано, что предварительная смена осей может отсутствовать или можно взаимно менять оси Y и X либо X и Z. Таким образом, не затрагивая первой ступени, можно размещать плоские фигуры в трех семействах координатных плоскостей: «фронтальных», «профильных» и «горизонтальных». Используется это и для задания «анизотропных» пространственных фигур, таких, как конусы или призмы.

Трехмерные аффинные преобразования первой ступени выполняются по тем же формулам, которые были приведены ранее, но коэффициенты (матрицы) для них подготавливаются отдельно, для чего используются такие операторы (подпрограммы), как перенос, поворот, масштабирование и др. На этом уровне можно размещать и плоские фигуры в наклонных плоскостях общего положения (по отношению к ОКС), а также объемные фигуры.

Библиотека подпрограмм для вычерчивания типовых фигур составляется в зависимости от потребностей пользователей, но обычно начиная с простейших (параллелепипеды, призмы, конусы и цилиндры, пирамиды, кривые поверхности некоторых типов, а также двумерные фигуры). Объемные тела для простоты обычно задаются для частного положения (например, когда ребра параллелепипедов параллельны координатным осям), остальное достигается с помощью преобразований.

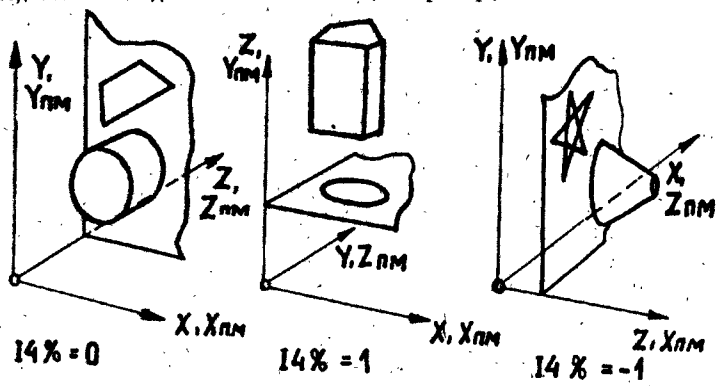


Рис. 8

Программы построения проекций и преобразования координат на БЕЙСИКе

Рассмотрим ряд подпрограмм*, позволяющих строить аксонометрические и перспективные проекции, которые могут служить основой мини-системы трехмерной графики. Читатель может добавлять свои подпрограммы, однако нельзя изменять системные переменные и рабочие величины.

Инициализирующая подпрограмма. Первой должна вызываться инициализирующая подпрограмма, которая, в частности, задает значения рабочих переменных, начальное состояние матриц преобразований и устанавливает положение координатной системы чертежа на экране, вычисляя значения WX,WY,WM,WN. Для этого перед вызовом подпрограммы в переменных X, Y и XX задаются (в системе чертежа КСЧ) значения координат точки, отображаемой в левом нижнем углу рабочего поля экрана, и первая координата какой-либо точки, отображаемой у правого края (запуск со строки 5000). Если вызвать подпрограмму со строки 4999, значения этих переменных прочитаются из оператора DATA. (Заметим, что оператор DIM при неоднократном вызове подпрограммы должен быть перенесен в основную программу. При необходимости в подпрограмму может быть добавлен оператор CLS.)

```
4999 READ X, Y, XX
5000 DIM AM!(12%),AO!(12%),AP!(13%)
5010 WM=256/(XX-X): WN=-WM/.7774: WX=-X*WM: WY=240-Y*WN
5014 RESTORE 5015
5015 DATA .01745,.1,1,0,0,0,0,0,1,0,0,0,0,1,1,1,1,1,1,1,10
5016 READ WU,WV,WW,I0%,I4%,I5%,I6%,I8%,I9%,XP,YP,ZP,Z,C%
5018 FOR I%=1% TO 12% : AM!(I%)=0% : AO!(I%)=0% : AP!(I%)=0% : NEXT
5020 READ AM!(1), AM!(6), AM!(11), AP!(1), AP!(6), KX%, KM%
5030 RETURN
```

Подпрограмма вычисляет коэффициенты преобразования из КСЧ в ФКС, причем WN определяется исходя из пропорций экрана и его размеров в пикселах. Для корректировки масштабов по осям КСЧ величину .7774 можно изменить. Подпрограмма также формирует матрицу тождественного аффинного преобразования первой ступени в массиве AM! и матрицу для построения главного вида как частного случая аксонометрии в массиве AP!. Приведем назначение «системных констант» и индексов-указателей.

- WU — коэффициент перевода из градусной меры в радианную (если его значение равно 1, углы для всех подпрограмм можно задавать в радианах).
- WV — приращение (в радианах) углового или «фазового» параметра для разбиения дуг окружностей, эллипсов и других кривых на

* Все приведенные далее программы соответствуют вильнюсской версии БЕЙСИКа БК-0010.01, за исключением записи нескольких операторов в одной строке. При вводе программ в БК нужно, соответственно, разбить разделенные двоеточиями операторы на несколько программных строк с их перенумерацией. — *Прим. ред.*

отрезки (при уменьшении этого значения возрастает точность черчения, но снижается скорость работы).

- WW — «линейное» приращение параметра для некоторых подпрограмм.
- I0% — признак «поднятия пера» (если он равен 0 — следующее заданное перемещение будет «холостым», иначе прочерчивается отрезок). Многие подпрограммы вычерчивания типовых фигур сами изменяют значение этого индекса.
- I4%, I5% и I6% — управляют преобразованиями (предварительной сменой осей, преобразованием из ЧКС или промежуточной КС в ОКС и построением проекций). При нулевых значениях, которые задаются при инициализации, преобразования отключены.
- I8% — подключает режим запоминания координат генерируемых точек в специальном массиве и, если нужно, периодическое переписывание последнего на магнитную ленту.
- I9% — позволяет отключить режим вывода на экран (при этом может продолжаться запоминание точек). При инициализации устанавливается единичное значение этого индекса, кроме того, обнуляются значения некоторых координат.
- KX% и KM% — текущий счетчик в массиве XY!, в котором производится запоминание точек, и его максимально допустимое значение. (Сам массив может иметь различную длину и задается, если нужно, в главной программе. Там же, после вызова инициирующей подпрограммы, перезадается значение KM.)
- C% — определяет цвет (или яркость) выводимых на экран элементов.

Преобразования и вывод на экран точек и отрезков (из которых слагаются все другие фигуры) осуществляются главным системным модулем преобразований и отработки (условное имя LX). Основная точка входа (со строки 5100) предусматривает задание координат конца выводимого отрезка или холостого хода в переменных X, Y, Z, а также нужного значения индекса «пера» I0%. Если Z не задано, сохраняется его старое значение, что соответствует вычерчиванию фигур в плоскости z=const.

Дополнительная точка входа (со строки 5099) предусматривает чтение трех координат и индекса «пера» из оператора DATA.

```
5099 READ X, Y, Z, I0%                    дополнительная точка входа
5100 W1=X-XP:W2=Y-YP:W3=Z-ZP
                        'вычисление компонент вектора перемещения
XP = X: YP = Y: ZP = Z
IF I4% <> 0 GOTO 5150                          'задана смена координат?
5110 IF I5% > 0 GOTO 5140   'задано преобразование 1-й ступени?
5111 XS = X: YS = Y: ZS = Z             'пересылка координат в ОКС
5115 IF I6% > 0 GOTO 5160             'задано построение проекции?
5116 XR = XS: YR = YS: ZR = ZS
                                  'пересылка координат в ПРКС,КСЧ
5120 XX%=WX+XR*WM: YY%=WY+YR*WN     'преобразование из КСЧ в ФКС
5122 IF I9% = 0 GOTO 5130                   'надо выводить на экран?
5124 IF I0% = 0 GOTO 5128                                'холостой ход?
5125 LINE -(XX%,YY%),C%: GOTO 5130                    'вывод отрезка
```



```
5128 PSET (XX%, YY%), C%                  'вывод точки (холостой ход)
5130 IF I8% = 0 GOTO 5190                 'надо запоминать координаты?
5132 XY!(KX%)=XX%: XY!(KX%+1)=YY%: KX%=KX%+2          'запоминание
IF KX% < KM% GOTO 5190                                    'проверка
PRINT "KX%>KM%": STOP                                     'останов
5140 IF I5% > 1 GOTO 5196              'тип преобразования 1 ступени?
XS=AM!(1%)*X+AM!(2%)*Y+AM!(3%)*Z+AM!(4%)
                                              'аффинное преобразование
YS=AM!(5%)*X+AM!(6%)*Y+AM!(7%)*Z+AM!(8%)
ZS=AM!(9%)*X+AM!(10%)*Y+AM!(11%)*Z+AM!(12%)
GOTO 5115
5150 VX = Z: IF I4% = -1 GOTO 5155          'блок смены координат
Z = Y: Y = VX: GOTO 5110
5155 Z = X: X = VX: GOTO 5110
5160 XR=AP!(1%)*XS+AP!(2%)*YS+AP!(3%)*ZS+AP!(4%)
                                        'преобразование 2-й ступени
YR=AP!(5%)*XS+AP!(6%)*YS+AP!(7%)*ZS+AP!(8%)
IF I6% = 3 GOTO 5120
ZR=AP!(9%)*XS+AP!(10%)*YS+AP!(11%)*ZS+AP!(12%)
ZQ=ZR+AP!(13%)                              '3-я координата в ПРКС1
IF I6% = 1 GOTO 5120
IF ZQ <= 0 THEN STOP                    'контроль на положение точки
XR=XR*AP!(13%)/ZQ: YR=YR*AP!(13%)/ZQ                'для перспективы
GOTO 5120
5190 X=XP: Y=YP: Z=ZP: RETURN
                                   'восстановление координат и выход
5196 STOP                          'резерв для преобразований пользователя
```

Алгоритм подпрограммы можно проследить по приведенным комментариям. Переменные X,Y,Z (и XP,YP,ZP) — координаты точки в частной координатной системе (ЧКС), XS,YS,ZS — в основной, XR,YR — координаты в системе чертежа (КСЧ) и (для аксонометрии) в пространственной системе, связанной с чертежом (ПРКС). ZR — третья координата в ПРКС. ZQ — в модифицированной ПРКС1.

При смене координатных осей и I4%=1% взаимно меняются оси Y и Z, при I4%=−1% — оси X и Z. При I5%=1% производится трехмерное аффинное преобразование, коэффициенты которого берутся из массива (рабочего поля) AM!. Если I5%>1%, то производится переход к строке 5196, где пользователь может записать свои формулы преобразований координат X, Y, Z в значения XS, YS, ZS, после чего управление надо передать строке 5115.

При I6%=1% или I6%=3% во второй ступени выполняется аксонометрическое преобразование по матрице AP!. В первом случае вычисляются все три, а во втором — только две первые координаты в ПРКС (они же — координаты в КСЧ). При I6%=2% строится перспективная проекция.

Если при построении перспективной проекции ZQ≤0 (точка в нейтральной плоскости или за ней), программа останавливается. Можно, однако, вместо оператора STOP записать переход к строке 5190, блокировав графический вывод (правда, при этом будут возникать отдельные погрешности чертежа). Программа может также прерываться по переполнению при близком расположении точек к нейтральной плоскости или

при излишне больших масштабах, так как целочисленные значения в переменных XX% и YY% не могут превышать 32767.

В показанном варианте, если I8%≠0, то будут запоминаться координаты в КСЧ. Можно осуществлять запоминание и в других системах. Кроме координат можно сохранять и значение индекса «пера» I0%. Если для конкретной программы запоминание вообще не нужно, соответствующие строки из подпрограммы (начиная с 5130) целесообразно убрать, оставив переход к строке 5190.

По мере надобности могут, конечно, изыматься или добавляться другие программные блоки, в чем и состоит преимущество открытых программных систем перед готовыми «черными ящиками». Так, в «большой» системе для мощных ЭВМ одновременно может быть доступен набор преобразований различных типов (двумерных и трехмерных), а также контроль поля чертежа и нейтральной плоскости с «правильным» отсечением элементов изображения. Возможна также программная генерация штриховых и штрихпунктирных линий, в том числе утолщенных, предварительная линейная интерполяция, нужная, в частности, при построении упоминавшихся выше криволинейных проекций, и т. д.

Описанная подпрограмма выполняет холостой ход или вычерчивает отрезок от отработанной (текущей) точки. Несложно составить подпрограмму **черчения отрезка, заданного полностью, с «автоматическим» включением и выключением «пера»**:

```
5199 READ X, Y, Z, XK, YK, ZK
5200 GOSUB 5100: I0% = 1: X = XK: Y = YK: Z = ZK: GOSUB 5100
I0% = 0: RETURN
```

В качестве примера приведем также подпрограмму **вычерчивания параллелепипеда частного положения** (с ребрами, параллельными координатным осям ЧКС). В переменных X,Y,Z задаются координаты одной из вершин, в XK, YK, ZK — длины трех ребер, отходящих от этой вершины (в том числе со знаками).

```
5499 READ X, Y, Z, XK, YK, ZK
5500 GOSUB 5550: Z = Z + ZK: GOSUB 5550: I0% = 0: Z = Z - ZK
X = X + XK: GOSUB 5580: Y = Y + YK: GOSUB 5580: X = X - XK
GOSUB 5580: Y = Y - YK: RETURN
5550 GOSUB 5100: X = X + XK: I0% = 1: GOSUB 5100: Y = Y + YK
GOSUB 5100: X = X - XK: GOSUB 5100: Y = Y - YK: GOSUB 5100
RETURN
5580 GOSUB 5100: I0% = 1: Z = Z + ZK: GOSUB 5100: I0% = 0
Z = Z - ZK: RETURN
```

Еще один пример — **вычерчивание конуса, цилиндра, правильной пирамиды или призмы**. Ось тела параллельна оси Z ЧКС, координаты центра первого основания — XC, YC и ZC, радиусы первого и второго оснований (для призм и пирамид имеется в виду описанный опорный цилиндр или конус) — R и R1, высота фигуры (в том числе со знаком) — H.

Для конуса или цилиндра чертится набор образующих (их число зависит от значения WV). Для призмы или пирамиды дополнительно

задаются AL — угол наклона радиуса, проведенного в одном из оснований в первую отрабатываемую вершину, и NT% — число сторон одного из оснований.

```
5599 READ XC,YC,ZC,R,R1,H              'доп.точка входа для конуса
5600 AL=0: NT%=PI/WV: GOTO 5612          'точка входа для конуса
5610 READ XC,YC,ZC,R,R1,H,AL,NT% 'доп.точка входа для пирамиды
5612 DA=PI/NT%                           'точка входа для пирамиды
Z=ZC: R2=R: FOR K%=1% TO 2%: A=0
5630 BT=AL*WU+A: GOSUB 5670: I0%=1%: A=A+DA
IF A<PI+PI+.001 GOTO 5630
5640 I0%=0%: R2=R1: Z=Z+H: NEXT
5650 A=0: DA=DA+DA
5652 BT=AL*WU+A: Z=ZC: R2=R: GOSUB 5670: I0%=1%
Z=Z+H: R2=R1: GOSUB 5670: I0%=0%: A=A+DA
IF A<PI+PI-.001 GOTO 5652
RETURN
5670 X=XC+R2*COS(BT): Y=YC+R2*SIN(BT): GOSUB 5100: RETURN
```

Теперь необходимо остановиться на **формировании матриц преобразований**. Описанный ранее **первый способ задания аксонометрической или перспективной проекции** (в нашем варианте они объединены) реализуется следующей программой:

```
7300 GOSUB 7380                          'точка входа п/пр AKS
AP!(1%)=V1*V3-V2*V4*V5: AP!(2%)=V2*V6
AP!(3%)=-V1*V4-V2*V3*V5: AP!(4%)=X
AP!(5%)=V4*V6: AP!(6%)=V5: AP!(7%)=V3*V6: AP!(8%) = Y
AP!(9%)=V1*V4+V2*V3: AP!(10%)=-V1*V6
AP!(11%)=V1*V3-V2*V4: AP!(12%)=Z: AP!(13%)=F
FOR I%=1% TO 13%: AP!(I%)=AP!(I%)*AM: NEXT
RETURN
7380 V1=COS(AL*WU): V2=SIN(AL*WU)     ' внутренняя подпрограмма
V3=COS(BT*WU): V4=SIN(BT*WU)
V5=COS(GM*WU): V6=SIN(GM*WU)
RETURN
```

Исходные данные: X,Y,Z — координаты начала ОКС, заданные в ПРКС, AL,BT,GM — углы трех поворотов, описанные относительно ее начала. Заметим, что преобразование из ЧКС в ОКС после этого не обязательно будет преобразованием подобия.

