ПЕРСОНАЛЬНЫЙ КОМПЬЮТЕР БК-0010 БК-0011м

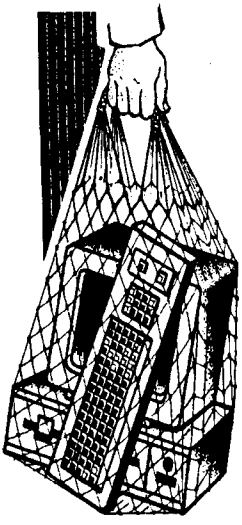
Приложение к журналу «ИНФОРМАТИКА И ОБРАЗОВАНИЕ»

3'94

УВАЖАЕМЫЕ ЧИТАТЕЛИ!

У нас в редакции вы можете приобрести отдельные выпуски журналов «Персональный компьютер БК-0010 — БК-0011М», «Персональный компьютер УКНЦ» и «Информатика и образование». Здесь же можно оформить полугодовую подписку на все перечисленные издания с получением экземпляров лично в редакции или по почте. У нас вы также можете приобрести различное программное и аппаратное обеспечение для IBM, БК и УКНЦ.

БК:



Дисковая операционная система NORD для БК-0010(.01) и БК-0011(М).

Универсальный программатор ППЗУ.

Профессиональный просмотрщик (вьювер) копий экрана БК-0010 для переноса графики на IBM.

Конвертор листингов на вильнюсском БЕЙСИКе в текстовые файлы формата EDASP и обратно.

Конвертор программ из внутреннего формата ФОКАЛа в листинги стандарта EDASP.

Резидентный драйвер ANIRAM-MIRIADA, дополняющий ОС ANDOS следующими возможностями:

— копирование экрана в файлы;

— сопровождение диалога с БК на принтере;

— поддержка дополнительного устройства «P:» — «принтер».

Широкий набор драйверов и утилит (в том числе для принтера D100) и многое другое.

IBM-совместимые:

Графический редактор с возможностью генерации программного кода для компиляторов Borland.

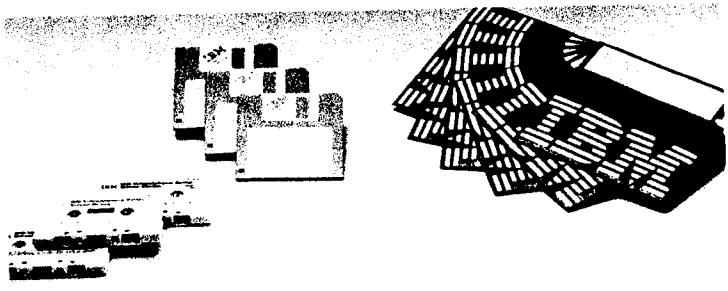
Система автоматической генерации спецификаций в соответствии с требованиями ГОСТ для ППП PCAD.

Геометрическая система проектирования обводов самолета ГЕМОС.

Самоучитель по MS-DOS — система АБАК.

Игра «Тайна шести сундуков» для изучения темы «Правила правописания падежных окончаний в русском языке».

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.



Телефон: 151-19-40
Факс: 208-67-37
E-Mail: mail@infoobr.msk.su

ПЕРСОНАЛЬНЫЙ КОМПЬЮТЕР БК-0010 БК-0011м

Приложение
к журналу
«ИНФОРМАТИКА
И ОБРАЗОВАНИЕ»

3'94 (4)

Издается с 1993 г.

В НОМЕРЕ

Москва «Информатика и образование» 1994

Авторы выпуска

Автухов Р.
Балло А. В.
Домолазов А. В.
Ермаков В. В.
Животовский П. П.
Зальцман Ю. А.
Карп А. П.
Ланеев А. С.
Милюков А. В.
Разбитной С. А.
Руденко В. В.
Сыченко А. Н.
Усенков Д. Ю.
Христофоров И. В.
Чумак А. В.
Щербина Н. Ю.
Юров В. П.

Редакторы: Васильев Б. М.
Усенков Д. Ю.

ISSN 0236-1809 «Библиотека журнала «Информатика и образование»
Свидетельство о регистрации средства массовой информации № 0110336
от 26 февраля 1993 г.

Телефон: (095) 151-19-40, 208-30-78
E-Mail: mail@Infoobr.msk.su
Факс: (095) 208-67-37

Продолжаем публикацию неформального учебника по программированию на ассемблере для начинающих пользователей БК-0010(.01). Первые уроки этого компьютерного языка см. в №1, №2 за 1994 г.

Ю. А. Зальцман,
г. Алма-Ата

МИКРОЭВМ БК-0010. АРХИТЕКТУРА И ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ АССЕМБЛЕРА

Способы адресации

Ранее мы установили, что в команде ассемблера ОПЕРАТОР указывает, ЧТО НАДО СДЕЛАТЬ, а операнд — С ЧЕМ ЭТО НАДО СДЕЛАТЬ. Как уже было сказано, поле операндов не обязательно (оно имеется не во всех командах). Но если их нет, это означает только то, что данный оператор в операндах не нуждается — они подразумеваются по умолчанию или производится обращение не к памяти ЭВМ, а к другим ее устройствам.

СПОСОБЫ (или МЕТОДЫ) адресации — это не что иное, как указание на ячейки памяти, с которыми должен манипулировать оператор. Таких методов в системе команд процессора БК вполне достаточно, чтобы обеспечить высокую гибкость в построении программы. Многие способы адресации, кроме указания на адреса памяти, еще и модифицируют их, т.е., по сути, выполняют функции дополнительных команд, а следовательно, значительно экономят память и повышают скорость исполнения программы. В большинстве случаев способы адресации одинаковы для всех команд, содержащих операнды (некоторые ограничения, относящиеся к отдельным операторам, будут оговорены особо).

Прежде чем перейти к изложению конкретных способов адресации, кратко напомним, какие в нашей ЭВМ вообще возможны операнды. Прежде всего, это ячейки памяти, лежащие в пределах адресного пространства ЭВМ (с адресами 0 — 177777). Адрес можно указывать с точностью до слова или до байта, но это уже скорее относится к формату записи оператора, а не операндов. Далее, это регистры общего назначения процессора (РОН): R0, R1, R2, R3, R4, R5, R6 и R7 (на языке ассемблера регистр R6 принято именовать SP, а R7 — PC). Напомним, что SP — это указатель вершины стека, а PC — счетчик команд (указатель адреса следующей команды). Подробно эти моменты рассматривались в разделах, посвященных архитектуре ЭВМ, сейчас же важно, что все регистры, в принципе, также представляют собой ячейки памяти.

Теперь можно перейти к описанию собственно способов адресации. Приводя примеры, будем стараться использовать только те два оператора (CLR и MOV), которые нам уже знакомы. Регистр общего назначения в общем виде будем обозначать RN (это может быть любой из РОН, однако использование в качестве операндов SP и PC может иметь некоторые особенности).

1. РЕГИСТРОВАЯ адресация (иногда ее называют еще РЕГИСТРОВОЙ ПРЯМОЙ, в отличие от косвенной). Операндом является один из РОН. Пример:

```
MOV R2,R4   Переслать (скопировать)
            ; содержимое R2 в R4
CLR R2      Очистить (обнулить) R2
MOV R4,SP   Переслать содержимое R4 в SP
```

Это самый простой способ адресации и, очевидно, в дальнейших пояснениях он не нуждается.

2. РЕГИСТРОВАЯ КОСВЕННАЯ адресация. В РОН содержится не само число, с которым нужно работать, а ЕГО АДРЕС, т.е. номер ячейки памяти, в которой оно находится. Обозначение: @RN или (RN). Значок «@» — это так называемое «коммерческое ЭТ», название длинное и труднопроизносимое, поэтому в программистском жаргоне его заменяет не слишком благозвучное, но зато удобное наименование «собачка». Обозначения @RN и (RN) абсолютно равнозначны, но лучше все же писать @RN, чтобы не путать его с адресацией с инкрементом или декрементом, о которых речь пойдет ниже. Примеры:

1) пусть в регистре R2 записано число 1000, тогда команда CLR @R2 обнулит ячейку 1000 (ту, адрес которой указан в R2);

2) пусть в R2 записано число 1000, а по адресу 1000 — число 1254. Тогда MOV @R2,R4 запишет в R4 число 1254, ведь именно его адрес указан в R2.

3. АВТОИНКРЕМЕНТНАЯ КОСВЕННАЯ адресация. Помимо основного действия (косвенного обращения к ячейке памяти) производится изменение (МОДИФИКАЦИЯ) адреса этого обращения (ячейки). Слово «ИНКРЕМЕНТ» означает увеличение чего-либо, в данном случае адреса ячейки памяти, к которой мы обращаемся (содержимого регистра, служащего указателем адреса). А «АВТО» — это указание на то, что данное увеличение происходит само, не требуя отдельной команды. Обозначение: **(RN)+**. То, что знак «+» стоит ПОСЛЕ имени регистра, указывает на порядок выполнения команды: сначала что-то делается с ячейкой, адрес которой помещен в регистр RN, а потом содержимое регистра увеличивается. На сколько — зависит от формы оператора: если он работает с байтом (т.е. записан с буквой «B»), то на 1 (переход к адресу следующего байта), а если со словом, то на 2 (переход к адресу следующего слова). Но из этого правила есть исключение: если в качестве указателя адреса используется регистр SP (адресуются ячейки стека), то автоинкремент делается ТОЛЬКО НА ДВА, независимо от формы оператора (то же относится и к регистру PC, но адресация через PC – особая тема для разговора).

Пример: пусть в регистре R2 записано число 1000. Тогда команда **CLR (R2)+** очистит СЛОВО по адресу 1000, а содержимое регистра R2 станет равно 1002. Если затем выполнить команду **CLRB (R2)+**, то очистится БАЙТ по адресу 1002 (младший байт слова), а содержимое R2 станет равно 1003 (R2 указывает на старший байт).

Данный способ адресации обычно применяется для работы с массивами слов или байтов. Если записать в регистр начальный адрес массива, а затем выполнить команду с автоинкрементной адресацией столько раз, сколько элементов в массиве, то команда (одна и та же, выполняемая, например, в цикле) каждый раз будет работать со следующим элементом — обнулять его, пересылать куда-либо и т.п. Если оператор имеет два операнда, как, например, MOV, то автоинкрементную адресацию можно использовать для каждого из них. Пусть в R2 записано число 1000, а в R4 — 2000. Тогда команда **MOV (R2)+,(R4)+** перешлет содержимое ячейки 1000 в ячейку 2000, после чего числа в R2 и R4 будут равны соответственно 1002 и 2002.

Отдельный, «хитрый», случай — автоинкремент регистра PC. Пусть, например, после команды хранится некоторая константа

```
MOV (PC)+,R4
<КОНСТАНТА>
```

Что произойдет? <КОНСТАНТА> запишется в регистр R4, ведь в момент исполнения команды в PC содержится адрес следующей команды, т.е. числа <КОНСТАНТА>! А затем PC увеличится на 2, т.е. управление будет передано уже на СЛЕДУЮЩУЮ после константы команду.

4. АВТОИНКРЕМЕНТНАЯ ДВОЙНАЯ КОСВЕННАЯ адресация. Этот способ не слишком отличается от предыдущего, только в регистре-указателе при этом содержится не адрес числа, с которым должен работать оператор, а АДРЕС АДРЕСА. Обозначение: **@(RN)+**.

Пример: пусть в ячейке 1000 записано число 1254, а по адресу 1254 — число 3333. Тогда, если в R2 хранится число 1000, то команда **MOV @(R2)+,R4** перешлет в R4 число 3333, а содержимое регистра R2 станет равным 1002.

Данный способ применяется относительно редко, но тем не менее он очень ценен, так как позволяет работать, например, с таблицами адресов. Пусть каждый элемент таблицы (а) является адресом одно-

И. В. Христофоров

ОПРЕДЕЛЕНИЕ ПОЛОЖЕНИЯ КУРСОРА В РЕЖИМЕ ГРАФ

Предлагаемое небольшое исследование касается вопроса определения позиции курсора в режиме ГРАФ по данным, хранящимся в системных ячейках БК-0010 (.01). На мой взгляд, эта информация будет интересной многим программистам БК, так как среди набора стандартных EMT-функций такая возможность имеется только для обычного текстового курсора (не в режиме ГРАФ).

При включенном режиме ГРАФ (когда в байте с адресом 50_8 записано ненулевое значение) содержимое ячейки 156_8 определяет положение прямоугольной области размером 8*10 точек (для режима 64 символа в строке) или 4*10 точек (для 32 символов), в которой находится курсор. Фактически данная ячейка содержит информацию о текстовой позиции, в пределах которой находится графический курсор. При этом ее текущие координаты хранятся в ячейке 156_8 в следующем формате:

биты 0-5 — координата X (от 0 до 77_8) для режима 64 символа в строке или 2*X от (0 до 37_8) для 32 символов;

биты 6-10 — координата Y (0-27_8).

Дальнейшее уточнение позиции графического курсора (в пределах уже найденной прямоугольной области) позволяют произвести ячейки 154_8, 166_8, 170_8 и 174_8. Младший байт ячейки 154_8 определяет относительную координату X в пределах области, старший (который можно рассматривать как отдельную ячейку с адресом 155_8 —

го из элементов таблицы (б). Данная адресация позволяет, «двигаясь» по элементам первой таблицы, проверять элементы второй, порядок следования которых может быть произвольным (не совпадать с порядком следования их адресов).

5. АВТОДЕКРЕМЕНТНАЯ КОСВЕННАЯ адресация. Если сообщить, что «ДЕКРЕМЕНТ» означает УМЕНЬШЕНИЕ чего-либо, то этот способ будет почти понятен, не правда ли? У него только два отличия от автоинкрементной косвенной адресации: во-первых, адрес в регистре-указателе не увеличивается, а уменьшается; во-вторых, это происходит не ПОСЛЕ выполнения команды, а ДО нее, на что указывает запись **-(RN)**, в которой знак «-» стоит перед обозначением регистра. В остальном же сходство с автоинкрементным способом полное, включая то, что при операциях с байтом модификация числа в регистре делается на 1, а со словом — на 2, и что регистр SP всегда модифицируется только на 2.

Пример: пусть в регистре R2 записано число 1002. Тогда команда **CLR -(R2)** очистит ячейку 1000 (ту, адрес которой на 2 меньше, чем записанное в регистре число), а содержимое регистра после операции, естественно, окажется равным тоже 1000 — оно стало таким еще ДО очистки ячейки. Видимо, дальнейших примеров не требуется, по аналогии с автоинкрементным способом все и так ясно. Этот способ позволяет делать, в общем, то же, что и автоинкрементный, но при работе с массивом его просмотр или пересылка производится «с конца», а перед началом операции в регистре должен быть записан адрес следующей за последним элементом массива ячейки.

Однако этот способ позволяет делать и нечто более интересное — например, перемещать числа в памяти. Пусть в R2 записано число 2004, а по адресу 2002 — число 1254. Тогда команда **MOV -(R2),-(R2)** поместит число 1254 в ячейку 2000, т.е. как бы «передвинет» его в памяти с большего адреса на меньший. (Разумеется, по адресу 2002 число 1254 тоже сохранится.) Как это произошло? Вначале содержимое R2 уменьшилось на 2, став равным 2002, и регистр указал на число 1254. Теперь оно должно быть записано по адресу второго операнда, но его указатель — тот же регистр, и опять вначале его содержимое уменьшится на 2, станет равно 2000, а уже потом туда будет записано число. Нетрудно догадаться, что таким способом можно «сдвигать» массивы в памяти после удаления в середине какого-либо элемента (аналогичное применение может найти и команда **MOV (R2)+,(R2)+**).

* * *

А теперь — маленькое развлечение и одновременно интересный пример по практике использования автодекрементной адресации. Запишем команду **MOV -(PC),-(PC)** (код 14747). Для большего эффекта поместим ее в старшие адреса ОЗУ, в ячейку 77776 (с помощью директив МСД это делается просто и безо всякого ассемблера: 77776A14747И), а затем передадим на нее управление (запустим «программу» по адресу этой команды: 77776G). Что случилось?! Экран мгновенно стал «полосатым», а ЭВМ... Она «мертва»: не отвечает на нажатие клавиш и даже на такое «мощное лечебное средство», как клавиша «СТОП», никак не реагирует! Только системный сброс (или выключение, а затем включение питания) «приводит ее в чувство». Что же это за «бомба», которая мгновенно парализовала все средства ЭВМ?

После запуска в регистре PC содержится адрес следующей команды — число 100000. При обращении к первому операнду содержимое PC уменьшается на 2, давая адрес 77776, а там записан код

Прим. ред.)— «ширину точек» и косвенным образом — режим 32/64 символа в строке (его значение равно, соответственно, 3 или 1).

В режиме 64 символа номер бита, содержащего 1, в байте 154_8 определяет относительную координату X текущей точки внутри прямоугольной области. Так, если содержимое байта 154_8 равно 00010000_2, курсор указывает на пятую точку по горизонтали внутри области. В режиме же 32 символа каждая точка кодируется парой битов. Например, если в байте 154_8 записано 11000000_2, это означает четвертую точку (в данном режиме биты могут устанавливаться в 1 только попарно: 0+1, 2+3, 4+5 и 6+7).

Роль байта с адресом 154_8 в БК следующая. После операции логического умножения на маску цвета (ячейка 214_8) значение накладывается на соответствующий байт экранной памяти, адрес которого, в свою очередь, определяется содержимым ячейки 170_8. При этом на экране в текущей позиции графического курсора формируется цветная точка. (В режиме 64 символа в строке маска цвета не используется.)

Теперь поговорим об определении вертикальной координаты графического курсора (Y). Ее абсолютное значение (от 0 до 357_8) хранится в битах 6—13 ячейки с адресом 166_8, а биты 6—9 ячейки 174_8 определяют относительную координату в пределах текущей текстовой позиции (прямоугольной области). И наконец, адрес текущего байта экранного ОЗУ, записанный в ячейке 170_8, позволяет однозначно получить абсолютную координату Y графического курсора, если учесть рулонное смещение.

команды — число 14747. А куда оно перепишется? По адресу второго операнда, но перед этим содержимое PC уменьшится еще на 2 и станет равно 77774, т.е. укажет уже на следующую («нижележащую») ячейку памяти. А потом, раз теперь в регистре PC адрес 77774, то туда и будет передано управление, а там снова та же команда, тот же «смертельный» код! Команда как бы «размножается» в памяти ЭВМ, причем со страшной скоростью: чтобы заполнить собой всю память, ей достаточно всего лишь 0.2 секунды! А почему ЭВМ отказывает? Это станет ясно, если вы вспомните, что в зоне адресов 0...777 размещается системная область и стек ЭВМ, ее «собственная память». А если стереть содержимое памяти любого «существа», пусть даже такого простого, как БК, это если не смерть, то безумие...

Но что же это все-таки за команда, зачем мы потратили на ее разбор столько времени, и для чего она может пригодиться? Вы, наверное, слышали о компьютерных вирусах, но, возможно, еще не встречались ни с одним из них. Так вот, позвольте представить: MOV -(PC),-(PC) — простейший вирус (точнее, еще не настоящий вирус, такие программы носят название «червяк»), вызванное им «заболевание» развивается молниеносно и, безусловно, «смертельно». Эта команда уничтожает всю информацию, хранимую в ОЗУ по адресам, меньшим, чем адрес самой команды. А зачем может понадобиться такая страшная команда? Существует ряд случаев, когда нужно срочно уничтожить хранимую в ОЗУ информацию. Например, чтобы в написанной вами игре после запуска разрешалось только определенное число попыток, а потом ее нужно было снова загружать с МЛ (что затруднит нахождение решений), поместите в любом месте программы этот вирус и, когда число попыток исчерпано, заставьте его «ожить» (такой «до поры до времени спящий» вирус носит название «троянский конь»). Сложнее, но тоже возможно реализовать активизацию вируса, например, по истечении заданного времени, после определенного числа копирований программы на МЛ (если она имеет встроенный копировщик) или при попытке что-либо изменить в программе. (Известно, что «настоящие» вирусы на IBM-совместимых компьютерах были изначально созданы для наказания «компьютерных воришек», взламывающих защиты от копирования. — *Прим. ред.*) Как видим, даже вирусы можно заставить работать с пользой. Но это еще не все. Изучая вирусы, создавая их и борясь с ними, программисты совершенствуются уже не в ремесле, а в искусстве. Последнее тоже приносит пользу, хотя отнюдь не перекрывает вред, приносимый вирусами. (Рассуждения о полезности вирусов составляют личное мнение автора статьи. На наш взгляд, написание вирусов можно рассматривать скорее как хулиганство и стремление напакостить всем без исключения (ведь вирус «бьет» бесприцельно!). Учиться же программировать лучше на других, не менее интересных и полезных, но безвредных вещах. — *Прим. ред.*)

Еще одно применение тот же принцип — «уничтожай, размножаясь» — находит в оригинальной компьютерной игре «бой в памяти»: в ОЗУ загружаются специальные «боевые программы», и особый монитор попеременно передает им управление, разрешая делать по нескольку «ходов» по очереди. Цель — «вывести из строя» программу противника. Эта игра была очень популярна среди программистов и любителей за рубежом несколько лет назад, но сейчас интерес к ней значительно упал. (О том, как реализовать на БК подобный «многопрограммный режим» с параллельной работой нескольких программ, можно прочитать в журнале «Информатика и образование» №2 за 1992 г. — *Прим. ред.*)

* * *

6. АВТОДЕКРЕМЕНТНАЯ ДВОЙНАЯ КОСВЕННАЯ адресация. Обозначение: **@-(RN)**. Если сделать уже известные нам поправки, что декремент — это уменьшение, и выполняется оно ДО обращения к ячейке, на которую указывает операнд, то из автоинкрементной двойной косвенной адресации легко получается автодекрементная. Далее, не детализируя, приведем пример. Пусть в регистре R2 записано число 1002, по адресу 1000 — число 1254, а по адресу 1254 — 3333. Тогда команда **MOV @-(R2),R4** запишет в R4 число 3333. Данный способ может быть применен для тех же целей, что и автоинкрементная двойная косвенная адресация, но дает возможность, скажем, просматривать таблицу адресов с конца.

Отметим, что такого, казалось бы, логичного способа как «двойная косвенная адресация», записать которую можно было бы, например, как **@(RN)**, не существует. А жаль — он бы тоже мог пригодиться... Но еще не все потеряно, мы можем «обмануть» ассемблер и все-таки задать адресацию таким образом! Как именно — увидим позже.

Кроме того, заметим, что все ранее рассмотренные способы адресации «вписывались» в формат самой команды, и, таким образом, она при переводе в машинный код занимала одно слово. Способы, описываемые ниже, таким преимуществом не обладают. Они требуют задания некоторых дополнительных параметров, в формат одного слова никак не укладывающихся. Такие команды занимают два, а то и три машинных слова, однако потеря объема памяти компенсируется еще большей гибкостью и универсальностью средств программирования на ассемблере. Но прежде чем перейти к описанию этих способов адресации, введем еще два общих обозначения: **X** — восьмеричное число, **МЕТ** — произвольное имя метки.

7. ИНДЕКСНАЯ адресация. Это разновидность косвенной адресации через регистр. Обозначение: **X(RN)** или **MET(RN)**. Стоящее перед регистром восьмеричное число X или имя метки MET называется ИНДЕКСОМ, указывает СМЕЩЕНИЕ от адреса, содержащегося в регистре, и заносится при трансляции вторым или третьим словом команды. Восьмеричное число может быть записано как без знака, так и (в последних версиях ассемблеров) со знаком «-» (в этом случае при трансляции оно автоматически переводится в дополнительный код). Вместо имени метки ассемблер тоже подставляет число — ее абсолютный адрес, полученный в результате трансляции и компоновки. Адрес же ячейки, к которой мы обращаемся, при данном способе адресации определяется как сумма смещения (индекса) и содержимого регистра RN (так называемой БАЗЫ). Пример: пусть в регистре R2 записано число 2000, тогда команда **CLR 1000(R2)** обнулит слово по адресу 3000, а **CLR -1000(R2)** — слово по адресу 1000.

Этот способ может быть использован, если при одном и том же содержимом регистра, которое нежелательно почему-либо изменять, нужно выполнить действия с несколькими ячейками, отстоящими от заданного в регистре адреса на некоторые фиксированные «расстояния». Например, если задать в R2 адрес какого-либо байта экранного ОЗУ, то команда **CLRB 100(R2)** очистит байт, находящийся в следующей телевизионной строке (под заданным), а **CLRB 1200(R2)** — байт, лежащий ниже на 10д строк.

Еще интереснее случай использования в качестве индекса имени метки. Предположим, что нам нужно вместо заносимых в регистр R2 чисел (например 0, 1, 2, ... 7) подставить значения 4, 3, 2, 1, 0, 7, 6 и 5 соответственно. Эта задача называется ПЕРЕКОДИРОВКОЙ. Можно решить ее «в лоб» — проверяя каждый раз содержимое регистра на соответствие ВСЕМ значениям ряда по очереди. Но такой способ ужасно неэкономичен как по времени, так и по занимаемому объему ОЗУ (представьте, что возможных кодов было бы 1000). Лучше применить ТАБЛИЧНЫЙ метод. Напишем программу, состоящую всего из одной (!) команды **MOVB TAB(R2),R2** и составим таблицу перекодировки, которая в нашем примере будет иметь вид:

```
TAB:  .B:4,3,2,1,0,7,6,5
```

(Забегая вперед, поясним, что псевдокоманда **.B:** позволяет записать ряд чисел в последовательные байты памяти, начиная с текущего адреса, в нашем случае — с адреса метки **TAB**.)

Поставленная задача решена. Пусть, например, в регистре R2 содержится число 3. Адрес первого операнда команды будет вычислен как сумма адреса метки TAB и содержимого регистра. Отсчитаем, начиная с нуля, третий байт таблицы — это число 1. Оно и будет занесено вместо исходного числа 3 в регистр R2 — перекодировка выполнена.

Вообще же круг подобных задач весьма широк — от замены символов текста (например, если набор символов, которые печатает принтер, ограничен или их порядок в знаковой таблице не соответствует принятому в БК-0010) до вычисления значений произвольной функции по ее аргументу табличным методом. (Закодировав значения синуса и косинуса для углов от 0 до 2π с некоторым шагом, таким способом можно реализовать вычисление координат точек окружности (или дуги) с заданным радиусом и, соответственно, ассемблерную подпрограмму для ее вычерчивания. — *Прим. ред.*)

8. ИНДЕКСНАЯ КОСВЕННАЯ адресация. Обозначение: **@X(RN)** или **@MET(RN)**. По смыслу она близка к предыдущей, но в ячейке, определяемой суммой индекса и содержимого регистра, теперь находится не само число-операнд, а ЕГО АДРЕС. С таким явлением — двойной косвенностью — мы уже встречались, поэтому приводить примеры не будем (читатель может придумать их сам), а лучше выполним данное ранее обещание — покажем, как реализовать через регистр двойную косвенную адресацию, которая в наборе команд ЭВМ «официально» не существует. Как уже говорилось, логично было бы представить запись такой адресации в виде **@(RN)**, мы же вместо этого запишем **@0(RN)**. Цель достигнута — мы обратились к ячейке с двойной косвенностью и без модификации содержимого регистра-указателя. Правда, за этот «обман» придется платить — команда занимает целых два машинных слова, хотя содержимое второго из них всего лишь нуль.

Индексная косвенная адресация применяется редко, но иногда без нее почти невозможно обойтись. Например, когда нужно не просто перекодировать последовательность чисел, а сравнить ее с другой последовательностью (возможно, тоже перекодированной!). С такой необходимостью вы непременно столкнетесь, если захотите написать программу сортировки по алфавиту русских слов — ведь символы русского алфавита в БК не упорядочены по возрастанию кодов.

9. НЕПОСРЕДСТВЕННАЯ адресация. Этот способ позволяет в качестве одного из операндов использовать число — константу, которая записывается как **#X** или **#MET**. Значок «#», называемый на программистском жаргоне «решеткой», в ассемблере как раз и обозначает «число». Восьмеричное число-операнд может быть без знака или со знаком «-», а имя метки ассемблер при трансляции и компоновке заменяет на ее абсолютный адрес, как и в двух предыдущих способах. Полученный при этом числовой операнд записывается вторым или третьим словом команды.

Ясно, что при этом способе часть команд, формально возможных, теряет смысл. Например, практически бесполезна операция CLR #1000. Что мы тут можем обнулить? Только второе слово самой команды, а зачем? Или, скажем, команда MOV #1000,#2000. Она всего лишь перешлет второе слово по адресу третьего. Трудно вообразить программу, в которой это требуется... Но пора, наверное, перейти к примерам «более практического свойства». Пожалуйста:

```
MOV   #1000,R2     Запись в R2 числа 1000
MOVB #40,R0        Запись в R0 числа 40
```

Обратим внимание, что во второй строке мы записываем число 40 в регистр R0 с помощью «байтовой» команды. Как это понимать, разве мы можем адресоваться не ко всему регистру, а к его части? Да, можем. Если при регистровой адресации применена байтовая команда, используется МЛАДШИЙ байт регистра. Раз так, то, казалось бы, его старший байт при этом меняться не должен. Но не тут-то было! Если с младшим байтом оперирует команда MOVB, происходит не всегда полезная вещь, называемая РАСПРОСТРАНЕНИЕМ ЗНАКА — все биты старшего байта регистра-приемника (второго операнда) принимают значение знакового разряда младшего байта (т.е. разряда 07). В приведенном примере (MOVB #40,R0) все разряды старшего байта R0 обнулятся, даже если мы не преследовали такой цели. Если выполнить команду MOVB #377,R0, в регистре R0 окажется не 377, а 177777 (во все разряды старшего байта R0 будут занесены единицы). А вот после MOV #377,R0 в R0 окажется число 377, как и следовало ожидать. Вообще же распространение знака возникает, если число в младший байт регистра будет записано с применением любого способа адресации в команде MOVB. Это, видимо, единственный случай, когда младший байт слова оказывает какое-то влияние на старший. (Кроме «явных» ассемблерных команд: обмена местами старшего и младшего байтов (SWAB) и НЕБАЙТОВЫХ команд смещения вправо и влево (ASL, ASR, ROL и ROR). — *Прим.ред.*) При байтовых операциях с ячейками ОЗУ такого, конечно же, не бывает.

Продолжим примеры. Нижеследующая запись уже в какой-то степени напоминает осмысленный текст программы:

```
MOV   #TEX,R2      Запись адреса метки TEX в R2
MOVB #101,(R2)+    и запись в последовательные
MOVB #102,(R2)+    ячейки памяти, начиная с метки
MOVB #103,(R2)+    TEX, чисел 101, 102 и 103
                   ; (кодов символов "A","B","C")
```

При написании программ часто приходится с помощью непосредственной адресации засылать в регистры (реже — в ячейки памяти) коды символов (как это и сделано в приведенном примере). Для этого их надо помнить наизусть, или, по крайней мере, иметь под рукой таблицу кодов. Развитые версии ассемблеров (в том числе, конечно, и МИКРО.10К) предоставляют дополнительное удобство – возможность записи в качестве операндов самих символов. Это тоже разновидность непосредственной адресации. Символы (они могут быть любыми) должны быть заключены в апострофы, а их число не должно быть больше двух.

```
MOV 'a',R0         Примеры записи в регистр R0
MOV '12',R0        кодов символов:
MOV 'A-',R0        "a", "1" и "2", "A" и "-"
```

Если при такой адресации в апострофах записан один символ, то его код заносится в младший байт регистра (или ячейки памяти), если два — код первого заносится в младший байт, а второго — в старший. (Коды, не имеющие символьного представления, например перемещения курсора, управление выводом на дисплей и т.д., приходится вводить «по старинке» — с помощью #<КОД>. — *Прим. ред.*)

10. АБСОЛЮТНАЯ адресация. Этот способ прост и прямолинеен: прямо указывается абсолютный адрес ячейки памяти, к которой мы обращаемся. Обозначение: @#X. Восьмеричное число X помещается при трансляции во второе или третье слово команды. Пример:

```
CLR   @#156        Очистка слова по адресу 156
CLRB  @#45         Очистка байта по адресу 45

MOV   @#100112,R2  Запись содержимого слова
                   ; по адресу 100112 в R2
MOV   #137,@#1000  Запись числа 137 по адресу 1000
```

Данный способ достаточно широко применяется для обращения к ячейкам вне самой программы, например по адресам системной области или ПЗУ. Обращаться же к ячейкам, находящимся в ОЗУ пользователя (или тем более в самой программе), удобнее с помощью ОТНОСИТЕЛЬНОЙ адресации.

11. ОТНОСИТЕЛЬНАЯ адресация. Этот способ является, пожалуй, самым распространенным при задании адресов переходов и вызове подпрограмм (что мы рассмотрим в свое время), а также при обращении к ячейкам памяти (переменным), расположенным в ОЗУ пользователя, особенно в пределах самой программы. Обозначение: МЕТ. При таком обращении ассемблер во время трансляции вычисляет СМЕЩЕНИЕ (разность между адресами текущей команды и метки МЕТ) и записывает это смещение вторым или третьим словом команды. При исполнении программы адрес операнда определяется как сумма текущего содержимого счетчика команд PC и смеще-

ния. Последнее вычисляется в дополнительном коде с учетом знака, поэтому такая адресация возможна как «вперед», так и «назад» относительно текущей команды. Не вдаваясь в тонкости, которые больше волнуют создателей ассемблер-систем, чем их пользователей, можно сказать, что относительная адресация — это просто обращение к ячейке памяти, помеченной меткой МЕТ. Для ассемблеров БК-0010, имеющих механизм локальных меток, необходимо иметь в виду, что адресация (не только в этом, но и во всех других случаях, когда в составе операндов употребляется имя метки) возможна только с использованием обычных меток (начинающихся с буквы), а локальные пригодны лишь для указания адресов переходов в операторах ветвления и цикла. Это связано в основном с тем, что отличить локальную метку (начинающуюся с цифры) от числа (например, индекса) ассемблер-система не может. Пример:

```
CLR   A1          Адресация с использованием
MOV   R2,BR2      меток:
MOV   RGB,(R5)+   A1, BR2, RGB, TYP
MOVB  #377,TYP
MOVB  TYP,R4
```

Особо отметим, что метки, упоминаемые в операндах (при любом способе адресации), должны быть ОПРЕДЕЛЕНЫ, т.е. должны присутствовать в тексте программы, помечая собой какие-либо строки (или же должны быть заданы ОПЕРАТОРОМ ПРЯМОГО ПРИСВАИВАНИЯ, о котором речь пойдет дальше). При использовании имен неопределенных (отсутствующих) меток правильная компоновка программы невозможна.

12. ОТНОСИТЕЛЬНАЯ КОСВЕННАЯ адресация. Аналогична предыдущему способу, но по адресу, определяемому меткой, находится не само число-операнд, а, как обычно в случаях косвенной адресации, его АДРЕС. Обозначение: @МЕТ. Применяется весьма редко, например для передачи управления по адресу, который перед этим занесен в определенную ячейку памяти (т.е. по значению переменной). Практически всегда этот способ адресации может быть заменен на другой, (например регистровую косвенную адресацию), но иногда он довольно удобен, так как не требует использования регистра.

* * *

На этом можно было бы считать изучение способов адресации законченным и в заключение привести сводную таблицу по всем рассмотренным способам. Но сначала сделаем заявление, на первый взгляд, противоречащее всему вышеизложенному: теоретически возможных способов адресации всего восемь, в общем виде все они так или иначе используют один из регистров общего назначения, т.е. являются разновидностями регистровой адресации. Но позвольте, скажете вы, как же тогда быть с тем, что мы рассмотрели 12 способов, и в четырех последних имена регистров вообще не фигурируют? Дело в том, что эти четыре способа являются не самостоятельными, а лишь разновидностями первых восьми. Они выделены потому, что имеют большое практическое значение, и по этой же причине в языке ассемблера для них введены особые, «нестандартные» обозначения. Между тем, смысл этих способов сводится к тому, что в качестве регистра-указателя используется R7 (или PC) — счетчик команд. Проиллюстрируем это на примере. Как мы уже знаем, непосредственная адресация обозначается как #X, где X — число-операнд (для простоты не будем касаться меток). Возьмем конкретную команду: **MOV #1000,R2**. При трансляции ассемблер заносит число 1000 во второе слово. Но мы уже сталкивались с случаем, когда, применяя регистр PC и автоинкрементную адресацию, можно было прочитать записанную после команды константу! Итак, непосредственная адресация — это вовсе не новый метод, а все тот же самый способ с автоинкрементом PC. И вместо MOV #1000,R2 вполне можно записать:

```
MOV (PC)+,R2
.#1000
```

(Здесь .#1000 – это запись числа 1000 по текущему адресу.)

Совершенно так же можно «развенчать» и остальные три способа — абсолютной, относительной и относительной косвенной адресации (предоставляем это сделать читателю). Все это всего лишь удобные приемы записи текста программ на языке ассемблера, частные случаи «настоящих» способов адресации с использованием PC. Именно так мы их и приведем в таблице.

Основные способы адресации

Номер (код)	Наименование	Обозначение
0	Регистровая (прямая)	RN
1	Регистровая косвенная	@RN
2	Автоинкрементная косвенная	(RN)+
3	Автоинкрементная двойная косвенная	@(RN)+
4	Автодекрементная косвенная	-(RN)
5	Автодекрементная двойная косвенная	@-(RN)
6	Индексная	X(RN) или MET(RN)
7	Индексная косвенная	@X(RN) или @MET(RN)

Способы адресации с использованием PC

Код	Наименование	Обозначение	
		стандартное	специальное
2	Непосредственная	(PC)+	#X
3	Абсолютная	@(PC)+	@#X
6	Относительная	MET(PC)	MET
7	Относительная косвенная	@MET(PC)	@MET

Как видим, действительно ничего принципиально нового в последних четырех способах адресации нет. Но их так называемая «стандартная» запись для программистов на ассемблере выглядит в высшей степени необычно и почти никогда не применяется.

Можно ли «изобрести» еще способы с использованием PC? Да, конечно. Но практического значения они не имеют и потому или не применяются, или мы их используем, даже не задумываясь, что у них есть столь «именитые родичи», имеющие отдельные «титулы» и «гербы». Продемонстрируем такие «открытия». Допустим, мы по аналогии захотели создать способ с применением PC, имеющий код 0. Что мы получим? Например, MOV PC,R4. Что же тут нового? Или, скажем, код 1: MOV @PC,R4 — так мы просто запишем в R4 код следующей команды. Это нужно? Мягко говоря, не часто... Код 4: MOV -(PC),-(PC) — вирус, с которым мы уже знакомы. И наконец, код 5: MOV @-(PC),R4 — эта команда перепишет в R4... число из ячейки с адресом, равным коду текущей команды! Чепуха, и ничего более.

Контрольные вопросы и задания

1. Что будет записано в регистр R0 командой MOV R1,R0, если в регистре R1 содержится число 5555?

— 5555.

2. Что будет записано в регистр R5 последовательностью команд:

```
MOV #1000,@#2500
MOV #40000,@#1000
MOV @#2500,R4
MOV R4,R5
MOV @R5,R5
```

— Число 40000.

3. Что будет записано в R0 последовательностью команд (считая, что метка MET определена и находится в ОЗУ):

```
MOV #1000,MET
MOV #3402,@#1000
MOV #377,@#3400
MOV @MET,R5
MOV -(R5),R0
```

— Число 377.

4. Каким будет содержимое R2 после выполнения каждой из команд: MOVB #160,R2; MOV #277,R2; MOVB #201,R2? (Чтобы не ошибиться, «нарисуйте» полученные числа в двоичной форме, поразрядно.)

— 160, 277, 177601.

5. Команда MOV -(PC),-(PC) «саморазмножается» в ОЗУ в сторону младших адресов («вниз» по памяти). Придумайте команду, которая будет исполнять «бег на месте», т.е. переписывать свой код по своему же адресу и передавать себе управление.

— MOV -(PC),@PC

6. Придумайте последовательность двух одинаковых команд, исполняющую «бег на месте» (обязательно с использованием кодов обеих команд).

— MOV @PC,-(PC)
MOV @PC,-(PC)

7. Придумайте команду, которая бы «саморазмножалась» по памяти «вверх».

— Такой команды нет.

8. Что запишет в R3 последовательность команд:

```
    MOVB #255,R0
    CLR MET
    MOVB R0,MET
    MOV (PC)+,R3
MET: CLR R0
```

— Число 255.

Оператор прямого присваивания. Выражения

После того, как изучены способы адресации, рассмотрим еще два тесно примыкающих к ним вопроса. Мы уже знаем, что метками в тексте программы помечаются некоторые строки и что в процессе трансляции и компоновки метке присваивается текущий адрес (т.е. адрес начала строки в которой она стоит). Но как быть, если имеется в виду адрес в ПЗУ или в системной области? Может быть, в таких случаях придется ограничиться только абсолютной адресацией, прямо указывая адрес? Это не всегда удобно, поэтому в ассемблере предусмотрен специальный механизм для ОПРЕДЕЛЕНИЯ меток, так называемый ОПЕРАТОР ПРЯМОГО ПРИСВАИВАНИЯ. Он записывается как MET=X, где X — восьмеричное число без знака. Этот

знака. Этот оператор нетранслируемый (при трансляции он не преобразуется в машинный код, а только присваивает заданной метке значение адреса X, которое заносится в таблицу меток и используется при дальнейшей трансляции и компоновке). Операторы прямого присваивания принято размещать в начале каждого отдельного модуля программы.

Что же можно определить с помощью этого оператора и для чего? Прежде всего, особенности работы ассемблер-системы таковы (не будем вдаваться в подробные объяснения, но сделано это для обеспечения перемещаемости программ), что обращение по адресам системной области (0 — 777) и ПЗУ (100000 — 177777) возможно только с применением абсолютной адресации или — косвенно — через регистр. Относительная адресация этих ячеек невозможна. Вот эти-то абсолютные значения адресов и могут быть приравнены меткам с помощью оператора прямого присваивания, причем при обращении по адресу определенной таким образом метки адресация все равно получается не относительная, а абсолютная. С другой стороны, оператор прямого присваивания не может приравнять метке адрес, лежащий в пределах ОЗУ пользователя — при трансляции и компоновке это обернется ошибкой. Пример (фрагмент листинга реальной программы — драйвера принтера):

```
REG=57          Признак регистра
CPR=256         Копия порта
POZ=374         Позиция печати
POR=177714      Порт ввода-вывода
; Выход в ФОКАЛ или МСД
;.....
CLRB REG        Рег. то, сбросить
;.....
CLR POZ         Обнулить позицию
;.....
CLR POR         Очистить порт
;.....
MOV R0,CPR      Запомнить символ
```

Здесь показано как определение меток в начале текста с помощью операторов прямого присваивания, так и дальнейшее использование этих меток. Отметим, что определенная таким способом метка может быть использована в составе команды с любым методом адресации (не только относительным) и ведет себя как обычная переменная. Например, если бы мы записали MOV #POR,R4, то тем самым число 177714, присвоенное метке POR, было бы записано в регистр R4.

Развитые версии ассемблеров допускают адресацию операндов с помощью не только меток, но и ВЫРАЖЕНИЙ, состоящих из ТЕРМОВ. (Термами называются входящие в состав выражения числа или символы, связанные знаками арифметических операций «+» или «-».) Количество термов в выражении для большинства версий ассемблеров не должно превышать двух. Выражение записывается как MET+X или MET-X, и в процессе трансляции вычисляется его значение. Пример (адрес метки TEX равен 1100):

```
CLR TEX       Очистка ячейки 1100
CLR TEX+100   Очистка ячейки 1200
CLR TEX-100   Очистка ячейки 1000
```

Аналогично можно задать операнды с помощью выражений и в других случаях, например:

```
CLR MET+10(R2)
MOV #MET-122,R2
CLRB @MET+43
```

(При трансляции и компоновке вторым словом данных команд записывается результат вычисления выражений.)

Контрольные вопросы и задания

1. Что запишется в регистр R1 в результате выполнения программы:

```
MET=254
MOV #MET+4,R0
MOV #155,-(R0)
MOVB @#256,R1
```

— Число 155.

2. Какая ячейка будет обнулена в результате выполнения программы:

```
MET=100
MOV #150,R0
MOV #1400,MET+156
CLR @MET+6(R0)
```

— Ячейка с адресом 1400.

(Продолжение следует)

Продолжаем публикацию заводских инструкций, известных среди пользователей БК под названием «ремонтная документация». Вниманию читателей предлагается описание устройства и функционирования БК-0010 (первая часть заводской документации, касающаяся собственно рекомендаций по наладке и ремонту, опубликована в № 2 за 1994 г.).

Напоминаем, что все приведенные ниже сведения относятся к первой модели БК-0010 (с пленочной клавиатурой). Для более поздних модификаций БК-0010.01 и БК-0011(М) возможны некоторые отличия.

ТЕХНИЧЕСКОЕ ОПИСАНИЕ БК-0010

МикроЭВМ «Электроника БК-0010» предназначена для использования дома, в учебных заведениях и учреждениях людьми различных профессий, как уже имеющими опыт работы с вычислительной техникой, так и встречающимися с ней впервые. Спектр возможных задач, решаемых с ее помощью, весьма широк:

- выполнение научно-технических, инженерных, математических и прочих расчетов;
- ведение домашнего хозяйства;
- обучение основам программирования;
- управление нестандартными периферийными устройствами посредством программируемого параллельного интерфейса (порта ввода-вывода);
- организация досуга (игры).

МикроЭВМ БК-0010 рассчитана на работу в помещениях с температурой воздуха от 5 до 40°С и относительной влажностью до 95% (при температуре 30°С).

Питание микроЭВМ (+5 В) осуществляется от сети переменного тока с частотой 50±1 Гц и напряжением 220 $\pm\frac{22}{33}$ В через выносной стабилизированный блок питания. Потребляемая мощность — не более 20 Вт.

Конструкция микроЭВМ

МикроЭВМ БК-0010 состоит из двух отдельных функциональных узлов (рис. 1):

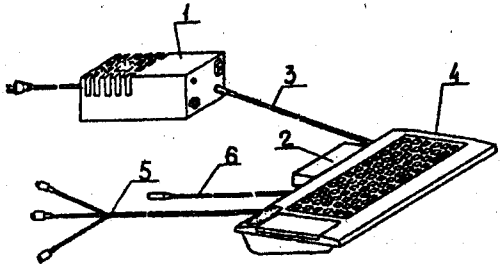


Рис. 1. Внешний вид микроЭВМ «Электроника БК-0010»: 1 – блок питания; 2 – блок нагрузки; 3 – кабель БП; 4 – информационно-вычислительное устойство; 5 – кабель магнитофона; 6 – кабель ТВ

- информационно-вычислительного устройства (ИВУ);
- блока питания.

Информационно-вычислительное устройство смонтировано на двух печатных платах, расположенных в пластмассовом корпусе друг над другом. На верхней плате размещена клавиатура, на нижней — собственно вычислитель. Крышка корпуса имеет окно (отсек пользователя) для доступа к сменным ПЗУ и переключателю «СТОП-ПУСК», служащему для перезапуска процессора.

Блок питания смонтирован в отдельном корпусе. Подключение к сети осуществляется шнуром с вилкой, а включение — тумблером, находящимся на передней стенке блока.

Принципы работы микроЭВМ

МикроЭВМ «Электроника БК-0010» представляет собой ЭВМ с микропрограммным управлением. Набор микрокоманд, реализующих операции, выполняемые процессором К1801ВМ1, хранится в его блоке микропрограммного управления.

Интерпретатор языка высокого уровня ФОКАЛ—БК-0010, размещенный в ПЗУ, преобразует каждый оператор языка ФОКАЛ в последовательности команд в машинных кодах.

Процессор (ПРЦ) выполняет арифметические и логические действия над адресами и операндами, хранящимися в его внутренних регистрах или в ОЗУ микроЭВМ. Под управлением системных программ процессор обращается к контроллерам внешних устройств и к порту ввода-вывода как к внешним регистрам и управляет работой этих устройств на программно-аппаратном уровне.

Ввод информации с клавиатуры микроЭВМ осуществляется при помощи контроллера клавиатуры, который преобразует сигналы клавиатурной матрицы в параллельный код КОИ-7 и обеспечивает его передачу процессору для последующей обработки.

Для отображения информации на экране видеомонитора (бытового телевизора) контроллер ТВ-приемника преобразует содержимое видеоОЗУ в последовательный код и формирует стандартный видеосигнал. Помимо этого контроллер ТВ-приемника

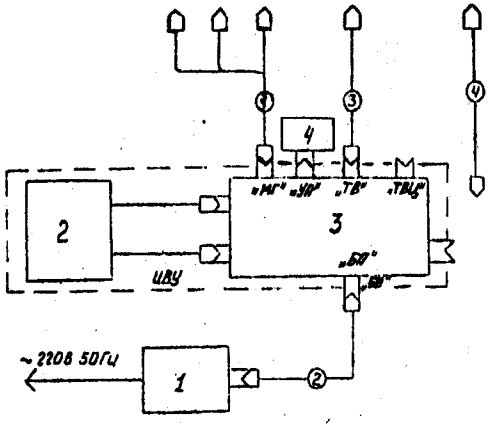


Рис. 2. Схема соединений БК-0010: 1 – блок питания; 2 – плата клавиатуры; 3 – плата вычислителя; 4 – блок нагрузки; (1)–(4) – кабель связи

осуществляет управление и регенерацию динамического ОЗУ микроЭВМ.

Для записи файлов на магнитную ленту и их последующего чтения контроллер бытового магнитофона преобразует выводимую информацию в последовательный широтно-импульсный модулированный сигнал.

Порт ввода-вывода обеспечивает связь микроЭВМ с устройствами пользователя (имеется 16 выходов и 16 входов).

Структура микроЭВМ

Схема соединения узлов микроЭВМ БК-0010 показана на рис. 2.

Плата клавиатуры предназначена для ввода информации в микроЭВМ. Она соединена с платой вычислителя с помощью разъемов XT1 и XT2. На плате установлено 92 переключателя типа ПКН-150-1.

Плата вычислителя является основной платой микроЭВМ. На ней установлены разъемы для подключения внешних устройств (бытового телевизионного приемника черно-белого изображения и кассетного магнитофона), резервный разъем для цветного телевизионного приемника (ТВЦ), разъемы УП для подключения устройств пользователя и XT8 (розетка типа РС-24) для установки третьего (резервного) модуля ПЗУ (ПЗУ пользователя).

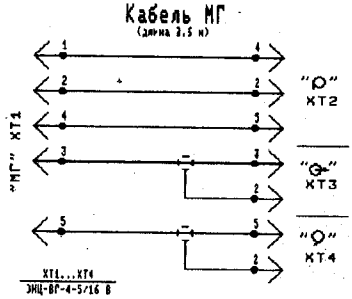


Рис. 3. Кабель магнитофона

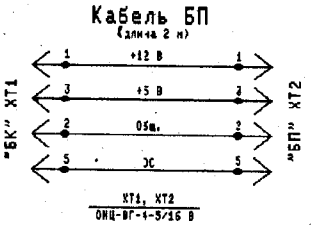


Рис. 4. Кабель питания

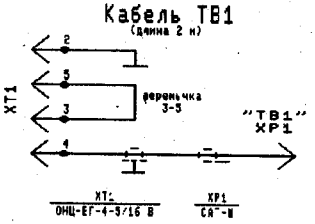


Рис. 5. Кабель ТВ (2 цвета)

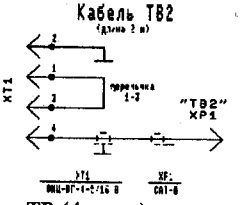


Рис. 6. Кабель ТВ (4 цвета)

Блок нагрузок предназначен для имитации подключения устройств пользователя к разъему УП микроЭВМ при контроле ее работоспособности.

Кабели связи. Кабель «МГ» (рис. 3) предназначен для подключения бытового кассетного магнитофона к разъему МГ платы вычислителя.

Кабель питания (рис.4) служит для соединения блока питания с платой вычислителя.

Кабель «ТВ1» (рис. 5) используется для подключения бытового телевизионного приемника черно-белого изображения к разъему ТВ платы вычислителя. Этот кабель содержит перемычку 3-5, обеспечивающую вывод графической информации с разрешающей способностью 512 точек в строке (две градации яркости).

Кабель «ТВ2» (рис. 6) служит для той же цели, что и «ТВ1», но в отличие от него содержит перемычку 1-3, обеспечивающую вывод 256 точек в строке (четыре градации яркости).

Структура вычислителя

Вычислитель включает в себя следующие узлы (основные показаны на рис. 7):

- процессор (ПРЦ);
- канал связи (магистраль, общая шина);

- схему запуска процессора (СЗП);
- схему синхронизации сигналов (ССС);
- контроллер клавиатуры (КК);
- схему прерывания (СП);
- постоянное запоминающее устройство (ПЗУ) — модули 1, 2 и 4 (модуль 3 — ПЗУ пользователя, для которого зарезервировано гнездо XT8 в отсеке пользователя);
- контроллер магнитофона (КМ);
- схему управления оперативным запоминающим устройством (УОЗУ);
- оперативное запоминающее устройство (ОЗУ);
- магистральный буферный регистр (МБР);
- сдвиговый регистр (СР);
- схему формирования видеосигнала (ФВС);
- генератор тактовых импульсов (ГТИ);
- параллельный программируемый интерфейс (порт ввода-вывода).

Процессор предназначен для вычисления адресов операндов, обмена информацией с другими устройствами, подключенными к каналу, обработки операндов, реакции на внешние воздействия (с порта ввода-вывода). Процессор является единственным активным устройством микроЭВМ, управляющим и обрабатывающим прерывания от пассивных устройств, которые могут передавать или принимать информацию только под управлением процессора.

Схема запуска процессора предназначена для запуска процессора при включении питания, а также для его останова и перезапуска при использовании переключателя «СТОП-ПУСК», контроле платы вычислителя или микроЭВМ в целом.

Канал микроЭВМ представляет собой систему сигнальных связей, назначение и физическая реализация которых закреплены стандартом интерфейса Q-bus (МПИ, ОСТ 11.305.903-80). Разъем XT3 («системная магистраль») предназначен для подключения стенда контроля микроЭВМ. (К этому же разъему подключаются контроллеры дисководов, блоки расширенного ОЗУ, а также блок МСТД с языком ФОКАЛ для модификации БК-0010.01. — *Прим. ред.*)

Нумерация и назначение сигналов разъема «системной магистрали» (XT3) приведены в табл. 1.

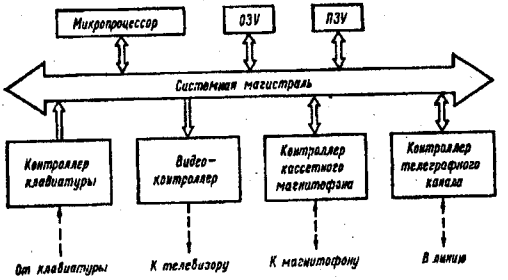


Рис. 7. Основные узлы вычислителя

Таблица 1

Номер контакта разъема XT3	Обозначение сигнала в магистрали	Обозначение в спецификации и условное обозначение процессора	Наименование сигнала магистрали
A1	QCT H	IRQ1	Останов
A2	Общий	—	Общий
A3	»	—	»
A4	+5 В (контроль)	—	Контроль источника питания +5 В
A5	ПРТ H	IRQ2	Требование прерывания
A23	ВВОД H	DIN	Ввод данных
A25	ДА18 H	AD18	Линия адреса данных
A26	ДА17 H	AD17	» » »
A27	ДА09 H	AD9	» » »
A28	ДА05 H	AD5	» » »
A31	ДА00 H	AD0	» » »
B3	Общий	—	Общий
B5	»	—	»
B4	ППР1 H	IAK1	Входной сигнал предоставления прерывания
B4	ППР1 H	IAK0	Выходной сигнал предоставления прерывания
B5	ТПР H	VIRQ	Требование прерывания
B7	ДА15 H	AD15	Линия адреса данных
B11	БАЙТ H	WTBT	Вывод байта
B19	СБРОС H	INIT	Первоначальная установка магистрали
B20	СИП H	RPLY	Сигнал синхронизации пассивного устройства (ответ)
B21	ВЫВОД H	DOUT	Вывод данных
B22	СИА H	SYNC	Сигнал синхронизации активного устройства
B23	ДА14 H	AD14	Линия адреса данных
B24	ДА13 H	AD13	» » »
B25	ДА10 H	AD10	» » »
B26	ДА08 H	AD8	» » »
B27	ДА06 H	AD6	» » »
B28	ДА04 H	AD4	» » »
B29	ДА02 H	AD2	» » »
B30	ДА03 H	AD3	» » »
B31	ДА01 H	AD1	» » »
B32	ДА07 H	AD7	» » »

Схема синхронизации предназначена для привязки сигналов IRQ1, IRQ2, VIRQ, ACLO и RPLY к моменту появления отрицательного фронта на выводе CLC процессора.

Контроллер клавиатуры устанавливает соответствие между нажатой клавишей и ее семиразрядным кодом. Прием кода осуществляется под управлением процессора по программе, называемой драйвером клавиатуры. Контроллер реализован на БИС К1801ВП1-014.

Схема прерывания обеспечивает аппаратную поддержку программной обработки прерывания по зависанию и позволяет отличить зависание при отсутствии адресуемой ячейки от возникающего при нажатии клавиши «СТОП».

Постоянное запоминающее устройство предназначено для хранения системных программ:

- модуль 1 — монитор и драйверы внешних устройств (БИС К1801РЕ1-017);
- модуль 2 — диалоговый язык ФОКАЛ (БИС К1801РЕ1-018);
- модуль 3 — устанавливаемая в резервный отсек микросхема ПЗУ с программами пользователя;
- модуль 4 — мониторная система диагностики (БИС К1801РЕ1-019).

Контроллер бытового магнитофона предназначен для аппаратной поддержки программно реа-

лизованных алгоритмов записи и чтения информации с магнитной ленты. Схема контроллера использует четыре разряда регистра начального пуска (РНП, адрес 177716_8).

Схема управления ОЗУ (контроллер ОЗУ) выполнена на основе БИС К1801ВП1-037 и обеспечивает регенерацию информации, хранящейся в ОЗУ. Одновременно эта схема является контроллером бытового телевизионного приемника.

Блок ОЗУ выполнен на основе 16 микросхем КР565РУ6 с динамическим хранением информации. Емкость блока ОЗУ составляет 16 Кб (1 Кб = 1024 байт) 16-разрядных слов.

Сдвиговый регистр предназначен для преобразования считанной из ОЗУ параллельной информации в последовательную для передачи ее на видеовход ТВ-приемника. Регистр выполнен на микросхемах К155ИР13.

Буферный регистр данных, выполненный на микросхемах К589ИР12, предназначен для временного хранения данных после завершения их выборки из ОЗУ до окончания передачи по каналу в активное устройство.

Схема формирования видеосигнала обеспечивает требуемые в соответствии с ГОСТ 7845-79 амплитудно-временные характеристики сигнала для его передачи на видеовход ТВ-приемника.

Генератор тактовых импульсов обеспечивает формирование сетки частот, необходимых для работы вычислителя.

Линии связи 16-разрядного программируемого параллельного интерфейса (порта) предназначены для подключения устройств пользователя (УП). Управление 16 линиями порта источника (ВД00-ВД15) и анализ состояния 16 линий порта приемника (ВВ00-ВВ15) осуществляются с помощью соответствующих операторов диалогового языка (путем обращения к регистру с адресом 177714_8). Дополнительные линии СБРОС Н и ПРТ Н могут быть использованы для начальной установки устройств пользователя и для прерывания основной программы по их запросу. Порт выполнен на микросхемах К589ИР12.

Устройство и работа узлов вычислителя*

Процессор (ПРЦ) выполнен на БИС К1801ВМ1 (D14). Специфичными применительно к микроЭВМ являются способ начального запуска процессора и использование выводов ACLO и DCLO для останова программы при настройке микроЭВМ или отладке программного обеспечения.

Генератор тактовых импульсов обеспечивает синхронизацию процессора и контроллера бытового телевизионного приемника. Схема генератора частотой 12 МГц и скважностью 2 состоит из следующих радиоэлектронных компонентов:

- микросхемы D5.1, D5.2, D5.3, D8.1, D8.2;
- кварцевый резонатор BQ1;
- резисторы R7, R10-R12, R15, R16;
- конденсатор C6.

Резисторы R7, R11 и конденсатор C6 в цепях обратной связи обеспечивают режим генерации инверторов D5.1 и D5.2. Резистор R16 — ограничивающий, а R10, R12 и R15 — нагрузочные. Инвертор D5.3 является буфером-формирователем. Микросхема D8 обеспечивает последовательное деление частоты до 6 МГц и 3 МГц. Сигналы этих частот поступают на входы CLC БИС D19 и БИС D14 соответственно, а также на схему синхронизации сигналов прерывания (D11 и D3.2).

Схема синхронизации сигналов прерывания предназначена для привязки сигналов IRQ1, IRQ2, VIRQ, ACLO и RPLY (выводы 31, 32, 35, 30, 39 БИС процессора D14) к моменту появления отрицательного фронта сигнала CLC (вывод процессора 01).

Схема синхронизации собрана на микросхемах D11, D3.2 и ограничивающих резисторах R21, R22, R28. Резистор R28 ограничивает величину тока в цепи «вывод D14/39 — вывод D3.2/9» при обращениях к системным регистрам процессора (в этих случаях вывод D14/39 является выходом).

Запуск процессора. В микроЭВМ БК-0010 выводы PA0 и PA1 процессора подключены к источнику питания, что устанавливает адреса регистров процессора в диапазоне 177700-177168 (см. табл. 2). При включении питания или при запуске переключателем SA1 процессор на микропрограммном уровне обращается в ячейку с адресом 177168 (регистр РНП) и проводит загрузку значения счетчика команд (СК) и регистра состояния процессора (РСП):

- старший байт СК (биты 15-8) <= РНП15-РНП8 = 10000000XXXXXXXX;
- младший байт СК (биты 7-0) <= 0 - считанные из регистра РНП значения битов РНП0-РНП7 (выше они обозначены «X») игнорируются и «микропрограммное» значение младшего байта СК (7-0) приравнивается нулю;
- РСП (биты 15-0) <= 340.

Схема запуска процессора предназначена для формирования одиночных положительных перепадов напряжения на выходах процессора D14/29 (DCLO) и D14/30 (ACLO) при его запуске или перезапуске (см. временную диаграмму на рис. 8).

* Монтажная и принципиальная схемы БК-0010 опубликованы в журнале «Персональный компьютер БК-0010 - БК-0011М» № 2 за 1994 г. — *Прим. ред.*

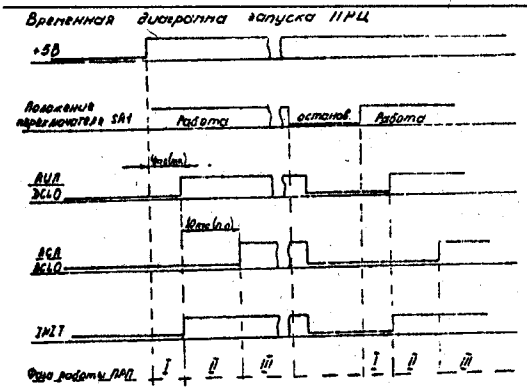


Рис. 8. Фазы запуска процессора: I – начальная установка; II – ожидание снятия сигнала аварии сетевого питания; III – работа по программе

Конденсатор C5 и резисторы R8, R9 обеспечивают длительность процесса установления сигнала DCLO. В это время происходит нормализация напряжения питания микроЭВМ, подготовка к работе ОЗУ и всех устройств системы. Это *первая фаза* начальной установки — сброс в исходное состояние блоков управления процессора, после чего он выставляет сигнал INIT (вывод D14/34).

Вторая фаза — ожидание сигнала ACLO.

Третья фаза — работа микропрограммы инициализации (чтение регистра начального пуска): по адресу 177716_8 читается число $10000000XXXXXXXX_2$, являющееся начальным адресом программы управления (монитора), и формируется значение счетчика команд (биты, обозначенные как «X», приравниваются нулю). В регистр состояния процессора загружается константа 340_8.

После этого анализируется состояние запросов на прерывание. Если незамаскированных прерываний нет, то происходит чтение первой команды и ее выполнение, иначе — отрабатывается процедура прерывания.

При переводе процессора в режим «СТОП» первым снимается сигнал DCLO, а затем ACLO. Временные интервалы при этом не нормируются.

Режим процессора «СТОП-ПУСК» переключается одним из трех способов:

- при включении питания;
- переключателем SA1;
- сменой логического уровня на выводе A1 разъема XT3 (ОСТ Н).

Включение питания *(SA1 в положении «ПУСК»)*. Если микроЭВМ была выключена (все конденсаторы схемы разряжены), то после включения и установления напряжения питания выход D6.2 триггера формирования DCLO (D6.1, D6.2, D6.3) и выход D2.3 схемы формирования ACLO (D6.4, D2.3) устанавливаются в «1».

Устанавливающим является пологий фронт сигнала в точке D6/09, обусловленный большой постоянной времени заряда C5. При достижении в этой точке уровня порога переключения схемы, равного примерно 0,5 от напряжения питания (через время $t \approx (R8+R9)*C5$ после включения питания) происходит переключение RS-триггера D6.2, D6.3.

Резистор R9 обеспечивает опережающее переключение инвертора D6.1 по отношению к D6.3, что гарантирует отсутствие многократных переключений триггера при пологом фронте сигнала на конденсаторе C5. На этом заканчивается первая фаза начальной установки.

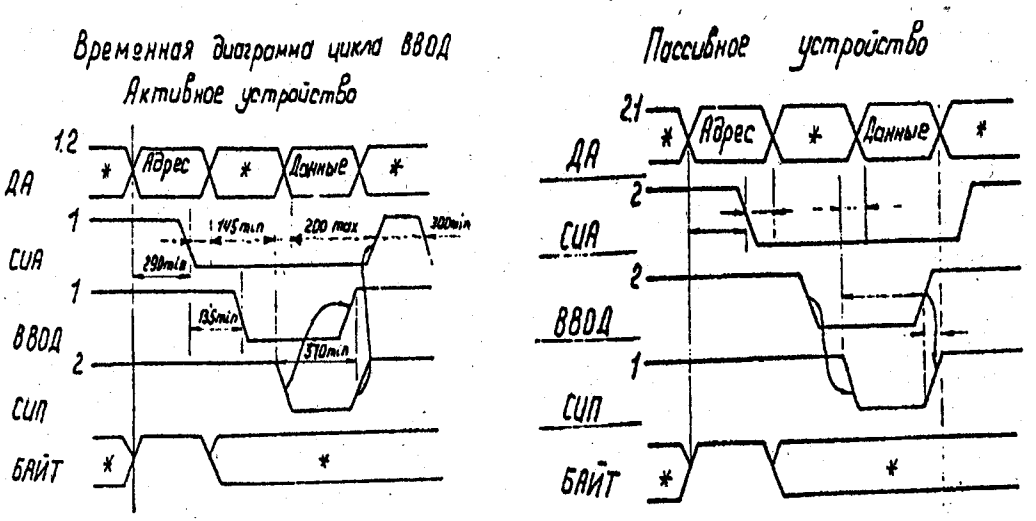


Рис. 9. Цикл «ВВОД»: 1 – передаваемый сигнал; 2 – принимаемый сигнал; * – уровень сигнала может быть любым

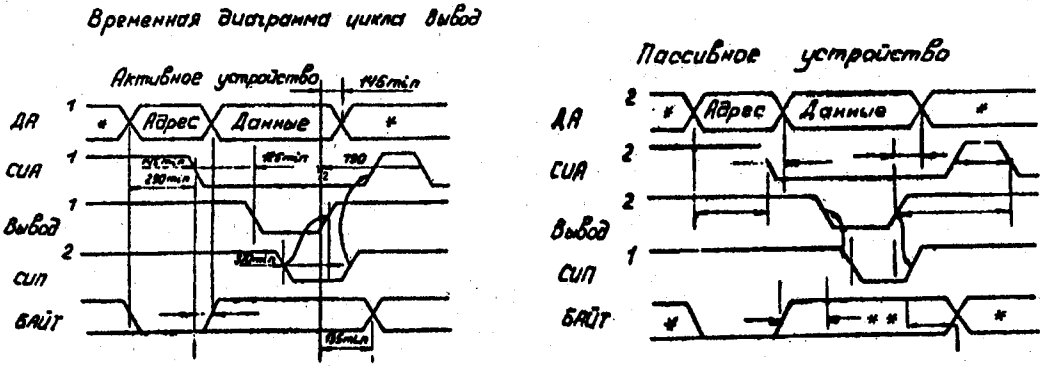


Рис. 10. Цикл «ВЫВОД»: 1 – передаваемый сигнал; 2 – принимаемый сигнал; * – уровень сигнала может быть любым; ** – сигнал устанавливается при байтовых операциях

RC-цепочка R14, C7 обеспечивает задержку t2 передачи положительного фронта в схеме формирования ACLO (см. рис. 8). Этой задержкой определяется длительность второй фазы начальной установки. Положительный перепад напряжения передается через схемы D6, D2.3 и D11 на вход ACLO — процессор включен.

Управление переключателем. Чтобы остановить работу процессора, нужно установить переключатель SA1 в положение «СТОП» (контакты 2 и 3 замкнуты). При этом конденсатор C5 разряжается, а схема запуска процессора устанавливается в исходное состояние.

Включение процессора осуществляется установкой переключателя SA1 в положение «ПУСК» (процесс запуска тот же, что и после включения питания).

Управление запуском процессора по линии ОСТ Н. Действие сигнала на линии ОСТ Н аналогично переключению SA1 («СТОП-ПУСК»), как это описано выше. Но в этом случае переключатель может быть установлен за пределами корпуса микроЭВМ (линия ОСТ Н соответствует контакту А1 разъема ХТ3).

Канал (магистраль) микроЭВМ содержит 26 линий связи (печатных проводников), из которых 16 (ДА00-ДА15) являются двунаправленными, т.е. информация по ним может как приниматься, так и передаваться относительно одного и того же устройства микроЭВМ.

Внутренняя магистраль (канал) не предназначена для расширения системы и не имеет шинных формирователей.

Временные диаграммы обмена информацией в канале при различных циклах обмена приведены на рис. 9-12.