Второй способ реализуется подпрограммой:

```
7430 DX=XS-XP: DY=YS-YP: DZ=ZS-ZP: V1=DX*DX+DZ*DZ
R1=SQR(V1): R2=SQR(V1+DY*DY): V3=DY/R2: V4=R1/R2
V5=-DX/R1: V6=-DZ/R1
AP!(1%)=V6 : AP!(2%)=0 : AP!(3%)=-V5
AP!(5%)=V3*V5: AP!(6%)=V4: AP!(7%)=V3*V6
AP!(9%)=V4*V5: AP!(10%)=-V3 : AP!(11%)=V4*V6: AP!(13%)=R2
FOR K%=1% TO 9% STEP 4%
AP!(K%+3%)=-XP*AP!(K%)-YP*AP!(K%+1%)-ZP*AP!(K%+2%): NEXT
FOR K%=1% TO 13% :AP!(K%)=AP!(K%)*AM: NEXT
RETURN
```

Здесь XP,YP,ZP — координаты главной точки картины и начала ПРКС в ОКС, XS,YS,ZS — координаты центра проецирования или направляющей точки на главном проецирующем луче, AM — масштабный множитель.

Наконец, приведем **три программы для управления преобразованиями первой ступени**. Они реализуют **операции параллельного переноса, масштабирования и вращения в разных модификациях**.

Оператор переноса изменяет матрицу текущего (предыдущего) преобразования так, что ЧКС без изменения направления осей и их масштабов перемещается параллельно самой себе. В частности, если предыдущее преобразование было тождественным или переносом из ЧКС в ОКС, переносом будет и новое преобразование. Однако его можно задать несколькими способами, так что подпрограмма фактически распадается на три. Для первого варианта, начинающегося со строки 7000, компоненты вектора переноса задаются в ОКС (переменные X,Y,Z). Для второго (со строки 7010) — вектор задается в ЧКС. В третьем случае (строка 7020) в тех же переменных задаются координаты начала ЧКС в ОКС.

```
7000 AP!(4%)=AP!(4%)+X: AP!(8%)=AP!(8%)+Y: AP!(12%)=AP!(12%)+Z
RETURN
7010 FOR K%=1% TO 9% STEP 4%
AM!(K%+3%)=AM!(K%)*X+AM!(K%+1%)*Y+AM!(K%+2%)*Z+AM!(K%+3%)
NEXT : RETURN
7020 AM!(4%)=X : AM!(8%)=Y : AM!(12%)=Z : RETURN
```

Масштабирование с множителями, равными 1 и −1, используется для построения симметричных частей объекта. Правда, для этого оси ЧКС должны быть взаимно ортогональны (иначе получится «квазисимметрия»). Второй вариант подразумевает как бы изменение масштаба по осям ОКС, вернее, поскольку мы полагаем ОКС неизменной, — «растяжение» или «сжатие» первичного пространства в направлении осей ОКС. При этом начало ЧКС и направления ее осей могут изменяться. Для обоих вариантов исходные данные размещаются в переменных X,Y,Z.

```
7100 J%=4%: J1%=1%: J2%=9%: GOTO 7110
7104 J%=1%: J1%=4%: J2%=4%
7110 FOR I%=1% TO J2% STEP J%
AM!(I%)=X*AM!(I%)
AM!(I%+J1%)=Y*AM!(I%+J1%)
AM!(I%+J1%*2%)=Z*AM!(I%+J1%*2%) : NEXT : RETURN
```

Для различных вариантов **вращений** можно привести подпрограмму **трехосного поворота**, которая располагает оси ЧКС с сохранением их взаимной ортогональности и единиц измерения под заданными углами. В отличие от поворотов в аксонометрии (где одна ось использовалась дважды) здесь вращения производятся последовательно на углы AL, BT, GM вокруг осей X, Y, Z текущей системы. Можно задать отличным от нуля один из трех углов и получить «одноосный поворот» вокруг любой из координатных осей. Начало ЧКС остается на прежнем месте, поэтому его нередко используют совместно с оператором переноса.

Листинг похож на текст подпрограммы задания аксонометрий:

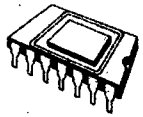

```
7200 GOSUB 7300                'см. внутреннюю п/пр для п/пр AKS
AM!(1%)=V3*V5 : AM!(2%)=-V3*V6 : AM!(3%)=-V4
AM!(5%)=V1*V6-V2*V4*V5: AM!(6%)=V1*V5+V2*V4*V6
AM!(7%)=-V2*V3: AM!(9%)=V2*V6+V1*V4*V5
AM!(10%)=V2*V5-V1*V4*V6: AM!(11%)=V1*V3 : RETURN
```

Таким образом, задавая преобразование первой ступени или проекцию, надо указать тип преобразования (значение индекса) и сформировать нужную матрицу. Для первой ступени типично периодическое изменение матрицы в процессе воспроизведения последовательных геометрических элементов или частей объекта.

Для пользователя БК-0010 основная проблема состоит в размещении подобных подпрограмм и основной программы в ОЗУ. При этом полезны специальные приемы экономии памяти, в том числе сокращение области символьных переменных и сдвиг части таблицы адресов в область экранного ОЗУ, а также заимствование части экранной памяти и работа с автономизированным «объектным кодом».

Иногда программу можно разделить на части, записывая промежуточные данные на магнитную ленту. Решение подобных задач облегчается при подключении дисковода или при переходе на более мощную ПЭВМ, например БК-0011(М).

HARD & SOFT



По многочисленным просьбам читателей публикуем материал о том, как изготовить самодельный «электронный ROM-диск» для БК-0010.01, обеспечивающий хранение и быструю загрузку пользовательских программ, в том числе на языке БЕЙСИК.

И. В. Канивец,

Санкт-Петербург

Модуль хранения программ для БК-0010.01

Бытовые компьютеры БК-0010.01 в силу их относительно низкой стоимости удобно использовать в качестве простых, удобных и надежных контроллеров для управления технологическими процессами, решения постоянного набора задач и обработки различной информации при применении их в составе устройств или приборов [1]. Однако все это справедливо лишь в том случае, если указанные программы не требуют загрузки с магнитной ленты, а хранятся в ПЗУ, так что от пользователя требуется лишь включить питание компьютера, выбрать в меню нужную программу и запустить ее.

Многочисленные практические решения этой задачи, представленные в различных публикациях, имеют, на мой взгляд, существенный недостаток: прошитая в ПЗУ программа пользователя располагается в области адресов, занимаемых, как правило, штатным БЕЙСИКом, тем самым исключается возможность использования программ, написанных на этом языке высокого уровня [2].

Предлагаемый модуль хранения программ не затрагивает аппаратные и программные ресурсы БК-0010.01. Все хранящиеся в нем программы располагаются вне адресного пространства компьютера. А в адресном пространстве (точнее, в области адресов системных регистров) располагается лишь небольшая (11 слов) загрузочная программа («голова загрузчика») и два регистра устройства. Суммарная емкость программ, хранящихся в модуле, зависит от используемых микросхем памяти и не превышает 128 Кб при использовании БИС РПЗУ УФ 27512.

Модуль подключается к разъему системной магистрали БК, электропитание его также осуществляется от компьютера, т. е. никаких переделок (и даже вскрытия корпуса) компьютера не требуется. Конструкция содержит 25 микросхем и другие не дефицитные элементы.

В основе работы модуля лежит принцип, изложенный в [3]. В микросхемах УФ РПЗУ D16, D17 прошиваются не только программы пользователя, но и сервисная программа «тело загрузчика», которая загружается в ОЗУ БК в первую очередь при передаче управления модулю хранения программ. «Тело загрузчика» выводит на экран список хранящихся в модуле прикладных программ, позволяя загрузить в ОЗУ любую из них и запустить.

Для чтения из УФ РПЗУ 16-разрядного слова необходимо загрузить его адрес в программно доступный регистр-счетчик адреса, собранный на микросхемах D12—D15. Обращение к нему осуществляется по адресу Аадр=177670, принадлежащему области системных регистров. Чтение данных из соответствующей ячейки памяти модуля осуществляется при обращении процессора к регистру данных по адресу Адан=177674. После окончания цикла чтения происходит автоматическое увеличение содержимого регистра-счетчика адреса на 1, т. е. через регистр данных становится доступным следующее слово памяти модуля без отдельного занесения его адреса в регистр адреса. Таким образом, обращаясь всего лишь к двум адресам в области системных регистров БК, можно прочитать информацию, прошитую в любой ячейке УФ РПЗУ.

Для хранения программы «голова загрузчика» длиной 22 байта используется ППЗУ К155РЕ3 (D18, D19), занимающее в адресном пространстве БК участок 177600—177624. Регистр D20 служит для хранения адреса при обращении к ячейкам данного ППЗУ. Резисторные сборки D22, D23 совместно с шинными драйверами D1—D4 типа К589АП26 образуют шину для чтения данных из УФ РПЗУ и ППЗУ в БК.

Микросхемы D5, D6, D7, D10 служат для дешифрации требуемых адресов. Микросхемы D8, D9, D11 и элементы микросхем D6, D10, D21, D24, D25 предназначены для формирования сигналов чтения из ППЗУ и УФ РПЗУ, а также для выполнения циклов чтения-записи на магистрали БК.

Для того чтобы в модуле допускалось использование микросхем УФ РПЗУ различной емкости, предусмотрена возможность коммутации их выводов перемычками в соответствии с табл. 1. Микросхемы ППЗУ D18, D19 с прошитой «головой загрузчика» впаиваются в плату, а микросхемы УФ РПЗУ D16, D17 устанавливаются на панельки.

На некоторых экземплярах БК для правильной и устойчивой загрузки программ требуется установка конденсаторов C1, C2, резисторов R1, R2 и диодов D1, D2.

Таблица 1

Тип микросхемы УФ РПЗУ	Положение перемычек
2764	К1—К5, К3—К6
27128	К1—К5, К3—К6
27256	К2—К5, К3—К6
27512	К2—К5, К4—К6

Перечень радиоэлементов, используемых в схеме, приведен в табл. 2.

Таблица 2

D1—D4	К589АП26
D5	К555ЛА2
D6	К555ЛИ6
D7	К555ИД7
D8, D9	К555ТМ2
D10	К555ЛИ3
D11	К555ЛА3
D12—D15	К555ИЕ7
D16—D17	i2764, i27128, i27256, i27512
D18, D19	К155РЕ3
D20	К555ТМ9
D21	К555ЛА8
D22, D23	МР1-4-9 680 Ом
D24	К555ЛН1
D25	К555ТЛ2
C1, C2	К10-17-0-5,6Н, подбирается от 0 до 5600 пф
C3, C4	К53-1 100 мкФ, 10%, 6В
C5—C22	К10-17, 0.1 мкФ
R1, R2	МЛТ-0,125, подбирается от 0 до 150 Ом, 5%
D1, D2	КД503А (устанавливается только совместно с R1, R2, C1, C2)
X1	СНП58-64/95х9Р-20-2