Все блоки, подключенные к каналу, имеют выходные каскады с открытым коллектором. Такое построение дает возможность параллельно подключать несколько устройств, не опасаясь сквозных токов через выходные каскады разных блоков от источника питания к общей шине. При этом активным в канале

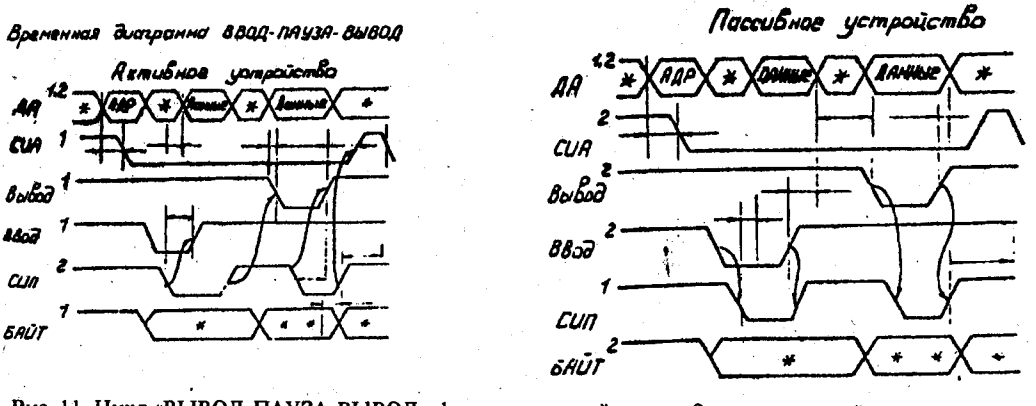


Рис. 11. Цикл «ВЫВОД-ПАУЗА-ВЫВОД»: 1 – передаваемый сигнал; 2 – принимаемый сигнал; * – уровень сигнала может быть любым; ** – сигнал устанавливается при байтовых операциях

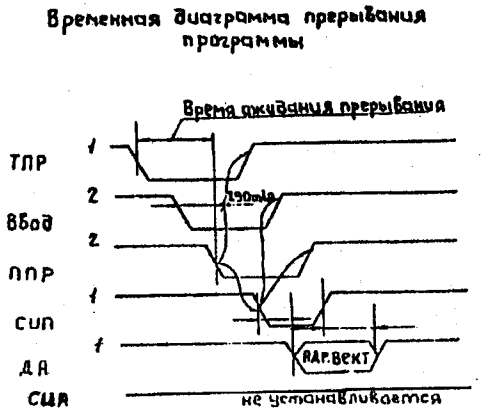


Рис. 12. Цикл «ПРЕРЫВАНИЕ»: 1 – сигнал, передаваемый устройством; 2 – сигнал, принимаемый устройством

Если процессор находится в режиме «СТОП», то на всех выводах разъема ХТ3 (за исключением СБРОС Н и ОСТ Н) установлен уровень напряжения питания, который обеспечивается нагрузочными резисторами 2.2 кОм, подключенными к шине питания.

Связь через канал замкнута, т.е. на управляющий сигнал, передаваемый процессором, должен поступать ответный сигнал от пассивного устройства (СИП Н).

Адресное пространство микроЭВМ. Постоянное и оперативное запоминающие устройства представлены на магистрали как совокупность ячеек памяти, каждая из которых имеет собственный адрес. Совместно с служебными регистрами микроЭВМ они образуют поле адресов от 0 до 2^{16}, схематически изображенное на рис. 13.

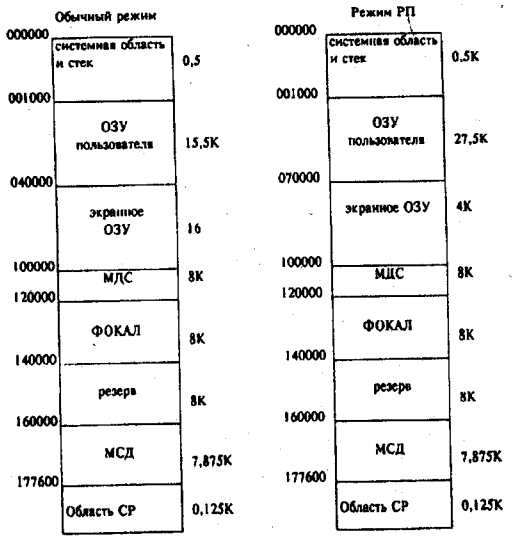


Рис. 13. Поле адресов БК-0010

Распределение адресного пространства:

- ОЗУ пользователя — в обычном режиме $0\text{-}37777_8$, в режиме РП $0\text{-}67777_8$;
- ОЗУ экранной памяти — в обычном режиме $40000 - 77777_8$, в режиме РП $70000 - 77777_8$;
- ПЗУ — $100000\text{-}177577_8$;
- системные регистры — $177600 - 177777_8$.

Адреса регистров процессора: 1777XX (два последних восьмеричных разряда адреса, обозначенные как «X», указаны в табл. 2).

Адреса системных регистров приведены в табл. 3.

Адреса прерывания и их приоритеты указаны в табл.4.

Системные регистры БК-0010 занимают, как уже было отмечено выше, область с адресами 177600 – 1777778, но далеко не все адреса этой области используются.

Регистр режима (состояния клавиатуры) служит для запрета/разрешения прерывания от клавиатуры и содержит информацию о поступлении нового кода клавиши. Назначение разрядов:

- РР0 – РР5 — не используются,;
- РР6 — разрешение прерывания от клавиатуры (0 – разрешено, 1 – запрещено), доступен по записи и чтению;
- РР7 — флаг состояния клавиатуры («несчитанного кода»), доступен только по чтению (1 – регистр данных готов к выдаче кода символа, 0 – регистр данных содержит предыдущий, уже считанный код);
- РР8-РР15 — не используются.

При начальном включении и после чтения кода из регистра данных РД бит РР7 сбрасывается, при нажатии на любую клавишу (кроме «НР», «СУ», «ПР», «ЗАГЛ», «СТР») — устанавливается.

Регистр данных (буфер клавиатуры) доступен по чтению, хранит код символа последней нажатой клавиши. Назначение разрядов:

- РД0 – РД6 — код символа (знаковая таблица КОИ7);
- РД7 – РД15 — не используются.

Регистры РР и РД входят в состав БИС К1801ВП1-014 (D4).

Регистр смещения (рулона) входит в состав БИС К1801ВП1-037 (D19), доступен по чтению и записи. Назначение разрядов:

- РС0 – РС7 — определяют адрес байта экранного ОЗУ, соответствующего началу первой телевизионной строки. Число 330_8 означает адрес 40000_8, его изменение на 1 соответствует увеличению или уменьшению адреса на 100_8,

Таблица 2

Наименование	Обозначение	Состояние PA0, PA1				Назначение
		00	01	10	11	
Режима	РР	00	20	40	60	Пользователю микроЭВМ недоступны.
Адр. прерывания	РАП	02	22	42	62	
Ошибки	РОШ	04	24	44	64	
Внешний регистр	ВР2	14	34	54	74	Порт ввода-вывода
Внешний регистр	ВР1	16	36	56	76	Регистр начального запуска

Примечание: конфигурация БК-0010 соответствует состоянию PA0, PA1 = 00.

Таблица 3

Наименование	Обозначение	Адрес (восьм.)
Режим клавиатуры	РР	177660
Буфер клавиатуры	РД	177662
Смещения (ROLL)	РС	177664
Начального запуска	РНП	177716

Таблица 4

Источник прерывания	Приоритет	Адрес вектора	Примечание
Зависание	1	4	
Ошибочный машинный код	2	10	
Бит Т ССП	3	14	
Сбой питания	4	24	
Радиальное статическое (клавиша «СТОП»)	5	4	Прерывание программы пользователя с клавиатуры
Радиальное динамическое	7	100	Прерывание по требованию внешнего устройства (подключенного к линии ПРТ Н)

- РС8 — не используется;
- РС9 — режим расширенной памяти (1 – выключен, 0 – включен);
- РС10 – РС15 — не используются.

При начальном включении микроЭВМ аппаратная установка регистра не производится.

Регистр начального пуска процессора (управления системными внешними устройствами) фактически состоит из двух регистров (с одним и тем же адресом 177716_8), один из которых доступен только по записи, а другой — только по чтению.

Назначение разрядов.

По чтению:

- РНП0, РНП1, РНП3, РНП8 – РНП14 — содержат «лог. 0»;
- РНП2 — сбрасывается при любом обращении к РНП по чтению (по окончании процесса чтения);
- РНП4 — бит линии радиального интерфейса (0 – прием «лог. 0» (если установлена перемычка S2 «ПрД»), 1 – прием «лог. 1»);
- РНП5 — бит линии чтения информации с магнитной ленты (0 – «лог. 0», 1 – «лог.1»);
- РНП6 — индикатор нажатия какой-либо клавиши (0 – клавиша удерживается нажатой, 1 – клавиша не нажата);
- РНП7 — константа «1», указывающая на отсутствие в системе команд расширенной арифметики (соответствующие этим командам коды вызывают прерывание по вектору 10_8, что позволяет программно сэмулировать операции расширенной арифметики, как это сделано, например, в интерпретаторе ФОКАЛа. — *Прим. ред.*);
- РНП15 — бит начального пуска процессора, константа «1» указывает на запуск с адреса 100000_8.

При считывании РНП в момент начального запуска значение, указанное в его младшем байте, не учитывается. Обращение к РНП происходит на микропрограммном уровне (сигнал ВР1 (SEL1) аппаратно инициирует чтение регистра).

По записи:

- РНП0, РНП1, РНП3, РНП8 – РНП15 — не используются;
- РНП2 устанавливается при любом обращении к РНП по записи;
- РНП4 — бит линии радиального интерфейса (0 – передача «лог. 0» (если установлена перемычка S3 «ПД»), 1 – передача «лог. 1»);
- РНП5, РНП6 — биты линии записи информации на магнитную ленту (от комбинации, заносимой в эти разряды, зависит уровень записываемого сигнала на контакте 5 разъема магнитофона XT4);
- РНП7 — бит управления двигателем магнитофона (0 – пуск, 1 – останов).

При перезапуске системы по клавише «СТОП» бит РНП7 микропрограммно устанавливается в 1.

(Реально управление двигателем магнитофона возможно только в том случае, если он оборудован дистанционным управлением, а штекер магнитофонного кабеля подключен в соответствующее гнездо магнитофона. В некоторых модификациях магнитофона «Электроника 302» дистанционное управление «встроено» в цепь внешнего микрофона, в гнездо которого и нужно включать данный штекер.— *Прим. ред.*).

Разряд РНП2 может быть использован как отличительный признак прерывания по клавише «СТОП» или по команде HALT при анализе причины прерывания в обработчике, адрес входа в который содержится в векторе 4:

— РНП2=0 — прерывание по зависанию (обращение к несуществующему регистру),

— РНП2=1 — прерывание клавишей «СТОП» или по команде HALT.

При нажатии на клавишу «СТОП» (или при отработке HALT) возникает радиальное прерывание IRQ1. Его микропрограммная обработка приводит к обращению по записи в РНП (в частности, для останова двигателя магнитофона), при этом значение РНП2 устанавливается в 1.

Далее на микропрограммном же уровне процессор обращается по записи по адресу 177676_8. Но в данной модификации микроЭВМ ячейка с таким адресом отсутствует, поэтому происходит обработка прерывания по зависанию (адрес вектора прерывания 4). В программе обработки прерывания по вектору 4 можно проанализировать значение разряда РНП2: если РНП2=1, то отрабатываются действия, соответствующие нажатию клавиши «СТОП». Если же РНП2=0, это означает отсутствие адресата (неисправность ячейки или регистра) или свидетельствует об ошибке в программе пользователя (обращение по несуществующему адресу).

(Именно так, с помощью анализа состояния бита РНП2, в режиме ТС реализован вывод сообщения «ЗВ <адрес>» при зависании. При нажатии же клавиши «СТОП» это сообщение не выдается. Приведем фрагмент на ассемблере, соответствующий дампу ПЗУ блока МСТД для БК-0010.01:

```
160722:  ....          ; адрес начала обработчика
         ....          ; прерывания по вектору 4
         BIT #4,@#177716 ; анализ бита РНП2
         BNE A         ;
; РНП2=0 - выдача сообщения "ЗВ" и адреса зависания
      A: JMP ...       ; РНП2=1 - выход в диалог ТС
```

Данный алгоритм может быть полезен при разработке различных отладчиков, дезассемблеров и т.д. — *Прим. ред.*)

Конструктивно регистр РНП состоит из трех частей: разряда РНП15, доступного только по чтению, который входит в состав БИС D19, разряда РНП2, собранного на микросхемах D7.1 – D7.4, D1.3, D1.4, D5.5, D9.1, D10.1 и разрядов РНП4 – РНП7, собранных на микросхеме D12.

Параллельный программируемый интерфейс (порт ввода-вывода) представлен двумя 16-разрядными регистрами, один из которых доступен только по записи (D15, D16), а другой только по чтению (D17, D18). Обращение к ним осуществляется с помощью сигнала SEL2 (ВР2).

Контроллер телевизионного приемника состоит из следующих элементов (рис. 14):

- БИС К1801ВП1-037 D19;
- блока ОЗУ экранной памяти DS1–DS16;
- буферного регистра данных D22, D23;
- сдвигового регистра D24, D25;
- схемы формирования видео- и синхросигнала D10.2, D5.6, D20, VT4-VT7; D10.3, D10.4, D9.2, D21, VT8, VT9.

Принципы работы контроллера. Регенерация хранимой в ОЗУ информации осуществляется путем чтения или записи ячеек ОЗУ. Для регенерации всех ячеек достаточно обратиться поочередно к каждой строке матрицы памяти (в строке достаточно прочесть одну любую ячейку). МикроЭВМ осуществляет поочередное чтение ячеек ОЗУ в соответствии с телевизионной разверткой.

Одновременно с процессом регенерации информации в ОЗУ производится регенерация телевизионного изображения — последовательное чтение слов из видеоОЗУ и выдача на экран ТВ. Каждой телевизионной строке соответствует 32 слова ОЗУ, т.е. 32*16=512 бит или 512 точек. Телевизионный экран состоит из 256 строк. Таким образом, всего на экран выводится 256*512=131072 точки или 131072/16 = 8192 слова экранного ОЗУ. За время обращения к 128 строкам матрицы памяти осуществляется регенерация ОЗУ объемом 16384 слов с выдачей 8192 слов из них на экран телевизионного приемника.

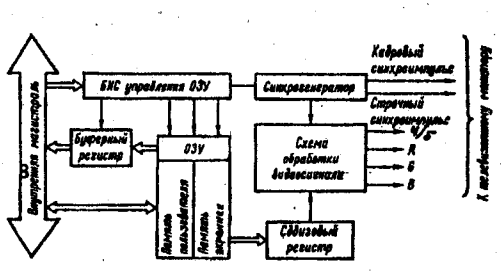


Рис. 14. Структурная схема контроллера ТВ-приемника

Регенерация ОЗУ происходит непрерывно и независимо от режима работы процессора.

Слово, считанное из видеоОЗУ, высвечивается на экране за время 2*0.66 мкс (при тактовой частоте 6 МГц 16 битов информации «выдвигаются» на экран примерно за 1.3 мкс).

Во временной диаграмме чтения ОЗУ в цикле регенерации остается интервал для дополнительного обращения к ОЗУ по произвольному адресу (по чтению или записи). Это время и предоставляется процессору. Рассогласований в работе схемы синхронизации, обеспечивающей регенерацию ОЗУ, при этом не происходит, так как в схему регенерации встроен узел-«арбитр», разрешающий обращение процессора строго в отведенное ему время. Таким образом, в течение 1.3 мкс осуществляется два цикла обращения к ОЗУ — чтения экранной памяти (регенерации содержимого ОЗУ и изображения на экране) и обращения процессора.

БИС К1801ВП1-037. Адресные входы БИС и управляющие выводы SYNC, DIN, WTBT, DOUT, RPLY подключены к каналу. Вывод CLC подключен к задающему генератору, A0-A6 — к адресным входам ОЗУ, RAS, CAS0, CAS1, WE — к управляющим входам ОЗУ. Вывод WTI подключен к выводу S0 сдвигового регистра, WTD — к выводу MD буферного регистра данных. Выводы BS и E подключены к CS (D4) и DIN (DS19) соответственно.

Буферный регистр данных (БРД) построен на микросхемах К589ИР12 (D22, D23) и предназначен для временного хранения информации, считанной из ОЗУ, с последующей ее выдачей в канал, когда последний свободен от другой информации. Управляющие входы микросхем К589ИР12 CS1, CS2, EW, CLR подключены к шине питания. В этом режиме управление осуществляется по входу MD: если на нем низкий уровень, то осуществляется запись информации со входов D1-D8 в регистр, а выходные каскады БРД отключены. Если же на входе MD высокий уровень, то отключены входы БРД, а выходы подключены к каналу. Входы MD БИС D22 и D23 соединяются с выходом WTD БИС D19, а входы D1-D8 — с соответствующими выходами ОЗУ. Выходы Q1-Q8 подсоединены к каналу.

Сдвиговый регистр данных (СР) построен на микросхемах К155ИР13 (D24, D25) и предназначен для преобразования параллельного кода, снятого с выходов ОЗУ, в последовательный, сдвигаемый тактовой частотой 6.0 МГц на вход схемы формирования видеосигнала. Информация, считанная с выходов DO ОЗУ и поданная на входы D0-D7, заносится в СР по сигналу WTI БИС D19. Тактовые импульсы положительным фронтом параллельно сдвигают два байта информации, которые снимаются одновременно с двух выходов Q0 микросхем К155ИР13 и подаются в схему формирования видеосигнала.

Схема формирования видео- и синхросигнала построена на микросхемах D5.6, D20, D10.2, D10.3, D10.4, D9.2, D21, транзисторах VT4-VT9 и резисторах R34-R49, R51-R58.

Схема обеспечивает три режима работы:

- формирование «черно-белого» видеосигнала с двумя градациями яркости и 512 точками в строке (если в кабеле XT9 установлена перемычка 3-5). Это основной режим;
- формирование «черно-белого» видеосигнала с четырьмя градациями яркости и 256 точками в строке (если в кабеле XT9 установлена перемычка 1-3);
- формирование сигналов для управления цветным телевизионным приемником по входам RGB (R — красный, G — зеленый, B — синий). Для реализации этого режима на плате вычислителя должен быть смонтирован разъем XT11 и использован кабель, предназначенный для подключения цветного телевизионного приемника. (Если в данной модификации микроЭВМ цветной режим не используется, то элементы VT4-VT7, VT11, R37-R40 и R42-R49 на плате вычислителя отсутствуют.)

Микросхемы D5.6, D10.2, D20, D10.3 представляют собой дешифратор на два входа, на которые подается информация со сдвигового регистра. Комбинации логических значений расшифровываются и, в зависимости от выбранного режима работы, интерпретируются как один из цветов R, G, B или как одна из градаций яркости.

Микросхемы D10.3, D10.4, D5.6 и D10.2 позволяют выделить один из двух битов, поступающих со сдвигового регистра. В течение положительного полупериода тактовой частоты 6.0 МГц выделяется нечетный бит информации (слова), в течение отрицательного полупериода — четный бит. Таким образом, два байта за восемь тактов выстраиваются в слово, и на экране высвечивается 16 точек, соответствующих этому слову информации (512 точек в строке, две градации яркости). Микросхема D9.2 обеспечивает привязку видеосигнала к тактовой частоте 12 МГц, выравнивая длительности импульсов, поступающих со сдвигового регистра. Микросхемы D21.2-D21.4 и резисторы R52-R54 позволяют в зависимости от комбинации битов на выходе СР менять сопротивление в цепи базы транзистора VT8 (через перемычку 1-3 кабеля XT9) и тем самым изменять яркость точки (четыре градации яркости, каждой точке соответствуют два бита, за счет чего ТВ-строка содержит 256 точек).

Транзисторы VT4-VT7 и резисторы R37-R40, R42-R49 осуществляют развязку сигналов (по схеме

эмиттерного повторителя) с выхода дешифратора на входы R, СИНХР, G и B ТВ-приемника.

Инверторы D21.1, D21.6 и резисторы R51, R55, R56, R52-R54 формируют синхросигнал, поступающий с выхода SINCO БИС D19, и обеспечивают его смешивание с видеосигналом, как этого требует ГОСТ 7845-79.

Контроллер кассетного накопителя на магнитной ленте (КНМЛ) построен по принципу программно-аппаратного управления. Программная часть (драйвер) осуществляет последовательную выдачу или прием информации. Аппаратная — преобразует уровни сигналов (при передаче — ослабляет импульсный сигнал до амплитуды 0.5-1 В, при приеме — усиливает сигнал с линейного выхода магнитофона до уровня логических переходов микроЭВМ).

Контроллер подключен к каналу с помощью микросхем D12, D13. Микросхема D12 представляет собой двунаправленный шинный формирователь, управляемый сигналами ВВОД Н, ВЫВОД Н и SEL1 (сигналом, разрешающим обращение к регистру РНП по записи или по чтению). В зависимости от режима (запись или чтение) на входы СВ или СА подается уровень общей шины, и, таким образом, выбирается направление прохождения сигнала:

- запись (выбран СВ) — направление от W к B;
- чтение (выбран СА) — направление от A к W.

При направлении от W к B информация из канала через D12 поступает на входы D БИС D13, хранится в ней и с выводов D13/13, D13/12, D13/11 и D13/10 поступает соответственно на перемычку S3, контакт 5 разъема XT4 и VT3 (для управления реле включения мотора магнитофона). При направлении же от A к W информация с контакта 3 разъема XT4, D2.5/5 и перемычки S2 передается соответственно через A БИС D12 в канал микроЭВМ.

Запись информации на магнитную ленту. Драйвер КНМЛ управляет разрядами регистра ВР2 (РНП5, РНП6), к которым подключен резистивный делитель. В зависимости от комбинации в разрядах РНП на выходе делителя (контакт 5 XT4) уровень сигнала может иметь различные значения (оптимальный уровень записи устанавливается регулятором уровня записи магнитофона, как это указано в инструкции по эксплуатации).

Запись с использованием двух разрядов РНП5 и РНП6 (четырех уровней сигнала) производится с целью компенсации нелинейности частотной характеристики тракта магнитофона и для исключения постоянной составляющей в спектре записываемого сигнала. Пример фрагмента записи приведен на рис. 15.

Чтение информации с магнитной ленты. Сигнал с линейного выхода магнитофона (амплитуда 0.25-0.3 В) подается через транзисторы VT1 и VT2 на вход A1 БИС D12. Драйвер КНМЛ производит

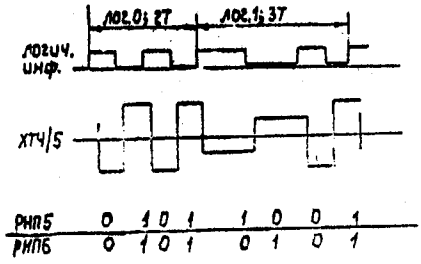


Рис. 15. Вывод на магнитную ленту

поиск синхроимпульса и анализ интервала времени между двумя синхроимпульсами. Большему интервалу (3Т) соответствует прием «лог. 1», меньшему (2Т) — прием «лог. 0». Затем драйвер формирует байт информации и заносит его в ОЗУ. По окончании приема (когда все слова записанного на ленте файла приняты) драйвер выдает сообщение об окончании чтения с магнитной ленты.

Программируемый параллельный интерфейс (порт ввода-вывода). Порт вывода построен на микросхемах К589ИР12 (D15, D16). Схема их включения отличается от используемой для буферного регистра данных заданием режима. Порт вывода имеет всегда активные выходы ВД00-ВД15, третье (отключенное) состояние отсутствует.

Обращение к порту вывода осуществляется по сигналу SEL2 БИС D14/08 и DOUT БИС D14/37. Сигнал SEL2 вырабатывается при обращении по адресу 177714_8. По сигналу DOUT производится собственно запись в порт информации, установленной в канале (входы D1-D8 БИС D15, D16). Выходы Q1-Q8 этих БИС подсоединены к внешнему разъему XT5, к которому могут быть подключены какие-либо устройства пользователя. Управление портом вывода — программное.

Входы порта ввода ВВ00-ВВ15 (микросхемы D17, D18) подключены к разъему XT5 и позволяют программно анализировать состояние устройств пользователя. Чтение этого состояния осуществляется по сигналам SEL2 и DIN БИС D14/38, информация передается в канал и далее по адресу в соответствии с программой пользователя.

Постоянное запоминающее устройство (ПЗУ) DS17-DS19 подключено к каналу в соответствии с требованиями технических условий на микросхему К1801РЕ1. Специфичным применительно к микроЭВМ является подключение вывода DIN (DS19/01) микросхемы DS19, который подключен не к магистральной линии DIN, а к выводу E БИС D19/37. Сигнал на входе DIN (DS19/01) разрешен в диапазоне адресов 0—177577_8, остальные адреса (177600-177777_8) зарезервированы для системных регистров (соответствующие ячейки БИС DS19 не используются).

(Окончание следует)

Микропроцессор К1801ВМ1. Справочные сведения

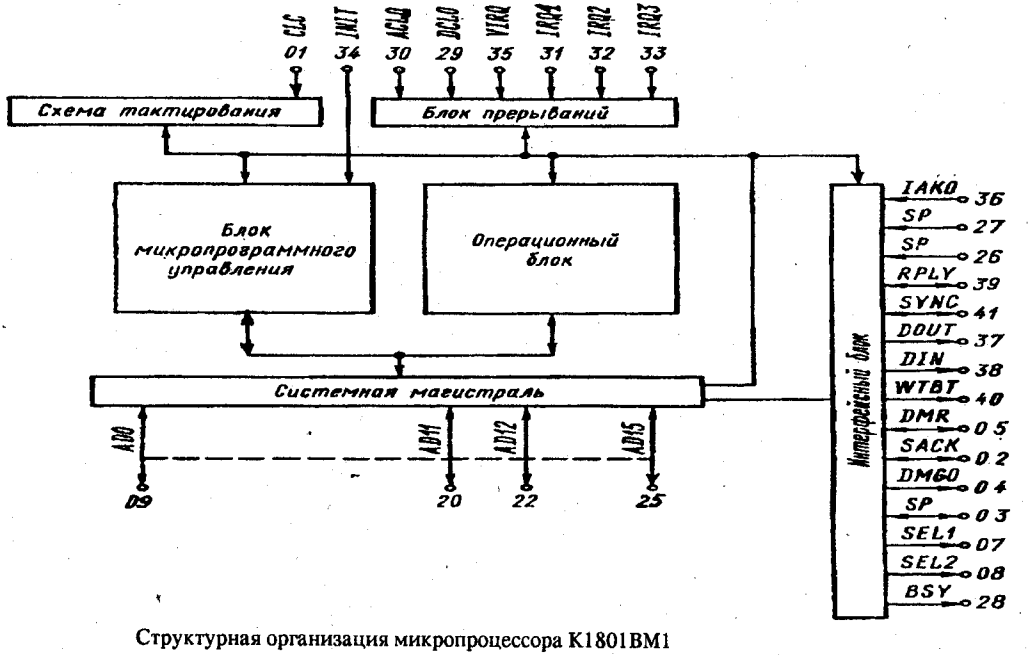
Однокристальный 16-разрядный микропроцессор К1801ВМ1 предназначен для выполнения следующих функций:

- вычисление адресов операндов и команд;
- обмен информацией с другими устройствами, подключенными к системной магистрали;
- обработка операндов;
- обработка прерываний от клавиатуры и устройств пользователя, подключенных к разъему порта ввода-вывода.

Процессор является единственным активным устройством микроЭВМ, управляющим циклами обращения к системной магистрали и обрабатывающим прерывания от пассивных устройств, которые могут посылать или принимать информацию только под управлением процессора.

Микропроцессор К1801ВМ1 работает в БК с тактовой частотой 3 МГц и содержит следующие основные функциональные блоки (см. рисунок):

- 16-разрядный операционный блок, служащий для формирования адресов команд и операндов, выполнения логических и арифметических операций, хранения операндов и результатов;
- блок микропрограммного управления, вырабатывающий последовательность микрокоманд, соответствующую коду принятой машинной команды. Этот блок построен на базе программируемой логической матрицы (ПЛМ), содержащей 250 логических произведений;
- блок прерываний, организующий приоритетную систему прерываний (прием и предварительная обработка внешних и внутренних запросов на прерывание);
- интерфейсный блок, обеспечивающий обмен информацией между микропроцессором и прочими устройствами, подключенными к системной магистрали. Этот же блок осуществляет арбитраж при операциях прямого доступа к памяти, формирует последовательность управляющих сигналов;
- блок системной магистрали, связывающий внутреннюю магистраль однокристального микропроцессора с внешней, управляющий усилителями приема и передачи ин-



Структурная организация микропроцессора К1801ВМ1

формации на совмещенные выводы адресов и данных;

- схема тактирования, обеспечивающая синхронизацию работы внутренних блоков микропроцессора.

Система команд, реализованная в ПЛМ блока микропрограммного управления микропроцессора К1801ВМ1, совпадает с системой команд наиболее распространенных отечественных мини- и микро-ЭВМ типа «Электроника 60» (ДВК-2, 3, 4 и т.п.) и практически аналогична принятой для компьютеров серии DEC. Предусмотрен также ряд специальных команд, предназначенных для работы с системным ПЗУ К1801РЕ1.

Сигналы **AD0-AD15** представляют собой адреса и данные, передаваемые по совмещенной системной магистрали. Передача адресов и данных по одним и тем же линиям связи обеспечивается путем разделения этих операций во времени.

Группа сигналов **SYNC, DIN, DOUT, WTBT, RPLY** служит для управления передачей информации по системной магистрали:

- SYNC - вырабатывается процессором как указание, что адрес находится на выводах системной магистрали, и сохраняет активный уровень до окончания текущего цикла обмена информацией;
- RPLY - вырабатывается пассивным устройством в ответ на сигналы DIN и DOUT. При отсутствии сигнала RPLY (т.е. когда выбранное устройство - регистр или ячейка памяти - не отвечает) процессор отсчитывает 64 такта синхрогенератора и затем отрабатывает прерывание по зависанию (вектор 4);
- DIN - предназначен для организации ввода данных (когда микропроцессор во время действия сигнала SYNC готов принять данные от пассивного устройства) и ввода адреса вектора прерывания (DIN вырабатывается совместно с сигналом IAK0 при пассивном уровне SYNC);
- DOUT - означает, что данные, выдаваемые микропроцессором, установлены на выводах системной магистрали;
- WTBT - указывает на работу с отдельными байтами и вырабатывается при обращении по нечетному адресу (операнд - старший байт) или при отработке байтовых команд.

Сигнал **VIRQ** является запросом на прерывание от внешнего устройства, информирующим микропроцессор о готовности устройства передавать адрес вектора прерывания. Если прерывание разрешено, то в ответ на этот сигнал процессор вырабатывает сигналы DIN и IAK0.

Сигнал **IRQ1** обеспечивает управление режимом «СТОП-ПУСК» процессора с внешнего переключателя аналогично использованию SA1 в отсеке пользователя. Низкий уровень сигнала (активный) соответствует положению «СТОП» SA1.

Сигналы **IRQ2** и **IRQ3** (последний в БК не используется) вызывают прерывания по фиксированным векторам 100_8 и 270_8 соответственно (при переходе из высокого уровня в низкий).

Сигнал предоставления прерывания **IAK0** процессор вырабатывает в ответ на внешний сигнал VIRQ. Сигнал IAK0 передается по очереди, начиная с устройства с максимальным приоритетом, ретранслируясь от одного устройства к другому в порядке уменьшения приоритетов. Устройство с наибольшим приоритетом из числа выставивших запрос на прерывание (сигнал VIRQ) запрещает дальнейшее распространение сигнала IAK0, таким образом запрещая на время обработки данного прерывания запросы от устройств с тем же или более низким приоритетом. Однако устройства с более высоким приоритетом могут прервать обработку повторным («вложенным») прерыванием.

Сигнал **DMR** вырабатывается внешним активным устройством, требующим передачи ему системной магистрали (режим прямого доступа к памяти). В ответ на него процессор устанавливает сигнал **DMGO**, предоставляющий системную магистраль внешнему устройству с наивысшим приоритетом из числа запросивших прямой доступ (механизм реализации приоритетов - тот же, что и для прерываний). Это устройство прекращает дальнейшее распространение сигнала DMGO и выставляет сигнал **SACK**, означающий, что устройство прямого доступа к памяти (ПДП) может производить обмен данными, независимо от процессора используя стандартные циклы обращения к системной магистрали.

Низкий уровень сигнала **BSY** означает, что микропроцессор начинает обмен по магистрали (т.е. что она занята для других устройств). Переход сигнала из низкого уровня в высокий указывает на окончание обмена.

Сигнал аварии источника питания **DCLO** вызывает установку микропроцессора в исходное состояние и появление сигнала INIT. Сигнал аварии сетевого питания **ACLO** вызывает переход микропроцессора на обработку прерывания по сбою питания (высокий уровень свидетельствует о нормальном сетевом напряжении). В БК-0010 сигналы ACLO и DCLO использованы нестандартно: для переключе-

Общие технические хаактеристики микропроцессора К1801ВМ1

Представление чисел	В дополнительном коде с фиксированной запятой
Виды команд	Безадресные, одноадресные, двухадресные
Виды адресации	Регистровая, регистровая косвенная, автоинкрементная, автоинкрементная косвенная, автодекрементная, автодекрементная косвенная, индексная, индексная косвенная
Количество регистров общего назначения	8
Количество уровней прерывания	4
Тип системной магистрали	Q-bus (МПИ, ОСТ 11.305.903-80)
Адресное пространство, Кб	64
Тактовая частота, МГц	До 5
Максимальное быстродействие при выполнении регистровых операций, оп./с	До 500000
Потребляемая мощность, Вт	Не более 1
Напряжение питания, В	+5(±5%)
Уровни сигналов, В: «лог. 0» (активный уровень)	Менее 0.5
«лог. 1»	Более 2.4
Нагрузочная способность по току, мА	3.2
Емкость нагрузки, пФ	До 100
Технология изготовления	N-МОП
Конструкция	Планарный металлокерамический корпус с 42 выводами

ния режимов «СТОП-ПУСК» аналогично IRQ1 - см. статью «Техническое описание БК-0010».

Сигнал **SEL1** инициализирует обращение к регистру управления системными внешними устройствами (адрес 177716_8), а сигнал **SEL2** - к регистру порта ввода-вывода (адрес 177714_8). Направление обмена данными между микропроцессором и регистрами определяется сигналами DIN или DOUT соответственно. Выставление сигнала RPLY от этих регистров не требуется. Длительности сигналов SEL1 и SEL2 совпадают с длительностью сигнала BSY.

Сигнал **INIT** является ответом микропроцессора на сигнал DCLO и используется, как правило, для установки периферийной части системы в исходное состояние.

Рекомендуемая литература

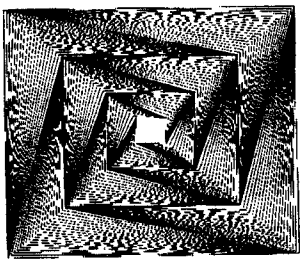
1. Микропроцессоры. Справочное пособие /Под ред. Овечкина Ю.А. Л.: Судостроение, 1987.

2. *Джхунян Б.Л.* Однокристальные микропроцессоры комплекта БИС серии К1801 //Микропроцессорные средства и системы. 1984. №4.

3. *Осетинский Л.Г., Осетинский М.Г., Писаревский А.Н.* ФОКАЛ для микро- и мини-компьютеров. Л.: Машиностроение, 1988.

4. *Конюшенко А.М.* Что может делать процессор 1801ВМ1 //Вычислительная техника и ее применение. 1991. №7.

5. *Лин В.* PDP-11 и VAX-11. Архитектура ЭВМ и программирование на языке ассемблера. М.: Радио и связь, 1989.