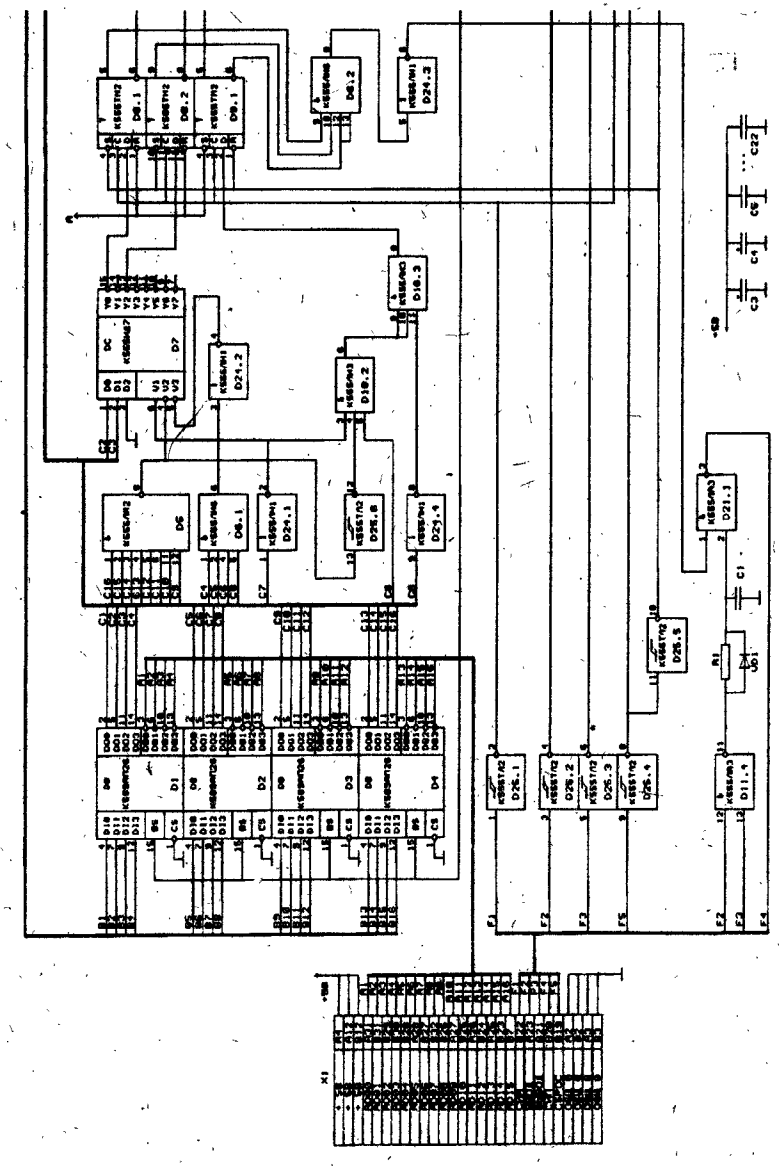
Программная поддержка

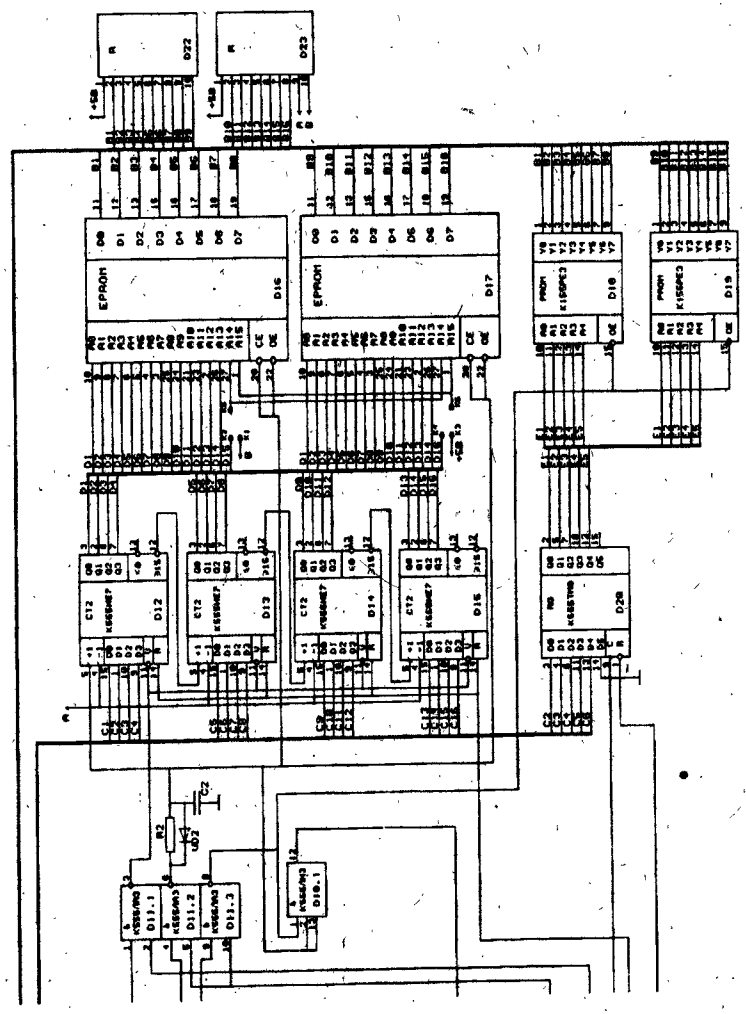
Как уже отмечалось выше, для запуска какой-либо программы, хранящейся в УФ РПЗУ, используется начальный загрузчик, состоящий из двух частей.

- *«Голова загрузчика»*. Прошивается в отдельном ППЗУ, отображаемом на неиспользуемую область адресов системных регистров БК (начиная с адреса 177600), и запускается из монитора командой S177600.

Функция «головы загрузчика» сводится к запуску программы, прошитой в микросхемах УФ РПЗУ с нулевого адреса. При этом считается, что в нулевом слове содержится начальный адрес загрузки этой программы в память БК, в первом — ее длина в байтах (обязательно четное число), а со второго слова следуют непосредственно коды программы.

Если в УФ РПЗУ предполагается хранить только одну программу пользователя (в машинных кодах), то она размещается, как описано выше. Если же в УФ РПЗУ необходимо хранить несколько различных программ, то с нулевого адреса должна располагаться сервисная программа «тело загрузчика».





Текст авторского варианта программы «голова загрузчика» вместе с кодами прошивки микросхем ППЗУ приведен в табл. 3.

Таблица 3

Адрес	Мнемоника	Ст.байт (D19)	Мл.байт (D18)
177600	MOV #177674,R5	025	305
177602		377	274
177604	CLR @#177676	012	037
177606		377	276
177610	MOV (R5),R1	023	101
177612	MOV R1,R2	020	102
177614	MOV (R5),R0	023	100
177616	ASR R0	006	200
177620	MOV (R5),(R1)+	023	121
177622	SOB R0,177620	176	002
177624	MOV R2,R7	020	207

- *«Тело загрузчика»*. Служит для вывода на экран списка хранящихся в УФ РПЗУ программ, позволяет выбрать и запустить любую из них. Для обеспечения корректной работы «тела загрузчика» информация в УФ РПЗУ должна быть прошита, как указано на рисунке внизу.

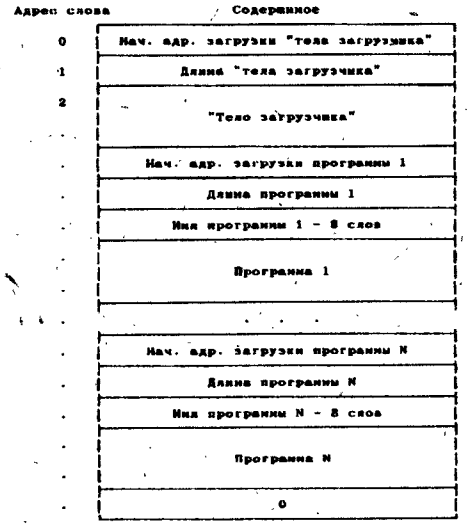


Схема распределения адресного пространства в прошивке УФ РПЗУ

Программы записываются друг за другом «по цепочке». Перед текстом каждой из них содержится служебная информация: адрес загрузки в память БК (2 байта), длина в байтах (2 байта), имя программы (16 символов, или 8 двухбайтных слов). За последней программой должно следовать по крайней мере одно нулевое слово — ограничитель.

Программа, размещающаяся в УФ РПЗУ первой и загружаемая с помощью «головы загрузчика» (в данном случае это — «тело загрузчика»), не содержит информации о своем имени.

Коды программ, предназначенных для прошивки в ППЗУ («голова

загрузчика») и в УФ РПЗУ, необходимо разделить на два массива (старшие и младшие байты), причем каждый массив отдельно прошивается в своей микросхеме.

Ниже приводится примерный текст программы «тело загрузчика», написанный на ассемблере МИКРО-11 или ему подобном. Программа перемещаема, может размещаться начиная с любого адреса и позволяет загружать из РПЗУ и запускать программы, написанные в машинных кодах или на языке БЕЙСИК.

Для БЕЙСИК-программы в РПЗУ прошивается листинг в формате записи на ленту командой CSAVE. Имя БЕЙСИК-программы, прошиваемое в РПЗУ, в позициях 15 и 16 должно содержать латинские заглавные символы BS.

После запуска «тело загрузчика» просматривает список прошитых в РПЗУ программ, выводит их имена на экран и запоминает адрес начала каждой программы в РПЗУ в таблице TABLE. Далее, используя клавиши «ВВЕРХ» и «ВНИЗ», можно выбрать требуемую программу и нажать «ВВОД».

Для программ в машинных кодах в последней строке экрана выводится прошитый в РПЗУ адрес загрузки в память БК. Если необходимо, его можно изменить, набрав требуемое значение и нажав «ВВОД». Для стирания последней введенной цифры используется клавиша «ЗАБОЙ». Если нажать «ВВОД», не вводя новый адрес загрузки, то программа будет загружена и запущена по прошитому в РПЗУ адресу.

Для БЕЙСИК-программ осуществляется следующий порядок действий. Драйвер магнитофона EMT 36 временно (на одно обращение) подменяется драйвером обслуживания модуля хранения программ. В «буфер клавиатуры» (используется возможность обработки программируемых ключей, предусмотренная в клавиатурном драйвере) заносится строка CLOAD"VM", после чего управление передается БЕЙСИК-системе. Драйвер клавиатуры эмулирует набор команды CLOAD"VM", и происходит обращение к драйверу обслуживания модуля хранения программ, который пересылает необходимый программный блок из РПЗУ в ОЗУ компьютера, эмулируя загрузку с магнитофона.

Примерный текст программы «тело загрузчика».

```
ADR  = 177670        ; Регистр адреса
DATA = 177674        ; Регистр данных

BEG  = 346
LEN  = 350
APZU = 376

MOV  #1000,SP        ; Начальный сброс
EMT  14              ; и установка режима экрана
MOV  #INF,R1
CLR  R2
EMT  20

MOV  #1,R5
```



```
     MOV  R5,ADR

     MOV  DATA,R0        ; Получить в R5 адрес начала
     ASR  R0             ;   информации о втором файле,
     ADD  R0,R5          ;   прошитом в РПЗУ
     INC  R5

     MOV  #10,R1         ; Координата X на экране
     MOV  #3,R2          ; Координата Y на экране

     CLR  R3

; -------- Вывод на экран имен прошитых в РПЗУ файлов --------

M2:  MOV  R5,ADR
     MOV  DATA,R0        ; Адрес загрузки текущего файла
     BEQ  CASE           ; 0 - конец цепочки файлов

     MOV  R5,TABL(R3)    ; Запомнить адрес начала текущего
     TST  (R3)+          ;   файла в РПЗУ

     MOV  DATA,R0        ; Длина текущего файла
     ASR  R0

     ADD  R0,R5          ; Адрес начала информации о
     ADD  #12,R5         ;   следующем файле в РПЗУ

     MOV  #10,R4         ; Длина имени файла (в словах)
     EMT  24             ; Установить позицию курсора

M1:  MOV DATA,R0         ; Вывод на экран имени очередного
     EMT  16             ;   файла
     SWAB R0
     EMT  16
     SOB  R4,M1

     INC  R2
     BR   M2

CASE: ASR  R3            ; Общее количество прошитых файлов

     MOV  #1,R2          ; Выбор первого файла
     MOV  #45616,R1

M4:  CALL INV            ; Инверсия имени выбранного файла

M3:  EMT  6
     CMP  R0,#32
     BEQ  UP
     CMP  R0,#33
     BEQ  DOWN
     CMP  R0,#12
     BEQ  ENTER
     BR   M3
```

```
UP:     CMP   R2,#1             ; Нажата клавиша "КУРСОР ВВЕРХ"
        BEQ   M3
        CALL  INV
        DEC   R2
        SUB   #1200,R1
        BR    M4

DOWN:   CMP   R2,R3             ; Нажата клавиша "КУРСОР ВНИЗ".
        BEQ   M3
        CALL  INV
        INC   R2
        ADD   #1200,R1
        BR    M4

INV:    MOV   R1,R0             ; Инвертирование текстовой строки
        MOV   #12,R4            ;   на экране
I2:     MOV   #22,R5
I1:     COM   (R0)+
        SOB   R5,I1
        ADD   #34,R0
        SOB   R4,I2
        RETURN

INF:    .B:233 .B:232 .B:224 .B:236 .B:222 .B:33
        .A:       Выберите программу :
        .B:221 .E

ENTER:  DEC   R2                ; Нужная программа выбрана
        ASL   R2
        MOV   TABL(R2),R5
        MOV   R5,ADR
        ADD   #12,R5
        MOV   R5,APZU           ; Адрес начала программы в РПЗУ

        MOV   DATA,BEG          ; Адрес загрузки программы
        MOV   DATA,LEN          ; Длина программы

        MOV   #352,R0           ; Перенос имени файла в область
        MOV   #7,R1             ;   ОЗУ драйвера магнитофона
M20:    MOV   DATA,(R0)+
        SOB   R1,M20
        MOV   #20040,(R0)+

        CMP   DATA,'BS'         ; Требуемая программа на БЕЙСИКе?
        BEQ   M5


; ------------------------ Не БЕЙСИК ----------------------

        CLR   R1                ; Установка курсора
        MOV   #26,R2
        EMT   24

        MOV   #SADD,R1          ; Вывод текста
```



```
        CLR   R2
        EMT   20

        MOV   BEG,R1            ; Вывод на экран восьмеричного
        ASL   R1                ;   адреса загрузки программы
        MOV   #5,R5             ;   прошитого в РПЗУ
M70:    MOV   #6,R0
        MOV   #3,R2
M68:    ASL   R1
        ROL   R0
        SOB   R2,M68
        EMT   16
        SOB   R5,M70

        MOV   #NADD,R1          ; Вывод текста
        CLR   R2
        EMT   20

        CLR   R1                ; Запрос с клавиатуры
        CLR   R2                ;   нового адреса загрузки
M26:    EMT   6
        CMP   #30,R0            ; Для сохранения прошитого
        BEQ   ZB1               ;  адреса загрузки нажать
        CMP   #12,R0            ;  "ВВОД", ничего больше
        BEQ   KN                ;  не набирая
        CMP   #5,R2
        BEQ   M26
        CMP   #67,R0
        BLO   M26
        CMP   #60,R0
        BHI   M26
        EMT   16
        SUB   #60,R0
        ASL   R1
        ASL   R1
        ASL   R1
        ADD   R0,R1
        INC   R2
        BR    M26

ZB1:    TST   R2                ; Стирание последнего набранного
        BEQ   M26               ;  символа - клавиша "ЗАБОЙ"
        EMT   16
        ASR   R1
        ASR   R1
        ASR   R1
        DEC   R2
        BR    M26

KN:     TST   R1
        BEQ   M30
        MOV   R1,BEG

M30:    MOV   #COD,R0
```

```
        CALL MOVING
        JMP  @#400

SADD:   .B:233
        .A:  Адрес загрузки (
        .B:40 .B:0
NADD:   .A: ) :
        .B:40 .B:232 .E

; ---------------- Обработка EMT36, БЕЙСИК --------------
M5:     MOV  #BAS,R0
        CALL MOVING
        MOV  #412,@#30
        MOV  #12, @#122        ; Длина текста
        MOV  #400,@#124        ; Адрес текста
        JMP  @#100270

MOVING: MOV  #400,R1           ; Подпрограмма пересылки
        MOV  #54 ,R2
M7:     MOV  (R0)+,(R1)+
        SOB  R2,M7
        EMT  14
        RETURN

COD:    MOV  BEG,R0
        MOV  LEN,R1
        ASR  R1
M6:     MOV  DATA,(R0)+
        SOB  R1,M6
        MOV  #100274,-(SP)     ; Адрес возврата в монитор
        MOV  BEG,PC

BAS:    .A:CLOAD"VM"
        .B:12
        MOV  R5,-(SP)
        MOV  2(SP),R5
        MOV  -(R5),R5

        CMP  #104014,R5
        BEQ  M14

        CMP  #104036,R5
        BNE  M10

        MOV  R0,-(SP)
        CLRB @#321             ; Ответ драйвера МГ
        MOV  BEG,R0            ; Адрес массива на чтение
        MOV  LEN,R5            ; Длина массива на чтение
        ASR  R5
        MOV  APZU,ADR
M12:    MOV  DATA,(R0)+
```



```
        SOB  R5,M12

        EMT  4
        MOV  (SP)+,R0
        MOV  #100112,@#30
        BR   M14

M10:    JSR  PC,@174000(R5)    ; Нормальная обработка EMT
M14:    MOV  (SP)+,R5
        RTI

TABL:   .E                     ; Таблица, в которой хранятся
                               ; адреса начала программ в РПЗУ
        END
```

Литература

1. *Полянский П. В., Ширковский Н. А.* «Электроника БК-0010» в системе управления технологическими объектами //Микропроцессорные средства и системы. 1987. N4. С. 33—34.

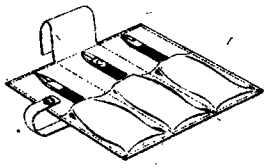
2. *Хабибулин Ю. Д.* Кассета ОЗУ-ПЗУ для БК-0010 //Вычислительная техника и ее применение. 1991. N11. С. 40—42.

3. *Шевкопляс Б. В.* Микропроцессорные структуры. Инженерные решения: Справочник. 2-е изд. М.: Радио и связь, 1990. С. 320.

Окончание статьи Б. Н. Сергеева «Практикум на ассемблере»

Продолжая в том же духе, найдем адреса и для остальных строк. При этом следует помнить, что каждая символьная строка имеет высоту по 10 ТВ строк. На экран выводится 24 строки (не считая служебной), а все рабочее поле экрана соответствует адресам 42000—77777:

Номер строки	Адрес	Номер строки	Адрес	Номер строки	Адрес	Номер строки	Адрес
1	42000	7	51400	13	61000	19	70400
2	43200	8	52600	14	62200	20	71600
3	44400	9	54000	15	63400	21	73000
4	45600	10	55200	16	64600	22	74200
5	47000	11	56400	17	66000	23	75400
6	50200	12	57600	18	67200	24	76600 ... 77777



О возможности повышения быстродействия БК за счет увеличения ее тактовой частоты (турбирования) сказано и написано уже достаточно много. Но в основном это разрозненные сведения, зачастую противоречащие друг другу, поэтому (а особенно «на фоне» слухов о «сожженных» при неумелой доработке процессорах) желающих турбировать собственные ЭВМ находилось немного. Надеемся, что данная статья поможет развеять сомнения и страхи и послужит хорошим практическим руководством для любителей БК.

С. М. Неробеев, А. В. Сорокин,

Москва

Увеличение тактовой частоты (турбирование) БК-0010(.01) и БК-0011(М)

Способы повышения тактовой частоты БК-0010(.01)

Существует два основных метода турбирования БК-0010(.01). Первый позволяет увеличить тактовую частоту вдвое — до 6 МГц, но такая доработка обеспечивает нормальное функционирование не всех экземпляров БК. Второй дает возможность увеличить тактовую частоту только до 4 МГц, но применим на любой БК-0010(.01).

Увеличение тактовой частоты до 6 МГц

Прежде всего следует пояснить, почему работа процессора К1801ВМ1 с тактовой частотой 6 МГц вообще возможна, ведь в документации на процессор в качестве предельного указано значение тактовой частоты 5 МГц. Дело в том, что процессор, как и любая другая микросхема, проектируется с определенными (причем весьма значительными) коэффициентами запаса по всем параметрам, в том числе и по тактовой частоте, что позволяет ему сохранять работоспособность при некотором превышении предельного значения частоты. Кроме того, при производстве микросхем с высокой степенью интеграции, в том числе и микропроцессоров, не удается изготовлять кристаллы с абсолютно идентичными параметрами. (Так, например, фирма Intel устанавливает максимальную тактовую частоту своих процессоров тестированием каждого конкретного экземпляра.) Поэтому существуют отдельные образцы процессора К1801ВМ1, сохраняющие работоспособность на частоте 8 МГц и даже выше.

Увеличение тактовой частоты до 6 МГц позволяет намного ускорить и качественно улучшить работу с диском, дает значительный выигрыш при

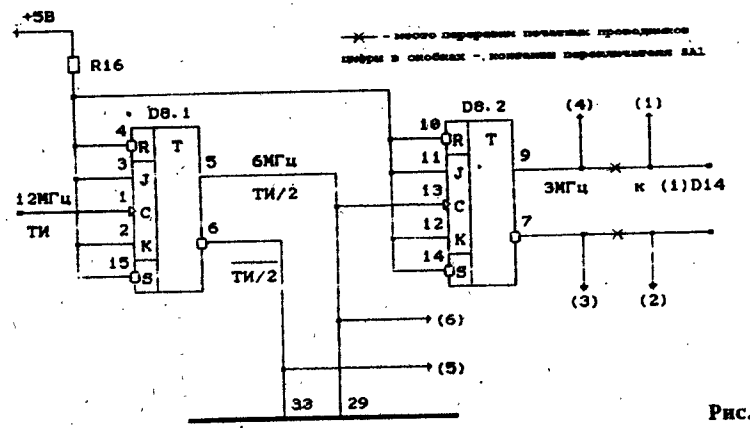


Рис. 1

проведении любых вычислений, обеспечивает намного лучшее воспроизведение оцифрованной музыки (через ЦАП типа Covox). Однако существует определенный круг программ, работа с которыми в турбо-режиме затруднена или нежелательна: ряд игровых программ (попробуйте-ка поиграть в «ТЕТРИС», если скорость падения фигур на наименьшем нулевом классе больше, чем до турбирования была на девятом), музыкальные редакторы и т. п. Кроме того, функционирование некоторых программ на тактовой частоте 6 МГц вообще невозможно (пример — магнитофонные копировщики).

Говоря о повышении тактовой частоты до 6 МГц, необходимо развеять два заблуждения, получивших широкое распространение среди пользователей БК. Первое из них состоит в том, что подобную доработку якобы можно провести только для процессора в керамическом корпусе, но не в пластмассовом. Ошибочность такого убеждения становится ясна, если протестировать процессоры в разных корпусах на заводском испытательном стенде: никаких различий (кроме типа корпуса) между ними нет. Сложности же в проведении доработки возникают вовсе не по вине процессора. Дело в том, что вычислительная машина — это очень сложный комплекс, состоящий из многих электронных узлов, каждый из которых имеет свое быстродействие. Общее же быстродействие вычислительной системы не может быть больше быстродействия самого медленного ее узла. В БК-0010(.01) таким «узким местом» являются микросхемы видеоконтроллера — контроллера ОЗУ* (К1801ВП1-037) и контроллера клавиатуры (К1801ВП1-014). При значительном повышении производительности процессора упомянутые БИС попросту не успевают обрабатывать поступающие от него команды. Согласовать работу этих микросхем

* Напоминаем читателям, что в БК функции регенерации ОЗУ и вывода изображения на дисплей совмещены в одной микросхеме. — *Прим. ред.*

с процессором можно путем подбора задержек соответствующих сигналов, но такая работа достаточно трудоемка и ее нельзя рекомендовать к широкому применению.

Вторым заблуждением является убеждение, что при подаче на процессор тактовой частоты 6 МГц и выше он может выйти из строя («сгореть»). На самом деле, в отличие от превышения питающего или входных напряжений, слишком большая тактовая частота может в крайнем случае вызвать лишь зависание процессора. Однако режим работы на частоте 6 МГц все же является нештатным, и если при 3 МГц гарантируется круглосуточная работа БК, то при столь же длительном функционировании компьютера на повышенной тактовой частоте возможен перегрев процессора и его выход из строя.

Теперь перейдем непосредственно к выполнению операции турбирования БК на 6 МГц. Данная доработка проста и не требует установки дополнительных микросхем. Одна из первых публикаций на эту тему появилась еще шесть лет назад — имеется в виду статья Д. Антонова «Увеличение тактовой частоты (турбо-режим)» в журнале «Информатика и образование», №2 за 1990 г., с. 46. Но, к сожалению, из-за допущенной там ошибки компьютер, турбированный по приведенной в «Информатике и образовании» схеме, работать не будет. В отличие от нее, приведенная ниже доработка свободна от ошибок и опробована на множестве экземпляров вычислительных машин (как БК-0010, так и БК-0010.01). Исходя из накопленного опыта, можно сделать вывод, что с тактовой частотой 6 МГц способна нормально функционировать каждая третья ЭВМ.

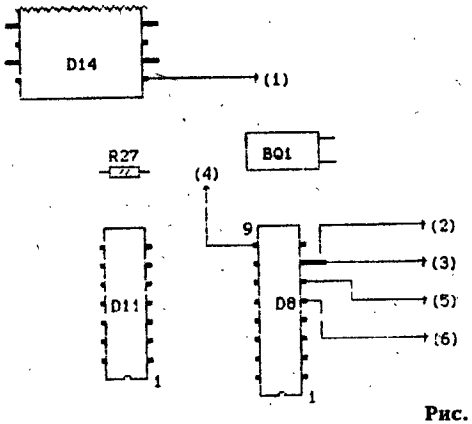


Рис. 2

Идея доработки состоит в следующем. В заводской конструкции тактовая частота 12 МГц от кварцевого генератора, прежде чем подаваться на процессор, дважды делится на два с помощью триггерных элементов микросхемы К531ТВ9П (D8): сначала до 6, а затем до 3 МГц. Если же снимать тактовые импульсы не со второго триггера (D8.2), а с первого (D8.1), то мы и получим требуемые 6 МГц. При этом нужно отсоединить не только прямой, но и инверсный выход элемента

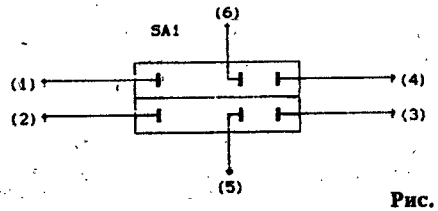


Рис. 3

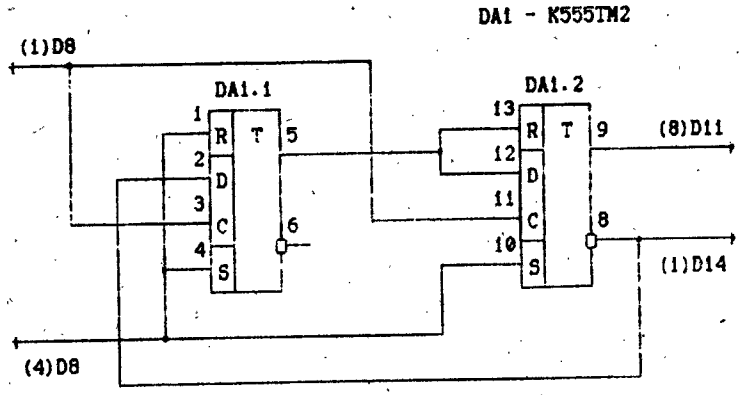


Рис. 4

D8.2, подключив вместо них соответствующие выходы D8.1. На рис. 1 показан фрагмент принципиальной схемы БК-0010 с изменениями, которые необходимо сделать для проведения турбирования. На системной плате вычислительной машины надо перерезать дорожку, идущую от девятого вывода микросхемы D8 к первому выводу процессора, а также аккуратно перерезать в месте соединения с платой и отогнуть седьмой вывод D8. Затем следует распаять провода от точек, указанных на рис. 2, к переключателю SA1, как показано на рис. 3. (Переключатель удобнее всего установить над разъемом порта ввода-вывода.)

Увеличение тактовой частоты до 4 МГц

Если попытка турбирования до 6 МГц, как описано выше, не увенчалась успехом, можно повысить тактовую частоту вычислительной машины до более скромного значения 4 МГц — и такое увеличение частоты качественно улучшает работу с дисководом и в значительной степени повышает быстродействие БК-0010(.01). Кроме того, заметим, что именно такая тактовая частота выбрана в качестве рабочей на БК-0011(М), так что можно установить данную частоту как постоянную. Преимущества такого подхода очевидны: не требуется сверлить дополнительные отверстия в корпусе под переключатель, значительно уменьшается количество монтажных проводов и их длина. Однако в некоторых случаях, например при воспроизведении мелодий, написанных для БК-0010 с тактовой частотой 3 МГц, желательно сохранить возможность включения стандартной тактовой частоты. Различия в проведении этих двух вариантов доработки будут оговорены ниже.

Турбирование на 4 МГц выполняется сложнее, чем в предыдущем случае, и требует установки дополнительной микросхемы. Суть доработки сводится к следующему: частота 12 МГц, минуя «штатные» делители, снимается непосредственно с тактового генератора и затем делится на

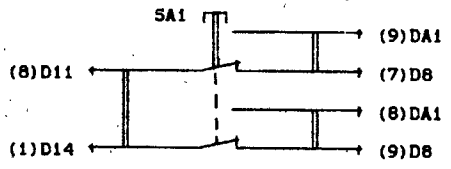


Рис. 5

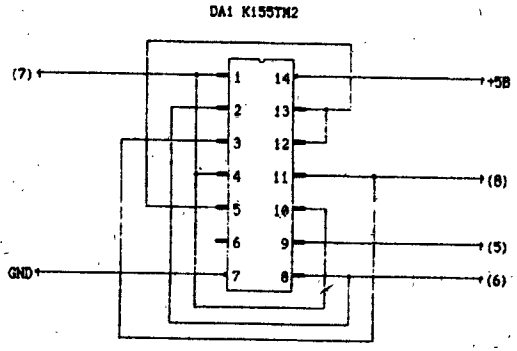


Рис. 6

три. Делитель может быть собран на микросхеме ТМ2 серий 155, 555, 1533. Схема доработки (для случая постоянного включения тактовой частоты 4 МГц) приведена на рис. 4. Если же вы хотите сохранить возможность включения стандартной тактовой частоты, то выводы 8 и 9 микросхемы DA1 следует подсоединить к переключателю, как показано на рис. 5. При проведении данной доработки первоначальная последовательность действий та же, что и для предыдущей: на системной плате надо перерезать дорожку, идущую от девятого вывода микросхемы D8 к первому выводу процессора, а также отсоединить от платы и отогнуть седьмой вывод D8. Микросхему DA1 следует установить на плате поверх D11, припаяв ее 7-й и 14-й выводы к соответствующим выводам нижней микросхемы, а остальные выводы DA1 отогнуть в горизонтальное положение. Затем производится распайка монтажных проводов согласно рис. 6 и рис. 7, на которых показаны, соответственно, микросхема DA1 и фрагмент системной платы БК-0010(.01). Соединение проводов (1) — (6) с переключателем выполняется в соответствии с рис. 3.

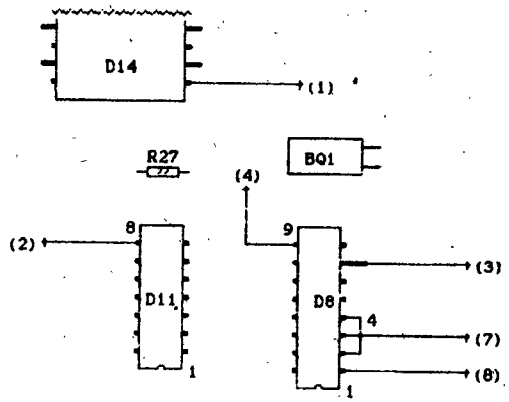
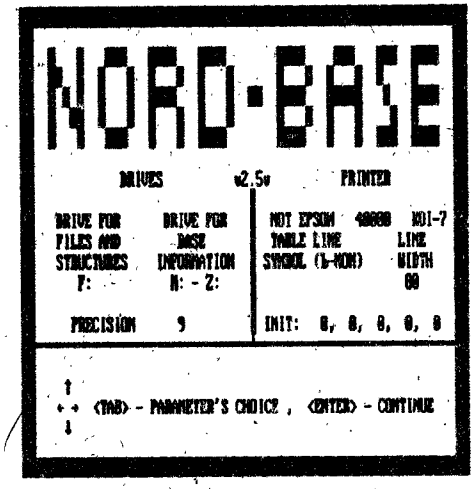


Рис. 7

Продолжение следует

А. Г. Прудковский,
Москва

База данных NORD-BASEv2.5v в системе NORD для БК-0011(М) с винчестером



Появление на БК жесткого диска стало причиной для создания винчестерной версии базы данных NORD-BASE, заставка которой представлена на рисунке. Хотя предыдущая ее версия (см. статью «NORD-BASEv2.3 — база данных для системы NORD на БК-0011(М)» в журнале «Персональный компьютер БК-0010 — БК-0011М», №6 за 1995 г.) также может работать на винчестере с помощью драйвера ABCD.COM, она имеет существенный недостаток, связанный с тем, что информация запоминается исключительно в материнском каталоге диска. Поскольку объем последнего ограничен двумя сотнями имен файлов, а емкость информационного файла базы данных NORD-BASE составляет 30 блоков (15 кб), получается, что диск может содержать не более 200*30=6000 блоков (3 Мб) информации. При максимальной же длине строки объем базы данных может достигать 10 Мб. В дисководном варианте это достигается путем записи одной базы данных на несколько дискет, на винчестере же потребовалось «научить» NORD-BASE работать с логическими дисками. Пользователю предлагается выбрать один логический диск (от F: до L:) для хранения файлов, макросов и структур и серию логических дисков (от N: до Z:) — для хранения собственно информации базы данных. Естественно, что для работы на дискетах винчестерная версия NORD-BASE малопригодна, так как не предусматривает задержки для смены диска.