Большинство пользователей БК-0010.01 «не очень-то жалуют» язык ФОКАЛ, чаще всего отдавая предпочтение БЕЙСИКу. Одной из причин такой «дискриминации» является отсутствие в ФОКАЛе литерных (текстовых) переменных и определенные сложности (особенно для начинающих программистов) при создании интерфейса «ФОКАЛ-программа — пользователь». Предлагаемая библиотека подпрограмм призвана значительно облегчить эту задачу и, как можно надеяться, послужит основой для множества различных полезных программ.

С. А. Разбитной,

г. Москва

ОРГАНИЗАЦИЯ ИНТЕРФЕЙСА НА ФОКАЛЕ

Предлагаемый пакет сервисных подпрограмм содержит некоторые средства ввода и вывода данных, включая как простые, обеспечивающие лишь лаконичность записи, так и достаточно сложные, облегчающие разработку прикладных программ. Среди них — очистка всего экрана и отдельно его нижней части, подача звукового сигнала, ввод одного символа, строки и числа с возможностью исправлений, табличный ввод и вывод данных, организация окон и меню.

Подпрограммы написаны на ФОКАЛе и занимают одну группу строк (1.1—1.99). При запуске программы, их вызывающей, командой G в строках 1.12—1.16 производится присвоение номеров строк для входа в подпрограммы их условным именам-переменным, а затем командой G 1.99 управление передается на пустую строку 1.99 и далее на начало основной программы.

Вызов подпрограмм осуществляется с помощью операторов D или G по их именам. Внутри подпрограмм используются переменные, имена которых содержат букву «О» (IO, O2, OG и т.п.). Имеется также переменная PI=3.14159, доступная в любом месте программы.

Пустая строка

Вызов: D O, G O или I(...)O.

Иногда в программе требуется передать управление на пустую строку (как правило, при выходе из подпрограммы с помощью логического оператора). Переменная О содержит номер пустой строки.

Очистка экрана

Вызов: D CL.

Эквивалентно нажатию клавиши «СБР» — экран полностью очищается.

Очистка нижней части экрана

Вызов: D CM.

Очищается «зона меню» — нижние строки экрана, начиная с двадцатой.

Очистка строки

Вызов: D CS.

Очищается строка справа от курсора.

Звуковой сигнал

Вызов: D BE.

Выводится пять «щелчков» (код 7).

Ввод символа без эхо-печати

Вызов: D SI.

Ввод кода нажатой клавиши в переменную СО без отображения на экране.

Ввод символа с эхо-печатью

Вызов: D SY.

Ввод кода нажатой клавиши в переменную СО с выводом введенного символа на экран.

Ввод строки

Вызов: D ST.

Ввод в массив СО кодов нажимаемых клавиш с отображением символов на экране. В процессе ввода можно исправлять ошибки, используя клавишу «ЗАБОЙ». Ввод завершается при нажатии клавиши «ВВОД». Перед обращением нужно задать значение переменной JO — индекса массива. Код первого символа будет присвоен переменной CO[JO+1]. После окончания ввода JO содержит номер кода последнего символа (13 — «ВВОД») в массиве СО, а CO[JO+1] — значение 32 (код пробела).

Ввод числа

Вызов: D NU.

Ввод строки символов (с помощью подпрограммы ST) и получение значения числа, записанного в начале этой строки. При вводе можно использовать «минус», цифры и десятичную точку («плюс» использовать нельзя). Любой другой символ воспринимается как конец записи числа. При вводе можно исправлять ошибки, используя клавишу «ЗАБОЙ». Значение числа будет присвоено переменной АО.

Ввод данных в таблице

Вызов: D TB.

На экран выводятся числа заданной пользователем произвольной таблицы. С помощью клавиш-стрелок «ВПРАВО», «ВЛЕВО», «ВВЕРХ» и «ВНИЗ» курсор подводится к любому из этих чисел. Их значения могут быть изменены путем ввода с клавиатуры. Выход из подпрограммы — после нажатия клавиши «СБР».

Таблица описывается следующим образом. Для каждой ее позиции требуется две строки программы, в первой из которых записаны операторы вывода значения переменной — элемента таблицы с пояснительной надписью в нужном месте экрана, а во второй — оператор присваивания этому элементу значения переменной AO (именно она вводится с клавиатуры), а также операторы, которые должны быть выполнены после ввода числа. Строки программы для всех позиций таблицы нумеруются подряд без пропусков с шагом 0.01. При обращении к подпрограмме переменной TH нужно присвоить значение номера первой строки описания таблицы, а переменной TK — значение номера последней строки.

Пример: объем параллелепипеда.

```
5.10 S TH=5.2;S TK=5.25;D TB
5.20 X FK(10,1);T "Длина=",L
5.21 S L=AO;D 5.3
5.22 X FK(10,2);T "Ширина=",B
5.23 S B=AO;D 5.3
5.24 X FK(10,3);T "Высота=",H
5.25 S H=AO;D 5.3
5.30 X FK(15,5);S V=L*B*H;
     T"Объем=",V
```

Теперь оператор D 5.1 выведет на экран три строки, в которых можно будет вводить значения переменных L, B и H (длина, ширина и высота). После каждого изменения будет заново вычисляться и печататься объем параллелепипеда. Ввод можно повторять многократно.

Организация таблицы в окне

Вызов: D WT.

В нескольких строках, образующих на экране окно, располагается таблица, имеющая заголовок и произвольное количество строк данных (их может быть больше, чем размер окна по вертикали). С помощью клавиш-стрелок «ВВЕРХ» и «ВНИЗ» можно перемещать курсор по строкам таблицы, а при подходе к границам окна ее текст смещается вверх или вниз. Строку, помеченную курсором, можно редактировать (после нажатия клавиши «ВВОД»). В конец таблицы можно добавлять новые строки (подвести курсор к последней строке, нажать стрелку «ВНИЗ» и после звукового сигнала нажать ее еще раз). Выход из подпрограммы — клавиша «СБР».

Перед первым обращением нужно задать переменные:

NS —количество строк данных в таблице;
NJ —номер строки на экране, в которую будет выводиться заголовок;
NI —количество строк на экране, отводимое для вывода данных (от NJ+1 до NJ+NI);
PB —номер строки программы, в которой записаны операторы вывода заголовка;
PS —номер строки программы с операторами вывода строки данных;
PW —номер строки программы с операторами редактирования строки данных.

Во время работы используются переменные:

I — номер текущей строки данных (помеченной курсором);
II — номер строки в окне, в которую выводится эта строка данных;
NJ+II — номер этой строки на экране.

Пример: таблица координат точек.

```
5.10 S NS=10;S NJ=0;S NI=5;
     S PB=5.2;S PS=5.3;
     S PW=5.4;D WT
5.20 T "Номер Координата X
     Координата Y"
5.30 T %4,I," ",%8.04,X[I],
     " ",Y[I]
5.40 X FK(0,10);D CS;G 5.5
5.50 T "I",I," X=";D NU;
     S X[I]=AO;T " Y=";
     D NU;S Y[I]=AO
```

Теперь оператор D 5.1 выведет на экран таблицу из пяти строк. Перемещая курсор, в ней можно просмотреть и изменить координаты десяти точек и добавить новые. Другие операторы: D 5.1;D CL;F I=1,NS;X FT(1,X[I],Y[I]), введенные в непосредственном режиме, вызовут эту таблицу, а после окончания работы с ней очистят экран и нарисуют на нем точки с заданными координатами. (Перед вводом этих операторов нужно хотя бы раз запустить программу с начала командой G с последующим остановом по клавише «СТОП». — *Прим.ред.*)

Организация меню

Вызов: G ME или D ME.

В нижней части экрана печатается меню — несколько пунктов, помеченных цифрами (от 1 до 9). При нажатии на цифровую клавишу осуществляется передача управления к соответствующему участку программы. Имеется возможность организации иерархической системы вложенных меню.

Перед обращением должны быть заданы переменные:

NM – количество пунктов меню;

GO – номер группы строк программы (целое число), в которой описано меню;

UM – номер группы строк программы, в которой описано меню более высокого уровня. При нажатии клавиши «СБР» управление будет передано в строку с номером UM+0.05.

На каждый пункт меню отводится несколько строк программы. В тех из них, номера которых оканчиваются на 0 (если GO=5, это 5.10, 5.20 и т.д.), записывается оператор печати названия пункта. При выборе в меню какого-либо пункта управление передается в строку, номер которой оканчивается на 1 (соответственно 5.11,5.21 и т.д.).

Пример: объем параллелепипеда.

```
5.05 S NM=4;S GO=5;G ME
5.10 T "Длина"
5.11 D CM;D 5.1;D NU;S L=AO;G 5.05
5.20 T "Ширина"
5.21 D CM;D 5.2;D NU;S B=AO;G 5.05
5.30 T "Высота"
5.31 D CM;D 5.3;D NU;S H=AO;G 5.05
5.40 T "Объем"
5.41 S V=L*H*B;D CM;D 5.4;T V
```

Оператор G 5.05 выведет на экран меню из четырех пунктов. При выборе любого из первых трех машина попросит ввести соответствующую величину и вернется в это же меню. При выборе четвертого пункта будет вычислен и выведен на экран объем параллелепипеда.

Листинг сервисных подпрограмм:

```
1.10 C СЕРВИСНЫЕ ПОДПРОГРАММЫ
1.12 S CL=1.2;S CM=1.21;S SI=1.22;S SY=1.23;S BE=1.24
1.14 S ST=1.25;S RB=1.27;S CS=1.28;S NU=1.3;S TB=1.4;S WT=1.6;S ME=1.8
1.16 S PI=3.14159;S O=1.99;G O
1.20 X FCHR[12]
1.21 X FK[0,20];X FCHR[153,27,153,27,153,27,153,26,26,26]
1.22 S CO=FCHR[-1]
1.23 S CO=FCHR[FCHR[-1]]
1.24 X FCHR[7,7,7,7,7]
1.25 S JO=JO+1;D SY;S CO[JO]=CO;I [-[CO-13]^2] 1.26;S CO[JO+1]=32
1.26 I [-[CO-24]^2] 1.25;S JO=JO-2;G 1.25
1.27 X FCHR[155]
1.28 X FCHR[153]
1.30 S AO=0;S JO=0;G 1.31
1.31 S ZO=1;D 1.32;S AO=AO*ZO
1.32 D ST;S JO=0;I [-[CO[1]-45]^2] 1.33;S JO=1;S ZO=-1;G 1.33
1.33 D 1.36;I [[48-CO]*[CO-57]] 1.34;S AO=AO*10+CO-48;G 1.33
1.34 I [-[CO-46]^2]O;S OO=1;G 1.35
1.35 D 1.36;I [[48-CO]*[CO-57]] O;S OO=OO/10;S AO=AO+OO*[CO-48];G 1.35
1.36 S JO=JO+1;S CO=CO[JO]
1.40 D 1.58;D 1.42;S GO=TH;D GO;G 1.44
1.42 F GO=TH,.02,TK;D GO
1.44 D SI;G 1.46
1.46 I [-[[CO-8]*[CO-26]]^2] 1.48;S GO=GO-.02;I [TH-GO] 1.57,1.57;S GO=TK-.01;G 1.57
1.48 I [-[[CO-25]*[CO-27]]^2] 1.5;S GO=GO+.02;I [GO-TK] 1.57,1.57;S GO=TH;G 1.57
1.50 I [-[CO-12]^2] 1.52;
1.52 I [[48-CO]*[CO-57]*[CO-45]^2] 1.56;D BE;T ":";D NU;G 1.54
1.54 S OO=GO+.01;D OO;D GO;S CO=27;G 1.48
1.56 D BE;G 1.44
1.57 D GO;G 1.44
1.58 X FK[0,20];T "← → ↑ ↓ 0..9 СБР"
1.60 I [NI-NS] 1.61;S NI=NS;G 1.61
1.61 X FK[0,NJ];D PB;D 1.63;D 1.72;S I=1;S II=1;G 1.64
1.63  F I=1,NI;X FK[0,NJ+I];D PS
1.64 X FK[0,NJ+I];G 1.65
1.65 D SI;G 1.66
1.66 I [-[CO-26]^2] 1.67;S II=II-1;S I=I-1;I [0-II] 1.71;S II=1;D 1.73;G 1.71
1.67 I [-[CO-27]^2] 1.68;S II=II+1;S I=I+1;I [II-NI-1] 1.71;S II=NI;D 1.75;G 1.71
```

```
1.68  I [-[CO-13]^2] 1.69;D PW;D 1.72;G 1.71
1.69  I [-[CO-12]^2] 1.7;D CM
1.70  D BE;G 1.65
1.71  X FK[0,NJ+II];D PS;X FK[0,NJ+II];G 1.65
1.72  D CM;T !,!," ↑ ↓  ВВОД СБР"
1.73  I [-I] 1.74;D BE;S I=1
1.74  X FK[0,NJ+NI];X FCHR[19];X FK[0,NJ+1];X FCHR[20];D PS
1.75  I [I-NS-1] 1.76;D BE;G 1.77
1.76  X FK[0,NJ+1];X FCHR[19];X FK[0,NJ+NI];X FCHR[20];D PS
1.77  I [-[FCHR[-1]-27]^2] O;S NS=NS+1;S I=NS;G 1.76
1.80  D CM;D 1.81;D 1.93;D 1.82;D 1.94;S OG=GO+OO/10+.01;G OG
1.81  X FK[0,20];T "═══════════════════════"
1.82  F OO=1,NM;S OG=1.83+.01*OO;D OG;G 1.83
1.83  X FCHR[156,OO+48,156];S OG=GO+OO/10;D OG
1.84  X FK[0,21]
1.85  X FK[0,22]
1.86  X FK[0,23]
1.87  X FK[O1,21]
1.88  X FK[O1,22]
1.89  X FK[O1,23]
1.90  X FK[20,21]
1.91  X FK[20,22]
1.92  X FK[20,23]
1.93  I [10-O1] O;S O1=10
1.94  S CO=FCHR[-1]-48;I [[CO+36]^2] O,1.95;I [-[CO-1]*[NM-CO]] 1.96,1.96;D BE;G 1.94
1.95  S GO=UM;S OO=.4
1.96  S OO=CO
1.99
```

А. Л. Карп
г. Целиноград

ЛИТЕРНЫЕ ВЕЛИЧИНЫ НА ФОКАЛЕ

Предлагаемая программа позволяет осуществлять ввод и вывод большого количества литерных величин в ФОКАЛе. Вводимые символы записываются в двумерный массив Y(I,J), где I — номер слова, а J — позиция символа в слове. Так как длина вводимых слов может быть различной, вычисляется ее максимальное значение (группа строк 7).

С помощью этой программы легко реализуется упорядочивание латинских слов по алфавиту (по первой букве). Для этого необходимо сравнивать первые символы слов с десятичными кодами латинских букв (строка 1.3 и подпрограмма 6).

```
01.01  E  ; С СОРТИРОВКА ЛИТЕРНЫХ
       ВЕЛИЧИН (LAT)
01.10  A "КОЛИЧЕСТВО ЛИТЕРНЫХ
       ВЕЛИЧИН",N;     F I=1,N; D 5
01.20  S  MAX=K(1); F I=1,N; D 7
01.30  F  L=65,90; F I=1,N; D 6
01.40  Q
05.10  C  ПОДПРОГРАММА ВВОДА СЛОВ
05.15  S  J=1
05.20  S  Y(I,J)=FCHR(FCHR(-1))
05.30  I  (Y(I,J)-13) 5.40,5.50,5.40
05.40  S  J=J+1; G 5.20
05.50  S  K(I)=J; T !; R
06.10  C  ПОДПРОГРАММА ВЫВОДА СЛОВ
       ПО АЛФАВИТУ
06.20  I  (Y(I,1)-L) 6.40,6.30,6.40
06.30  T  !; F Z=1,MAX-1; X FCHR(Y(I,Z))
06.40  R
07.05  C  ПОДПРОГРАММА ОПРЕДЕЛ.
       ДЛИНУ МАКСИМ. СЛОВА
07.10  I  (K(I)-MAX) 7.30,7.30,7.20
07.20  S  MAX=K(I)
07.30  R
```

Программу легко модернизировать и для упорядочивания русских слов. Для этого необходимо добавить строки 1.04–1.09, в которых в массив B в алфавитном порядке вводятся коды русских букв:

```
01.04 S  B(1)=225;S B(2)=226;S B(3)=247;
  S B(4)=231;S B(5)=228;S B(6)=229;C А-Е
01.05 S  B(7)=246;S B(8)=250;S B(9)=233;
  S B(10)=234;S B(11)=235;S B(12)=236;C Ж-Л
01.06 S  B(13)=237;S B(14)=238;S B(15)=239;
  S B(16)=240;S B(17)=242;C М-Р
01.07 S  B(18)=243;S B(19)=244;S B(20)=245;
  S B(21)=230;S B(22)=232;C С-Х
01.08 S  B(23)=227;S B(24)=254;S B(25)=251;
  S B(26)=253;S B(27)=249;C Ц-Ы
01.09 S  B(28)=248;S B(29)=255;S B(30)=252;
  S B(31)=224;S B(32)=241;C Ь-Я
```

Соответственно, требуется изменить две уже существующие строки:

```
01.30 F L=1,32; F I=1,N; D 6
06.20 I (Y(I,1)-B(L)) 6.40,6.30,6.40
```

Примечание редактора

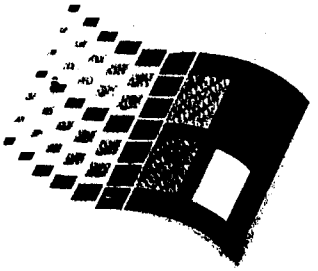
Предлагаемый способ работы с литерными величинами на ФОКАЛе, наверное, заинтересует многих БКманов: несмотря на широкую популярность вильнюсского БЕЙСИКа и появление других языков программирования (АЛМИК, ЛОГО, СИ Цаплева А.В., ФОРТ, две версии ПАСКАЛЯ и т.д.) ФОКАЛ еще пользуется среди программистов БК-0010(.01) определенным успехом. Но в ФОКАЛе, к сожалению, не реализованы многие удобные функции: звук, развитая графика, подключение кодовых подпрограмм типа USR и, в частности, работа с символьными переменными.

Сама по себе идея хранения кодов символов в массиве не нова (см. хотя бы «МикроЭВМ в учебных заведениях» // Под ред. Преснухина Л.Н. М.: Высшая школа, 1988 г.—кн.8. С.76; кстати там же на странице 74 высказан еще один любопытный вариант — запись слова в числовую переменную). Однако предложенная автором статьи реализация этой идеи выделяется из ряда ранее опубликованных возможностью хранения и обработки в одном и том же «комплекте» подпрограмм любого числа слов. (Увы, ФОКАЛ не умеет передавать в подпрограммы имена массивов в качестве формальных аргументов, и для разных массивов пришлось бы дублировать подпрограммы.) Кроме того, автором предложен простой (хотя и медленный) способ сортировки по алфавиту во время вывода слов на экран. Следует отметить, что можно вводить и целые предложения — пробел как разделитель между словами здесь равноправен с прочими символами. Вообще же при использовании данного метода можно применять любые знаки: русские и латинские, строчные и заглавные, полуграфику и даже коды, не имеющие графического представления (например, курсорные «стрелки»). Что же касается требования использовать при работе с программами только заглавные латинские (первый авторский вариант) или русские буквы, оно порождено исключительно особенностями реализации сортировки по алфавиту.

Однако, как известно, любую программу можно улучшить. Например, удобнее было бы использовать для хранения количества букв в каждом слове незадействованные пока ячейки Y(I,0), высвободив таким образом массив K(I) для других целей. (Можно при желании и количество слов хранить в переменной Y(0,0), освободив N и размещая в массиве Y всю символьную информацию, какая только есть в программе.) Кроме того, параметр MAX (максимальная длина слова) в большинстве случаев не нужен. В качестве иллюстрации приведем оптимизированный вариант первого листинга:

```
01.01E;  С Работа с литерными величинами
   (ЛАТ) - вариант от редакции
01.10 A "Количество слов",N; F I=1,N; D 5
01.30 F L=65,90; F I=1,N; D 6
01.40 Q
05.10 C Ввод слов
05.15 S J=1
05.20 S Y(I,J)=FCHR(FCHR(-1))
05.30 I (Y(I,J)-13) 5.40,5.50,5.40
05.40 S J=J+1; G 5.20
05.50 S Y(I,0)=J; T !; R
06.10 C Вывод слов по алфавиту (вариант LAT)
06.20 I (Y(I,1)-L) 6.40,6.30,6.40
06.30 T !; F Z=1,Y(I,0)-1; X FCHR(Y(I,Z))
06.40 R
```

В заключение отметим такой любопытный факт. Та же идеология — массив, хранящий коды символов, — используется и в таком мощном языке, как СИ на IBM-совместимых компьютерах. Оказывается, ФОКАЛ-то наш не так уж и прост, вон у него какие «богатые родственники за границей»! Только код конца строки в ФОКАЛе не 0, а 13. Так что, посидев пару-другую часов с БК, любой мало-мальски подготовленный пользователь сможет написать для ФОКАЛа целую библиотеку «символьных функций»: копирование строк, сравнение, поиск вхождений и т.п. — не хуже, чем в Си. .Так что — дерзайте!



Одним из критериев качества современной программной разработки является внешнее оформление. И это не только чисто эстетические соображения. Пользователи IBM-совместимых компьютеров (которые сегодня стало общепринятым приводить в качестве «положительного примера») знают, насколько облегчается удобство работы с программой, снабженной оконным интерфейсом, подробными Help-ами и развитой системой меню. Да и ошибок при этом пользователи (в особенности начинающие) делают гораздо меньше. Самый показательный пример — практически повсеместный переход от «чистой» DOS к Norton Commander и далее к Windows.

Д. Ю. Усенков,
г. Москва

БК-0010: ЕЩЕ НЕ WINDOWS, НО ...

БК-0010 по сравнению с IBM не может похвастаться большим объемом памяти, так что надеяться на появление для нее версии Windows могут лишь самые отчаянные оптимисты. Но хотя бы простейшая реализация окон и меню доступна и БКманам. Для этого служат предлагаемые подпрограммы на ассемблере стандарта М18.

Организация оконного интерфейса

Производится вывод окна заданных размеров с «тенью» в указанном месте экрана без сохранения прежнего изображения.

Пример вызова (сообщение об ошибке чтения):

```
    MOV #10,&XX     ;координаты верхнего левого
    MOV #12,&YY     ; угла: (&O10,&O12)
    CLR &CVF        ;печать тени
    CLR &CVT        ; в виде черных
    MOV ' ',&TS     ; пробелов
    JSR R7,WINDOW   ; вызов подпрограммы
    .@OKERCH        ;ссылка на метку текста окна
    .#20            ;размеры окна
    .#3             ; &O20*&O3
    .#177777        ;красный фон,
    .#125252        ; зеленая надпись
    EMT 6           ;ждем нажатия клавиши

OKERCH: .A:+- - - - - - - - - - - - - - -+
        .A: | ОШИБКА ЧТЕНИЯ. |
        .A:+- - - - - - - - - - - - - - -+
        .E
&XX:    .#0
&YY:    .#0
&CVF:   .#0
&CVT:   .#0
&TS:    .#0
```

Листинг подпрограммы:

```
; Подпрограмма "WINDOW"
;- - - - - - - - - - - - - - - - - - - - - - - - - - - -
; Входные данные:
;   1) перечисляемые:
;     .@TEKST - метка текста окна
;     .#X
;     .#Y       - (X,Y) размеры окна
;     .#******* - цвет фона окна
;     .#******* - цвет текста окна
;
;   2) задаваемые операторами MOV
;       &XX     - координаты левого
;       &YY     - верхнего угла окна
;       &CVF    - цвет фона "тени"
;       &CVT    - цвет символа "тени"
;       &TS     - код символа для "тени"
;- - - - - - - - - - - - - - - - - - - - - - - - - - - -
; Текст окна набирать вместе с рамкой
; (полуграфика), потоком, строка за строкой,
; без разделяющих знаков "перевод строки",
; в конце текста обязателен код "0"
;- - - - - - - - - - - - - - - - - - - - - - - - - - - -
; На выходе сохраняются все регистры и
; переменные, кроме &XX и &YY
;- - - - - - - - - - - - - - - - - - - - - - - - - - - -
WINDOW: MOV R0,-(R6)
        MOV R1,-(R6)
        MOV R2,-(R6)
        MOV R3,-(R6)
        MOV R4,-(R6)
        MOV R5,-(R6)
        MOV @#212,-(R6)
        MOV @#214,-(R6)
        MOV 20(R6),R0
        MOV R0,R1
        ADD (R0)+,R1
```

```
        MOV (R0)+,%X
        MOV (R0)+,R3
        MOV R3,%Y
        MOV (R0)+,@#212
        MOV (R0)+,@#214
        MOV R0,20(R6)
        MOV R1,-(R6)
1:      MOV &XX,R1
        MOV &YY,R2
        EMT 24
        MOV (R6)+,R1
        MOV %X,R2
        EMT 20
        MOV R1,-(R6)
        INC &YY
        SOB R3,1
        TST (R6)+
        INC &XX
        MOV &XX,R1
        MOV &YY,R2
        EMT 24
        MOV &CVF,@#212
        MOV &CVT,@#214
        MOV %X,R1
        MOV &TS,R0
2:      EMT 16
        SOB R1,2
        DEC %Y
        MOV %Y,R1
3:      MOV #34,R0
        EMT 16
        MOV &TS,R0
        EMT 16
        SOB R1,3
        MOV (R6)+,@#214
        MOV (R6)+,@#212
        MOV (R6)+,R5
        MOV (R6)+,R4
        MOV (R6)+,R3
        MOV (R6)+,R2
        MOV (R6)+,R1
        MOV (R6)+,R0
        RTS R7
;
%X:     .#0
%Y:     .#0
;-------------------------------
```

Реализация вертикальных и горизонтальных меню

Две другие почти однотипные подпрограммы позволяют реализовать вертикальные и горизонтальные меню. Пользователь может, перемещая «подсветку» клавишами «СТРЕЛКА ВВЕРХ» и «СТРЕЛКА ВНИЗ» (для горизонтального меню — соответственно, «ВЛЕВО» и «ВПРАВО»), выбрать нужный пункт. После нажатия клавиши «ВВОД» управление возвращается вызвавшей программе, причем в регистре R0 передается номер выбранного пункта (считая с единицы).

Следует заметить, что в этих подпрограммах не реализуется вычерчивание рамки меню. Для этого нужен отдельный предварительный вывод «пустого» окна.

Пример вызова вертикального меню (горизонтальное — аналогично):

```
MOV #177777,&CVT  ; красный текст
MOV #125252,&CVF  ; на зеленом фоне
MOV #52525,&CVK   ; с синим указателем
MOV #100,&ZDLIT   ; длительность и частота звука
MOV #160,&ZCH     ; при нажатии на клавиши
JSR R7,MENUV      ; вызов подпрограммы
.@MENU1           ; текст пунктов меню
.#6               ; длина пункта - 6 символов
.#4               ; верхний левый угол
.#4               ; меню - (4,4)
.#4               ; количество пунктов меню - 4
CMP R0,#1         ; анализ номера
BNE HELP          ; выбранного пункта
.....
&XX:      .#0
&YY:      .#0
&CVF:     .#0
&CVT:     .#0
&TS:      .#0
&CVK:     .#0
&ZDLIT:   .#0
&ZCH:     .&0
MENU1:    .A: HELP ПУНКТ1ПУНКТ2 QUIT
          .B:40.E
```

Листинги подпрограмм:

```
; Подпрограмма вертикального меню
; без рамки (перемещаемая)
;-------------------------------
; Входные данные: R0 - исходная позиция
; указателя
;   1) перечисляемые:
;      .@TEKST  - метка текста меню
;      .#******* - длина пункта меню(симв)
;      .#******* - позиция верхнего левого
;      .#******* - угла меню
;      .#******* - количество пунктов меню
```

```
;  2) задаваемые оператором MOV
;     &CVF    - цвет фона меню
;     &CVK    - цвет указателя
;     &CVT    - цвет текста
;     &ZDLIT - длительность подзвучки
;     &ZCH    - частота подзвучки
;-------------------------------
; Выходные данные: R0 - номер выбранной
; позиции
;-------------------------------
; Текст меню набирается потоком, по порядку
; следования пунктов с их пробелами,
; в конце код 0
;-------------------------------
; Сохраняет значения регистров кроме R0
;-------------------------------
MENUV:  MOV R1,-(R6)
        MOV R2,-(R6)
        MOV R3,-(R6)
        MOV R4,-(R6)
        MOV R5,-(R6)
        MOV R0,-(R6)
        MOV 14(R6),R0
        MOV R0,R1
        ADD (R0)+,R1
        MOV (R0)+,R2
        MOV (R0)+,R3
        MOV (R0)+,R4
        MOV (R0)+,R5
        MOV R0,14(R6)
        MOV (R6)+,R0
        CMP R0,R5
        BHI 1
        TST R0
        BEQ 2
        BR 3
2:      MOV #1,R0
        BR 3
1:      MOV R5,R0
3:      MOV R3,-(R6)
        MOV R4,-(R6)
        JSR R7,MENUV1
        MOV (R6)+,R4
        MOV (R6)+,R3
        MOV R0,%X2
4:      EMT 6
        MOV R0,-(R6)
        MOV R2,-(R6)
        MOV R3,-(R6)
        MOV R1,-(R6)
        MOV R4,-(R6)
        MOV R5,-(R6
        MOV &ZCH,R3
        MOV &ZDLIT,R2
        JSR R7,@#102062
; звук с меняющейся частотой
        MOV (R6)+,R5
        MOV (R6)+,R4
        MOV (R6)+,R1
        MOV (R6)+,R3
        MOV (R6)+,R2
        MOV (R6)+,R0
        CMPB R0,#32
        BNE 5
        DEC %X2
        BGT 6
        MOV R5,%X2
        BR 6
5:      CMPB R0,#33
        BNE 7
        INC %X2
        CMP %X2,R5
        BLOS 6
        MOV #1,%X2
6:      MOV %X2,R0
        MOV R3,-(R6)
        MOV R4,-(R6)
        JSR R7,MENUV1
        MOV (R6)+,R4
        MOV (R6)+,R3
        BR 4
7:      CMPB R0,#12
        BNE 4
        MOV %X2,R0
        MOV (R6)+,R5
        MOV (R6)+,R4
        MOV (R6)+,R3
        MOV (R6)+,R2
        MOV (R6)+,R1
        RTS R7
;
MENUV1: MOV R0,-(R6)
        MOV R1,-(R6)
        MOV R2,-(R6)
        MOV @#212,-(R6)
        MOV @#214,-(R6)
        MOV &CVT,@#214
        MOV #1,%X1
10:     CMP %X1,R0
        BEQ 11
        MOV &CVF,@#212
        BR 12
11:     MOV &CVK,@#212
12:     MOV R0,-(R6)
```

```
                MOV R5,-(R6)
                MOV R2,-(R6)
                MOV R1,-(R6)
                MOV R3,R1
                MOV R4,R2
                EMT 24
                INC R4
                MOV (R6)+,R1
                MOV @R6,R2
                EMT 20
                MOV (R6)+,R2
                MOV (R6)+,R5
                MOV (R6)+,R0
                INC %X1
                CMP %X1,R5
                BLOS 10
                MOV (R6)+,@#214
                MOV (R6)+,@#212
                MOV (R6)+,R2
                MOV (R6)+,R1
                MOV (R6)+,R0
                RTS R7
                .E;
;
%X1:            .#0
%X2:            .#0
;--------------------------------

;Подпрограмма горизонтального меню
; (перемещаемая)
; Входные данные:
; R0 - исходная позиция указателя
;
;   1) перечисляемые:
;     .@TEKST  - метка текста меню
;     .#******* - длина пункта меню (симв)
;     .#******* позиция левого верхнего
;     .#******* - угла меню
;     .#******* - количество пунктов меню
;
;   2) задаваемые оператором MOV
;      &CVF   - цвет фона меню
;      &CVK   - цвет указателя
;      &CVT   - цвет текста меню
;      &ZDLIT - длительность подзвучки
;      &ZCH   - частота подзвучки
; ---------------------------------------
; Выходные данные: R0 - текущая позиция
; указателя
; ---------------------------------------
; Текст набирать потоком, по порядку
; следования пунктов, с их пробелами,
; в конце - код 0.
; ---------------------------------------
; Сохраняет регистры кроме R0
; ---------------------------------------
MENUG:          MOV R1,-(R6)
                MOV R2,-(R6)
                MOV R3,-(R6)
                MOV R4,-(R6)
                MOV R5,-(R6)
                MOV R0,-(R6)
                MOV 14(R6),R0
                MOV R0,R1
                ADD (R0)+,R1
                MOV (R0)+,R2
                MOV (R0)+,R3
                MOV (R0)+,R4
                MOV (R0)+,R5
                MOV R0,14(R6)
                MOV (R6)+,R0
                CMP R0,R5
                BHI 1
                TST R0
                BEQ 2
                BR 3
2:              MOV #1,R0
                BR 3
1:              MOV R5,R0
3:              JSR R7,MENUG1
                MOV R0,%X3
4:              EMT 6
                MOV R0,-(R6)
                MOV R2,-(R6)
                MOV R3,-(R6)
                MOV R1,-(R6)
                MOV R4,-(R6)
                MOV R5,-(R6)
                MOV &ZCH,R3
                MOV &ZDLIT,R2
                JSR R7,@#102062
; звук с меняющейся частотой
                MOV (R6)+,R5
                MOV (R6)+,R4
                MOV (R6)+,R1
                MOV (R6)+,R3
                MOV (R6)+,R2
                MOV (R6)+,R0
                CMPB R0,#10
                BNE 5
                DEC %X3
                BGT 6
                MOV R5,%X3
                BR 6
5:              CMPB R0,#31
                BNE 7
                INC %X3
```

```
        CMP %X3,R5
        BLOS 6
        MOV #1,%X3
6:      MOV %X3,R0
        JSR R7,MENUG1
        BR 4
7:      CMPB R0,#12
        BNE 4
        MOV %X3,R0
        MOV (R6)+,R5
        MOV (R6)+,R4
        MOV (R6)+,R3
        MOV (R6)+,R2
        MOV (R6)+,R1
        RTS R7
;
MENUG1: MOV R0,-(R6)
        MOV R1,-(R6)
        MOV R2,-(R6)
        MOV @#212,-(R6)
        MOV @#214,-(R6)
        MOV R1,-(R6)
        MOV R2,-(R6)
        MOV &CVT,@#214
        MOV R0,-(R6)
        MOV R5,-(R6)
        MOV R3,R1
        MOV R4,R2
        EMT 24
        MOV (R6)+,R5
        MOV (R6)+,R0
        MOV (R6)+,R2
        MOV (R6)+,R1
        MOV #1,%X4
10:     CMP %X4,R0
        BEQ 11
        MOV &CVF,@#212
        BR 12
11:     MOV &CVK,@#212
12:     MOV R0,-(R6)
        MOV R5,-(R6)
        MOV R2,-(R6)
        EMT 20
        MOV (R6)+,R2
        MOV (R6)+,R5
        MOV (R6)+,R0
        INC %X4
        CMP %X4,R5
        BLOS 10
        MOV (R6)+,@#214
        MOV (R6)+,@#212
        MOV (R6)+,R2
        MOV (R6)+,R1
        MOV (R6)+,R0
        RTS R7
        .E
;
%X3:    #0
%X4:    #0
;-------------------------------
```

Н. Ю. Щербина, В. В. Руденко

СОЗДАНИЕ ИНТЕРАКТИВНЫХ МЕНЮ НА БЕЙСИКЕ

Одной из многих проблем, интересующих наших читателей, является реализация в программах на БЕЙСИКе меню, в котором можно было бы выбирать требуемую команду, наводя на соответствующий пункт «курсор»-подсветку с помощью клавиш-стрелок. В этой статье предлагается два варианта реализации меню с инверсным указателем, присланные читателями журнала.