Пример разбивки винчестера для работы с базой NORD-BASEv2.5v

При работе с винчестером очень важна предварительная разметка на нем «физических» устройств. Вообще говоря, система NORD поддерживает диски максимальной емкости до 64000 блоков (32 Мб), однако

работы базы данных размер дисков следует оценивать точнее. Так, если на данном устройстве предполагается хранить только одну базу данных с длиной строки, например, 1 кб, то общий объем информации не превысит 10 Мб плюс несколько мегабайт на структуры и файлы, так что размечать диск надо на фрагменты не более чем по 12 Мб (24000 блоков). После разметки винчестера на «физические» устройства на каждом из них в системе NORD следует построить структуру логических дисков. В приведенном выше примере нужно выделить диск F: на 1 Мб (2000 блоков) для структур и файлов, диски N:, O: и P: на (30*199+20)=5990 блоков каждый (из них 20 резервируется на каталог логического диска, причем одна строка каталога зарезервирована на подкаталог /DEL для удаленных файлов) и диск Q: на (30*71+20)=2150 блоков. Тогда диски N:, O:, P: и Q: позволяют хранить 199*3+71=668 информационных файла по 15 кб каждый, что составляет 20120 блоков (10.06 Мб) информации базы данных. Саму программу NORD-BASEv2.5v следует запускать с того же устройства, например из материнского каталога, который будет «распоряжаться» оставшимся мегабайтом «физического» устройства.

Если же вы собираетесь хранить на одном «физическом» устройстве несколько баз данных, обслуживаемых программой NORD-BASEv2.5v, то разбивка винчестера должна быть, разумеется, другой. Возможен даже случай, когда требуется использовать весь максимальный объем в 32 Мб, но даже тогда не потребуется задание всех возможных логических дисков, так как для разметки 32 Мб на фрагменты по 5990 блоков (2.995 Мб) достаточно десяти букв алфавита (например, от N: до W:), а остальные 2.01 Мб составят диск F: (файлы, структуры базы и материнский каталог устройства).

Примечание. При задании логических дисков система NORD требует указания длины этих дисков в восьмеричном виде ($5990_{10}=13546_8$, $2150_{10}=4144_8$), причем длина логического диска в восьмеричном представлении не должна быть кратной 100_8 — такова особенность системы NORD.

* * *

Итак, винчестерная версия базы NORD-BASEv2.5v может работать в полном объеме на жестком диске, полностью используя его возможности. Допустима даже работа нескольких баз данных на одном устройстве для создания отчетов с перекрестными ссылками.

В заключение отметим, что в винчестерной версии NORD-BASEv2.5v произведена модификация интерфейса базы в соответствии с рекомендациями П. П. Животовского (см. «Персональный компьютер БК-0010 — БК-0011М», №4 за 1995 г.): после запуска программа сразу входит в меню на считывание структуры и только после нажатия клавиши «КТ» переходит в режим NEW (создание новой структуры). Создан также дисковый (не винчестерный) вариант NORD-BASE с модифицированным интерфейсом, получивший имя NORD-BASEv2.4.

ОБМЕН ОПЫТОМ

Д. Ю. Усенков,
Москва

Запуск на исполнение произвольного фрагмента другой программы

Дефицит свободной памяти на БК-0010(.01), как известно, проблема номер один. Бороться с этим недостатком можно различными способами — от установки дополнительного ОЗУ до разнообразных алгоритмических хитростей, позволяющих уменьшать объем создаваемой программы. Но расширение ОЗУ, как и любая другая аппаратная доработка, требует значительных затрат денег (при покупке готового блока) или сил и времени (при его самостоятельном изготовлении). К тому же в последнем случае нужны хотя бы минимальные навыки работы с паяльником. Поэтому БКманы чаще всего предпочитают экономить байты за счет чисто программистских изысков. А из них самый простой и доступный даже для начинающих — это использование в своей программе вызовов «системных» подпрограмм (прошитых в ПЗУ или имеющихся в дисковой операционной системе). О назначении и правилах обращения к тем или иным подпрограммам монитора, БЕЙСИКа, ФОКАЛа и других «базовых» программных модулей сказано и написано уже немало. Но бывают случаи, когда нужный модуль «чужой» программы представляет собой не «цельную» подпрограмму, а ее фрагмент. Наглядный пример — подпрограмма монитора, производящая запрос имени файла и его чтение с магнитной ленты или диска (адрес входа @#100536, в R5 указывается адрес загрузки). Довольно быстро пользователи БК нашли другой адрес входа (@#100552), позволяющий записывать файл с запросом его имени (правда, буфер EMT 36 при этом нужно заполнять вручную). Но в обоих случаях мы можем изменять только адрес ВХОДА в подпрограмму, ВЫХОД же из нее всегда производится исключительно по достижении «финального» RTS PC. А что делать, если нужна только функция запроса имени (без файловых операций)? Ведь вставить в ПЗУ команду RTS в нужное нам место невозможно!

А что, если вообще отказаться от стандартного механизма «JSR—RTS» и поискать другой метод вызова фрагментов программ на исполнение? Например, попытаться использовать «отладочное» прерывание по Т-биту для контроля текущего исполняемого адреса и отлова заданной точки выхода. Идея, несомненно, достаточно интересна, чтобы попытаться

реализовать ее на практике. Разработанный автором короткий листинг универсальной подпрограммы — драйвера вызова на исполнение произвольного фрагмента другой программы представлен ниже. Его вызов производится командой JSR PC,SSE (предварительно нужно занести в ячейки ADRSTR и ADREND адреса первой и последней команд требуемого фрагмента соответственно).

```
; SCREW (c)          SUBROUTINE EXECUTOR              1995
;   Универсальный модуль вызова на исполнение фрагментов
;    других программ с произвольными граничными адресами
;
; ВХОДНЫЕ ДАННЫЕ:
;   ADRSTR - адрес первой команды вызываемого фрагмента;
;   ADREND - адрес последней команды фрагмента
;
; ВНИМАНИЕ! Данный модуль несовместим с программами,
;           использующими режим трассировки
;
; ---------------------------------------------------------
SSE:      MOV @#14,%OLDT    ; сохранить старый вектор
          MOV #BPTSET,@#14  ; адрес установщика Т-бита
          BPT               ; вызов установщика
  ; блок отслеживания адреса исполняемой команды
TRSSSE:   CMP @SP,ADREND    ; проверка достижения конца фрагмента
          BLOS BPTS1        ; еще нет - восстановить Т-бит
                            ;  и продолжить
          MOV #BPTRES,@#14  ; адрес блока сброса Т-бита
          BPT               ; вызов блока сброса
          RTS PC            ;  и выход из модуля
  ; блок установки Т-бита
BPTSET:   MOV ADRSTR,@SP    ; готовить возврат на нач. адрес
BPTS1:    BIS #20,2(SP)     ; установить Т-бит в ССП в стеке
          MOV #TRSSSE,@#14  ; вектор -> на блок слежения
          RTT               ; выход на нач. адр. с T=1
  ; блок сброса Т-бита
BPTRES:   BIC #20,2(SP)     ; сбросить Т-бит в ССП в стеке
          MOV #%OLDT,@#14   ; восстановить старый вектор
          RTI               ; выход со сбросом Т-бита
  ; используемые ячейки памяти
ADRSTR:   .#0               ; адрес начала фрагмента
ADREND:   .#0               ; адрес конца фрагмента
%OLDT:    .#0               ; для сохранения старого вектора Т
; ---------------------------------------------------------
```

В этой статье обсуждается работа с российскими принтерами МС-6312, МС-6313, СМ-6337 и МС-6304, а также с импортными моделями «ROBOTRON», «EPSON», «STAR», «IBM» и др.

А. Г. Прудковский,
Москва

Драйвер принтера для БК

Распайка принтера

Первое, что необходимо сделать после покупки принтера, — это подключить его к параллельному порту компьютера. К сожалению, для этой операции до сих пор существуют два стандарта, отличающиеся распайкой сигналов «ЗАНЯТО» и «СТРОБ». История возникновения такой неоднозначности следующая: при появлении БК-0010 для этих сигналов были выбраны (уж и не знаю кем) контакты В31 и А28 и этот стандарт соблюдался до появления в продаже компьютера БК-0011М с подсоединяемым к порту блоком КМ, который был настроен на контакты В29 и А25. Кто придумал блок КМ и для какой цели, неизвестно. Этот блок значительно ограничил возможности подключения внешних устройств к порту и ввел в обиход «новый» стандарт подключения принтера. Современные программы обычно имеют самонастраивающийся на распайку драйвер, и поэтому выбор подключения достаточно безразличен, но «старые» программы могут требовать ТОЛЬКО старую распайку, а ПЗУ БЕЙСИКа в БК-0011М — только новую.

Обозначение сигнала		Контакт порта	
		Новая распайка	Старая распайка
*	Д0	А16	А16
Д	Д1	А13	А13
А	Д2	В12	В12
Н	Д3	В10	В10
Н	Д4	В5	В5
Ы	Д5	В7	В7
Е	Д6	В6	В6
*	Д7	А7	А7
ЗАНЯТО	ЗП-ПН	В29	В31
СТРОБ	СТР	А25	А28
ОБЩИЙ		А18, А19, В18, В19	А18, А19, В18, В19

Основной драйвер принтера

Итак, кабель принтера распаян. С точки зрения программиста мы при этом получили связь с принтером через ячейку порта @#177714, причем информация передается через младший байт порта, а сигнал строба (так же как и сигнал занято) — через старший байт: это восьмеричное число 400 для старой распайки или 40000 — для новой. Подпрограмма WD для передачи байта на принтер может быть следующей:

```
WDI:  JSR R5,WD0        ;дополнительный вход для передачи ESC
      .#33              ;последовательностей
WD0:  MOV (R5)+,R0      ;доп. вход для передачи служебных байтов
WD:   MOV R1,-(SP)      ;основной вход, R0 — код печатаемого символа
      MOV R4,-(SP)      ;запоминание регистров R1 и R4 в стеке
      MOV #177714,R1    ;занесение адреса порта в R1
      CLR (R1)          ;очистка порта
1:    MOV (R1),R4       ;запрос сигнала "ЗАНЯТО" с учетом 2-х типов
      BIC #137377,R4    ; распайки и установка строба в R4
      BEQ 1             ; в соответствии с видом сигнала "ЗАНЯТО"
      MOV R4,-(SP)      ;занесение строба в стек
      BISB R0,(SP)      ;добавление в младший байт информации из R0
EPS:  NOP               ;COMB (SP) - для "EPSON"-стандарта
      MOVB (SP),(R1)    ;формирование переднего фронта импульса
      MOV (SP),(R1)     ;добавление строба к сигналу
3:    BIT R4,(R1)       ;ожидание, пока принтер примет сигнал
      BNE 3
      MOVB (SP)+,(R1)   ;сброс строба
      CLR (R1)          ;очистка порта, формирование заднего фронта импульса
      MOV (SP)+,R4      ;восстановление регистров R4,R1
      MOV (SP)+,R1
      RTS R5            ;выход из подпрограммы
```

Содержимое ячейки EPS в представленном драйвере зависит от стандарта, принятого на вашем принтере: «EPSON»-стандарт обычно требует инвертирования сигнала относительно строба, тогда в эту ячейку следует поставить команду COMB (SP). Отечественный же стандарт этого не требует, и можно оставить пустую команду NOP. Стандарт конкретного принтера обычно указан в документации к нему, так, принтеры МС-6312 обычно работают в отечественном стандарте, а МС-6313 — уже в новом «EPSON»-стандарте. Работа с подпрограммой WD будет описана ниже.

Компьютер как «пишущая машинка» (русификация принтера)

Чтобы ознакомиться с работой драйвера WD, напишем простейшую программу, превращающую компьютер в пишущую машинку.

```
      JSR R5,WDI    ;задание начальной последовательности ESC @
      .#100         ;или 33,100, инициализирующей принтер
```



```
      ..........    ;здесь можно вставить и другие команды,
      ..........    ;настраивающие принтер, устанавливающие
      ..........    ;размер и тип шрифта, расстояние между строк
      ..........    ;и так далее. На принтере МС-6304 вместо
      ..........    ;инициализации необходимо выполнить команды:
                    ;       JSR R5,WD0  - аннулирование
                    ;       .#30
                    ;       JSR R5,WD0  - установка шрифта КОИ-8
                    ;       .#23
A:    EMT 6         ;получение кода с клавиатуры
      CMPB R0,#12
      BNE 2
      JSR R5,WD     ;перевод строки при нажатии клавиши "ВВОД"
      JSR R5,WD0    ;возврат каретки (эта команда обязательна
      .#15          ;не для всех принтеров)
      BR A
2:    JSR PC,WDU    ;обращение к программе русификации, если
                    ;принтер не работает в кодах КОИ-8.
      JSR R5,WD     ;печать кода на принтере
      BR A
```

Программа русификации зависит от кодировки русских букв на вашем принтере. Так, если он работает в кодах КОИ-7, то эта программа может быть следующей:

```
WDU:  MOVB R0,-(SP)   ;запоминание кода
      JSR R5,WDI      ;задание последовательности ESC 122,
      .#122           ; включающей выбор одного из шрифтов КОИ-7
      CLR R0          ;0 - для латинских букв
      TSTB (SP)
      BGE 1
      INC R0          ;1 - для русских букв
1:    JSR R5,WD       ;посылка кода 0 или 1 на принтер
      MOVB (SP)+,R0   ;восстановление начального кода
      RTS PC
```

Большинство русифицированных западных моделей, а также принтеры МС-6313 и СМ-6337 имеют режим работы в кодировке ГОСТ. Работа в нем предпочтительнее, так как в нем есть символы полуграфики. Этот режим устанавливается ключами принтера или специальной командой после команды инициализации. Программа русификации для шрифта ГОСТ может быть следующей:

```
WDU:  TSTB R0
      BGT 1
      BIC #177600,R0
      CMPB R0,#40
```

```
    BLO 1
    SUB #40,R0          ;либо:  ADD PC,R0       ;для
    MOVB #TAB(R0),R0    ;       ADD (PC)+,R0    ;перемещаемого
1:  RTS PC              ;       .@TAB-36        ;драйвера
                        ;       MOVB (R0),R0
                        ;   1:  RTS PC
```