Н.Ю. Щербина (г. Львов) предлагает использовать для «подсветки» текущего пункта меню небольшую программу на ассемблере:

```
        MOV ADR,R0
        MOV XX,R1
        MOV YY,R2
M1:     MOV R1,R3
M2:     COMB (R0)+
        SOB R3,M2
        ADD #100,R0
        SUB R1,R0
        SOB R2,M1
        RTS R7
```

Здесь ADR — адрес в ОЗУ экрана левого верхнего угла инвертируемого участка, XX — ширина инвертируемого участка в байтах, YY — высота инвертируемого участка в точках.

Соответственно, полный алгоритм реализации меню будет таким:

1. Вывести текст меню (или пиктограммы).

2. Установить начальные координаты «курсора» меню.

3. Произвести инверсию участка меню с текущими координатами.

4. Войти в режим ожидания нажатия соответствующей клавиши управления.

5. По нажатию клавиши произвести повторную инверсию ранее выбранного участка меню.

6. Изменить координаты в соответствии с нажатой клавишей.

7. Если нажата клавиша «ВВОД», организовать выход из меню в нужное место программы.

8. Для других клавиш — повторить алгоритм с п.3.

В.В. Руденко (г. Ивантеевка Московской обл.) предлагает другой вариант — именно с использованием режима вывода на экран инверсного текста (CHR$(&H9C)). В программе используются переменные I%, I5%, I1% и MA$. Переменная I% является счетчиком положения курсора в меню и с нее происходит съем информации при нажатии клавиши «ВВОД».

В строках 1300, 1400, 1500, 1600 и т.д. происходит позиционирование и задание информации для вывода на экран. Строки 1000-1055 — инициализация меню. Если добавить строку «1056 RETURN», то эту часть программы можно использовать самостоятельно для вывода на экран меню и присвоения начального значения счетчику I%. Обращение к этой подпрограмме: GOSUB 1000.

Вторая часть программы (строки 1140-1240) реализует цикл опроса клавиатуры и перемещения курсора. Строка 2000 осуществляет переход на подпрограммы 2100, 2110 и т.д. в зависимости от выбранного пользователем пункта меню.

Данный демонстрационный вариант программы выводит пять записей и при нажатии клавиши «ВВОД» показывает цифру, соответствующую номеру выбранной строки меню:

```
1000 I%=1%
1005 LINE (45,70)-(140,140),3,B
1010 ? CHR$(156)
1020 ON I% GOSUB 1300,1400,1500,1600,1700
1030 ? CHR$(156)
1040 FOR I5%=1% TO 4%
1050 ON I5% GOSUB 1400,1500,1600,1700
1055 NEXT I5%
1140 I1%=I%
1150 MA$=INKEY$
1160 IF MA$="" THEN 1150
1170 IF ASC(MA$)=26 TH I%=I%-1%
1175 IF ASC(MA$)=10 TH GOSUB 2000
1180 IF ASC(MA$)=27 TH I%=I%+1%
1190 IF I%>5 TH I%=1%
1195 IF I%<=0% TH I%=5%
1200 ON I1% GOSUB 1300,1400,1500,1600,1700
1210 ? CHR$(156)
1220 ON I% GOSUB 1300,1400,1500,1600,1700
1230 ? CHR$(156)
1240 GOTO 1140
1300 LOCATE 8,8
1310 ? "SAVE"
1320 RETURN
1400 LOCATE 8,8+1
1410 ? "LOAD"
1420 RETURN
1500 LOCATE 8,8+2
1510 ? "EDIT"
1520 RETURN
1600 LOCATE 8,8+3
1610 ? "REN "
1620 RETURN
1700 LOCATE 8,8+4
1710 ? "EXIT"
1720 RETURN
2000 ON I% GOSUB 2100,2110,2120,2130,2140
2010 RETURN
2100 LOCATE 15,8
2101 ? "1"
2102 RETURN
2110 LOCATE 15,8
2111 ? "2"
2112 RETURN
2120 LOCATE 15,8
2121 ? "3"
2122 RETURN
2130 LOCATE 15,8
2131 ? "4"
2132 RETURN
2140 LOCATE 15,8
2141 ? "5"
2142 RETURN
```

Примечание редактора

Н.Ю. Щербина предлагает использовать для выделения текущей позиции меню инверсию соответствующих байтов экранного ОЗУ с помощью команды ассемблера COMB, причем, как могут заметить внимательные читатели, в предложенной им подпрограмме используется по сути тот же алгоритм, что и при работе со спрайтами. Идея здесь предельно проста: на экран выводится любой текст, выглядящий как ряд пунктов меню, а затем отдельно подсвечивается нужный его участок. При этом, во-первых, не требуется переписывать заново сам текст меню, что при использовании инверсии символов замедляет работу. Во-вторых, можно вместо текстовых надписей использовать в качестве пунктов меню графические картинки-пиктограммы. И в-третьих, данный способ применим в более сложных случаях, например для организации входящих сегодня «в моду» ключевых слов, когда в каком-либо тексте (скажем, в подсказке) на отдельные слова, выделенные другим цветом, можно навести подсветку и по нажатию клавиши «ВВОД» получить разъяснение данного слова.

Однако рассматриваемый вариант имеет и недостатки. Во-первых, приходится вести пересчет текстовой позиции нужной строки на экране в значение адреса в экранном ОЗУ (а если при этом нужно учитывать рулонное смещение, то работа еще больше усложняется). И во-вторых, если выделять таким способом области большого размера, появляется довольно неприятное «мерцание» подсветки, вызванное немгновенным инвертированием выбранного участка.

Программа, предложенная В.В. Руденко, проще и реализована исключительно на БЕЙСИКе, чему будут благодарны многие читатели, не знакомые пока с программированием на ассемблере. Программа позволяет организовать на экране вертикальное меню из пяти пунктов, она хорошо документирована автором, так что практически для любого знающего БЕЙСИК пользователя не составит особого труда приспособить ее для своих нужд или создать по тому же принципу горизонтальное меню. Однако написана она крайне неоптимально. Поэтому хотелось бы предложить свой вариант программы для тех же целей. В ней используется несколько отличающийся от предыдущего варианта алгоритм, что позволило значительно сократить ее объем. Кроме того, предусмотрена возможность работы в цветном режиме.

Программа содержит в себе две подпрограммы, служащие для вывода вертикального меню на экран и управления им. Они полностью универсальны и могут быть использованы в любой БЕЙСИК-программе, в том числе и для реализации вложенных меню (путем многократного вызова). Строки с 10 по 260 служат демонстрацией правил использования подпрограмм.

Исходные данные для вывода меню содержатся в следующих переменных:

MU$() — массив строк-пунктов меню;

KS% — общее количество пунктов меню;

C1%, C2% и C3% — цвета текста, фона и подсветки меню соответственно;

XM% и YM% — координаты текстовой позиции левого верхнего угла меню;

PZ% — возвращаемый в основную программу номер выбранной позиции меню.

Значения переменных MU$(), KS%, C1%, C2%, C3%, XM% и YM% необходимо задать перед вызовом подпрограмм, а значение PZ% подпрограмма возвращает после нажатия пользователем клавиши «ВВОД».

Строки, заносимые в массив MU$, желательно выровнять по длине за счет добавления пробелов, чтобы улучшить вид меню.

Цвета, задаваемые в C1%, C2% и C3%, кодируются следующим образом: 0% — черный, &O177777 (–1%) — красный, &O125252 — зеленый, &O52525 — синий. В приведенной программе заданы красный цвет текста, зеленый фон и синий курсор подсветки.

```
 10 CLS
 20 DIM MU$(5)
 30 FOR I%=1 TO 5
 40 READ MU$(I%)
 50 NEXT I%
 60 DATA LOAD,SAVE,EDIT,"REN ",EXIT
 70 READ KS%,C1%,C2%,C3%,XM%,YM%
 80 DATA 5,&O177777,&O125252,
    &O52525,10,10
 85 GOSUB 200
 90 GOSUB 1100
100 ? AT(0,0);"Выбрана строка: ";PZ%;
110 GOTO 90
200 C4%=PEEK(&O212)
205 POKE &O212,&O125252
210 FOR I%=XM%-1 TO XM%+4
220 FOR J%=YM%-1 TO YM%+5
230 ? AT(I%,J%);" ";
240 NEXT J%,I%
250 POKE &O212,C4%
260 RETURN

1000 C4%=PEEK(&O212) ' вывод меню на экран
1010 C5%=PEEK(&O214)
1020 POKE &O214,C1%
1030 FOR I%=1 TO KS%
1040 IF I%=PZ% THEN POKE &O212,C3%
     ELSE POKE &O212,C2%
1050 ? AT(XM%,YM%+I%-1);MU$(I%)
1060 NEXT I%
1070 POKE &O212,C4%
1080 POKE &O214,C5%
1090 RETURN

1100 PZ%=1 ' управление меню
1110 GOSUB 1000
1120 CH$=INKEY$
1130 IF CH$="" GOTO 1120
1140 CH%=ASC(CH$)
1150 IF CH%=10% THEN RETURN
1160 IF CH%=26% THEN PZ%=PZ%-1%
1170 IF CH%=27% THEN PZ%=PZ%+1%
1180 IF PZ%<=0 THEN PZ%=KS%
1190 IF PZ%>KS% THEN PZ%=1
1200 GOTO 1110
```

Дадим некоторые пояснения к работе подпрограмм. Основной является подпрограмма GOSUB 1100:

строка 1100 — исходное положение указателя меню – на первом пункте;

1110 — вывод текста меню на экран;

1120-1140 — запрос команды пользователя (в CH% находится код нажатой клавиши);
1150-1170 — действия по нажатию клавиш «ВВОД», «СТРЕЛКА ВВЕРХ» и «СТРЕЛКА ВНИЗ» соответственно;
1180-1190 — «закольцевание» меню.

(При повторном вызове подпрограммы со строки 1110 положение указателя меню определяется значением PZ%, заданным в основной программе, либо, если оно не изменялось, подсветка указывает на пункт, выбанный при предыдущем вызове меню).

Подпрограмма GOSUB 1000 предназначена для вывода текста меню вместе с курсором подсветки на экран:

1000-1010 — сохранение в буферных переменных текущего цвета текста и фона;
1020 — установить заданный цвет текста;
1030-1060 — вывод строк-пунктов меню в цикле одного под другим, причем при совпадении I% и PZ% текст печатается на одном фоне, а при несовпадении — на другом (таким образом имитируется выделение текста курсором-указателем);
1070-1080 — восстановление исходных цветов текста и фона.

(Примечание: подпрограмма в строках 200—260 относится к демонстрационной части программы и служит для создания «окна» для вывода меню, чтобы улучшить его внешний вид.)

Среди периферийных устройств, которые можно подключить к БК, числится и мышь, носящая «космическое» имя «Марсианка». Работать с ней весьма удобно, особенно в среде графических редакторов (например, с GRAF и БК-PAINT).
К сожалению, разработчики мыши не потрудились не только создать для нее хотя бы простейший драйвер, но и достаточно подробно описать требуемый алгоритм его работы (хорошо хоть с подключением к разъему БК-0010(.01) сложностей нет – существует единственный, «заводской», стандарт распайки,– не то что для джойстиков!). А реализовать в своей программе «мышиный» интерфейс ой как хочется!

Д. Ю. Усенков,
г. Москва

МЫШКА ДЛЯ БКшки

Оказывается, это не так уж сложно. Предлагаемые библиотечные подпрограммы (ассемблер стандарта М18) позволяют сделать это с минимальными трудностями. Данные листинги лучше всего оттранслировать в объектный код и затем подгружать при компиляции ваших программ командой LI.

Функции низкого уровня

Этот набор простейших функций, занимая небольшой объем, позволяет тем не менее использовать мышь достаточно эффективно.

Чтение состояния мыши (INMOUS)

Производит считывание пересылаемой мышью на порт ввода-вывода БК информации в регистр R0, после чего инициализирует порт. Эта подпрограмма предназначена не столько для вызова пользователем, сколько для обслуживания других подпрограмм «мышиной» библиотеки.

Проверка кнопок (MOUKEY)

Проверяет, нажата ли в данный момент хотя бы одна их кнопок мыши. В регистр R0 возвращается 0, если кнопки не нажаты, или ненулевое число — в противном случае.

Прямой доступ к мыши (MOUSMV)

Преобразует слово состояния мыши в массив байтов, избавляя таким образом пользователя от необходимости перебирать биты по маске с помощью оператора BIT. Доступ возможен как по метке начала массива (MOUSIS) и номеру элемента, так и по метке каждого отдельного байта:

MOUSUP – движение мыши вверх;
MOUSRT – движение вправо;
MOUSDN – движение вниз;
MOUSLT – движение влево;
MOUSKL – нажатие левой клавиши;
MOUSKR - нажатие правой клавиши.

Факт движения или нажатия отображается числом −1 (1777778) в соответствующем байте, в противном случае его значение нулевое. Следует отметить, что ненулевыми могут быть сразу несколько элементов массива, например при движении мыши влево вверх с двумя одновременно нажатыми кнопками минус единице равны MOUSUP, MOUSLT, MOUSKL и MOUSKR.

Эмуляция клавиатуры (MOUSYM)

Весьма удобным может оказаться дублирование клавиатуры с помощью мыши. Данная подпрограмма возвращает в регистре R0 код клавиши, соответствующей движению мыши (в том числе диагональных стрелок). Если же мышь неподвижна, R0 равно нулю. Дополнительно в R1 возвращается код, соответствующий кнопкам мыши: правая – «ПРОБЕЛ», левая – «ВВОД», обе одновременно – «КТ».

Подпрограмма эмуляции клавиатуры позволяет реализовать простейший способ «подключения» мыши к программе на ассемблере. Для этого достаточно после опроса клавиатуры (убедившись, что на ней ни одна клавиша не нажата), вызвать данную подпрограмму и, если R0 не равно 0, передать его значение на дальнейшую обработку, как если бы была нажата соответствующая клавиша. Аналогично можно обрабатывать и кнопки мыши (регистр R1).

Переназначение клавиш (MOUSTD)

На IBM-совместимых компьютерах часто используют в качестве клавиши «ВВОД» правую кнопку. Сторонники этого стандарта могут, один раз вызвав эту подпрограмму, симметрично поменять назначение кнопок мыши в предыдущей функции MOUSYM. Повторный вызов возвращает прежние назначения.

Листинг:

```
MOUSE.SYS
Библиотека функций «низкого уровня»
;для управления мышью
;-----------------------------------------
INMOUS:     MOV #177714,R0
; Читает слово из порта в R0
            CLR @#177714
            MOV #10,@#177714
            RTS R7
;-----------------------------------------
MOUKEY:     JSR R7,INMOUS
; Проверяет, была ли нажата хоть одна
; из кнопок мыши:
; нет - R0 равно 0
; да - R0 не равно 0
            BIT #40,R0
            BNE 1
            BIT #100,R0
            BNE 1
            CLR R0
            RTS R7
1:          MOV #177777,R0
            RTS R7
;-----------------------------------------
MOUSMV:     JSR R7,INMOUS
; Расширяет 7 младших битов порта до
; 7-байтного массива. Ячейки массива
; содержат 0 или -1, если соответствующие
; биты равны 0 или 1
            MOV R1,-(R6)
            JSR R7,%A61
            .@MOUSIS
            MOV R2,-(R6)
            MOV #7,R2
1:          ROR R0
            BCS 2
            CLRB (R1)+
            BR 3
2:          MOVB #377,(R1)+
3:          SOB R2,1
            MOV (R6)+,R2
            JSR R7,%A61
            .@MOUSIS
            MOV R1,R0
            MOV (R6)+,R1
            RTS R7
;-----------------------------------------
MOUSIS:     MOUSUP:  .B:0  ; мышь вверх
            MOUSRT:  .B:0  ; мышь вправо
            MOUSDN:  .B:0  ; мышь вниз
            MOUSLT:  .B:0  ; мышь влево
                     .B:0  ; р е з е р в
            MOUSKL:  .B:0  ; левая кнопка
            MOUSKR:  .B:0  ; правая кнопка
                     .E
;-----------------------------------------
```

```
            BEQ 2
            INC R1
            SOB R2,1
            CLR R0
            BR 3
2:          MOVB (R1)+,R0
3:          BIC #177637,@R6
            MOV #3,R2
4:          CMPB @R6,(R1)+
            BEQ 5
            INC R1
            SOB R2,4
            CLR R1
            BR 6
5:          MOVB (R1),R1
6:          MOV (R6)+,R2
            MOV (R6)+,R2
            RTS R7
;-------------------------------------------

%TABL3:     .B:1.B:32    ; UP
            .B:2.B:31    ; RIGHT
            .B:4.B:33    ; DOUN
            .B:10.B:10   ; LEFT
            .B:3.B:35    ; UP-RIGHT
            .B:11.B:34   ; UP-LEFT
            .B:3.B:36    ;DOUN-RIGHT
            .B:14.B:37   ; DOUN-LEFT
            .B:100.B:40  ; RIGHT KEY (пробел)
            .B:40.B:12   ; LEFT KEY (ввод)
            .B:140.B:3   ; LEFT+RIGHT KEYS (КТ)
;-------------------------------------------
MOUSTD:     MOV R1,-(R6)
; Симметрично меняет назначение кнопок
; мыши в предыдущей функции (правая - ввод,
; левая - пробел, обе одновременно - КТ)
; Повторное использование возвращает
; вас к прежнему стандарту
            JSR R7,%A61
            .@%TABL3+20
            MOVB @R1,-(R6)
            MOVB 2(R1),@R1
            MOVB (R6)+,2(R1)
            MOV (R6)+,R1
            RTS R7
;-------------------------------------------
%A61:       MOV @R6,R1
; Запись адреса перемещаемой метки в R1
            ADD @R1,R1
            ADD #2,@R6
            RTS R7
```

Функции высокого уровня

Этот набор подпрограмм позволяет совместно с библиотекой низкого уровня реализовать высокоразвитый интерфейс работы с мышью. Для их использования необходим режим 32 символа в строке и предварительная нормализация рулонного смещения экрана двукратной подачей через EMT 16 кода 214_8 (или записью в регистр смещения @#177664 числа 1330_8). Кроме того, нужно обязательно прилинковать к создаваемой программе рассмотренную ранее библиотеку функций низкого уровня. Следует заметить, что работа с кнопками мыши здесь не реализована, для этого нужно использовать подпрограммы библиотеки низкого уровня.

Инициализация интерфейса (MOUINI)

Эта подпрограмма обязательно должна быть вызвана в начале работы с мышью. Она гасит курсор, одновременно устанавливая для него стандартную конфигурацию, и обнуляет текущие координаты мыши.

Установка/сброс курсора мыши (MOUCUR)

В зависимости от переданного в регистре R0 значения выключает (если R0=0) или включает курсор мыши. Одновременно устанавливается слово статуса курсора, доступное пользователю по метке MCURST. (ВНИМАНИЕ! Не пытайтесь изменять значение статуса, записывая в ячейку MCURST другое число. Это может привести к сбою функционирования программы.)

Установка/съем координат курсора мыши (MOUSXY)

Если переданное подпрограмме значение R0 равно нулю, то в паре регистров (R1,R2) возвращаются текущие координаты курсора мыши. Иначе курсор переносится в точку, координаты которой заданы в (R1,R2).

Съем текстовой позиции (MOUSTX)

Текущие координаты курсора мыши (в пикселах) пересчитываются в координаты соответствующей текстовой позиции (R1,R2). Эта подпрограмма удобна, когда нужно определить, например, на какой пункт меню указывает курсор мыши.

«Мышь в прямоугольной области» (MOUINB)

Проверяет, находится ли курсор мыши в заданной прямоугольной области (пары регистров (R1,R2) и (R3,R4) — координаты верхнего левого и нижнего правого углов, соответственно). В R0 возвращается 0, если курсор мыши вне области, и −1, если внутри нее. В отличие от предыдущей, эта подпрограмма удобна для реализации пиктографического меню.

Обновление координат (MOUSFR)

Эта подпрограмма без параметров, обращаясь к порту, определяет текущее состояние мыши и перемещает курсор. Таким образом, достаточно через небольшие промежутки времени в цикле вызывать «между делом» эту подпрограмму, чтобы полностью «автоматизировать» движение мышиного курсора.

(ПРИМЕЧАНИЕ. Пользователи, работающие с БК-0011(М), могут модернизировать предложенный листинг, вызывая данную подпрограмму через прерывание таймера. Это позволит полностью автоматизировать (уже без кавычек!) работу с мышью. Что же касается БК-0010(.01), желающие могут попробовать реализовать то же самое по методу «параллельного исполнения программ» (см. «Информатика и образование» №2 за 1992 г.)

Шаг перемещения курсора мыши (MOUSTP)

В данной версии библиотеки шаг перемещения курсора предполагается неизменяемым (равен 4 пикселам). Именно это значение подпрограмма возвращает в R0.

Вывод курсора в стандарте «Microsoft» (CUMSET)

Эта служебная подпрограмма производит вывод на экран курсора в виде стрелки, почти такой же, к какой привыкли пользователи IBM. В данной версии используется быстрый спрайтовый вывод, который, однако, ограничивает шаг курсора значением 4. (У автора имеется другая версия, в которой используется более медленный режим ГРАФ, зато реализована возможность менять шаг с помощью MOUSTP. Желающие могут попытаться модифицировать подпрограмму вывода спрайта для обеспечения возможности задания произвольного шага перемещения курсора.)

Альтернативный вид курсора (%AMOCR)

Позволяет задать другую форму курсора мыши (например, в виде «указующего перста»), для чего нужно передать в R0 адрес байтового массива с его изображением (пример записи для стандартного курсора см. в ассемблерном листинге). Значение R0=0 вновь устанавливает стандартный курсор. (ПРИМЕЧАНИЕ. Желательно погасить курсор мыши на время смены его вида.)

Гашение курсора мыши (CUMRST)

Эта служебная подпрограмма гасит курсор путем переписывания на его место ранее сохраненного «фонового» изображения. Следует отметить, что по этой причине при любом изменении содержимого экрана (запись байтов в экранное ОЗУ, рисование и вывод текста) необходимо гасить курсор с помощью MOUCUR.

Листинг:

```
; MOUSE2.LIB (спрайтовый, быстрый вариант)
; Библиотека функций высокого уровня для
; работы с мышью
;-----------------------------------------
; Обязательно наличие режима 32 символа
; в строке и отсутствие рулонного смещения
; (начало экрана = 40000)
;
; При использовании обязательно наличие
; библиотеки функций мыши
; низкого уровня (MOUSE.SYS)
;-----------------------------------------
MOUINI:     MOV R1,-(R6)
; Инициализация мыши (гасит курсор,
; устанавливает координаты мыши в 0 и
; стандартный курсор)
            CLR MCURST
            CLR %MOUSX
            CLR %MOUSY
            JSR R7,%A61
            .@%A65
            MOV R1,AMOUCR
            MOV (R6)+,R1
            RTS R7
;-----------------------------------------
MOUCUR:     TST R0
; Установка/сброс индикации курсора
; мыши и установка ячейки статуса
; R0=0 - сброс курсора;
; R0<>0 - установка курсора
            BEQ 1
            TST MCURST
            BNE 2
            JSR R7,CUMSET
2:          MOV #177777,MCURST
            RTS R7
1:          TST MCURST
            BEQ 3
            JSR R7,CUMRST
3:          CLR MCURST
            RTS R7
;
MCURST:     .#0
; Слово статуса курсора (0 - погашен,
; -1 - включен)
;-----------------------------------------
MOUSXY:     TST R0
; Установка/съем координат курсора мыши:
; R0=0 - координаты возвращаются в
```

```
; паре регистров (R1=X,R2=Y)
; R0<>0 - установка новых координат
; по значениям (R1,R2) с переносом
; курсора, если он включен
                BNE 1
                MOV %MOUSX,R1
                MOV %MOUSY,R2
                RTS R7
1:              CMP R1,#370
                BLOS 123
                MOV #370,R1
123:            BIC #3,R1
                CMP R2,#356
                BLOS 1234
                MOV #356,R2
1234:           TST MCURST
                BNE 2
                MOV R1,%MOUSX
                MOV R2,%MOUSY
                RTS R7
2:              CLR R0
                JSR R7,MOUCUR
                MOV R1,%MOUSX
                MOV R2,%MOUSY
                INC R0
                JSR R7,MOUCUR
                RTS R7
;
%MOUSX:         .#0
%MOUSY:         .#0
; Ячейки координат мыши, к ним
; можно обращаться непосредственно
;-----------------------------------------
MOUSTX:         MOV %MOUSX,R1
; Возвращает в паре (R1,R2) координаты
; текущей текстовой позиции
                CLC
                ROR R1
                ASR R1
                ASR R1
                MOV %MOUSY,R2
                MOV R0,-(R6)
                CLR R0
2:              SUB #12,R2
                BMI 1
                INC R0
                BR 2
1:              MOV R0,R2
                MOV (R6)+,R0
                RTS R7
;-----------------------------------------
MOUINB:         CMP %MOUSY,R2
; Проверяет координаты курсора мыши
; на нахождение внутри прямоугольной
; зоны, заданной парами регистров
; (R1,R2) и (R3,R4) (левый верхний и
;  правый нижний углы) и
; возвращает результат в R0:
; 0 - вне зоны; -1 - внутри зоны
                BLO 1
                CMP %MOUSY,R4
                BHI 1
                CMP %MOUSX,R1
                BLO 1
                CMP %MOUSX,R3
                BHI 1
                MOV #177777,R0
                RTS R7
1:              CLR R0
                RTS R7
;-----------------------------------------
MOUSFR:         MOV R0,-(R6)
; Обновление координат курсора мыши
; в соответствии с ее манипуляциями.
; Перемещение курсора, если он включен.
; Функция без параметров
                MOV R5,-(R6)
                JSR R7,INMOUS
                MOV R0,R5
                BEQ 6
                CLR -(R6)
                TST MCURST
                BEQ 1
                INC @R6
                CLR R0
                JSR R7,MOUCUR
1:              MOV R5,R0
                BIT #1,R0
                BEQ 2
                SUB #4,%MOUSY
; шаг жестко задан =4 и неизменяем
                BGE 2
                CLR %MOUSY
2:              BIT #2,R0
                BEQ 3
                ADD #4,%MOUSX
```

```
                CMP %MOUSX,#370
                BLOS 3
                MOV #370,%MOUSX
3:              BIT #4,R0
                BEQ 4
                ADD #4,%MOUSY
                CMP %MOUSY,#356
                BLOS 4
                MOV #356,%MOUSY
4:              BIT #10,R0
                BEQ 5
                SUB #4,%MOUSX
                BGE 5
                CLR %MOUSX
5:              TST (R6)+
                BEQ 6
                CLR R0
                INC R0
                JSR R7,MOUCUR
6:              MOV (R6)+,R5
                MOV (R6)+,R0
                RTS R7
;-----------------------------------------
MOUSTP:         TST R0
; Оставлено для совместимости с версией
; MOUSE1.LIB, но шаг не меняет,
; а возвращает всегда 4
                BEQ 1
                RTS R7
1:              MOV #4,R0
                RTS R7
;-----------------------------------------
CUMSET:         MOV R5,-(R6)
; Служебная подпрограмма вывода курсора
; мыши в стандарте Microsoft
;  по текущим координатам с сохранением фона
                MOV R4,-(R6)
                JSR R4,@#110346
                JSR R7,%A61
                .@%A64
                MOV R1,R0
                MOV %MOUSX,R1
                CLC
                ROR R1
                ASR R1
                MOV #42000,R3
                ADD R1,R3
                MOV %MOUSY,R2
                MOV #6,R1
1:              ASL R2
                SOB R1,1
                ADD R2,R3
                MOV R3,-(R6)
                MOV #2,R1
                MOV #20,R2
; 2 байта * 20 строк
2:              MOV R1,R4
3:              MOVB (R3)+,(R0)+
                SOB R4,3
                ADD #100,R3
                SUB R1,R3
                SOB R2,2
                MOV AMOUCR,R0
; адрес спрайта мыши
                MOV @R6,R3
                MOV #2,R1
                MOV #20,R2
22:             MOV R1,R4
23:             TST R3
                BMI 24
                BICB (R0)+,(R3)+
                SOB R4,23
                ADD #100,R3
                SUB R1,R3
                SOB R2,22
24:             MOV AMOUCR,R0
                ADD #40,R0
                MOV (R6)+,R3
                MOV #2,R1
                MOV #20,R2
32:             MOV R1,R4
33:             TST R3
                BMI 34
                BISB (R0)+,(R3)+
                SOB R4,33
                ADD #100,R3
                SUB R1,R3
                SOB R2,32
34:             JSR R4,@#110362
                MOV (R6)+,R4
                MOV (R6)+,R5
                RTS R7
;-----------------------------------------
AMOUCR:         .#0     ; адрес спрайта-курсора
%A64:           .+50    ; буфер для сохранения фона
%A65:           .#77    ; прорисовка курсора:
```

```
                .#377
; 20 по 2 байта - тень от курсора (негатив)
                .#1777
                .#7777
                .#37777
                .#177777
                .#177777
                .#177777
                .#177777
                .#37777
                .#37777
                .#177477
                .#177400
                .#177600
                .#177600
                .#0
                .#0
; 20 по 2 байта - курсор (позитив)
                .#6
                .#36
                .#176
                .#776
                .#3776
                .#17776
                .#77776
                .#3776
                .#3636
                .#17006
                .#17000
                .#74000
                .#74000
                .#0
                .#0
;-----------------------------------------
%AMOCR:     TST R0
; Установка альтернативного вида курсора.
; В R0 - адрес спрайтового массива
; в данном формате.
; R0=0 - стандартный курсор
            BEQ 1
2:          MOV R0,AMOUCR
            RTS R7
1:          MOV R1,-(R6)
            JSR R7,%A61
            .@%A65
            MOV R1,R0
            BR 2
;-----------------------------------------
CUMRST:     MOV R5,-(R6)
; Служебная подпрограмма стирания
; курсора мыши путем переписывания
; ранее сохраненного в буфере фона
; в область с текущими координатами
            MOV R4,-(R6)
            JSR R4,@#110346
            JSR R7,%A61
            .@%A64
            MOV R1,R0
            MOV %MOUSX,R1
            CLC
            ROR R1
            ASR R1
            MOV #42000,R3
            ADD R1,R3
            MOV %MOUSY,R2
            MOV #6,R1
1:          ASL R2
            SOB R1,1
            ADD R2,R3
            MOV #2,R1
            MOV #20,R2
2:          MOV R1,R4
3:          TST R3
            BMI 4
            MOVB (R0)+,(R3)+
            SOB R4,3
            ADD #100,R3
            SUB R1,R3
            SOB R2,2
4:          JSR R4,@#110362
            MOV (R6)+,R4
            MOV (R6)+,R5
            RTS R7
;-----------------------------------------
; ВНИМАНИЕ! При автономном использовании
; подпрограммы CUMRES нужно внимательно
; следить за соответствием координат восстанов-
; ления фона ранее использованным координа-
; там при выводе курсора.
;-----------------------------------------
; При использовании прямой записи байт в
; экранное ОЗУ, вывода символов и прочем
; изменении изображения на экране желательно
; выключать курсор мыши на время изменения
; и затем включать его вновь.
;
; Библиотека MOUSEV2 несовместима с альтер-
; нативной версией MOUSEV.
```

ОБМЕН ОПЫТОМ

А. В. Балло

В СТРОКЕ 80 СИМВОЛОВ

Может быть, кого-либо заинтересует приведенная здесь простая программа, позволяющая получать дамп оперативной памяти БК в формате, похожем на принятый в ОС RT11 и ее аналогах, за исключением распечатки в коде RADIX50. Дамп может быть выведен как на экран, так и на принтер с интерфейсом ИРПР (например, МС6302), причем формат вывода (80 символов в строке) на экране и на бумаге совершенно идентичен. Подпрограмма вывода символов на экран в 80 (до 84) столбцов расположена с адреса <416+адрес загрузки> и может быть использована для построения различного рода редакторов и СУБД, в которых для совместимости с другими ЭВМ должно быть 80 символов в строке.

Программа является перемещаемой, запускается она с адреса загрузки, ее длина 1120. После запуска запрашивается адрес начала дампа, который следует набрать без исправлений (а если допущена ошибка, нужно нажать «ПРОБЕЛ» и ввести адрес заново) и закончить клавишей «ВВОД». После этого на запрос распечатки на принтер следует нажать «ВВОД», если ДА, или любую другую клавишу, если НЕТ. После вывода 25 строк дампа программа ждет нажатия клавиши. Если необходимо изменить адрес, нажмите «КТ». Для продолжения листания дампа — любую другую. Последняя строка текущего экрана повторится как первая на следующем (соответственно, на принтере в момент смены экрана будут отпечатаны две одинаковые строки — *Прим. ред.*)

Листинг программы дампа ОЗУ в машинных кодах:

нач. адрес: 001000 длина: 001120

010701	062701	000012	005002	104020	000421	106214	062101
071144	071545	035163	052000	020157	071160	067151	024164
062531	026563	042474	052116	051105	024476	000077	004737
100472	005002	104020	104006	020027	000012	001004	012767
010046	000752	000403	012767	000207	000742	012702	000031
012700	000014	004767	000266	010504	010501	004767	000162
012700	000057	004767	000246	012703	000010	012401	004767
000150	012700	000040	004767	000224	077310	012700	000052
004767	000212	010504	012703	000020	112400	120027	000040
103406	200271	000200	103405	120027	000240	103002	012700
000056	004767	000150	077317	020227	000001	001404	012700
000052	004767	000130	062705	000020	012700	000012	004767
000554	077266	104006	020027	000003	001632	162705	000020
000675	010146	010246	006301	000411	010146	010246	005000
006301	006100	062700	000060	004767	000042	012702	000005
005000	006301	006100	006301	006100	006301	006100	062700
000060	004767	000010	077214	012602	012601	000207	004437
110346	010546	004767	000426	004767	000010	012605	004437
110362	000207	042700	177400	020027	000012	001541	020027

000014	001561	020027	000040	103407	020027	000200	103407
020027	000240	103401	000406	012700	000020	000405	162700
000020	000402	162700	000060	006300	010001	006300	006300
060001	062701	112036	016705	000342	016704	000340	006304
012702	000012	112100	010003	042703	177677	006203	050300
042700	177700	060407	000403	000404	000415	000432	150015
000434	010003	000303	006203	006203	150315	006200	006200
150065	000001	000422	010003	006303	006303	006303	006303
150365	000001	006200	006200	006200	006200	150065	000002
000404	006300	006300	150065	000002	062705	000100	077256
026727	000172	000003	103434	005067	000162	062767	000003
000152	016700	000146	042700	177700	020027	000072	103422
042767	000077	000126	062767	001200	000120	026727	000114
077200	103410	012767	040000	000102	005067	000100	000403
005267	000072	000207	012701	040000	012702	020000	005021
077202	012767	040000	000044	005067	000042	000207	010046
005737	177714	001775	052700	000400	010037	177714	005737
177714	001375	005037	177714	012600	000207	057044	000003

Контрольная сумма: 033561

Примечание редактора

Предложенная вниманию читателей программа проверена на БК-0010.01. Из недостатков можно указать разве только на отмеченный в тексте статьи факт повтора последних строк каждого экрана на принтере.

Кроме того, проверка работоспособности программы производилась только в режиме вывода на экран (по причине отсутствия в редакции принтера МС6302). Взамен можно предложить читателям ассемблерную запись содержащейся в программе дампа ОЗУ подпрограммы печати на принтер. Пользователи, имеющие опыт работы с принтером и знающие ассемблер, смогут сами решить, подходит ли имеющийся у них принтер для этой программы, либо адаптировать ее под свой принтер. Подпрограмма расположена, начиная с адреса <1056+начальный адрес>. (Судя по приведенному листингу, подпрограмма печати соответствует стандарту для принтеров типа МС6312, но, возможно, неработоспособна с принтерами типа МС6313, требующими инвертирования информационного байта.)

Сама по себе программа дампа оперативной памяти, возможно, мало кого заинтересует — то же самое можно сделать с помощью отладчиков типа DEBU10 (которые, кроме кодов и текста, выводят еще коды RADIX и ассемблерные мнемоники — именно с помощью DEBU10 были получены не предоставленные автором ассемблерные листинги). Однако весьма интересной является подпрограмма вывода символов по 80 знаков в каждой строке экрана. Фактически она представляет собой альтернативу стандартной функции EMT 16 и построена по весьма любопытному алгоритму. Конечно, плотно «стиснутый» символ к символу текст на экране читается хуже, чем обычный, но сама по себе идея вывода по 80 символов в строке наверняка найдет горячих поклонников.

```
; Подпрограмма печати на принтер
2056:   MOV  R0,-(R6)
2060:   TST  @#177714
        BEQ  2060
        BIS  #400,R0
        MOV  R0,@#177714
2076:   TST  @#177714
        BNE  2076
        CLR  @#177714
        MOV  (R6)+,R0
        RTS  R7
```

К сожалению, предложенный автором кодовый дамп весьма трудно читать и переводить. Поэтому приведем свой ассемблерный листинг данной подпрограммы, являющийся некоторой незначительной модификацией предложенного автором, который можно будет использовать в других программах. Имеющиеся в

нем комментарии помогут вам детально разобраться в сути алгоритма. Но некоторые вещи требуют, на наш взляд, дополнительных разъяснений.

Подпрограмма OUTCHR обеспечивает обработку переданного ей в младшем байте регистра R0 кода, извлекает из прошитого в ПЗУ монитора шрифта нужное изображение (непечатаемые коды выводятся пробелами) и выводит его на экран. При этом рулонное смещение не учитывается (!), поэтому перед использованием данной подпрограммы его нужно нормализовать (например, двухкратной подачей через EMT 16 кода 214_8) и при работе избегать прокручивания экрана.

Так как при предложенном способе вывода на экран каждый символ должен занимать по ширине менее одного байта, часть символов должна быть разделена между двумя экранными байтами. Соответствующий участок подпрограммы (с весьма нестандартным использованием команд ADD ...,R7 и списка BR в качестве аналога БЕЙСИКовского ON ... GOTO...) обеспечивает разделение изображения символа на части и вывод его в соседние байты.

Две ячейки ОЗУ, задействованные в программе, хранят: ячейка ADRESS — текущий адрес (байт, соответствующий верхнему левому углу очередного знакоместа) в экранном ОЗУ, ячейка TMPRL — некоторую рабочую информацию (номер выбираемого из списка оператора BR в соответствии с одной из четырех возможных ситуаций при выводе символов).

Немного, пожалуй, стоит поговорить и о стандарте записи знакогенератора в БК. Изображения всех символов в виде спрайтов размером 1 байт × 10 графических строк (всего 12 байт в восьмеричном счислении) хранятся в ПЗУ БК, начиная с адреса #112036. Так как последовательность кодов в БК дважды прерывается непечатаемыми (от 0 до 37 и от 200 до 237, все коды восьмеричные), удобно разделить все печатаемые символы на две группы(два банка). Первый банк символов содержит знаки, цифры и латинские буквы. Второй — полуграфику и кириллицу. В ПЗУ же, начиная с адреса #112036, в начале записаны изображения печатаемых в режиме «БЛОК РЕД» стрелок (коды редактирования и управления курсором), затем следуют спрайты для первого банка символов, а затем без какого-то ни было интервала спрайты второго банка (т.е. сразу же за символом «ЗАБОЙ»-«квадратик» следует «Пи»). Поэтому, как и сделано в предложенной подпрограмме, переданный код символа нужно вначале преобразовать. «ПРОБЕЛ» в ПЗУ знакогенератора стоит на 20 месте от начала списка, поэтому непечатаемые коды заменяем байтом 20. Если переданный код символа находится в интервале от 40 до 177 (первый банк символов), вычитаем из него 20, тогда бывший код 40 будет в ПЗУ соответствовать пробелу. Следовательно, для кодов от 240 до 377 мы должны вычесть 60, чтобы «ликвидировать разрыв» между спрайтами первого и второго банка символов (разрыв в 40 кодов плюс 20 в начале). Теперь полученный из кода порядковый номер изображения нужно умножить на 10 и прибавить к нему адрес #112036, получив адрес спрайта символа. Приведенный алгоритм может понадобиться и в других случаях, например, для получения увеличенных букв без предварительного вывода стандартного символа через EMT 16 в буферную область экрана (так сделано, например, в БК-PAINT).

Ассемблерный листинг подпрограммы вывода 80 символов в строке:

```
;       Подпрограмма вывода символа на экран с плотностью 80
;       символов в строке (модификация предложенной А.В. Балло)
;
;       Входные параметры:      R0 - код символа (мл. байт)
;       Выходные параметры:     нет
;       Операционные регистры:  сохраняются в стеке
;
OUTCHR: MOV  R5,-(R6)
        JSR  R4,@#110346
; R0-R4 в стек
        BIC  #177400,R0
; выделить в R0 младший байт
        CMP  R0,#12
; код Перевода строки
        BEQ  A1760
        CMP  R0,#14
; код Перевода формата (СБР)
        BEQ  A2026
        CMP  R0,#40
        BCS  A1510
; код < #40
```

```
                CMP   R0,#200
                BCS   A1516
    ; #40 < код < #200
                CMP R0,#240
                BCS   A1510
    ; #200 < код < #240
                BR    A1524
    ; код > #240
A1510:          MOV   #20,R0
    ; непечатаемый код заменить ПРОБЕЛом
                BR   A1530
A1516:          SUB   #20,R0
    ; первый банк символов
                BR   A1530
A1524:          SUB   #60,R0
    ; второй банк символов
A1530:          ASL   R0
    ; R1=R0*10
                MOV   R0,R1
                ASL   R0
                ASL   R0
                ADD   R0,R1
                ADD   #112036,R1
    ; адрес спрайтов-символов в ПЗУ
                MOV   ADRES,R5
    ; адрес в видеоОЗУ для выводимого символа
                MOV   TMPRL,R4
    ; вспомогательное значение
                ASL   R4
                MOV   #12,R2
    ; в символе 10 графических строк
A1564:          MOVB   (R1)+,R0
    ; очередной байт символа
                MOV   R0,R3
    ; "подтягивание" символов друг к другу
                BIC   #177677,R3
    ; вывод символа ...
                ASR  R3
                BIS   R3,R0
                BIC   #177700,R0
                ADD R4,R7
    ; аналог ON ... GOTO ...
                BR   A1616
                BR   A1622
                BR   A1646
                BR   A1702
A1616:          BISB   R0,@R5
                BR   A1712
A1622:          MOV   R0,R3
                SWAB   R3
                ASR   R3
                ASR   R3
                BISB   R3,@R5
                ASR   R0
                ASR   R0
                BISB   R0,1(R5)
                BR   A1712
A1646:          MOV   R0,R3
                ASL   R3
                ASL   R3
                ASL   R3
                ASL   R3
                BISB   R3,1(R5)
                ASR   R0
                ASR   R0
                ASR   R0
                ASR   R0
                BISB   R0,2(R5)
                BR   A1712
A1702:          ASL   R0
                ASL   R0
                BISB   R0,2(R5)
A1712:          ADD   #100,R5
    ; следующая графическая строка символа
                SOB   R2,A1564
                CMP   TMPRL,#3
                BCS   A2020
                CLR   TMPRL
                ADD   #3,ADRES
    ; переход на следующую текстовую
    ; позицию и проверка, не нужно ли
    ; перейти на следующую строку...
                MOV   ADRES,R0
                BIC   #177700,R0
                CMP   R0,#72
                BCS   A2024
A1760:          BIC   #77,ADRES
    ; переход на следующую строку
    ; текста, в т.ч. и при подаче
    ; кода "Перевод строки"
```

```
                ADD   #1200,ADRES
                CMP   ADRES,#77200
                BCS   A2024
        ; проверка на конец экрана
                MOV   #40000,ADRES
        ; вывод производится поверх старого
        ; с начала экрана
                CLR   TMPRL
                BR    A2026
A2020:          INC   TMPRL
A2024:          BR    QUIT
A2026:          MOV   #40000,R1
        ; очистка экрана по коду СБР
                MOV   #20000,R2
A2036:          CLR   (R1)+
                SOB   R2,A2036
                MOV   #40000,ADRES
                CLR   TMPRL
QUIT:           JSR   4,@#110362
        ; восстановить R0-R4 из стека
                MOV   (R6)+,R5
                RTS   R7
        ; выход из подпрограммы
        ;
ADRES:          .#40000
        ; адрес для вывода символа в экран
TMPRL:          .#0
        ; вспомогательные данные
;-------------------------------------------------------------
;
;       ВНИМАНИЕ! Для правильной работы подпрограммы необходимо
;       предварительно привести рулонное смещение в исходный вид
;       (начало экрана должно совпадать с адресом #40000)
;
```

В. В. Ермаков,

с. Урелики Магаданской обл.

«ГОВОРЯЩАЯ ПРОГРАММА» НА БЕЙСИКЕ БК-0010.01

Предлагаю простую и короткую программу, которая научит ваш компьютер говорить (точнее, повторять слова, произнесенные вами в микрофон). Принцип ее работы можно уподобить магнитофону, где вместо записывающей и воспроизводящей головок – соответственно 5 и 6 разряды регистра управления системного внешнего устройства (адрес 177716_8), а вместо магнитной ленты – последовательные ячейки ОЗУ.

Программа, коды которой записаны в операторах DATA, загружается с адреса 1000_8 (стек БЕЙСИКа). Для работы с ней необходимо подключить к компьютеру магнитофон, включить его на запись, нажав предварительно клавишу «ПАУЗА», установить уровень записи в среднее положение, запустить программу с адреса 1000_8 и произнести в микрофон желаемые слова. Адрес запуска программы для воспроизведения записи — 1052_8. Увеличение задержки (содержимое R5) позволяет записать в память больше слов, жертвуя при этом качеством их воспроизведения.

Нижеприведенная БЕЙСИК-программа является иллюстрацией работы «синтезатора речи», который в данном случае оформлен в виде двух USR-подпрограмм. (Можно, однако, использовать кодовую программу и вне БЕЙСИКа, например в режиме ТС.)

```
10 DATA &O12701,&O20000, &O12704,
   &O20, &O5000, &O12705,&O10,&O241,
   &O32737,&O40,&O177716,&O1401,&O261,
   &O6100, &O77501,&O77413,&O10021,
   &O20127, &O40000,&O2756,&O207
20 DATA &O12701,&O20000, &O12704, &O20,
   &O12100, &O12705,&O10, &O6300,
   &O103004, &O12737,&O100,&O177716,
   &O403,&O12737,&O0,&O177716,&O77501,
   &O77415,&O20127,&O40000,&O2755,
   &O207
30 FOR I%=&O1000 TO &O1124 STEP &O2
40 READ J%
50 POKE I%,J%
```

```
 60 NEXT
 70 DEF USR1=&O1000
 80 DEF USR2=&O1052
 90 CLS
100 PRINT "Если готовы, нажмите любую
    клавишу"
110 I$=INKEY$
120 IF I$="" THEN 110 ELSE PRINT "Гово-
    рите в микрофон"
130 A$=CHR$(&O234)
140 IF INP(&O177716,&O40)=&O40 THEN 140
    ELSE PRINT A$; "Идет запись"; A$
150 A=USR1(A)
160 PRINT "Запись окончена"
170 ' .... строки программы ....
180 IF INKEY$="" GOTO 180 ' ожидание
    нажатия клавиши
190 A=USR2(A) ' вывод речи
200 ' .... строки программы ....
```

Голос записывается в ОЗУ с адреса 20000_8 по 40000_8, поэтому перед запуском БЕЙСИК-программы нужно обязательно освободить место в памяти, выполнив в непосредственном режиме оператор CLEAR,&O20000. Затем программа запускается командой RUN. БК «заговорит» после выполнения функции A=USR2(A).

А. В. Домолазов,
г. Казань

ПОДПРОГРАММА РАСПОЗНАВАНИЯ СИМВОЛОВ НА ЭКРАНЕ

В некоторых версиях БЕЙСИКа существует оператор, который позволяет по двум координатам определить код символа, который находится в соответствующей позиции экрана. Приведенная ниже программа выполняет аналогичную задачу на БК: получив адрес ячейки видеопамяти, она определяет код символа, находящегося в этом месте (если он там есть), — иначе говоря, распознает буквы на экране видеомонитора. Такой способ удобен для текстовых редакторов: можно дать пользователю полную свободу в наборе строки, а потом по буквам ее загрузить, вместо того чтобы писать громоздкую программу по обслуживанию управляющих клавиш.

Вывод символов в БК осуществляется командой EMT 16. Их изображения хранятся в ПЗУ с адреса 112036 (все адреса восьмеричные). При отработке EMT 16 вычисляется адрес нужного символа и его изображение (размером 1*10 байт) копируется в экранное ОЗУ.

Основная идея предлагаемой программы — сравнение данных в экранном ОЗУ (копии) с данными ПЗУ (оригиналом). Адрес символа в экранном ОЗУ можно вычислить так:

```
MOV #X,R1              ; координата X
MOV #Y,R2              ; координата Y
EMT 24
MOV @#160,R3           ; в R3 адрес символа в экранном ОЗУ
ADD #100,R3
```

Так как первый байт копии всегда нулевой (как и байт оригинала), он вообще не проверяется, чтобы не занимать лишнего времени.

```
; ПРОГРАММА 'LETTER'
; используются регистры R0,R1,R2,R3,R4,R5
; входные данные:  R1 - адрес символа в экранном ОЗУ
; выходные данные: R5 - определенный программой
;                       код символа
```

```
LETTER:  MOV #40,R5        ; R5 - код символа
         MOV #112277,R2    ; R2 - адрес символа
A:       MOV #10,R0        ; R0 - min число
         MOV R1,R3         ; значащих байтов
         MOV R2,R4
B:       CMPB (R3),(R4)+   ; копия и оригинал
         BNE D             ; равны?
         ADD #100,R3       ; если да, то
         SOB R0,B          ; следующий байт
L1:      CMPB #200,R5      ; код 8-разрядный?
         BHI C             ; нет
         ADD #40,R5
C:       RTS PC            ; RETURN
D:       ADD #12,R2        ; если нет, то
         INC R5            ; следующий символ
         BIT #400,R5       ; символы кончились?
         BEQ A             ; еще нет
         MOV #40,R5        ; непонятный символ
         BR C              ; заменили пробелом
         HALT
```

Замечания к программе

1. Коды с 200 по 240 (это «ИНДСУ», «БЛОКРЕД», «ГРАФ», «ЗАП», «ШАГ» и т.д.) не имеют прорисовки, т.е. при их обработке драйвер ТВ-приемника на экран ничего не выводит (если не считать индикации в служебной строке). Из-за этих «пустых символов» возникает смещение на 40 байт, которое необходимо учесть при определении кода (метка L1).

2. Первые 160 байт, считая с 112036, изображают управляющие клавиши в режиме «БЛОКРЕД». Их можно опустить, взяв сразу адрес пробела — 112276 (так как первый байт нулевой, в программе берется адрес 112277).

Примечание редактора

Действительно, в версиях БЕЙСИКа для IBM имеется функция N=SCREEN(X,Y,Z), возвращающая код символа, изображенного на экране в позиции (X,Y). Но на IBM реализация этой функции проста: в текстовом режиме ее «видео-ОЗУ» содержит в явном виде массив кодов и их атрибутов символов. На БК же режим вывода всегда «графический», экранное ОЗУ содержит изображение символов, а их коды нигде не сохраняются (если только их сохранение не предусмотрено в самой программе пользователя). Так что единственный способ реализации подобной функции на БК — побайтное сравнение изображения символов (как спрайта размером 1*10 байт) с содержимым знакогенератора. Предложенная программа использует именно этот способ.

Программа может быть полезна не только при создании текстовых редакторов, но и, например, при печати на принтере копии текста на экране (текст печатается гораздо быстрее, чем графика) или для сохранения содержимого экрана при реализации «выпадающих окон» (для массива кодов символов требуется намного меньший объем памяти, чем для изображения того же текста в виде спрайта, как это делается обычно).

Однако у данного метода имеются и свои недостатки. Во-первых, программа способна распознавать только стандартный текст в режиме 64 символа в строке. Если же в указанной позиции экрана находится графическое изображение, или если текст был после вывода на экран каким-либо образом преобразован (превращен в «курсив», «сжат» либо «растянут» по ширине или по высоте и т.п.), или если стандартный текст выведен в режиме инверсии, подчеркивания либо даже просто в режиме 32 символа в строке, программа не может его определить и возвращает в R5 код пробела. Во-вторых, она путает символы с одинаковым начертанием (например, русские буквы «А», «О», «В», «Е» и т.д. будут определены ею как соответствующие латинские). Это происходит из-за того, что при сравнении программа перебирает массив знакогенератора (оригиналы) в порядке возрастания кодов символов и при совпадении сразу же прекращает поиск.

Кроме того, следует предостеречь читателей: при вызове подпрограммы LETTER в регистр R1 нужно занести в качестве адреса в экранном ОЗУ значение, полученное по методике, рекомендуемой автором статьи, или по формуле: &O42000+Y*&O1200+X+&O100, где &O42000 — адрес верхнего левого угла экрана с учетом служебной строки (&O40000 — адрес верхнего левого угла служебной строки), X и Y — координаты текстовой позиции экрана, а &O100 прибавляется для пропуска нулевого байта. Если же смещение &O100 не прибавить, программа работать не будет.

Р. Автухов,
г. Липецк

ПОДПРОГРАММА ДЛЯ ОЧИСТКИ ЭКРАНА

Предлагаемая небольшая перемещаемая подпрограмма позволяет осуществить красивую очистку экрана.

```
START:  MOV #40000,R0       ; начало экрана
        MOV #77776,R1       ; конец экрана
B:      MOV #52525,(R0)+    ; синий цвет
        MOV #177777,-(R1)   ; красный цвет
        MOV #200,R5
A:      SOB R5,A
        CMP R0,#60000       ; середина экрана
        BNE B
        CLR (R1)
D:      CLR (R0)+
        CLR -(R1)
        MOV #200,R5
C:      SOB R5,C
        CMP #40000,R1
        BNE D
        HALT
```

А. Н. Сыченко,
г. Новосибирск

ПОДПРОГРАММА ВЫВОДА ДЕСЯТИЧНЫХ ЧИСЕЛ

Думаю, что предлагаемая маленькая подпрограмма для вывода на экран десятичных чисел значительно облегчит жизнь многим программистам БК. Подпрограмма перемещаема, ее длина не превышает 52 байтов.

```
; Подпрограмма печати десятичного числа, содержащегося
; в регистре R4
;
DES:  MOV R7,R2      ; вычисление адреса
      ADD #40,R2     ; констант для перевода
      MOV #5,R3      ; цикл из 5 итераций
2:    MOV #57,R0     ; R0 - код цифры
1:    INC R0         ; увеличить R0
      SUB @R2,R4     ; вычесть константу и, если
      BHIS 1         ; больше или равна, повторить
      EMT 16         ; печать цифры на экране
      ADD @R2,R4     ; коррекция числа
      ADD #2,R2      ; адрес константы увеличить на 2
      SOB R3,2       ; цикл по регистру R3
      RTS R7         ; возврат из подпрограммы
```

Примечание редактора

Работа предлагаемой программы основана на наиболее простом алгоритме: реализации деления на 10 в четвертой, третьей и т.д. степени и выводе целой части результата каждого деления в качестве десятичной цифры. Само же деление реализуется путем вычитаний из исходного заранее «заготовленных» чисел, соответствующих десятичным 10000, 1000, 100, 10 и 1, с подсчетом количества выполненных вычитаний. Когда же содержимое R4 (оно является остатком от деления на соответствующую степень 10) ока-

```
NOP         ;
.#23420     ; десятичные константы
.#1750
.#144
.#12
.#1
```

зывается меньше, чем текущее вычитаемое, из таблицы извлекается следующее число. Оператор NOP нужен исключительно для того, чтобы при вычислении в R2 адреса начала таблицы прибавляемое во второй строке число #40 было «круглым». Вместо NOP можно записать любой оператор ассемблера, занимающий в кодовом тексте 2 байта, например .#0. Можно и вовсе убрать оператор, заменив число #40 во второй строке на #36 (при этом экономится 2 байта).

При использовании подпрограммы нужно помнить следующее.

— Содержимое исходного и операционных регистров (R0, R2, R3 и R4) не сохраняется.

— Содержимое R4 интерпретируется как беззнаковое целое число в диапазоне от 0 до 65535 (в восьмеричном представлении — от #0 до #177777 соответственно).

— «Пропуск» незначащих нулей слева не производится.

А. С. Ланеев,

г. Ташкент

ДЖОЙСТИК В ВИЛЬНЮССКОМ БЕЙСИКЕ

Известно, что общение с играми да и с другими программами становится приятнее (и безопаснее для клавиатуры БК), если используется управление от джойстика. Его реализация на БЕЙСИКе БК-0010.01 оказалась несложным делом. Примером этого является небольшая программа, позволяющая рисовать с помощью джойстика:

```
 5 ' (c) 1990 LANEEW
10 X=122
20 Y=122
30 CLS
40 IF PEEK(&O177714)=1 THEN R=-1 ' влево
50 IF PEEK(&O177714)=4 THEN P=-1 ' вверх
60 IF PEEK(&O177714)=8 THEN P=1 ' вниз
70 IF PEEK(&O177714)=2 THEN R=1 ' вправо
80 PSET (X,Y),1
90 X=X+R
100 Y=Y+P
110 IF X<10 THEN X=240
120 IF X>240 THEN X=10
130 IF Y>230 THEN Y=10
140 IF Y<10 THEN Y=230
150 PSET (X,Y),2
160 R=0
170 P=0
180 GOTO 40
```

Как видим, определение положения джойстика сводится к анализу содержимого порта УВВ (регистр с адресом 177714_8), что нетрудно сделать с помощью условного оператора IF PEEK(&O177714)=<константа> THEN <номер строки для обработки>. В приведенном выше примере используются только простейшие движения джойстика (вверх, вниз, вправо и влево), но аналогичным способом можно реализовать реакцию на диагональные наклоны рукоятки и нажатие кнопок (в том числе одновременно с наклоном рукоятки).

Следует отметить, что поскольку общепринятого стандарта подключения джойстика к БК не существует, константы в операторах IF будут зависеть от конкретной распайки и, в общем случае, могут оказаться различными для разных пользователей. (Значения констант в приведенной программе соответствуют следующей распайке: «вверх»—B23, «вниз»—B17, «влево»—B24, «вправо»—A24, «общий»—A18.) Определить значения для своего джойстика вы можете с помощью простейшей программки:

```
10 KOD=PEEK(&O177714)
20 IF KOD <> 0 THEN ? KOD
30 GOTO 10
```

После ее запуска и какой-либо манипуляции с джойстиком (действия рукояткой или нажатия на кнопки) по экрану начинает «бежать» число — это и есть значение константы для данного действия.

И последнее. Если вы хотите написать программу «для широкого использования», можно предусмотреть в ее начале операцию «настройки» управления от джойстика, как это сделано в большинстве игр в машинных кодах. Для этого нужно предложить пользователю выполнить элементарные движения джойстиком, на которые должна впоследствии реагировать программа, запомнить содержимое регистра &O177714 для каждого движения (если джойстик в данный момент не используется, содержимое регистра равно нулю), а позже использовать в операторах IF вместо констант переменные с запомненными значениями.

П. П. Животовский,
г. Мурманск

ОДНОВРЕМЕННОЕ ПОДКЛЮЧЕНИЕ К БК-0011М НЕСКОЛЬКИХ ПЕРИФЕРИЙНЫХ УСТРОЙСТВ

Среди домашних компьютеров БК имеет наиболее развитую периферию (5- и 8-контактный джойстик, мышь, принтер, графопостроитель). Их подключение к БК по отдельности неоднократно рассматривалось в журналах (например, в «Информатике и образовании»). Однако при использовании в процессе работы сразу всех этих устройств частые перестыковки могут привести к повреждению разъема «УП», а, кроме того, необходимо отключать питание БК во время переключения.

Предлагаемая аппаратная доработка поможет избежать этих неудобств при работе с несколькими периферийными устройствами. В состав комплекта БК-0011М входит блок «КМ», который практически не используется, так как обслуживает только два устройства, да и то под управлением не очень удачной ОС. Выньте из него плату (пустой корпус блока «КМ» можно также купить на радиорынке) и вставьте ответную часть разъема «УП». В качестве последнего можно использовать нагрузочное устройство, убрав из него перемычки, либо разъемы, входящие в комплект мыши или графопостроителя. В верхней части блока установите свои разъемы по количеству имеющихся периферийных устройств (любые, желательно с минимальными габаритами и количеством контактов, соответствующим подключаемому устройству). После распайки согласно приведенным ниже таблицам вставьте блок в разъем «УП». Теперь к промежуточным разъемам можно подключить все устройства одновременно или подключать и отключать их по мере необходимости, не выключая питания БК.

Распайка переходного устройства к БК-0011М

ПРИНТЕР МС6313М.02

Разъем принтера	Вилка РШ2Н-1-30	Розетка РГ1Н-1-5	Разъем "УП" БК0011М
23	1	1	А28 (А25)
3	2	2	А16
5	3	3	А13
7	4	4	В12
9	5	5	В10
11	6	6	В5
13	7	7	В7
15	8	8	В6
17	9	9	А7
21	11	11	В31 (В29)
4,6	16	16	А11

ПЛОТТЕР XY4140

Разъем плоттера	Вилка РШ2Н-1-30	Розетка РГ1Н-1-5	Разъем "УП" БК0011М
1	2	2	А16
2	3	3	А13
3	4	4	В12
4	5	5	В10
5	1	1	А23
5	9	9	А7
6	16	16	А18

МЫШЬ "МАРСИАНКА"

Разъем мыши	Вилка РШ2Н-1-30	Розетка РГ1Н-1-5	Разъем "УП" БК0011М
1	7	7	В24
2	2	2	А24
3	6	6	В23
4	3	3	В17
5	1	1	А20
6	4	4	В22
7	16	16	А19
8	16	16	А19
9	5	5	В10
10	13	13	А8

ДЖОЙСТИК 1

Разъем джойстика	Вилка РШ2Н-1-30	Розетка РГ1Н-1-5	Разъем "УП" БК0011М
1 (Вверх)	6	6	В24
2 (Общий)	16	16	В11
3 (Вправо	2	2	А24
4 (Вниз)	4	4	А23
5	Не используется		
6 (Влево)	3	3	В17
7 (Кнопка	1	1	А20

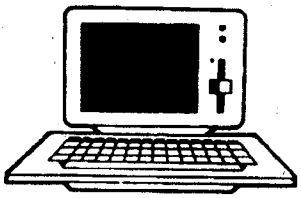
ДЖОЙСТИК 2

Разъем джойстика	Вилка РШ2Н-1-30	Розетка РГ1Н-1-5	Разъем "УП" БК0011М
1 (Вверх)	6	6	В30
2 (Общий)	16	16	В18
3 (Вправо)	3	3	В22
4 (Вниз)	4	4	В23
5			
6 (Влево)	2	2	А32
7 (Кнопка)	1	1	А23

ДЖОЙСТИК 3 (герконовый)

Разъем мыши	Вилка РШ2Н-1-30	Розетка РГ1Н-1-5	Разъем "УП" БК0011М
1 (Вверх)	6	6	В24
2 (Общий)	16	16	В19
3 (Вправо)	2	2	А24
4 (Вниз)	4	4	В23
5 (Кнопка 1)	11	1	А20
6 (Влево)	3	3	В17
7 (Кнопка 2)	7	7	В22

СПРАВОЧНОЕ БЮРО



В этой статье приведены краткие сведения о назначении системных ячеек БК-0010(.01), расположенных по адресам 000-377$_8$ и используемых подпрограммами ПЗУ монитора. Более подробное описание имеется в журнале «Вычислительная техника и ее применение», № 3 за 1992 г., с.16.

А. В. Чумак,
г. Комсомольск-на-Амуре

СИСТЕМНЫЕ ЯЧЕЙКИ ПЭВМ БК-0010

004—006*: PC и PSW прерывания по ошибке передачи данных, при нажатии клавиши «СТОП», по команде процессора HALT или из-за команд, нарушающих четность PC и SP.

010—012: PC и PSW прерывания по неправильным командам ЦП. Это свойство может быть использовано для реализации «дополнительных» команд (псевдокоманд), обрабатываемых через данное прерывание. Пример — «команды плавающей арифметики» в интерпретаторе ФОКАЛа.

014—016: PC и PSW прерывания по биту T — trace, позволяющему отслеживать процесс выполнения фрагментов программ при их отладке.

020—022: PC и PSW прерывания по команде IOT.

024—026: PC и PSW прерывания по аварии питания.

030—032: PC и PSW прерывания по команде EMT. Диспетчер EMT-команд расположен по адресу 100112, а таблица адресов подпрограмм, вызываемых командой EMT, — 100004—100110.

034—036: PC и PSW прерывания по команде TRAP. Прерывание TRAP стандартными драйверами не используется, а в системах ФОКАЛ и Бейсик применяется в служебных целях (для вывода символов с кодами 0—177 и сообщений об ошибке).

040—056: флаги (байты) состояния дисплея:

040 — режим 32/64 символа в строке;

041 — инверсия фона экрана;

042 — режим расширенного ОЗУ;

043 — РУС/ЛАТ (этот байт служит также маской для формирования кодов русских и латинских букв из кодов клавиатуры, его стандартное значение — 200$_8$ в режиме РУС, 0 — ЛАТ);

044 — режим ПОДЧеркивания символов;

045 — режим ИНВерсии символов;

046 — режим ИНДикации Символов Управления;

047 — режим БЛокировки РЕДактирования;

050 — ГРАФический режим;

051 — режим ЗАПиси;

052 — режим СТИРания (при одновременной установке флагов СТИР и ЗАП работает режим ЗАП; ГРАФ индицирует включение одного из этих графических режимов);

053 — режим 32/64 символа в служебной строке;

054 — режим ПОДЧеркивания символов в служебной строке;

055 — режим ИНВерсии символов в служебной строке;

056 — режим погашенного курсора.

Значение любого из этих флагов, равное -1 (кроме РУС/ЛАТ), свидетельствует о включенном режиме. Установка/отмена режима осуществляется побитной инверсией соответствующего флага командой COM FLAG (кроме РУС/ЛАТ).

060-062: PC и PSW прерывания по нижнему регистру клавиатуры. Адрес обработки — 101136.

100-102: PC и PSW пользовательского прерывания IRQ2 по сигналу, поданному на контакт ПРТ порта ввода-вывода. БЕЙСИК устанавливает его на команду RTI (адрес 135070). ФОКАЛ — на остаток своей функции FCLK (адрес 132226).

Драйвер клавиатуры

104: Код последней нажатой клавиши.

105: Флаг несчитанного кода. Содержит последний введенный с клавиатуры, но еще не считанный программой код. При чтении обнуляется (для EMT 6). До тех пор пока значение этой ячейки не равно 0, нажатие всех прочих клавиш игнорируется.

106: Слово задержки при ожидании отпускания клавиши «ПОВТ».

110: Флаг нажатия клавиши «ПОВТ».

111: Количество выдаваемых функцией EMT 6 при нажатии клавиши ТАБ пробелов. (При ненуле-

* Здесь и далее все адреса – восьмеричные.

вом значении байта функция EMT 6 всегда выдает код пробела.)

112—120: 32/64-битовое поле. Каждый установленный бит — затабулированная позиция экрана. Для режима 32 символа в строке слова 116 и 120 не используются.

122: Количество выдаваемых символов строки ключа. (Нулевое значение — отсутствие «активного» ключа.)

124: Адрес очередного печатаемого символа строки ключей.

126—150: Массив адресов упакованных строк ключей.

Драйвер дисплея

152: Флаг нормализации положения курсора. Подпрограмма, осуществляющая нормализацию, располагается по адресу 102414 и производит коррекцию координат курсора при выходе из рабочего поля (а также если установлен флаг 152), привязку координат (X,Y) курсора (ячейка 156) к значению ячейки 160.

154: Маска для формирования точек, рисуемых курсором в графическом режиме. Возможное применение аналогично ячейке 155.

155: Маска для формирования точек графики. В режиме 64 символа в строке используется нулевой бит, при 32 — два младших бита. Если же установить в 1 остальные биты, то можно проводить линии увеличенной толщины (3—8).

156: Позиция курсора в формате $X+100_8*Y$ (для 64 символов в строке) и $X*2+100_8*Y$ (для 32 символов в строке).

160: Адрес верхнего левого (в режиме 32 символа в строке) угла изображения курсора. При печати символа для формирования его изображения сначала используется эта ячейка, а потом осуществляется привязка ячейки 156 к 160, поэтому для установки курсора лучше изменять ячейку 160, в противном случае — пользоваться нормализацией курсора.

162: Смещение курсора после печати символа. При режимах 32/64 ее содержимое равно 2 или 1. Чтобы курсор после печати символа смещался в другом направлении, надо записать в ячейку 162 такие значения:

режим 64 символа в строке:

1	— вправо;	177777	— влево;
100	— вниз;	177700	— вверх;

режим 32 символа в строке:

2	— вправо;	177776	— влево;
100	— вниз;	177700	— вверх.

Значение, большее 1 или 2, дает печать символов вразрядку.

164: Используется при нормализации курсора, в подпрограммах перевода режимов, служит ограничением сверху для ячейки 156. Устанавливая ячейку 164 и другие, можно изменить размер рабочей области экрана. В обычном режиме равна 300, в РП — 400.

166: Расстояние от начала рабочего поля до курсора в режиме графики.

170: Адрес байта, в котором находится центр графического курсора. В обычном режиме 42000, в РП — 72000 (аналогично ячейке 160).

172: Количество графических точек, заданное в режиме ГРАФ цифровыми клавишами.

174: Координаты (X,Y) курсора в режиме графики (аналогично ячейке 156).

176: X-координата графического курсора.

200: Y-координата графического курсора.

202: Адрес начала поля служебной строки. Для обычного режима вывода 40000, для РП — 70000.

204: Ширина поля служебной строки, т.е. величина, которую надо прибавить к содержимому ячейки 202 для получения адреса начала рабочей области экрана. Обычно равна 2000. При нулевом значении служебная строка не изображается.

206: Размер области видеоОЗУ в байтах, обычно 40000, в РП — 10000.

210: Размер рабочей области экрана (без учета служебной строки), обычно 36000, т.е. 360 (240 дес.) телевизионных строк, в режиме РП — 6000 и 60 (48) соответственно. Устанавливая ячейки 164, 170, 202, 204, 206 и 210, можно добиться необходимого размера рабочей области экрана, чтобы использовать видеоОЗУ для хранения программ или данных (так, например, поступает система MIRAGE).