Здесь #TAB — адрес таблицы длиной 140_8 байт, в которой дано соответствие русских символов принтера и БК. Такую таблицу нетрудно составить, сравнивая описания кодов в инструкции к БК и к принтеру. Для кодировки ГОСТ эта таблица будет следующей:

```
TAB: .B:24,250,3,241,203,251,243,224,226,6,240,246,201,370
     .B:252,225,247,367,232,371,5,244,236,245,4,242,237,207
     .B:230,215,366,233,356,320,321,346,324,325,344,323,345,330
     .B:331,332,333,334,335,336,337,357,340,341,342,343,326,322
     .B:354,353,327,350,355,351,347,352,316,260,261,306,264,265
     .B:304,263,305,270,271,272,273,274,275,276,277,317,300,301
     .B:302,303,266,262,314,313,267,310,315,311,307,312
```

Графическая копия экрана

Если вам нужно распечатать на принтере графическую копию экрана, то можно воспользоваться приведенной ниже программой (если на вашем принтере имеется установка стандартного перевода строки, ее следует отключить):

```
AA:     MOV @#177664,-(SP)  ;расчет начального адреса экрана
        ADD #50,(SP)        ; с учетом его рулонного сдвига
        SWAB (SP)
        CLRB (SP)
        ROR (SP)
        ROR (SP)
1:      JSR R5,WDI          ;инициализация принтера
        .#100
2:      JSR R5,WDI          ;установка интервала между
        .#63                ; строками на 30/216 дюйма
        JSR R5,WD0
        .#30
        MOV #40,R5          ;высота экрана 40*10=400 (восьм.) строк
        TSTB @#42           ;при обнаружении режима РП
        BEQ 7               ; его высота уменьшается до
        MOV #7,R5           ; 7*10=70 (восьм.) строк
7:      JSR R5,WDI          ;начало основного цикла, выдача
        .#52                ; * графической команды ESC
        JSR R5,WD0          ;тип графики может быть различным
777:    .#4                 ; **  от 0 до 7 (см. инструкцию принтера)
```



```
        JSR R5,WD0          ;задание длины строки как
        .#0                 ; **    2*400+0=1000 (восьм.) точек
        JSR R5,WD0
        .#2                 ; **
        MOV #100,R4         ; **  задание длины строки 100 (восьм.) байт
6:      MOV #1,R3           ;проверка начиная с бита 1
5:      MOV #200,R1         ;установка начиная со старшего бита
        MOV (SP),R2         ;текущий адрес экрана
        CLR R0              ;очистка регистра сигнала
4:      BIC #140000,R2      ;модификация адреса с учетом рулонного
        BITB R3,40000(R2)   ; сдвига и проверка граф. точки
        BEQ 3               ;если точка есть, то установка бита
        BISB R1,R0          ; в сигнальный регистр R0
3:      ADD #100,R2         ;переход на следующую строку
        ASR R1              ;сдвиг установочного бита
        BNE 4               ;от 200 до 1
        JSR R5,WD           ;передача сигнала на принтер
        ASLB R3             ;изменение проверочного бита от 1
        BNE 5               ; до 200
        INC (SP)            ;переход к следующему байту
        SOB R4,6            ;цикл по количеству байт в строке
        ADD #700,(SP)       ; **. сдвиг на 8 строк (адрес в (SP)
        JSR R5,WD0          ; уже сдвинут на 100 байт в цикле по R4)
        .#12                ;перевод строки принтера
        JSR R5,WD0          ; (на 216/30 дюйм)
        .#15
        SOB R5,7            ;цикл по строкам
        JSR R5,WDI          ;завершающая инициализация принтера
        .#100               ; (на всякий случай)
        TST (SP)+
        RTS PC
```

Примечание. На принтерах с узкой кареткой графический режим 0 (задается на метке 777:) не позволяет распечатать всю ширину экрана, поэтому в каждой графической строке приходится пропускать последние 4 байта. В связи с этим надо изменить и программу (участок, помеченный двумя звездочками — «**»):

```
        JSR R5,WD0          ;задание длины строки как
        .#340               ; 1*400+340=740 (восьм.) точек
        JSR R5,WD0
        .#1
        MOV #74,R4          ;задание длины строки 74 (восьм.) байт
```

Еще одну команду придется изменить в том месте, где рассчитывается следующий адрес экрана через 8 строк от прежнего. Так как мы не обработали последние 4 байта, то к адресу в (SP) следует добавить число, на 4 большее прежнего (в листинге строка также отмечена «**»):

```
        ADD #704,(SP)       ; сдвиг на 8 строк
```

Вышеприведенная программа работает на всех принтерах со стандартной системой команд и 8-игольной головкой. Для нестандартных принтеров программа может быть другой, хотя, по сути, последовательность основных операций останется той же. Так, для принтера МС-6304 с 7-игольной головкой и оригинальной системой команд эта программа будет выглядеть так:

```
AA:     MOV @#177664,-(SP)  ;расчет начального адреса экрана
        ADD #50,(SP)        ; с учетом его рулонного сдвига
        SWAB (SP)
        CLRB (SP)
        ROR (SP)
        ROR (SP)
        JSR R5,WD0          ;аннулирование принтера
        .#30
        JSR R5,WDI          ;установка головки на позицию 0 по X
        .#122
        JSR R5,WD0
        .#0
        MOV #45,R5          ;высота экрана 45*7=403 (восьм.) строк
        TSTB @#42           ;при обнаружении режима РП
        BEQ 7               ; его высота изменяется до
        MOV #10,R5          ; 10*7=70 (восьм.) строк
7:      MOV #74,R4          ;задание длины строки 74 (восьм.) байт
                            ;правые 4 байта экрана не помещаются
                            ; на узкой каретке принтера
        CLR R1              ;обнуление регистра подсчета пробелов
6:      MOV #1,@#302        ;проверка начиная с бита 1
5:      MOV #200,R0         ;установка начиная со старшего бита
        MOV (SP),R2         ;текущий адрес экрана
        CLR R3              ;очистка регистра сигнала
4:      BIC #140000,R2      ;модификация адреса с учетом рулонного
        BITB @#302,40000(R2) ; сдвига и проверка граф. точки
        BEQ 3               ;если точка есть, то установка бита
        BISB R0,R3          ; в сигнальный регистр R0
3:      ADD #100,R2         ;переход на следующую строку
        ASR R0              ;сдвиг установочного бита
        CMPB R0,#1          ; от 200 до 2
        BNE 4
        TSTB R3
        BNE 1
        CMPB R1,#177        ;если сигнал нулевой и количество
        BLO 2               ; пробелов в R1 менее 177, то обход
                            ; выдачи сигнала (для ускорения печати)
1:      JSR R5,WDI          ;сдвиг головки по X на R1 пробелов
        .#131
```



```
        MOVB R1,R0
        JSR R5,WD
        JSR R5,WDI          ;команда выдачи семиточечного
        .#117               ; столбца
        CLR  R1
        MOV R3,R0
        JSR R5,WD           ;передача сигнала на принтер
2:      INCB R1             ;подсчет количества пробелов
        ASLB @#302          ;изменение проверочного бита от 1
        BNE 5               ; до 200
        INC (SP)            ;переход к следующему байту
        SOB R4,6            ;цикл по количеству байт в строке
        ADD #604,(SP)       ;сдвиг на 7 строк (адрес в (SP)
                            ; уже сдвинут на 74 байта в цикле по R4)
        JSR R5,WDI          ;перевод головки на 7 позиций
        .#132               ; вниз по оси Y
        JSR R5,WD0
        .#7
        JSR R5,WDI          ;установка головки на 0 позицию по X
        .#122
        JSR R5,WD0
        .#0
        SOB R5,7            ;цикл по строкам
        JSR R5,WD0          ;пропуск строки
        .#12
        TST (SP)+
        RTS PC
```

В качестве домашнего задания для начинающих программистов предлагаю написать аналогичные программы выдачи копии экрана с поворотом на 90°, кстати, это единственная возможность получения ПОЛНОЙ копии экрана в режиме 0 на стандартных принтерах с узкой кареткой, а также на принтере МС-6304.

Теперь обсудим вопрос об использовании программы выдачи графической копии экрана. Конечно, ее можно применять как обычную подпрограмму. Но для получения копии из произвольных программ ее лучше оформить в виде резидентного драйвера, работающего от того или иного внешнего сигнала. Для этой цели наиболее удобен сигнал прерывания от параллельного порта (ячейка @#270). Программа начальной загрузки драйвера может быть следующей:

```
        JSR PC,U            ;установка вектора @#100
        MOV PC,R0           ;перехват команды EMT 14,
        ADD (PC)+,R0        ; которая очищает ячейку @#100
        .@W+2               ;на БК-0010, где ячейку 100014 монитора
        MOV @#100014,W+2    ; менять нельзя, аналогичный
        MOV R0,@#100014     ; перехват можно сделать по вектору 30
        HALT
W:      JSR PC,@#100140     ;инициализация монитора
```

```
U:    MOV PC,R0          ;восстановление вектора 100
      ADD (PC)+,R0
      .@AU+2
      MOV R0,@#100
      RTS PC
AU:   MOV @#4,-(SP)      ;запоминание вектора СТОП
      JSR R4,@#100346    ;запоминание всех регистров
      MOV R5,-(SP)
      MOV SP,AW+2        ;запоминание стека
      MOV PC,-(SP)       ;установка вектора СТОП на
      ADD (PC)+,(SP)     ;адрес AW для  возможного
      .@AW+2             ;выхода из драйвера AA
      MOV (SP)+,@#4      ;по нажатию на клавишу "СТОП"
      JSR PC,AA          ;обращение к графическому драйверу
AW:   MOV #0,SP          ;восстановление стека
      MOV (SP)+,R5       ; и регистров
      JSR R4,@#110362
      MOV (SP)+,@#4      ;восстановление вектора СТОП
      RTI                ;выход из прерывания
```

Теперь осталось оборудовать порт кнопкой, соединяющей контакты B2 и B18 («общий»). При нажатии этой кнопки на принтер выдается копия содержимого экрана*. Если не получится, попробуйте проделать то же самое с вектором 100, при этом сигнал «общий» надо подавать на контакт B1 с одного из контактов порта: A18, A19, B18, B19.

* Работа с прерыванием по вектору 270 редакцией не проверялась.

*** ВНИМАНИЕ! ОПЕЧАТКА ***

В статье Б. Н Сергеева «Практикум на ассемблере» (журнал «Персональный компьютер БК-0010 — БК-0011М», №4 за 1995 г.) в разделе «6. Полезные подпрограммы монитора БК» (с. 21, слева вверху) ошибочно указан адрес прошитой в ПЗУ подпрограммы генерации звука с возрастающей частотой. Правильное значение адреса входа в данную подпрограмму — 102062.

Приносим читателям свои извинения.

С. Лашкевич,

г. Борисов Минской обл.

Самодельная лента для матричного принтера

Предлагаю несколько полезных советов для владельцев матричных принтеров (МС-6313, D100, D100M, СМ-6337 и др.). При интенсивной работе красящая лента быстро изнашивается (ее ресурс составляет примерно 2—3 млн. отпечатанных символов), после чего печать становится нечеткой и ленту приходится менять на новую. Но можно восстановить «выбитую» красящую ленту и тем самым сэкономить немало денег. Поместите израсходованную ленту в герметически закрывающуюся емкость (например, стеклянную банку), на дно которой положите пропитанную керосином вату, затем уложите ленту и сверху опять вату с керосином. Плотно закройте банку и оставьте на 2—3 дня, после чего просушите ленту и ее можно снова использовать по назначению.

Если есть необходимость, можно самому изготавливать ленты для принтера. Из прочной нейлоновой или плотной капроновой ткани (годятся ленты для девчоночьих бантиков) вырежьте ленту шириной 13 или 16 мм (в зависимости от типа принтера) и окрасьте ее, пропитав тонером и просушив.

Рецепты приготовления тонера:

1. 30 частей мыла (в порошке или кусочками), 24 части дистиллированной воды и 8 частей глицерина растворить при нагревании и добавить 4 части любой анилиновой краски для окрашивания тканей, растворенной в 5 частях воды и 6 частях любого спирта.

2. 100 частей глицерина нагреть в фарфоровой чашке и постепенно добавить 100 частей тончайшего порошка анилиновой краски любого цвета. При охлаждении масса становится пескообразной, к ней при постоянном помешивании добавить небольшими порциями воду до исчезновения зернистости и образования блестящей жидкости.

(В зависимости от интенсивности анилиновой краски ее количество может составлять от 4 до 10% смеси.)

Для принтера D100M можно использовать только «фирменную» ленту, снабженную на концах специальными заклепками для автоматического изменения направления перемотки. Если у вас такой ленты нет, можно использовать обычную, надев на ее концы с отступом примерно по 10 см самодельные заклепки. (Идеально подходят одежные кнопки малого размера. Они защелкиваются по краям ленты и великолепно выполняют требуемую от них функцию.)

ЕСТЬ ИДЕЯ

Листая страницы старых журналов, иной раз можно натолкнуться на что-нибудь интересное, незаслуженно забытое в наши дни. Вот, например, материал об изобретении ленинградского инженера Г. Загорельского — гезотайпе, приставке для быстрой печати к электрическим пишущим машинкам, — опубликованный в журнале «Изобретатель и рационализатор», № 11 за 1972 г., и затем перепечатанный в «Науке и жизни» (к сожалению, в последнем случае от журнала сохранилось лишь несколько листков с названной статьей, и определить номер и год его выхода в свет теперь не представляется возможным). В то время о «персоналках» большинство из нас и слыхом не слыхивало, а до «рождения» самой первой модели БК-0010 оставалось целых 13 лет. Но ведь БК по возможностям подключения разнообразных периферийных устройств намного опережает любые электрические пишущие машинки, а ожидаемые от гезотайпа удобства (в частности, значительное увеличение скорости печати) достаточно привлекательны, чтобы вновь вспомнить об этом изобретении. Предлагаем вашему вниманию сокращенную перепечатку статьи из «Науки и жизни» (ее автором был специальный корреспондент журнала «Изобретатель и рационализатор» А. Ратов) и надеемся, что наши БКманы сумеют найти применение этой интересной идее.

Гезотайп: клавиатура на новом принципе

Каждый, кому доводилось работать с пишущей машинкой, наверное, замечал, что ее стандартная клавиатура практически не позволяет обеспечить максимально возможную скорость печати. Главная причина этого — нерациональная организация необходимых для этого движений пальцев: прицельных ударов, направленных сверху вниз. Загорельский же решил сделать такую клавиатуру, на которой набирать информацию можно, двигая пальцами влево и вправо и пересекая при этом датчики букв — движения для человека гораздо более привычные. (Попробуйте постучать пальцами по столу, а потом сделайте несколько движений, как будто стряхиваете пыль со стола, и вы убедитесь, что второй вид движений намного легче, удобнее и быстрее.) То же самое очевидно из рис. 1, где показаны траектории движения руки при написании слова «дина» карандашом, печати на обычной пишущей машинке и наборе с помощью гезотайпа.

Одна из моделей прибора, гезотайп-Д, имела плоскую панель с чуть выступающими над ней шестью валиками-датчиками, четыре из которых расположены столбиком в центре, а два по бокам. Если посмотреть на каждый валик с торца, то формой он напоминает гриб. Палец оператора, пересекая датчик, поворачивает его, а тот краем «шляпки» замыкает либо правую, либо левую пару установленных рядом с ним контактов. Соответствующий электрический сигнал, пройдя несложную схему гезотайпа, поступает на электрическую машинку типа «Консул», а та печатает нужную букву.

Пересекая поочередно и справа и слева четыре центральных валика, мы набираем восемь знаков (рис. 2, а). Если палец пройдет сначала через боковой датчик, а затем через центральные, то получим еще восемь знаков (рис. 2, б). Движение от центра через боковой к средним (боковой при этом пересечется дважды) даст еще восемь (рис. 2, в). И наконец, путь пальца от центра через боковой датчик и далее, не касаясь его, через средние даст еще восемь знаков (рис. 2, г). Итого — 32 знака русского алфавита.

Как показали опыты, всего за 20 часов (т. е. менее чем за неделю) любой новичок может научиться набирать на гезотайпе более 400 знаков в минуту — со скоростью очень квалифицированной машинистки!

Выбирая соответствия между буквами и движениями, Загорельский основывался на статистическом анализе сочетаний букв в русском тексте. Например, буквам Г, Л, М, Х и другим, чаще всего встречающимся не в начале слова, соотнесены движения, начинающиеся от центра. Гласные О, Е, И, А, с которых многие слова начинаются, принадлежат центральным датчикам.

Датчики гласных букв расположены слева, согласных — в основном справа, что позволяет набирать их одним движением, так как в большинстве случаев в тексте они следуют друг за другом.

Существует и еще один прием для увеличения скорости. Пересекая одновременно двумя пальцами — указательным и средним — боковой валик и центральный второй снизу, за одно движение получаем набор букв ПР. Подобным же образом набираются и иные часто встречающиеся

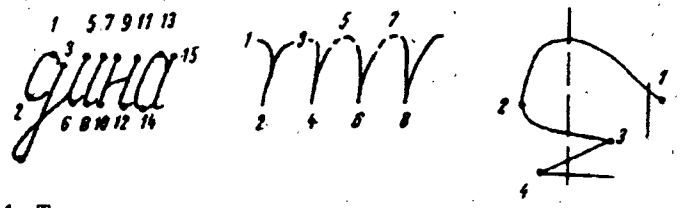


Рис.1. Траектории движения руки при написании слова «дина» карандашом, на обычной пишущей машинке, с помощью гезотайпа

пары — КТ, СТ, ТР и др. Можно также использовать две панели гезотайпа для каждой руки. И тогда вполне реально достигнуть скорости набора 800 знаков в минуту, т. е. практически скорости речи.

Как сделать гезотайп

Для изготовления гезотайпа вам потребуются: гетинаксовая пластина размером 20×100×250 мм, шесть эбонитовых кулачков (валиков) грибовидной формы в разрезе, три проволочные оси, десять диодов типа Д9Д, четыре реле РЭС-10 с сопротивлением обмоток 630 Ом и немного монтажного провода.

В гетинаксовой пластине надо вырезать шесть отверстий, как показано на рис. 2 вверху. В эти отверстия установите кулачки на осях. Добейтесь, чтобы они легко поворачивались, когда вы касаетесь их пальцем. С обратной стороны пластины приклейте эпоксидной смолой. Вы получите рабочее поле с шестью информационными зонами. Общий вид панели гезотайпа в разрезе, с установленными кулачками и контактами показан на рис. 3, а схема распайки контактов — на рис. 4.

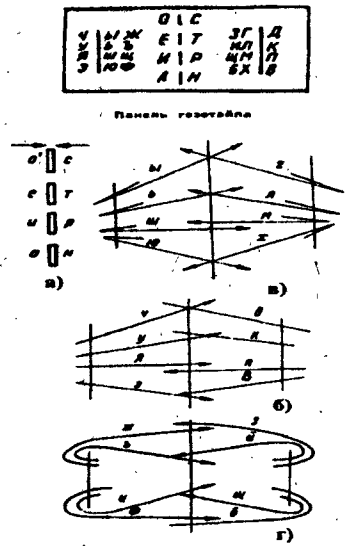


Рис. 2. На рабочем поле гезотайпа шесть датчиков. Но несложная система последовательных касаний их дает набор всех букв русского алфавита.

Схема преобразователя для управления электрической пишущей машинкой «Консул-254» или «Оптима-Электрик» дана на рис. 5. Выходы 1—8 и I—IV подключаются к столбцам и строкам матрицы, элементами которой являются диод и электромагнит, соответствующий определенной букве электрической пишущей машинки.

Устройство работает так. Пересекая верхнюю центральную зону с левой стороны, замыкаем контакты К5, «Плюс» напряжения питания подключается к столбцу 5 (Ы, Ж, Ч, О). И так как «минус» был подан через контакты реле Р2, Р4 по каналу IV, то срабатывает электромагнит буквы О. Одновременно через диод Д5 подается напряжение на обмотку реле Р1, которое обрывает подачу «минуса» с задержкой, достаточной для срабатывания электромагнита. (Время задержки зависит от емкости конденсатора и времени его заряда.) Когда палец оторвется от кулачка, контакт К5 размыкается, реле Р1 отключается, резистор подсоединяется к конденсатору, который разряжается через него. Схема возвращается в начальное состояние.

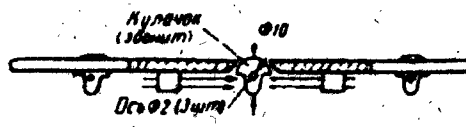


Рис. 3. Разрез панели гезотайпа

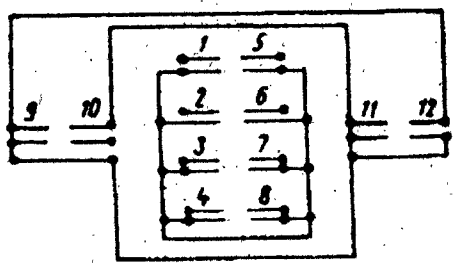


Рис. 4. Электрическая схема контактов кулачков панели

Если предварительно пересечена боковая зона с внутренней и (или) наружной стороны, то кулачок боковой зоны собственными контактами блокирует реле Р2 и (или) Р3 и Р4, подключая «минус» напряжения питания к другим строкам. Далее при срабатывании определенного центрального датчика печатается соответствующий знак и осуществляется сброс блокировки за счет уменьшения напряжения. Вот, собственно, и все, что надо знать, чтобы самому построить гезотайп.

Расположение и величина информационных зон выбраны на основании экспериментов со многими испытуемыми. Очевидно, и вам придется не раз примериться, прежде чем будет найдено наилучшее расположение датчиков для руки.

Запомнить расположение датчиков и порядок движений нетрудно — достаточно потренироваться с недельку. Лучше всего придумать для себя мнемонические правила. Можно, например, основные восемь букв запомнить с помощью слова «осетрина», которое они образуют. Практика показала, что скорость письма на гезотайпе в 400—500 знаков в минуту абсолютно реальна и достижима для каждого.

Напоминаем, что под рубрикой «ЕСТЬ ИДЕЯ» планируется публиковать интересные идеи, задумки, предложения относительно различного аппаратного и программного обеспечения. Если вы придумали что-то интересное, но не располагаете временем либо необходимыми знаниями, чтобы довести это до законченной разработки, или если вы имеете и то, и другое, но вам не хватает фантазии, рубрика «ЕСТЬ ИДЕЯ» всегда к вашим услугам! (Подробности о рубрике, ее целях и задачах см. в № 2 за 1994 г.)

Приглашаем всех читателей к сотрудничеству.

НЕОЧЕВИДНОЕ, НО ВЕРОЯТНОЕ

Д. Ю. Усенков,
Москва

Компьютер БК: «окно в четвертое измерение»

Не думаю, что реакция читателей на строку, вынесенную в заголовок, будет однозначной. Одни равнодушно скользнут взглядом мимо, другие только саркастически усмехнутся, а кто-то, может быть, и поспешит разменивать тысячные банкноты, чтобы рвануть в недавнее советское прошлое за дешевой колбасой. Последних придется слегка огорчить: превратить БКшку в «машину времени», способную переносить пользователя в прошлое или будущее, нам пока не удастся. Но компьютер способен помочь в не менее интересном исследовании — моделировании поведения четырехмерного объекта с точки зрения наблюдателя из «нашего» мира. Может быть, кому-то приведенные в статье рассуждения покажутся чисто умозрительными, а сама тема — чересчур оторванной от практики, но это не уменьшает интересности самой задачи.

Итак, каковы «начальные условия»? Пусть имеется некий объект, способный не только перемещаться по привычным для нас трем измерениям пространства, но и свободно изменять скорость течения собственного времени. Прежде всего, необходимо выбрать «опорную» систему отсчета. Очевидно, что пытаться рассматривать поведение объекта относительно его самого бесполезно. Представьте себе наблюдателя, сидящего в замкнутой комнате. Если время меняет свой ход одновременно во всех точках комнаты (четвертое измерение «изохронно»), наблюдатель ничего не заметит. Ведь насколько быстрее (или медленнее) будут идти часы, капать вода из крана или взмахивать крыльями попугай в клетке под потолком, ровно настолько изменится и скорость восприятия действительности у наблюдателя. (Этот эффект можно считать своего рода аналогом принципа эквивалентности инертной и гравитационной массы в теории относительности.) Значит, нужно отслеживать поведение объекта именно относительно нашего мира с «нормальным» течением времени. (Заметим кстати, что параметр, названный ранее «скоростью течения времени», и есть сама четвертая координата, а не изменение координаты, подобно скорости в обычном понимании этого слова.)

Задавшись «начальными условиями» и «системой отсчета», посмотрим теперь, что необходимо определить. Очевидно, мы должны найти проекцию четырехмерной траектории движения объекта (в пространстве и времени) на «наше» пространство-время. Сделать это проще всего

путем моделирования — расчета дискретных положений объекта с некоторым шагом по времени.

Но чтобы построить проекцию траектории, вначале надо определить саму эту траекторию. Поэтому в качестве аргумента функции возьмем собственные временные параметры объекта: начало отсчета времени, конец отсчета и шаг (в «его» секундах). Траекторию перемещения будем задавать в параметрической форме — в виде системы из четырех уравнений (здесь TS — собственное время объекта):

```
X=F1(TS) - координата X;
Y=F2(TS) - координата Y;
Z=F3(TS) - координата Z;
T=F4(TS) - ход нашего времени.
```

Как и при исследовании вида алгебраических функций, будем рассчитывать значения координат X, Y, Z и T с некоторым шагом, заполняя выделенный для этой цели массив. Как легко видеть, отсчеты «нашего» времени (T) в этом массиве окажутся записанными вперемешку (как и должно быть, если разрешить объекту произвольно перемещаться в прошлое или будущее с любой «скоростью»). Теперь же мы должны пересортировать массив, расставив его элементы по возрастанию величины T, чтобы получить «проекцию» времени объекта на «наше». Затем остается только выводить на экран символическое изображение объекта по координатам (X,Y,Z) с пошаговым отсчетом T.

Перейдем сразу к листингу программы на БЕЙСИКе — будем просматривать ее строки и одновременно обсуждать алгоритм.

```
10 ' Моделирование четырехмерных процессов
20 DIM A(100,4)
30 CLS
40 GOSUB 980 ' заставка
50 ? AT(10,10);"ЖДИТЕ..."
60 LET IN=1 ' начало отсчета собств. времени объекта
70 LET IK=100 ' конец отсчета собств. времени объекта
80 LET IS=1 ' шаг отсчета собств. времени объекта
90 II=IN ' счетчик секунд объекта
100 ? AT(1,15);"1-й этап: расчет координат     ";
    CHR$(147);AT(0,16);
110 FOR I=0 TO (IK-IN)\IS
120 GOSUB 750   ' расчет X,Y,Z,T
130 A(I,1)=X ' координата X
140 A(I,2)=Y ' координата Y
150 A(I,3)=Z ' координата Z
160 A(I,4)=T ' "наши" секунды
170 II=II+IS
180 ? "#";
190 NEXT
200 ' массив заполнен; сортировка
```

```
210 XM=0 ' будущий максимум X
220 YM=0 ' будущий максимум Y
230 ZM=0 ' будущий максимум Z
240 ? AT(1,15);CHR¤(145);"2-й этап: поиск максимумов    ";
    CHR¤(146);AT(0,16);
250 FOR I=0 TO (IK-IN)\IS
260 ? "#";
270 IF XM<A(I,1) THEN XM=A(I,1)
280 IF YM<A(I,2) THEN YM=A(I,2)
290 IF ZM<A(I,3) THEN ZM=A(I,3)
300 NEXT I
310 ? AT(1,15);CHR¤(145);"3-й этап: сортировка массива  ";
    AT(0,16);
320 FOR I=0 TO (IK-IN)\IS
330 ? "#";
340 FOR J=I+1 TO (IK-IN)\IS
350 IF A(I,4)<=A(J,4) GOTO 370
360 GOSUB 1000  ' перестановка эл-тов местами
370 NEXT J,I
380 ' массив пересортирован; вывод на экран
    (для простоты двумерный)
390 CLS
400 BEEP
410 BEEP
420 BEEP
430 BEEP
440 BEEP
450 ? AT(10,10);"ГОТОВО."
460 ? " Для пуска нажмите любую клавишу"
470 IF INKEY¤="" GOTO 470
480 CLS
490 MX=100/XM ' коэфф. масштаба по X
500 MY=100/YM ' коэфф. масштаба по Y
510 GOTO 560
520 LINE (25,120)-(225,120),2
530 LINE (125,20)-(125,220),2
540 ' вычерчены оси X и Y
550 RETURN
560 FOR I=0 TO ((IK-IN)\IS)
570 CLS
580 GOSUB 520 ' вычертить оси
590 IF A(I,4)=0 GOTO 670
600 ? AT(0,0);"T=";INT(A(I,4))
610 XX=INT(A(I,1)*MX)+125
620 YY=120-INT(A(I,2)*MY)
630 LINE (XX-3,YY-3)-(XX+3,YY+3),1,B
640 IF INT(A(I,4))<>INT(A(I+1,4)) GOTO 670
650 I=I+1 ' это делается, если
660 GOTO 610 ' неск. позиций массива "одновременны"
```



```
670 ? AT(1,1);"Нажмите любую клавишу"
680 FOR K=1 TO 1000 ' задержка
690 IF INKEY¤<>"" GOTO 710 ' выход из задержки
    при нажатии любой клавиши
700 NEXT K
710 NEXT I
720 ? AT(1,1);"     Все выведено!    "
730 END
740 ' ------------------------------------------------
750 ' подпрограмма расчета "проекционных"
    характеристик объекта
760 ' РАСЧЕТ X
770 X=X+3 ' равномерное движение относительно оси X
    со "скоростью" 3 ед. в 1 "собств." сек.
780 ' РАСЧЕТ Y
790 IF II<20 THEN Y=0 ELSE 810  ' первый этап -
    движение по горизонтали
800 GOTO 860
810 IF II<40 THEN Y=Y+2 ELSE 830  ' второй этап - "взлет"
    на 2 ед. в 1 "собств." сек.
820 GOTO 860
830 IF II<60 THEN Y=Y-2 ELSE 850  ' третий этап -
    "снижение" на 2 ед. в 1 "собств." сек.
840 GOTO 860
850 Y=0 ' четвертый этап и до конца - по горизонтали
860 ' РАСЧЕТ Z
870 Z=0 ' для упрощения - все время постоянное
880 ' РАСЧЕТ T
890 IF II<50 THEN T=T+1 ELSE 910   ' вначале собственное
    время объекта идет синхронно с нашим
900 GOTO 960
910 IF II<80 THEN T=T-1 ELSE 930 ' вторая часть пути -
    уход в прошлое
920 GOTO 960
930 T=T+1 ' а дальше - снова синхронно с нами
940 ' РАСЧЕТЫ ОКОНЧЕНЫ
950 ' ------------------------------------------------
960 RETURN
970 ' подпрограмма-заставка
980 CLS
990 ? AT(8,7);"SCREW (c) - 1995"
1000 ? AT(2,10);"Моделирование четырехмерных"
1010 ? AT(10,11);"объектов"
1020 BEEP
1030 FOR I=1 TO 2000 ' задержка
1040 IF INKEY¤<>"" GOTO 1060  ' прерывание ожидания
    при нажатии любой клавиши
1050 NEXT
1060 CLS
```

```
1070 RETURN
1080 AA=A(I,1)
1090 A(I,1)=A(J,1)
1100 A(J,1)=AA
1110 AA=A(I,2)
1120 A(I,2)=A(J,2)
1130 A(J,2)=AA
1140 AA=A(I,3)
1150 A(I,3)=A(J,3)
1160 A(J,3)=AA
1170 AA=A(I,4)
1180 A(I,4)=A(J,4)
1190 A(J,4)=AA
1200 RETURN
```