212: Маска для формирования фона. Используется при очистке экрана, формировании символов и т.д.

214: Маска для формирования изображения графики, символов в режиме 32 символа в строке, подчеркивания и т.д.

216: Маска для формирования фона служебной строки. Используется аналогично ячейке 212.

220: Маска для формирования линии служебной строки. Очистив эту ячейку и нормализовав служебную строку кодом 236, можно погасить линию служебной строки. Это слово также служит флагом для заполнения кодом 30 или 10020 (в зависимости от режима 32/64) одной телевизионной строки ниже сплошной линии (разметка знаковых позиций).

222: Увеличивается на 1 при печати символа, обнуляется при нажатии клавиши и последующей печати символа, т.е. равно числу напечатанных при нажатии клавиши символов минус 1.

224: При нажатии клавиши происходит прерывание от клавиатуры, обрабатываемое по адресам 101136 или 101362. При этом вызывается подпрограмма подзвучки по адресу 102032, которая в зависимости от флага 224 вызывает подпрограмму нормализации режимов служебной строки по адресу 110536.

Последняя проверяет текущие режимы дисплея и устанавливает соответствующие индикаторы в служебной строке. В этой подпрограмме ячейка 224 служит счетчиком. Ячейку 224 устанавливают коды, изменяющие режимы дисплея. Во многих программах нормализация служебной строки является нежелательной, поэтому после выдачи управляющих кодов лучше очистить ячейку 224 во избежание нормализации при случайном нажатии клавиш.

226—252: Используются на БК, оснащенных локальной сетью.

254: Задержка при передаче на ТЛГ-линию. Ее содержимое в зависимости от режима представлено в таблице.

256: Зарезервирована в качестве буфера для записи выводимой в порт ввода-вывода информации (автоматическое сохранение не производится).

260: Адрес обработки программами пользователя нажатия клавиш. При нулевом содержимом обработки не происходит.

262: Флаг передачи вместо кода 12 (ПС) значения 15 (ВК). Эта возможность практически не используется программами на БК, кроме интерпретатора ФОКАЛа.

264: Адрес последнего массива данных, загруженного в ОЗУ с магнитофона или линии.

266: Длина последнего массива данных, загруженного в ОЗУ с магнитофона или линии.

274—276: PC и PSW прерывания по нижнему регистру клавиатуры. Стандартный адрес обработки 101362.

300: Подпрограммами чтения с магнитофона используется как флаг чтения в инверсном сигнале.

301: Байт ответа драйвера магнитофона.

302: Флаг фиктивного чтения.

304: Смещение байта данных при чтении с магнитофона. Можно организовать чтение массивов «наоборот» или вразрядку.

306: Используется для сохранения адреса блока параметров.

310: Используется для сохранения SP.

312: Контрольная сумма массива.

314: Средняя длительность сигнала при чтении (для контроля). Вычисляется как средняя длительность 200 сигналов настроечной последовательности, умноженная на 1.5.

Таблица

Номер	Скорость	Задержка
1	4800	36
2	2400	103
3	1200	213
4	600	435
5	300	1100
6	150	2206

Параметры магнитофона

Размещение блока данных для функции EMT36 с адреса 320 рекомендуется в Руководстве пользователя и является стандартным.

320: Байт команды.

321: Байт ответа.

322: Адрес массива для записи и чтения (при нулевом значении считывание файла производится по адресу, хранящемуся в его «заголовке» на магнитной ленте).

324: Длина массива для записи.

326—345: Имя массива для записи или чтения (20_8 символов).

346: Адрес текущего массива.

350: Длина текущего массива.

352: Имя текущего массива (20_8 символов)

372—376: При работе с локальной сетью используются программы, зашитые в ПЗУ ФОКАЛа или БЕЙСИКа:

ФОКАЛ — вектор прерывания 360 (канал связи 11). Адрес обработки — 136730;

БЕЙСИК — вектор прерывания 320 (канал связи 7). Адрес обработки — 146404.

Примечание редактора

Для сохранения и последующего восстановления содержимого регистров R0-R5 можно рекомендовать такие последовательности команд ассемблера:

```
; сохранение в стеке
MOV R5,-(R6)
MOV R4,-(R6)
JSR R4,@#110346
```

```
; восстановление из стека
JSR R4,@#110362
MOV (R6)+,R4
MOV (R6)+,R5
```

А. В. Милюков,
г. Москва

НЕКОТОРЫЕ ПРОГРАММЫ МОНИТОРА БК-0010

Кроме предоставляемых пользователю «стандартных» EMT-функций монитор содержит ряд подпрограмм «для собственных нужд». Их также можно (при некоторых условиях) использовать и в своих программах.

- Подпрограмма @#100460 выводит на экран текст, адрес начала которого задан в R3. Последний байт текста должен быть нулевым.
- Подпрограмма @#100472 помещает в R5 восьмеричное число, введенное с клавиатуры. Исправления при помощи клавиши «ЗАБОЙ» возможны, однако после этого не всегда число на экране совпадает с записываемым в R5. Окончание ввода — нажатие клавиши «Ввод». (Подробности о работе этой подпрограммы см. Вычислительная техника и ее применение. 1990. №12. С.34. — *Прим. ред.*)
- Подпрограмма @#102032 выдает щелчок и возвращает в R0 код из регистра 177662_8. Одновременно производится установка индикаторов в служебной строке, если содержимое ячейки 224_8 не равно нулю.
- Подпрограмма @#104116 умножает содержимое регистра R3 на 4 и заносит в R2 значение произведения R3*5.
- Подпрограмма @#107030 умножает содержимое регистра R1 на 8, прибавляет к содержимому R0 значение R1*10 и заносит результат в ячейку 172_8.
- Подпрограмма @#111576 окрашивает в цвет фона десять соседних строк телевизионного растра. Маска находится в ячейке @#212, а адрес первого байта строки растра (кратный 100_8), начиная с которой идет окрашивание, — в R3. Таким способом можно очистить как весь экран, так и любую его часть, равную по высоте одной символьной строке, не прибегая к циклической печати пробелов. Если адрес в R3 превышает 100000_8, из него вычитается содержимое ячейки @#206 (обычно равное 40000_8).
- Подпрограмма @#116622 возвращает в R0 значение контрольной суммы массива длиной R4 байт с начальным адресом, хранящимся в R5. Содержимое R2 не сохраняется.

От редакции

ИСПОЛЬЗОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ БК-0010.01, ЗАШИТОГО В ПЗУ, И НЕ ТОЛЬКО...

Вопрос об использовании защитых в ПЗУ программ ставился на страницах журналов уже неоднократно. Например, в «Вычислительной технике» № 8 за 1990 г. было рассказано об использовании стандартной подпрограммы вывода символов на экран для вывода спрайтов, о подпрограммах сохранения содержимого регистров в стеке, файлов с магнитофона и т.п.

Предлагаем вниманию читателей еще ряд сведений по той же теме – результаты «микроисследования» ПЗУ, установленных в блоке МСТД для к БК-0010.01 (эта информация верна также и для БК-0010, но там ПЗУ размещено внутри корпуса компьютера).

1. В программах в машинных кодах, выполняющих какие-либо расчеты, обмен данными с магнитофоном и прочие действия, требующие диалога с пользователем, часто предусматривается вывод текстовых сообщений об ошибках. Такие тексты обычно хранятся в ОЗУ вместе с кодами самой программы и печатаются на экране с помощью функции EMT 20. Но это слишком «роскошный» способ диалога с пользователем — ведь тексты занимают довольно много места в памяти, а ей БК и так небогат. Поэтому разработчики программ частенько ограничиваются очень короткими, но, увы, не слишком вразумительными фразами. Однако для сообщения об ошибках можно использовать тексты, записанные в ПЗУ ФОКАЛа, тогда «и волки будут сыты и овцы целы».

Ниже приводится таблица адресов текстов сообщений об ошибках с указанными номерами. (Соответствие номеров ошибок и самих текстов сообщений можно посмотреть в прилагаемом к БК «Руководстве пользователя по ФОКАЛу».)

Номер	Адрес (восьм.)	Номер	Адрес (восьм.)
00	135040	14	135726
01	135064	15	135756
02	135116	16	136032
03	135173	17	136104
04	135213	18	136157
05	135240	19	136214
06	135274	20	136264
07	135371	21	136353
08	135431	22	136402
09	135511	23	136433
10	135534	24	136456
11	135601	25	136516
12	135632	26	136546
13	135657	27	136600

Все тексты сообщений об ошибках в ФОКАЛе оканчиваются нулевым байтом, поэтому указывать длину строки не требуется. Для вывода на экран можно воспользоваться подпрограммой, также находящейся в ПЗУ ФОКАЛа и имеющей начальный адрес 163270_8 (вызов производится командой JSR R7,@#163270, перед этим в регистр R4 нужно занести адрес начала текстовой строки) или последовательностью команд:

```
MOV #нач.адрес, R1
CLR R2
EMT 20
```

Изменяя же значение адреса начала текста и явным образом задавая в R2 его длину, можно выводить только часть прошитого в ПЗУ сообщения. Например, задав R1=136032 и R2=26, можно выдать на экран фразу «слишком много символов».

Аналогичным образом можно при желании вывести на экран и текст подсказки по ФОКАЛу (вызываемый обычно командой HELP): его начальный адрес — 136650_8.

2. Приведенные ниже подпрограммы ПЗУ вы можете вызывать из своей программы в машинных кодах (команда ассемблера JSR R7,@#адрес):

Адрес (восьм.)	Назначение подпрограммы
163220	Вывод на экран восьмеричного числа, записанного в R4
163270	Вывод на экран строки текста. Начальный адрес в R4, строка заканчивается нулевым байтом
165464	Пуск мотора магнитофона (если он оборудован дистанционным управлением)
165446	Останов мотора магнитофона
164774	Вывод текста в служебную строку. Перед вызовом в R2 записывается адрес начала строки текста, а в R1 — число, на 1 меньшее ее длины. (ВНИМАНИЕ! Особенность данной подпрограммы: строка текста должна быть записана в ОЗУ «наоборот» — справа налево. Например, чтобы вывести текст «ПРИМЕР», нужно записать его в ОЗУ как Р,Е,М,И,Р,П и занести в регистры значения R2=адр.начала, R1=5. Текст будет выведен в командную строку с левого края, без «окантовки» пробелами.)

3. Программа расчета контрольной суммы для фрагмента ОЗУ (R1=начальный адрес, R2= длина):

```
        CLR R4
A$:     MOV @R1,R0
        ADD R0,R4
        ADC R4
        ADD #2,R1
        SOB R2,A$
; печать значения
; контрольной суммы
        JSR R7,@#163220
```

4. Подпрограмма ввода восьмеричного числа с клавиатуры:

```
        CLR R5
A$:     EMT 6
        CMP R0,#12
        BEQ B$
        ASL R5
        ASL R5
        ASL R5
        BIC #177770,R0
        ADD R0,R5
        BR A$
B$:     RTS R7
```

После возврата из этой подпрограммы введенное число находится в R5. (ВНИМАНИЕ! Здесь не предусмотрена защита от ошибочного ввода цифр 8, 9 и нецифровых символов, а также набора более шести символов. Окончание ввода — клавиша «ВВОД».)

5. Возможно, для кого-либо окажется интересной информация о расположении в ПЗУ программ-тестов БК-0010:

Номер теста	Адрес (восьм.)
1	170262
2	166742
3	171170
4	166466
5	173232

6. В заключение дадим небольшой совет пользователям БК-0010.01, работающим на Вильнюсском БЕЙСИКе. БК распознает как имена переменных, так и некоторые команды языка по двум первым символам. При этом, если команда, записанная двумя символами, первая в строке, БК сам «дописывает» недостающие знаки. (В этом можно убедиться, задав команду LIST.) С другой стороны, имена переменных, начинающихся с соответствующей пары букв и стоящих первыми в строке (в операторах присваивания, в которых опущен оператор LET), ошибочно могут быть приняты за команду БЕЙСИКа, что вызывает сообщение «ОШИБКА 2» (синтаксис) и недоумение пользователя: откуда в правильно набранной (а затем и в уже исправленной) строке вдруг появляется «лишняя» команда? Единственное средство в таких случаях (помимо переименования злополучной переменной во всей программе) — это использование оператора LET, тогда имя переменной БК с командой БЕЙСИКа уже не перепутает.

Команда	Пара букв	Команда	Пара букв	Команда	Пара букв	Команда	Пара букв
AUTO	AU	CSAVE	CS	INPUT	IN	PAINT	PA
BEEP	BE	DIM	DI	KEY	KE	POKE	PO
BLOAD	BL	DRAW	DR	LLIST	LL	PSET	PS
BSAVE	BS	END	EN	LPRINT	LP	RUN	RU
CALL	CA	FIND	FI	MONIT	MO	SAVE	SA

СПРАВОЧНЫЙ ЛИСТОК

А. Конюшенко
= UC2AAU =
220030 Минск-30.я.1
БЕЛАРУСЬ

КОДЫ МАШИННЫХ КОМАНД 1801ВМ1 (PDP-11)			
00 00 00 HALT	00 4R DD JSR	10 00 00 +XXX BPL	
00 00 01 WAIT	00 50 DD CLR	10 04 00 +XXX BMI	
00 00 02 RTI	00 51 DD COM	10 10 00 +XXX BHI	
00 00 03 BPT	00 52 DD INC	10 14 00 +XXX BLOS	
00 00 04 IOT	00 53 DD DEC	10 20 00 +XXX BVC	
00 00 05 RESET	00 54 DD NEG	10 24 00 +XXX BVS	
00 00 06 RTT	00 55 DD ADC	10 30 00 +XXX BCC, BHIS	
	00 56 DD SBC	10 34 00 +XXX BCS, BLO	
(00 00 07 - 00 00 11) [1a]	00 57 DD TST	10 40 00 - 10 43 77 EMT	
	00 60 DD ROR	10 44 00 - 10 47 77 TRAP	
	00 61 DD ROL	10 50 DD CLRB	
00 00 12 START	00 62 DD ASR	10 51 DD COMB	
	00 63 DD ASL	10 52 DD INCB	
(00 00 13 - 00 00 15) [1b]	00 64 NN MARK	10 53 DD DECB	
		10 54 DD NEGB	
	(00 65 SS) (MFPI) [2]	10 55 DD ADCB	
00 00 16 S	(00 66 DD) (MTPI) [3]	10 56 DD SBCB	
		10 57 DD TSTB	
(00 00 17 - 00 00 77) [1c]	00 67 DD SXT	10 60 DD RORB	
		10 61 DD ROLB	
00 01 DD JMP	(00 70 00 - 00 77 77)[4]	10 62 DD ASRB	
00 02 0R RTS		10 63 DD ASLB	
	01 SS DD MOV	10 64 SS MTPS	
(00 02 1R тех.контроль)	02 SS DD CMP		
(00 02 2N тех.контроль)	03 SS DD BIT	(10 65 SS) (MFPD) [9]	
(00 02 3N) (SPL)	04 SS DD BIC	(10 66 DD) (MTPD) [10]	
	05 SS DD BIS		
00 02 40 NOP	06 SS DD ADD	10 67 DD MFPS	
00 02 41 - операции с			
- 00 02 77 разрядами PSW	(07 0R SS) (MUL) [5]	(10 70 00 - 10 77 77) [11]	
00 03 DD SWAB	(07 1R SS - 07 3R SS)[6]		
00 04 00 +XXX BR		11 SS DD MOVB	
00 10 00 +XXX BNE	07 4R SS XOR	12 SS DD CMPB	
00 14 00 +XXX BEQ		13 SS DD BITB	
00 20 00 +XXX BGE	(07 50 0R - 07 50 3R)[7]	14 SS DD BICB	
00 24 00 +XXX BLT	(07 50 40 - 07 67 77)[8]	15 SS DD BISB	
00 30 00 +XXX BGT		16 SS DD SUB	
00 34 00 +XXX BLE	07 7R NN SOB	(17 00 00 - 17 77 77) [12]	

Область [1a-c] в системе команд PDP-11 не используется; у процессора 1801ВМ1 в этой области имеются команды START и S (шаг), но вследствие отсутствия системных регистров, БК-0010(01) по этим командам отрабатывает вектор 4.

[2-3] команды процессоров с MMU
[4, 8, 11] резервный код
[5] команда умножения (реализована в 1801ВМ1Г)
[6] расширенный набор команд арифметики
[7, 12] команды процессора с плавающей точкой
[9,10] команды процессоров с 22-разрядным MMU

2

УСЛОВНЫЕ ОБОЗНАЧЕНИЯ

В кодах машинных команд

```
 *  = 0  для операций над словами
      1  для операций над байтами
SS  = поле адресации операнда-источника
DD  = поле адресации операнда-приемника
 R  = РОН  N 0...7 (3 бита)
XXX = смещение (-128,...,+127; 8 бит)
 N  = число, 3 бита
NN  = число, 6 бит
```

В обозначениях операций

```
(N) = содержимое ячейки или рег.N
 s  = операнд-источник (scr)
 d  = операнд-приемник (dst)
 r  = содержимое регистра
<=  = становится равным
 X  = относительный адрес
 %  = определение регистра
```

ЛОГИЧЕСКИЕ ОПЕРАЦИИ

```
∧ = логическое И
∨ = логическое ИЛИ
\\ = исключающее ИЛИ
¬ = НЕ
```

ОПЕРАЦИИ НАД РАЗРЯДАМИ PSW

```
* = установка/сброс по результату
- = состояние разряда не меняется
0 = сброс
1 = установка
```

МЕТОДЫ АДРЕСАЦИИ

		Мнемоника		МЕТОД R
0	регистровая	R	(R)	- операнд
1	косвенная регистровая	(R) или @R	(R)	- адрес операнда
2	автоинкрементная	(R)+	(R)	- исполняемый адрес (R) <= (R)+2 (или 1)
3	косв. автоинкрементная	@(R)+	(R)	- адрес адреса (R) <= (R)+2 (или 1)
4	автодекрементная	-(R)	(R) <= (R)-2 (или 1) до исп.ком.	(R) - исп-й адрес
5	косв. автодекрементная	@-(R)	(R) <= (R)-2 (или 1) до исп.ком.	(R) - адрес адреса
6	индексная	X(R)	(R)+X	- адрес
7	косв. индексная	@X(R)	(R)+X	- адрес адреса

С ИСПОЛЬЗОВАНИЕМ СЧЕТЧИКА КОМАНД

			МЕТОД 7
2	непосредственная	#n	операнд n в следующем за командой слове
3	абсолютная	@#A	адрес A в следующем за командой слове
6	относительная	A	адр.операнда = адр.ком.+4+X (+6+X)
7	косв. относительная	@A	адр.адреса опер. = адр.ком.+4+X (+6+X)

ВЕКТОРЫ ПРЕРЫВАНИЯ

Источник прерывания	Адрес вектора
Зависание при передаче данных по каналу или от клавиши <СТОП>	000004
Резервный код команд	000010
Прерывание по Т-разряду	000014
Прерывание по команде IOT	000020
Авария сетевого питания (аппаратно не поддержан)	000024
Прерывание по команде EMT	000030
Прерывание по команде TRAP	000034
Прерывание от клавиатуры	000060
Сигнал IRQ2	000100
Прерывание от клавиатуры (коды нижнего регистра)	000274

3

РАБОТА с ПОДПРОГРАММАМИ

00 00 00	HALT	останов (в БК-0010 при попытке записи PSW в несуществ. регистр 177676,происх.прер.4)	(177716) <= 000010\/(177716) (177676) <= PSW <= (160004) PC <= (160002)
00 00 01	WAIT	пауза - ожидание прерывания	
00 00 02	RTI	возврат из прерывания	PC <= (SP)+ PSW <= (SP)+
00 00 03	BPT	отладочное прерывание вектор 14	-(SP) <= PSW <= (16) -(SP) <= PC <= (14)
00 00 04	IOT	вызов системы ввода-вывода вектор 20	-(SP) <= PSW <= (22) -(SP) <= PC <= (20)
00 00 05	RESET	сброс магистрали и процессора	
00 00 06	RTT	возврат, с запретом прерывания по T-разряду до исп.сл.команды	PC <= (SP)+ PSW <= (SP)+
00 01 DD	JMP	безусловный переход	PC <= d
00 02 0R	RTS	возврат из подпрограммы	PC <= R <= (SP)+
00 02 40	NOP	нет операции (пустая команда)	
00 4R DD	JSR	вызов подпрограммы	-(SP)<= R <= PC <= d
00 64 NN	MARK	восстановление стека	SP <= PC + (2 x NN) PC <= R5 <= (SP)+
07 7R NN	SOB	выч.1 и ветвл.,если (R#) не 0	R# <= R# - 1 PC <= PC - (2 x NN)
10 40 00 - 10 42 77	EMT	вызов ОС, вектор 30	-(SP)<= PSW <= (32) -(SP)<= PC <= (30)
10 44 00 - 10 47 77	TRAP	общего назн.,вектор 34	-(SP)<= PSW <= (36) -(SP)<= PC <= (34)
10 64 SS	MTPS	запись PSW	PSW <= s
10 67 DD	MFPS	чтение PSW	d <= PSW

ПЕРЕХОДЫ ПО УСЛОВИЮ (ВЕТВЛЕНИЯ): Bxx <адр.перехода>

Базовый КОП	+/-	X X X
15	8 7	0

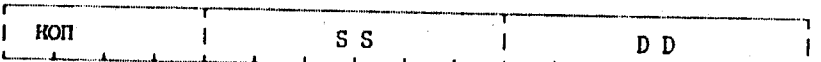
Если условие удовлетворяется, то (PC) <= (PC) + (2 x XXX)

ВОСЬМЕРИЧНЫЕ V	=ТРИАДЫ					
0	0	0	0	10 20 00 +XXX	BVC	арифм.перепол.отсутствует V = 0
				10 24 00 +XXX	BVS	произошло арифм.перепол. V = 1
1	0	0	1	10 30 00 +XXX	BCC	перенос отсутствует C = 0
				10 34 00 +XXX	BCS	произошел перенос C = 1
				Переход по знаку		
2	0	1	0	10 00 00 +XXX	BPL	знак плюс N = 0
				10 04 00 +XXX	BMI	знак минус N = 1
3	0	1	1	00 20 00 +XXX	BGE	больше или равно(нулю) N\/V = 0
				00 24 00 +XXX	BLT	меньше (нуля) N\/V = 1
4	1	0	0	00 30 00 +XXX	BGT	больше (нуля) Z\/(N\/V) = 0
				00 34 00 +XXX	BLE	меньше или равно (нулю) Z\/(N\/V) = 1
5	1	0	1	Переход без знака		
				10 10 00 +XXX	BHI	больше C\/Z = 0
6	1	1	0	10 14 00 +XXX	BLOS	меньше или равно C\/Z = 1
				10 30 00 +XXX	BHIS	больше или равно (= BCC) C = 0
7	1	1	1	10 34 00 +XXX	BLO	меньше (= BCS) C = 1

ОДНООПЕРАНДНЫЕ КОМАНДЫ: OPR dst

				N Z V C
00 03 DD	SWAB	перестановка байтов		* * 0 0
*0 50 DD	CLR(B)	очистка	(d) <= 0	0 1 0 0
*0 51 DD	COM(B)	побитная инверсия	(d) <= (¬d)	* * 0 1
*0 52 DD	INC(B)	прибавление 1	(d) <= (d) + 1	* * * -
*0 53 DD	DEC(B)	вычитание 1	(d) <= (d) - 1	* * * -
*0 54 DD	NEG(B)	изменение знака	(d) <= -(d)	* * * *
*0 55 DD	ADC(B)	прибавить перенос	(d) <= (d) + C	* * * *
*0 56 DD	SBC(B)	вычесть перенос	(d) <= (d) - C	* * * *
*0 57 DD	TST(B)	проверка	(d) <= (d)	* * 0 0
*0 60 DD	ROR(B)	циклич.сд.вправо	=> C,d	* * * *
*0 61 DD	ROL(B)	циклич.сд.влево	C,d <=	* * * *
*0 62 DD	ASR(B)	априм.сд.вправо	(d) <= (d)/2	* * * *
*0 63 DD	ASL(B)	априм.сд.влево	(d) <= 2(d)	* * * *
00 67 DD	SXT	расширить знак	N = 0, (d) <= 0	0 1 0 -
			N = 1, (d) <= 177777	1 0 0 -

ДВУХОПЕРАНДНЫЕ КОМАНДЫ: OPR src dst; OPR src R; OPR R dst



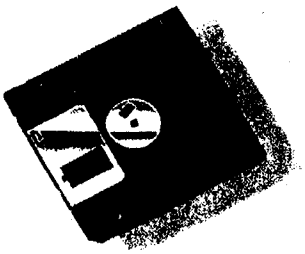
				N Z V C
*1 SS DD	MOV(B)	переслать	(d) <= (s)	* * 0 -
*2 SS DD	CMP(B)	сравнить	(s) - (d)	* * * *
*3 SS DD	BIT(B)	проверить разряды	(s)/\(d)	* * 0 -
*4 SS DD	BIC(B)	очистить разряды	(d) <= (¬s)/\(d)	* * 0 -
*5 SS DD	BIS(B)	установить разряды	(d) <= (s)\/(d)	* * 0 -
06 SS DD	ADD	сложить	(d) <= (s) +(d)	* * * *
07 4R SS	XOR	исключающее ИЛИ	(s) <= (r)\\(s)	* * 0 -
16 SS DD	SUB	вычесть	(d) <= (d) -(s)	* * * *

ОПЕРАЦИИ с РАЗРЯДАМИ PSW:

ЛОГИЧЕСКИЕ ОПЕРАЦИИ:

(s)	0	1	0	1
(d)	0	0	1	1
(s)\/(d)	0	0	0	1
(s)/\(d)	0	1	1	1
(s)\\(d)	0	1	1	0

			Очистить					
00 02 41	CLC	C	-	-	-	0		
00 02 42	CLV	V	-	-	0	-		
00 02 44	CLZ	Z	-	0	-	-		
00 02 50	CLN	N	0	-	-	-		
00 02 57	CCC	N Z V C	0	0	0	0		
		Установить						
00 02 61	SEC	C	-	-	-	1		
00 02 62	SEV	V	-	-	1	-		
00 02 64	SEZ	Z	-	1	-	-		
00 02 70	SEN	N	1	-	-	-		
00 02 77	SCC	N Z V C	1	1	1	1		



В выпуске № 1 за 1993 год были подробно рассмотрены операционные системы для БК-0010(.01): ANDOS, MKDOS, NORD, NORTON, MDOS, DOSB10. Рассмотрим некоторые особенности их работы на БК-0011(М)*, а также системы, работающие только на БК-0011(М): RT-11 и АО-DOS.

В. П. Юров,
г. Москва

ДИСКОВЫЕ ОПЕРАЦИОННЫЕ СИСТЕМЫ ДЛЯ БК-0011(М)

Практически* все операционные системы, ранее рассмотренные нами применительно к БК-0010(.01), почти одинаково работают как на БК-0011, так и на БК-0011М. Отличия этих компьютеров проявляются только при использовании некоторых программ, особенно работающих в среде эмуляции БК-0010(.01). (Эти отличия в дальнейшем не оговариваются.)

Следует также отметить, что операционных систем для БК создано более чем достаточно и многие из них достаточно надежны, чтобы удовлетворить взыскательного пользователя. И хотя некоторые ОС даже не имеют собственного командного монитора (ANDOS, MKDOS, NORD), но за счет развитых «нортон»-подобных оболочек для работы с файлами они вполне могут конкурировать с ОС, имеющими командный монитор (RT-11, AO-DOS, MicroDOS, DOSB10). Поэтому при сравнении тех из них, которые обладают достаточной надежностью в работе, на первый план выступают доступность пользователю, качество оболочек для работы с файлами, а также обеспеченность системными программами и утилитами.

Операционная система RT-11 (ОС БК-11)

Операционная система RT-11 входит в заводской комплект поставки БК-0011(М) и снабжается подробной документацией (в нескольких книгах). Но эта документация рассчитана на подготовленных пользователей, для начинающих же она выглядит как абракадабра. (Отдельные пояснения, изложенные в дополнительных листах, также входящих в комлект заводской поставки, для большинства все равно остаются непонятными.) Ниже будут даны некоторые рекомендации для начинающих, но прежде всего — ряд общих замечаний.

RT-11 — довольно развитая операционная система, способная работать на различных компьютерах. Она имеет собственный командный монитор и работает в диалоговом режиме (сообщения и запросы выводятся на английском языке). Система выполняет более 40 команд монитора, которые, в свою очередь, могут быть расширены с помощью ключей, уточнены или объединены для выполнения групповых операций. В ОС RT-11 могут быть созданы исполняемые командные файлы. Способ хранения информации — блочный, загрузочный сектор и директорий находятся на нулевой дорожке. Максимальная длина имени файла вместе с расширением — 9 символов, дата и время его создания отображаются в директории. При записи и копировании одноименные файлы записываются вместо уже существующих без перезапроса. Длина файла неограничена, несколько файлов могут быть объединены в один. При копировании на магнитофон начальный адрес файла становится равным 40000. Система может работать в режимах 32, 64 или 80 символов в строке.

Главный (и решающий) недостаток этой системы состоит в том, что на БК она работает очень ненадежно: постоянно происсходят различные сбои и зависание системы, а также повреждения нулевой дорожки с загрузочным сектором и директорием. Этот недостаток не устранен и в модернизированных версиях. Редко кому удается с первого раза переписать целиком дискету по файлам или секторам, особенно если она была записана на другом дисководе. При этом процесс копирования прерывается при первой же ошибке считывания или записи вместо того чтобы предложить пользователю выбор: повторить операцию, игнорировать ошибку или прервать копирование. Правда, в системе есть возможность установить режим игнорирования ошибок, но тогда при ошибках чтения они окажутся в записанной копии, а при ошибке записи выполнение операции все равно будет прекращено без предупреждений.

Кроме этого, RT-11 по сравнению с другими, более надежными, системами работает очень медленно, несмотря на то, что двигатели дисковода для ускорения работы не отключаются долгое время после выполнения операции. (Такое решение, имеющее лишь некоторое значение при отдельных опе-

* Под обозначением «БК-0011(М)» следует понимать обе модификации — БК-0011 и БК-0011М.

рациях и только для «тихоходных» дисководов, не только сохранилось в модернизированных версиях, но неожиданно оказалось примененным в новой версии MKDOS'а.)

Система состоит из основного файла и ряда дополнительных, подгружаемых по мере необходимости. Такое построение используется довольно часто, но у многих пользователей вызывает нарекания: если у вас один дисковод, то все эти файлы приходится хранить на каждой дискете. (Даже для того чтобы считать директорий дискеты, на ней должен быть соответствующий системный файл.) Большинство других систем на БК состоят из одного (основного) файла, который после загрузки находится в ОЗУ, постоянное же наличие в дисководе системной дискеты при этом не требуется.

Новые версии RT-11 (они имеют русское обозначение «ОС БК-11», но диалог все равно осуществляется на английском языке) мало отличаются от исходной и также ненадежны.

Использование системы без оболочки сегодня выглядит анахронизмом. Развитая система команд удобна для программистов, но для большинства пользователей она только затрудняет использование системы, так как эти команды необходимо запомнить и каждый раз вводить с клавиатуры. Другое дело, когда система имеет развитую оболочку для работы с файлами. В этом случае все потребности удовлетворяются простым манипулированием курсором, а вся умственная энергия пользователя полностью направляется на работу с программой. Для четвертой версии ОС разработана «нортон»–подобная оболочка InterCommander (см. выпуск № 1 за 1993 г.), которая делает работу более удобной, но по своим возможностям значительно уступает аналогичным оболочкам в других системах, а из-за низкой надежности самой RT-11 преимущества использования подобной оболочки невелики. Оболочка InterCommander, в соответствии с «традициями» системы, состоит из шести файлов, в то время как в других ОС для этого вполне хватает одного основного и (в некоторых случаях) одного или двух дополнительных. Более того, все преимущества оболочки сводятся на нет тем, что она защищена от перезаписи на другую дискету. Конечно, эта защита — мера вынужденная (дабы хоть как-то защититься от пиратства), но для добросовестных пользователей это обернулось весьма значительным неудобством. Тем же, кто пользуется RT-11 для работы с текстовыми файлами или программирования либо использует программное обеспечение от БК-0010(.01) (а последних большинство), наличие оболочки практически не поможет в работе, так как она будет осуществляться в редакторе или в среде БК-0010(.01), где оболочка не действует.

Слишком низкая надежность рассматриваемой системы не позволяет рекомендовать ее для работы на БК, особенно для начинающих пользователей, несмотря на ряд ее потенциальных достоинств. На этом можно было бы поставить точку, тем более что для начинающих система, подобная RT-11, слишком сложна. (Точнее говоря, система RT-11 ничуть не сложнее, чем, например, MS–DOC для IBM PC или операционные системы для других компьютеров. Но ее изучение целесообразно для программистов и совсем не обязательно для пользователей.)

Как же быть тем, кто вынужден работать с этой системой? Проще всего приобрести другую систему. Сложнее дело обстоит для тех, кого привлекли возможности RT-11, отсутствующие в других ОС. Мне удалось выделить только пять преимуществ RT-11, имеющих значение для пользователей: обмен программами с другими типами компьютеров, использующими RT-11; работа с локальной сетью; работа с длинными файлами; использование некоторых программ, отсутствующих на БК в других системах, и датирование файлов.

Что касается обмена с другими компьютерами, использующими RT-11, следует иметь в виду, что значительную часть программ с них все равно не удастся запустить на БК. Возможность же работы с локальной сетью имеет значение только для школ, для индивидуальных пользователей это несущественно. Работа с длинными файлами имеет особое значение для текстовых файлов. Поясним это на примере текстового редактора EDIK, рассчитанного на работу в среде RT-11. Используя возможности операционной системы, он позволяет заранее выделять под файл необходимую область на дискете, своевременно автоматически записывать создаваемый текстовый файл, размеры которого выходят за пределы отведенной под него оперативной памяти, а также выводить на экран и редактировать одновременно два текстовых файла. (Но даже в тех случаях, когда эти преимущества являются решающими, лучше работать в текстовом редакторе EDASP, не говоря уже о BIGRED или Vortex!, чем в устаревшем EDIK в ненадежной системе. Даже последняя версия EDIKM не может по основным показателям сравниться с редакторами Vortex! или BIGRED.)

Из программ, разработанных только для RT-11, имеют особое значение языки программирования ФОРТРАН и ПАСКАЛЬ. Остальные программы, которые существуют только для RT-11, — это старые, значительно уступающие аналогичным на БК-0010(.01). Они, как правило, не имеют описаний, без которых использовать их практически невозможно. Что, например, можно сделать с программой, если после ее запуска на экране появляется только элементарное приглашение (обычно знак «*») и отсутствуют меню и подсказки? А ведь по такому принципу построены все программы в RT-11. В то же время уровень программирования, достигнутый сейчас на БК в других ОС, достаточно высок, и большинство современных программ имеют дружественный интерфейс, благодаря которому работа с ними даже без описания не представляет трудностей, тогда как для многих программ в RT-11 неизвестно даже их назначение.

Что же касается последнего из отмеченных преимуществ — датирования файлов, то это хотя и приятная мелочь, но большинство пользователей лучше

обойдется без нее или введет иной способ маркировки своих файлов, чем станет ради этого рисковать их потерей (тем более что в новой версии ANDOS предусмотрено и датирование).

И, наконец, несколько советов начинающим. Вообще-то эти советы должны были бы быть в руководстве к компьютеру, но их там нет, а, кроме того, очень часто у пользователей (особенно в школе) по различным причинам не оказывается никакой документации.