- Строка 20: резервируется массив из 100 записей, каждая из которых состоит из четырех элементов — координат X, Y, Z и T. К сожалению, в БЕЙСИКе не реализован настоящий механизм записей (как в СИ или ПАСКАЛе), но сама идея — представить двумерный массив как одномерный из одинаковых по структуре элементов, каждый из которых состоит из нескольких переменных, — нам очень пригодится.
- Строки 30—50 и подпрограмма в строках 980—1070: чисто «оформительские» элементы. Впрочем, строки 1030—1050, где наиболее простым способом реализуется пауза заданной длительности, при желании прерываемая «досрочно» нажатием любой клавиши, могут оказаться полезными при написании других БЕЙСИК-программ.
- Строки 60—80: задание исходных значений. Переменная IN содержит номер начальной «собственной» секунды объекта, IK — конечной, IS — шаг. Для простоты выбираем «линейный» интервал наблюдения от первой секунды до сотой с шагом в одну секунду.
- Строка 90—190: цикл расчета координат. В операторе FOR (строка 110, а также 250, 320, 340 и 560) нам удобнее использовать некоторую «отвлеченную» переменную I, всегда изменяющуюся с единичным шагом (чтобы не оставлять пустые ячейки в массиве). Поэтому введем дополнительную переменную II для отсчета «собственных» секунд объекта (строка 90). Количество же проходов FOR-цикла можно рассчитать по простой формуле — разделив на величину шага разность конечного времени и начального: (IK–IN)\IS. Отметим, что здесь использовано целочисленное деление, так как IK может оказаться не кратно шагу (если он отличается от единицы).

В цикле производится вызов подпрограммы (строки 750—960), в которой даны зависимости траектории движения объекта. В качестве примера в листинге приведены уравнения для одного из наиболее интересных случаев (из найденных автором). Для простоты пространственное перемещение объекта выберем не очень запутанным. Пусть координата X (строка 770) на каждом шаге изменяется на 3 единицы (т. е. относительно горизонтальной оси объект движется равномерно). Движение относительно вертикальной оси (координата Y) разделяется на четыре этапа:

— первые 20 секунд Y не изменяется (строка 790);

— следующие 20 секунд Y возрастает на каждом шаге на 2 единицы (строка 810);

— в течение еще 20 секунд Y на каждом шаге уменьшается на 2 единицы (строка 830);

— наконец, последние 40 секунд Y опять остается постоянной и равной нулю (строка 850).

Чтобы не заниматься рисованием на экране трехмерной системы координат, ограничимся плоской (двумерной) траекторией — суть моделируемого процесса от этого не изменится. Поэтому координата Z (строка 870) всегда будет равна нулю и введена в программу только для универсальности (желающие могут модифицировать демонстрационную часть программы для построения изометрии и добавить изменение координаты Z).

Исходя из рассмотренных уравнений, видим, что пространственная траектория движения объекта относительно «его» собственного времени должна выглядеть следующим образом: объект движется в некоторой плоскости вначале горизонтально, затем совершает «прыжок» и вновь возвращается к горизонтальному перемещению.

Что же касается расчета координаты T («наше» время относительно собственного времени объекта), он требует большей внимательности. «Временная» траектория объекта разделяется на три этапа:

— первые 50 «собственных» секунд (строка 890) идут синхронно с «нашими» (т. е. объект движется в будущее с единичным шагом, как и внешний наблюдатель);

— следующие 30 «собственных» секунд (строка 910) объект перемещается в прошлое относительно наблюдателя, причем «абсолютная скорость» течения времени объекта также единичная (поэтому T на каждом шаге уменьшается на единицу);

— последние 20 «собственных» секунд (строка 930) объект вновь движется синхронно с наблюдателем, как и на первом этапе.

После возврата из подпрограммы вычисленные значения X, Y, Z и T заносятся в ячейки массива, формируя «запись» с порядковым номером I. Строки 100 и 180, а также 240, 260, 310 и 330 образуют на экране динамический индикатор обработки информации, чтобы пользователь мог визуально наблюдать его ход (и чтобы хоть как-то скрасить довольно долгое ожидание окончания расчетов).

- Строки 210—300: определение максимальных значений X, Y и Z. Они нужны исключительно для автоматического масштабирования координат при выводе условного изображения объекта на экран в демонстрационной части программы.
- Строки 310—370: сортировка массива по возрастанию значений T (четвертый элемент записей) по широко известному «пузырьковому» методу. Для удобочитаемости программы перестановка элементов массива вынесена в подпрограмму (строки 1080—1200).
- Строки 390—500: все готово для наблюдения результатов моделирования, о чем сообщает пятикратный писк пьезодинамика. Экран очищается, и рассчитываются коэффициенты масштабирования MX и MY (по осям X и Y соответственно).

Далее по оператору GOTO мы «перепрыгиваем» через блок вычерчивания координатных осей, который оформлен в виде подпрограммы. (Ревнители структурного программирования могут переписать программу, вынеся этот блок в конец листинга.)

- Строки 560—730: собственно демонстрационная часть и завершение работы программы. Организуется цикл перебора элементов уже отсортированного массива и для каждого значения T на экране рисуется прямоугольник (условное обозначение объекта). Предварительно выполняется расчет экранных координат XX и YY с учетом масштабных коэффициентов (строки 610 и 620), а значение T выводится в верхней строке экрана. Обратите особое внимание на строки 650 и 660: при движении объекта то в прошлое, то в будущее (образование «временной петли») одному и тому же значению T может соответствовать НЕСКОЛЬКО положений в пространстве.

Когда изображение (или изображения) объекта в тот или иной момент «нашего» времени T нарисовано, делается небольшая пауза (для ускорения вывода на экран можно прервать ее нажатием на любую клавишу), а затем экран очищается и выводится изображение для следующего шага.

Запустим программу и посмотрим, что увидел бы «наш» наблюдатель при заданных в строках 750—960 уравнениях перемещения объекта. Сначала, до 18-й секунды (время здесь и далее — «наше») все идет как обычно: объект смещается по горизонтальной прямой. Но вот начиная с 19-й секунды справа появляется еще один объект, который тут же раздваивается. Теперь на экране присутствуют одновременно три объекта — «копии», из которых один движется по заданной нами траектории, другой перемещается по той же траектории навстречу первому, а третий смещается по горизонтали и на 41-й секунде скрывается за правым краем экрана. Оставшиеся две «копии» встречаются на траектории и исчезают.

Кто-то из читателей, возможно, скажет: ну и что — нарисовать любое количество движущихся по экрану прямоугольников может даже начинающий программист на БЕЙСИКе. Но ведь мы сами задали в уравнениях пространственное движение ОДНОГО объекта, а его «фантомы» образовались самопроизвольно — как результат нарушения причинно-следственных связей! Читатели могут сами поэкспериментировать с программой, задавая различные уравнения четырехмерной траектории. Может быть, кому-нибудь удастся наблюдать еще более удивительные эффекты.

В заключение хотелось бы отметить очень часто встречающийся в сообщениях очевидцев появления «летающих тарелочек» факт: НЛО могут совершенно произвольно «дублироваться», вновь сливаться, исчезать и появляться, резко менять направление движения в пространстве и т. д. Материальные тела, обладающие определенной массой, инерцией и прочностью, вряд ли способны к таким «пертурбациям». Но как это похоже на то, что мы только что наблюдали на модели! Так, может быть, и НЛО являются «фантомами» четырехмерного «нечто», порожденными неоднозначностью проекции «чужого», меняющегося времени на «наше»?

Уважаемые читатели!

Мы обращаемся ко всем, кто так или иначе связан с БК: к изготовителям самого компьютера и периферийных устройств для него, к кооперативам и клубам пользователей БК, ко всем профессиональным разработчикам программного обеспечения и аппаратных устройств к БК и к программистам-любителям, к нашим авторам, наконец, просто к читателям журнала. Мы с большим вниманием относимся к любому вашему письму, статье, телефонному звонку — даже к сердитым, критическим, пустым и невыполнимым по запросам.

Однако, приглашая вас к сотрудничеству, следует поговорить и о том, как осуществить это сотрудничество наиболее рациональным путем, чтобы снизить затраты сил и времени как на подготовку материалов, так и на их публикацию и одновременно уменьшить вероятность появления ошибок.

Рекомендация тем, кто пишет нам впервые. Статью нужно начинать с объяснения, для кого и для чего она написана, на каком конкретно компьютере вы работаете. При описании программ нужно указать, чем они отличаются от ранее опубликованных (если таковые были), описать их особенности. Не бойтесь перегрузить материал подробностями и формулами: важно, чтобы все было понятно даже новичку, впервые увидевшему БК. Лишнее редакция уберет.

Всех, обращающихся в редакцию, просим указывать в письме полностью и разборчиво: имя, фамилию и отчество, адрес (с индексом), паспортные данные, телефон (служебный и домашний), год рождения. Просим также предоставлять нам (письменно) право помещать в конце вашей статьи сведения об авторе (адрес, телефон и т.п.) или, наоборот, предупреждать, что этого делать не следует.

Что касается содержания присылаемых материалов, прежде всего для нас (и для читателей!) интересны статьи, содержащие работоспособные листинги программ и (или) принципиальные схемы аппаратных разработок. Не менее важны материалы по методике алгоритмизации и программирования и прочие «теоретические изыски, опирающиеся на практику»: справочные данные, исследования схемотехники БК и содержимого его ПЗУ, опыт работы с БК и периферией и др.

Лучше всего отпечатать текст статьи на принтере через два интервала. Иллюстрации и таблицы желательно выполнять каждую на отдельном листе. То же относится и к листингам программ. Отправляемые материалы (2 экз. — на бумаге) должны быть выверены (программы — отлажены). Необходимо оставить у себя копию.

Из программ в первую очередь рассматриваются распечатанные на принтере. Все, что вручную переписано с экрана, как правило, содержит ошибки, а следовательно, может потребовать серьезной доработки, что, в свою очередь, еще больше увеличивает время подготовки статьи к печати.

Известно, что при перенаборе ошибки неизбежны. Поэтому лучше всего предоставлять редакции файлы на магнитных носителях (носители возвращаются автору). Тексты статей могут быть записаны на дискетах IBM 3,5' и 5,25' (Лексикон, Word), на дискетах БК (операционные системы ANDOS, MKDOS, NORD или Нортон) или на кассетах БК (в двух последних случаях — текстовые редакторы TED, VorteX! или EDASP). Если вы не имеете возможности преобразовывать в обычный текст листинги на БЕЙСИКе и ФОКАЛе, а также кодовые программы в текстовые таблицы машинных кодов, лучше всего переслать на диске или кассете сами программы. При всем этом на дискете желательно иметь резервную копию (в специально созданном для этого подкаталоге или под другими именами), на кассете же наличие не менее двух копий обязательно.

Аналогичным способом могут быть подготовлены иллюстрации к статье. Формат их представления — файл, содержащий копию экранного ОЗУ (начиная с адреса 40000 или 42000, т. е. с учетом служебной строки или без нее), причем рулонное смещение должно быть нормализовано (адрес 40000 должен соответствовать верхнему левому углу изображения). При выполнении рисунков в графических редакторах серии GRAF или БК-PAINT редакции могут быть переданы файлы, записанные в формате этих программ. И наконец, при подготовке иллюстраций на IBM-совместимых компьютерах желательно применять форматы .PCX (черно-белый вариант) или TIFF. При использовании иных графических форматов или векторных редакторов типа AUTOCAD или PCAD необходимо обязательно предостиавить редакции четкую распечатку каждой иллюстрации на бумаге; во всех прочих случаях наличие распечатки рисунков также желательно.

Почтовый адрес: 125315, Москва, а/я 17.
Телефон: (095) 292-53-87
E-Mail: mail@infoobr.msk.su

СОДЕРЖАНИЕ

ПЕРСОНАЛЬНЫЙ КОМПЬЮТЕР БК-0010 – БК-0011М

Главный редактор
Васильев Б. М.

Редактор
Усенков Д. Ю.

Корректор
Антонова В. С.

Компьютерная верстка
Усенков Д. Ю.

Наш адрес:
Москва, м. «Охотный ряд», ул. Тверская, д. 5/6, Международная Академия Информатизации, комн. 308

Адрес для переписки:
125315, Москва, а/я 17

Телефон:
(095) 292-53-87

E-Mail (RELCOM):
mail@infoobr.msk.su

Указанный выше адрес для переписки действителен до 1.06.96 г. Новый почтовый адрес будет опубликован в одном из следующих выпусков.

ПЕРСОНАЛЬНЫЙ КОМПЬЮТЕР БК-0010 - БК-0011М

Подписано в печать с оригинал-макета издательства «Компьютика» 10.02.96.
Формат 60×90 1/16. Бумага офсетная.
Усл.печ.л. 6,5. Заказ №
Цена 6500 руб. (по подписке).
В розничной продаже цена договорная.

Отпечатано в типографии «Принт-Сервис»

Планируемое содержание выпуска 2'96:

- Описание драйверно-мониторной системы БК-0011
- Самодельный фотопистолет для БК
- Новая операционная система DX-DOS
- Многофункциональный АЦП для БК
- Принтер D100: руководство пользователя
- Использование прерывания по Т-разряду в копировщиках

Редакция журнала «Персональный компьютер БК-0010 — БК-0011М» с целью изучения покупательского спроса начинает прием от читателей предварительных заказов на брошюру «Ассемблерный листинг и описание работы драйверно-мониторной системы компьютера БК-0010(.01)». Ориентировочная стоимость издания — 10000 руб. (без учета почтовой пересылки), приблизительный срок выхода из печати (при наличии достаточного количества заказов) — декабрь 1996 г.

Заказы на брошюру
можно присылать по адресу:
125315, Москва, а/я 17
или оформить лично в помещении редакции.

Уважаемые читатели!

В порядке эксперимента редакцией выпущено дисковое приложение к шестому выпуску журнала за 1995 г. На диске 800 кб (формат ANDOS) содержатся работоспособные листинги опубликованных в №6'95 программ и авторская документация по описываемым программным разработкам (NORD-BASE v2.3, NORD для винчестера, БК-PageMaker и ANIRAM ZIPPER). По вопросам приобретения дискового приложения обращайтесь в редакцию по телефону (095) 292-53-87.