- Прежде чем загружать дискету с операционной системой, желательно защитить ее нулевую дорожку с помощью утилиты DIRKEEP2.MVI (см. № 1 за 1993 г.) и целиком скопировать дискету (сделать это лучше с помощью соответствующих копировщиков в других системах). Если же такой возможности нет, то хотя бы заклейте вырез справа на дискете (защита от записи) с помощью специальных наклеек, продающихся вместе с дискетами, или любой липкой непрозрачной лентой. Это спасет дискету от повреждений при загрузке или при ошибочных командах, но не оставляйте ее в дисководе при его включении или выключении — в эти моменты может быть повреждена даже дискета с заклеенным вырезом!
- Загрузка дискеты на БК-0011М осуществляется автоматически после включения компьютера и дисковода. Однако, если компьютер после включения вышел в БЕЙСИК, то его нужно перевести в монитор командой MONIT (не забудьте подтвердить команду нажатием клавиши «ВВОД». После выхода в монитор достаточно нажать клавишу «В», и дискета начнет загружаться. (На БК-0011 надо набрать команду 160000G «ВВОД».) При успешной загрузке появится запрос даты (в некоторых версиях система сразу выйдет в командный монитор ОС). Дважды нажмите «ВВОД», и через некоторое время появится сообщение об ошибке (из-за того, что дискета заклеена), на которое не надо обращать внимание, и приглашение к диалогу. В различных версиях ОС БК-11 это приглашение выглядит по-разному, но вы его легко узнаете. Чтобы полностью снять сомнения в правильной загрузке системы, наберите команду DIR SY: «ВВОД» или DIR BY0: «ВВОД», после которой на экране должен появиться список файлов, содержащихся на этой дискете. Если он появился, то первый шаг сделан и все остальное зависит от вашей сообразительности. Описывать подробно дальнейшие шаги здесь не представляется возможным. Но обязательно попытайтесь (теперь уже с помощью средств RT-11) в первую очередь сделать копию системной дискеты на ДРУГУЮ дискету, предварительно отформатированную и инициализированную.

AO-DOS (v.1.76)

Эта операционная система совместима с MicroDOS, похожа на нее по своей структуре и внешнему виду командного монитора, имеет во многом одинаковые команды, но более надежна и в ее комплект входит «нортон»-подобная оболочка DOS-SHELL. Директорий дискеты дополнительно записывается на дублирующую дорожку. Он может содержать 100 имен файлов (в MicroDOS до 200). В AO-DOS, так же как и в MicroDOS, могут быть созданы исполняемые BAT-файлы, но одноименные файлы переписываются вместо прежних без перезапроса. Так же как и в MDOS, можно загрузить файл с указанного адреса или записать требуемую область памяти, но кроме этого имеются дополнительные возможности работы с памятью (просмотр и редактирование). При групповых операциях копирования системой самостоятельно (без перезапроса) исключаются из намеченного пользователем списка те файлы, при чтении которых произошел сбой. Максимальная длина имени файла 14 символов, наименьший адрес загрузки — 400. Все команды системы, в том числе форматирование и инициализация, а также редактирование дискеты и работа с памятью, включены в основной файл. Возможность перезаписи файлов с магнитофона на дискету не предусмотрена, но может быть осуществлена с помощью отдельного копировщика. В системе поддерживается работа с электронным диском (RAM disk), образуемым в области оперативной памяти компьютера.

RAM disk — это область оперативной памяти компьютера, выделенная для временного хранения файлов, с которой можно работать как с диском. В описаниях к операционным системам для БК электронный диск часто называют виртуальным диском (Virtual disk). Ошибки в этом нет, поскольку виртуальный диск — это более широкое понятие. Но виртуальный диск может быть образован и на физическом диске. Поскольку на БК уже существует и то и другое, во избежание недоразумений виртуальный диск, образованный в области оперативной памяти, будем называть электронным диском, а образованный на дискете — логическим.

В электронном диске до выключения питания компьютера могут храниться файлы, с которыми можно выполнять те же операции, что и на дискете. Использование электронного диска удобно при перезаписи программ на одном дисководе, а также для временного хранения файлов, используемых в данном сеансе работы, особенно тех, которые не могут работать резидентно.

В системе AO-DOS не предусмотрена процедура создания загружаемых дискет (отсутствует соответствующая команда или утилита для этих целей), хотя такие дискеты без труда могут быть созданы опытным пользователем. (Кстати, система снабжена лишь краткой документацией без рекомендаций для начинающих.)

«Нортон»-подобная оболочка DOS-SHELL, содержащаяся в отдельном файле и подгружаемая с дискеты, позволяет выводить файлы в одной панели в одну или две колонки, имеет подгружаемый редактор и меню пользователя, позволяет сортировать файлы по имени или размеру, но по своим воз-

можностям значительно уступает аналогичным оболочкам в ANDOS и MKDOS.

В целом операционная система AO-DOS хотя и превосходит по возможностям MicroDOS (если не считать отсутствия перезапроса при записи одноименых файлов и вдвое меньшего объема каталога), но уступает таким системам как ANDOS, MKDOS и NORD. Она плохо поддержана утилитами, в ней нет выхода к штатному БЕЙСИКу и не обеспечена связь с форматом дисков IBM PC, которую, однако, можно осуществить через другие системы и их утилиты. Рассмотренная система полезна тем, кто по каким-то причинам предпочитает работать в формате MicroDOS и кому при этом необходим командный монитор.

MicroDOS

Работа этой системы на БК-0011(М) ничем не отличается от таковой на БК-0010(.01). Послужив стандартом для многих других систем, она в настоящее время не имеет каких-либо существенных преимуществ перед AO-DOS.

NORTON

Работа этой ОС на БК-0011(М) также не отличается от функционирования на БК-0010(.01), за исключением отдельных версий, в которых на БК-0010(.01) не работает функция записи файлов с дискеты на магнитофон. Эта система также устарела, но в виде файла (как утилиту) ее удобно держать на дискетах вместе с AO-DOS, MKDOS и NORD, используя иногда для перезаписи файлов с магнитофона на диск и с диска на магнитофон.

Кроме того, с помощью этой системы можно оценить корректность директория дискеты. Дело в том, что системы, работающие в формате MicroDOS, имеют некоторые отличия в его структуре. После удаления файлов, записи на их место других и т. п. при просмотре дискеты в другой (совместимой) системе директорий может перестать считываться совсем или частично. Чтобы не проверять такую дискету во всех системах, достаточно проверить читаемость директория в NORTON.

NORD-11

Для БК-0011(М) выпускается специальная версия NORD-11 v2.16, отличающаяся от аналогичной для БК-0010(.01) (NORD-10 v2.16) набором утилит, а также возможностью работы с электронным диском и со штатным БЕЙСИКом БК-0011М. Сегодня в версию 2.16 (без изменения ее номера) внесены некоторые изменения, улучшившие работу с файлами данных на ФОКАЛе и с утилитами, обеспечивающими связь NORD с ANDOS и MS-DOS на IBM PC. Для БК-0011(М) выпущена также версия NORD-11 v2.17, предназначенная для совместной работы с базой данных, которая продается вместе с этой системой (работает только в ней), сама же операционная система от версии 2.16 ничем не отличается.

NORD позволяет работать не только с поддиректориями и электронным диском, но и с логическими дисками, образованными на дискете. Их количество на одной дискете или на разных ограничено только числом незанятых алфавитных клавиш (до 20). К этим дискам можно программно обращаться как к самостоятельным дисководам, в их директории (содержащем до 200 имен файлов) также могут быть образованы поддиректории, в остальном же эти диски для пользователя ничем не отличаются от обычных поддиректориев.

По ряду технических причин не каждая дискета, отформатированная на БК в формате MS-DOS, будет читаться на IBM PC, и наоборот. В комплект NORD входит несколько утилит, по-разному подготавливающих дискеты формата MS-DOS. Одна из этих утилит может быть также использована для перезаписи файлов из ANDOS в NORD (напоминаю, что формат записи NORD совместим с форматом MicroDOS).

В NORD достаточно развитая, но несколько уступающая AO-DOS система команд для создания исполняемых BAT-файлов. NORD наиболее полно поддерживает ФОКАЛ, в числе входящих в ее комплект различных утилит и программ имеется и компилятор ФОКАЛа.

NORD имеет достаточное количество поклонников, но в целом она продолжает производить впечатление некоторой незаконченности. Конечно, речь идет о мелочах, но когда таких мелочей много, то в целом хорошая работа выглядит неряшливо, особенно на фоне постоянно улучшающихся других операционных систем для БК. (Авторское описание ОС NORD с точки зрения программиста приведено в статье разработчика этой системы А. Г. Прудковского, опубликованной в журнале «Персональный компьютер БК-0010—БК-0011М» № 2 за 1994 г. Кроме того, практически закончена разработка третьей версии ОС NORD с языком программирования ПАСКАЛЬ, описание которой будет опубликовано в одном из следующих номеров. — *Прим. ред.*)

DOSB10 (v2.0)

В настоящее время разработана новая версия (2.0) этой системы, значительно отличающаяся от предыдущей и пригодная для работы на БК-0011(М). В ней реализована возможность работы с поддиректориями, с электронным диском и с логическими дисками. Кроме того, реализована возможность создания и запуска исполняемых командных файлов. В комплект входят утилиты, позволяющие обмениваться (в обоих направлениях) файлами, записанными в форматах RT-11, MicroDOS и ANDOS. Сообщения системы по выбору пользователя могут выдаваться на английском или на русском языке. Все выбранные установки запоминаются системой (в том числе шрифты и количество символов в строке — 32 или 64). В дальнейшем предполагается заменить оболочку системы на новую со значительно расширенными возможностями.

DOSB10 имеет развитой командный монитор, часть команд которого включена в основной файл

системы, а другая реализуется с помощью подгружаемых утилит. По идеологии построения эта ОС в значительной мере напоминает RT-11, но ее работа на БК более надежна. В то же время она хотя и не столь медлительна, как RT-11, но по оперативности работы с диском значительно уступает другим системам. Утилиты, которые входят в комплект DOSB10 (также как и аналогичные в комплекте RT-11), не только не имеют даже скромного оформления, но для их загрузки и работы с ними необходимо использовать команды и параметры, набираемые с клавиатуры.

Тем не менее следует отметить, что эти замечания нисколько не умаляют ту огромную и плодотворную работу, которую пришлось выполнить разработчику DOSB10. В то же время ориентация на более современные решения в ряде случаев могла бы сделать работу с системой более удобной для пользователей. Впрочем, в ее комплект входят исходные тексты утилит на ассемблере, которые помогут пользователям, владеющим ассемблером, доработать эти утилиты.

В DOSB10, как и в большинстве рассматриваемых систем, имеются значительные возможности для ее улучшения, но (на момент написания этой статьи) она уступает таким системам как ANDOS и MKDOS, в основном за счет более развитых оболочек и утилит последних. В то же время эта система будет полезна тем, для кого необходим развитой командный монитор и возможность обмена файлами с RT-11, а также занимающимся копированием файлов с магнитофона на дискету и обратно, поскольку это единственная система из числа работающих на БК-0011(М), которая допускает 16 символов в имени файлов (т.е. как на магнитофоне), и, следовательно, никакое имя файла не исказится при копировании. Средства же копирования файлов, предоставляемые этой системой, вполне удовлетворительны. Имеется также возможность при копировании файлов с диска на магнитофон устанавливать на БК-0011(М) формат записи, принятый на БК-0010(.01), чего нет в других системах. Через DOSB10 наиболее удобно осуществлять связь практически со всеми операционными системами, существующими на БК. (Авторское описание новой версии DOSB10 читайте в № 2 за 1994 г. — *Прим. ред.*)

ANDOS

Эта система неоднократно упоминалась (в частности, в журнале «Персональный компьютер БК-0010 — БК-0011М» № 1 за 1993 г. в статье автора ОС ANDOS А. М. Надежина — *Прим. ред.*), и, как ранее было отмечено, она наиболее примитивна по набору команд, но и наиболее надежна из существующих. Имеется много программ, которые разработаны специально для нее. С точки зрения пользователя функционирование этой системы на БК-0011(М) практически ничем не отличается от БК-0010(.01), за исключением возможности работы с электронным диском и штатным БЕЙСИКом БК-0011М.

Чтобы не повторять все, что ранее говорилось об ANDOS, отметим лишь кратко совершенно новые возможности, которые предоставлены пользователю в версии ANDOS 2.50 после появления «нортон»-подобной оболочки Disk MASTER, особенно для работы с файлами. Путем элементарных манипуляций все файлы или только их часть, находящаяся ниже курсора, могут быть не только рассортированы в алфавитном порядке по именам или расширениям, но и перемещены в любое место директория, выбранное пользователем. Все эти изменения могут быть временными (до перезагрузки дискеты) или зафиксированными постоянно. Таким образом, файлы могут быть размещены в любом удобном для пользователя порядке. Образованные группы файлов можно вызывать на экран, используя фильтр (шаблон имени и расширения). Курсор всегда возвращается на файл, с которым осуществлялись какие-либо операции. В оболочке возможно листание директория, а также быстрое перемещение курсора в его начало или конец. Кроме того, при групповой операции последовательность копирования файлов может быть выбрана пользователем вне зависимости от их реального размещения в директории. (Еще одно, не столь очевидное на первый взгляд, удобство, предоставленное оболочкой Disk MASTER, в отличие от SHELL, состоит в следующем. Операция удаления группы файлов выполняется за один прием (при одном обращении к дискете), а не отдельно для каждого файла. — *Прим. ред.)*

Дополнительные удобства обеспечены также за счет введения пользовательского меню, что вместе с программируемыми ключами частично компенсирует отсутствующую в ANDOS возможность создания исполняемых BAT-файлов. В оболочке можно сравнивать директории дискет, просматривать дискеты в формате MicroDOS или текстовые файлы на них, а также переписывать файлы в формат ANDOS или запускать их. Однако загружать текстовые файлы с дискет в формате MicroDOS в редакторы нельзя. Следует также обратить внимание на то, что некоторые многофайловые программы (а иногда и однофайловые с защитой имени) при перезаписи из формата MicroDOS в ANDOS могут перестать запускаться. Такие файлы рекомендуется переписывать через магнитофон. (Ряд других дополнительных удобств в ANDOS обеспечивается с помощью резидентного драйвера ANIRAM – MIRIADA. Подробности об этом драйвере см. журнал «Персональный компьютер БК-0010 — БК0011М» № 1 за 1994 г., стр. 33, а также на второй стороне обложки. — *Прим. ред.*)

На момент написания статьи Disk MASTER — это лучшая оболочка для работы с файлами на БК, превосходящая другие оболочки практически по всем показателям и во многом компенсирующая отсутствующую пока в ANDOS возможность работы с поддиректориями.

Что касается утилит, в версию 2.50 входит модернизированный форматировщик, работающий в турборежиме и использующий наиболее грамотный

алгоритм форматирования. Некоторая некорректность при инициализации дискет, имевшая место в предыдущей версии, устранена, все операции осуществляются достаточно надежно и быстро.

К сожалению, из комплекта версии 2.50 исключена оболочка SHELL. Конечно, по своим возможностям она уступает Disk MASTER, но с ее помощью в ряде случаев удобно осуществлять обмен файлами с магнитофоном (тогда как работа с ним в Disk MASTER'е вообще не предусмотрена!). Кроме того, в ANDOS до сих пор не реализована иногда необходимая возможность записи области памяти в файл.

В настоящее время готовится к выпуску давно обещанная третья версия ANDOS, работающая с поддиректориями (к сожалению, по данным, полученным от А. М. Надежина, формат поддиректорий в ней не совместим с таковым в MS-DOS. — *Прим. ред.*). Предполагается, что в ней будет сохранено все то, что имеется в оболочке Disk MASTER.

MKDOS

Сейчас в продажу поступила версия MKDOS 2.0, существенно отличающаяся от предшествующих. Ее принципиальное отличие заключается в возможности работы с поддиректориями, кроме того, в комплект входит новая «нортон»-подобная оболочка MCommander.

При загрузке на БК-0011(М) автоматически формируется электронный диск, в который помещается командный файл, позволяющий создавать простейшие (по набору команд) исполняемые файлы.

Форматирование осуществляется с помощью утилиты, в которой впервые на БК применены решения, позволившие значительно увеличить скорость форматирования. В настоящее время это наиболее быстрый форматировщик на БК. Дополнительно в нем имеется возможность выбора одного из трех алгоритмов форматирования, а также пометки поврежденных блоков, если используется некачественная дискета.

При работе с двумя дисководами копирование файлов также осуществляется наиболее быстро, при этом операция выполняется корректно, при копировании одноименных файлов производится перезапрос, а при ошибках чтения или записи предлагается возможность повторить операцию, продолжить или отказаться от ее выполнения.

Эта система наиболее надежна по сравнению с другими, работающими в формате MicroDOS, и имеет примерно одинаковую надежность с ANDOS. (Сравнить надежность MKDOS и ANDOS довольно сложно: повреждение дискет в этих системах происходит очень редко и не всегда ясно, что явилось причиной этого — случайный сбой дисковода, дефект дискеты или некорректность работы системы. В этих системах наиболее правильно и быстро работают форматировщики, быстро и без ошибок происходит переименование и удаление файлов, а также их копирование.) Некоторые огрехи, которые имеются в версии MKDOS 2.0, в основном обусловлены недоработками оболочки и вскоре наверняка будут устранены, поскольку эта система постоянно развивается. (Например, за время написания статьи уже появилась откорректированная версия оболочки v2.01). Предполагается также включение в систему дополнительных утилит, расширяющих ее возможности.

MKDOS и ANDOS — это две конкурирующие между собой системы, претендующие, несмотря на отсутствие в них обеих командного монитора, на вытеснение остальных систем. Очевидно, для того, чтобы это стало реальностью, обе эти системы должны превосходить остальные по всем показателям, или, по крайней мере, по наиболее важным. Однако определить такой перечень наиболее важных показателей практически невозможно: всегда найдутся люди, для которых приоритеты ценностей будут иными. Даже, казалось бы, такой важный показатель, как надежность системы, для кого-то может оказаться менее существенным, чем какой-либо другой. Поэтому ниже, в дополнение к сказанному ранее, будет приведено подробное сравнение этих ОС (точнее не самих систем, а их оболочек и утилит) между собой и с другими системами.

Отсутствие командного монитора в обеих системах частично компенсируется развитыми оболочками и дополнительными утилитами, в которых можно сделать практически то же, что и в командных мониторах, но более простыми методами. Пожалуй, единственное, что может иметь значение для многих пользователей, — это отсутствие возможности записать в файл область памяти. Такая необходимость возникает, например, при зависании текстового редактора. Для сохранения текста в этом случае можно было бы записать в файл область памяти, в которой он находится.

Поддиректории, которые используются в NORD и MKDOS, не являются настоящими поддиректориями в принятом значении этого термина: все имена файлов все равно хранятся в основном директории и в разных поддиректориях нельзя хранить одноименные файлы. Если выйти из оболочки и просмотреть директорий диска, то будут показаны все файлы в том порядке, в котором они были записаны на дискету. Структура же логического диска, принятая в NORD, позволяя увеличить допустимое количество имен файлов на диске, делает его более похожим на диск с «настоящими» поддиректориями. Файлы на логическом диске можно увидеть, только войдя в его директорий. Но, с другой стороны, 200 имен в каталоге для дискеты емкостью 800 Кб вполне достаточно, так как в реальной ситуации переполнение такого директория маловероятно. Даже на системной дискете NORD, на которой помещено много коротких утилит и подпрограмм, 160 файлов целиком заполнили дискету. В ANDOS же, директорий которого допускает не более 112 имен файлов, его переполнение — достаточно частое явление при программировании и работе с текстовыми файлами.

С помощью поддиректориев облегчается сортировка файлов: группы файлов, объединенные по какому-нибудь признаку, отделяются друг от друга, доступ к ним становится более удобным (но не всегда, так как при этом требуется указызывать путь доступа к файлу), а их наличие легко поддается визуальному контролю. Но почти такой же эффект может быть достигнут за счет развитых средств сортировки файлов, предоставляемых оболочками, причем в этом случае не требуется указывать путь доступа к файлу.

В таблице приведено сравнение оболочек в системах ANDOS, MKDOS, NORD и AO-DOS. Из нее видно, что наименьшие возможности сортировки имен файлов предоставляет оболочка NORD. Что же касается оболочек MKDOS и ANDOS, то здесь сравнение неоднозначно: оболочка MKDOS по одному показателю превосходит оболочку ANDOS (наличие поддиректориев), но уступает по пяти другим. (Здесь, конечно, следует иметь в виду, что в таблице невозможно указать все детальные отличия оболочек.) Эти последние с учетом возможностей, предоставляемых пользовательским меню, которое имеется в оболочке ANDOS, могут оказаться более значимыми, чем отсутствующая возможность работы с поддиректориями. Особенно важным оказалось свободное перемещение имен файлов в директории и возможность их сортировки ниже курсора, а также закрепление на дискете сделанных изменений. Кроме того, хотя в оболочке MKDOS и имеется функция пользовательского меню, но по своей сути оно таковым не является. Через это меню может быть вызвана только подпрограмма (драйвер), позволяющая просматривать директории дискет ANDOS, а также копировать с них файлы, если они не фрагментированы (т.е. записаны подряд), на дискеты MKDOS и просматривать текстовые файлы.

В оболочке же ANDOS файлы с дискеты в формате MKDOS (MicroDOS) не только могут быть просмотрены и скопированы без подгрузки дополнительных подпрограмм, но и запущены. А за счет пользовательского меню могут загружаться любые программы и выполняться некоторые команды. Например, можно установить курсор на текстовый файл и через пользовательское меню загрузить его в текстовый редактор (имя выбранного файла автоматически подставляется в запрограммированную в меню команду вместо шаблона).

В MKDOS пока отсутствует возможность подготовки дискет в формате MS-DOS, как это сделано в NORD или ANDOS. В то же время у MKDOS по сравнению с ANDOS имеется преимущество, которое для многих покажется существенным: почти неискаженные имена файлов и их последовательный способ записи.

Дополнение

За время подготовки статьи к печати произошли некоторые изменения. Так, в настоящее время поступила в продажу давно ожидаемая новая версия

Сравнение пользовательских характеристик оболочек в системах ANDOS, MKDOS, NORD, AO-DOS

Параметр	Оболочки			
	DM	MC	NORD	D-SHELL
Поддиректории	–	+	+	–
Фильтр	+	+	–	–
Сравнение директориев	+	+	–	–
Сортировка всех имен файлов или их расширений по алфавиту	+	+	–	+
Сортировка имен файлов, расположенных ниже курсора,(или их расширений) по алфавиту	+	—	–	–
Перемещение имен файлов в директории	+	—	–	–
Возврат курсора на прежнее место после завершения операций или выхода из программы	+	–	–	+
Закрепление изменений в директории	+	–	–	–
Листание директория	+	+	+	+
Быстрое перемещение в конец и начало директория	+	+	–	–
Две колонки в каждой панели	–	–	–	+
Меню пользователя	+	–	+	+
Перемещение курсора	Быстрое	Быстрое	Медленное	Среднее

Примечание. Знаком «+» отмечено наличие параметра, а знаком «–» — отсутствие.

ANDOS — v.3.1, в которой устранены некоторые ошибки и недоработки. Важно отметить, что авторы этой системы наиболее тщательно следят за корректностью ее работы и поэтому она поступает в продажу наиболее доработанной.

В новой версии ANDOS сохранены абсолютно все преимущества предыдущей и достигнуты новые. Среди последних отметим лишь следующие: работа с поддиректориями, возможность просмотра и работы со всем директорием диска без учета поддиректориев, датирование диска и файлов, возможность сортировки файлов по датам их создания и работа с «флайером» (в прикладной программе, например, в текстовом редакторе, можно вызвать директорий диска и, перемещая по нему курсор, выбрать нужный файл для загрузки).

К недостаткам новой версии можно отнести то, что в ней не работает часть программ, написанных для v.2.5, а также неоправданно сложный алгоритм создания системной дискеты: необходимо вначале отформатировать или инициализировать дискету с помощью утилиты ANFORMAT, затем выйти из нее и переписать основной файл системы, а потом вновь загрузить утилиту ANFORMAT и создать системную дискету. И наконец, еще одна неожиданность не из ряда приятных: все сообщения системы стали выводиться на английском языке.

Что же касается MKDOS, то уже в его ранних версиях была замечена некоторая некорректность работы с директорием диска, о которой ранее не упоминалось в надежде, что авторы ее устранят. Однако они не придали этому значения, что привело к значительным погрешностям в версии 2.10.

После удаления файлов, их перемещения по поддиректориям и записи новых файлов директорий дискеты искажается: в нем появляются «фиктивные» файлы с нелепыми именами и адресами, а также возможно «самодеятельное» их помечивание как выделенных. Директорий может по-разному выглядеть в различных окнах оболочки или вне оболочки и принимать другой вид в зависимости от способа перезагрузки дискеты. Последние могут также не читаться в ОС NORTON, а в различных эмуляторах MDOS — читаться с различными искажениями. Дальнейшая работа с такой дискетой часто становится невозможной. Все это ставит надежность системы MKDOS значительно ниже, чем ANDOS.

Замечено и еще одно неприятное явление, когда из программы, загруженной с левого окна оболочки, выход происходит в правое окно и туда же перемещается директорий из левого окна, или же курсор не возвращается на прежнее место. Предполагается, что к моменту опубликования статьи будет разработана новая версия MKDOS (и оболочки), в которой отмеченные ошибки будут устранены и введены дополнительные преимущесства для пользователей. Но пока ANDOS лидирует, и, если бы не всегда удобное изменение имени файла и отсутствие возможности записи содержимого памяти в виде файла, об остальных системах можно было бы забыть.

И, наконец, еще одно важное замечание. При работе с различными системами, поддерживающими формат MDOS, было обнаружено, что некоторые файлы на дискетах автора статьи, имеющие нечетную длину, перестали запускаться. Это произошло из-за их перемещения во время копирования на один байт вверх. Сделала ли это какая-либо система (MDOS, NORTON, NORD или MKDOS) или в этом повинен какой-то копировщик — установить не удалось. Исправить ошибку можно различными способами, но проще всего сделать это с помощью копировщика CF50, входящего в комплект MKDOS (а также распространяемого как отдельная программа). Для этого необходимо загрузить в него файл, средствами копировщика переместить его на один байт вниз, удалить лишний байт, а затем вновь записать файл на диск. Например, для адреса загрузки 1000 нужно указать адрес перемещения 777, а затем, используя функцию копировщика для снятия автозапуска, удалить лишний байт. Если файл имеет автозапуск и его адрес загрузки, например, равен 760, то при перемещении нужно указать адрес равным 757. Перед записью файла автозапуск лучше удалить, а если он необходим, то установить его повторно средствами копировщика. (Поскольку при перемещении файлов вверх может потеряться последний байт, то принципиально возможно, что некоторые программы после их перемещения вниз все равно не будут работать, но это маловероятно.)

Заключение

В настоящее время для БК-0011(М) существует единственная профессиональная операционная система RT-11, но она в основном устарела и по многим показателям уступает любительским системам, разработанным прежде всего для БК-0010(.01). Среди них лидируют ANDOS и MKDOS, работающие в разных форматах, но допускающие обмен файлами между ними. Эти системы продолжают совершенствоваться, но уже сейчас они способны удовлетворить большинство пользователей. Обе они работают как на БК-0010(.01), так и на БК-0011(М). В состав этих систем входит документация, достаточная для их освоения даже неподготовленным пользователем. Потенциальная возможность изменения ситуации имеется также для DOSB10 при ее значительном совершенствовании (повышение быстродействия, замена оболочки и упрощение утилит).

Такие системы как KMON, TurboDOS, MicroDOS и NORTON полностью вытеснены другими. А системы NORD, AO-DOS, DOSB10 и RT-11 (ОС БК11) могут использоваться только при необходимости применить функцию, не реализованную в других системах.

ГЛЮКАДЕМИЯ

Если бы строители строили здания так же, как программисты пишут программы, первый залетевший дятел разрушил бы цивилизацию.

Из «Законов Мерфи»

Слово «глюк» в сознании нынешних хакеров — заядлых компьютероманов ассоциируется отнюдь не с фамилией известного австрийского композитора. На жаргоне пользователей ЭВМ и программистов оно обозначает ошибки, время от времени обнаруживаемые в общераспространенных программах. Причиной подобных «мелких и не очень» пакостей чаще всего является невнимательность разработчиков ПО, не утруждающих себя тщательным тестированием своих творений. Впрочем, отчасти виноват в этом и «пиратский» способ распространения последних — как правило, это «сшитые на живую нитку» полуфабрикаты, наскоро созданные для сиюминутных и достаточно специфических целей и позже списываемые друг у друга «на халяву».

Открывая на страницах журнала рубрику «Академия глюков» (или попросту «Глюкадемия»), приглашаем всех наших читателей сообщать об ошибках, обнаруженных в достаточно широко известных программах, и о том, как эти ошибки можно устранить. Надеемся, что авторы критикуемых программ не обидятся, а, напротив, примут к сведению замечания пользователей и исправят ошибки при создании следующих версий.

Ждем ваших писем, а пока «для затравки» предлагаем короткую заметку.

ПОЛЕЗНАЯ... ОШИБКА

Текстовый редактор TED8 содержит небольшую полезную ошибку. Не удивляйтесь! Именно ошибку, и именно полезную. При наборе нового текста, если последняя введенная строка — текущая, клавиша «ВС» (удаление текущей строки) удаляет ее из текста, но не из буфера ввода. Так что если сразу же нажать клавишу «КУРСОР ВВЕРХ», стертая строка восстановится на экране. Таким образом, если при наборе нового текста вы по ошибке нажали «ВС» и строка исчезла, нужно сразу же нажать клавишу «СТРЕЛКА ВВЕРХ», и строка вновь появится на своем месте. Однако следует помнить, что этот фокус срабатывает только в том случае, если «ВС» была ошибочно нажата один раз и если после нее не нажимались другие (кроме «СТРЕЛКИ ВВЕРХ») клавиши.

«Мужчин от мальчиков отличает стоимость их игрушек»

Роберт Фрост

ИГРА «САПЕР»

Игра «САПЕР» построена по типу реализованной на IBM PC игры «МИННОЕ ПОЛЕ», однако она несколько упрощена по сравнению с прототипом: нет возможности установки флажков и не производится автоматическое открывание всех пустых клеток (без мин и цифр).

Перед вами минное поле (зеленый прямоугольник), на некоторых его квадратиках расставлены мины. В верхнем левом углу поля выводится указатель, который можно перемещать от одного края до другого (снаружи поле ограничено бордюром из таких же зеленых квадратиков, но туда указатель не допускается). Нажатием клавиши «ВВОД» можно открыть квадратик, на котором находится указатель, и тогда:

если на нем была мина, то вы взрываетесь, а игра заканчивается;

если мины нет, этот квадратик считается разминированным, и в нем выводится цифра, указывающая, сколько мин находится в соседних клетках (сверху, снизу, справа, слева и по диагоналям). Если в соседних клетках мин нет, открытый вами квадратик остается пустым.

Определяя логическим путем по выводимым цифрам местонахождения мин (а кое-где придется положиться и на фортуну), вы должны разминировать все поле (открыть все клетки, в которых нет мин) и при этом не подорваться.

Управление игрой

Управление игрой возможно одновременно с клавиатуры и мыши «Марсианка» со стандартной распайкой для БК-0010(.01).

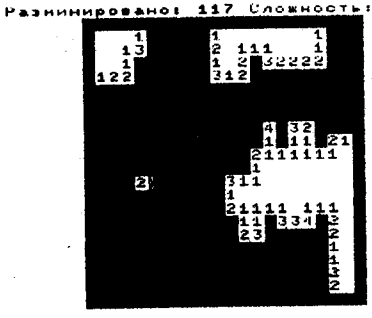
При выдаче программой запросов «PRESS ANY KEY» для продолжения работы можно нажимать любую клавишу или любую из кнопок мыши.

Движение указателя по полю производится клавишами-стрелками или с помощью мыши. Выбор (открывание) клетки — клавиша «ВВОД» или правая кнопка мыши. Клавиша «КТ» или обе кнопки мыши, нажатые одновременно, — прерывание игры. Левая кнопка мыши задействована только при ответе на «PRESS ANY KEY».

ВНИМАНИЕ! Диагональные перемещения мыши и ее движение с одновременным нажатием кнопок в данной версии не распознаются. Если вам кажется, что программа перестала реагировать на мышь, нужно вести ее ровнее. Кроме того, БЕЙСИК медлителен, и на слишком быстрые движения мыши реакция также довольно плохая.

```
 5 ' ИГРА«САПЕР»
10 DIM P%(21%,21%)
20 CLS
30 GOSUB 1470
40 FOR I%=0%TO21%
50 FOR J%=0%TO21%
60 P%(I%,J%)=0
70 NEXT J%,I%
80 FOR I%=1%TOSL%*10%+20%
90 X%=INT(RND(1%)*20%+1%)
100 Y%=INT(RND(1%)*20%+1%)
110 IF P%(X%,Y%)<>0% TH90
120 P%(X%,Y%)=1%
130 NEXT I%
140 KO%=0%
150 GOSUB 1670
160 IF INKEY$="" TH1910
170 CLS
180 FOR J%=0%TO21%
190 FOR I%=0%TO21%
200 GOSUB 820
210 NEXT I%,J%
220 XM%=1%
230 YM%=1%
240 GOSUB 850
250 CH$=INKEY$
```

```
260 ? AT(0%,0%);CHR$(147%);
    "Разминировано:";CHR$(145%);
    KO%;AT(19%,0%);CHR$(146%);
    "Сложность:";CHR$(145%);SL%;
270 IF CH$="" TH GOS650
280 IF CH$="" TH250
290 CH%=ASC(CH$)
300 IF CH%<>8% TH350
310 GOSUB 850
320 IF XM%>1% TH XM%=XM%-1%
330 GOSUB 850
340 GOTO 250
350 IF CH%<>25% TH400
360 GOSUB 850
370 IF XM%<20% TH XM%=XM%+1%
380 GOSUB 850
390 GOTO 250
400 IF CH%<>26% TH450
410 GOSUB 850
420 IF YM%>1% TH YM%=YM%-1%
430 GOSUB 850
440 GOTO 250
450 IF CH%<>27% TH500
460 GOSUB 850
470 IF YM%<20% TH YM%=YM%+1%
480 GOSUB 850
490 GOTO 250
500 IF CH%=3% TH1150
510 IF CH%<>10% TH250
520 IF P%(XM%,YM%)=1% TH1090
530 IF PEEK(17984%+(XM%+ 6%)*2%+
    (YM%+1%)*640%)=-21846%
    TH KO%=KO%+1%
540 IF KO%=(400%-(SL%*10%+20%))
    TH1270
550 SUM%=P%(XM%-1%,YM%-1%)+
    P%(XM%,YM%-1%)+
    P%(XM%+1%,YM%-1%)
560 SUM%=SUM%+P%(XM%-1%,YM%)
    +P%(XM%+1%,YM%)
570 SUM%=SUM%+P%(XM%-1%,YM%+
    1%) +P%(XM%,YM%+1%)+
    P%(XM%+1%,YM%+1%)
```




```
580 IF SUM%<>0% TH610
590 GOSUB 780
600 GOTO 250
610 CH$=CHR$(SUM%+48%)
620 GOSUB 990
630 GOTO 250
650 L%=PEEK(-52%)
660 POKE -52%,0%
670 POKE -52%,8%
680 IF L%=1% TH CH$=CHR$(26%)
690 IF L%=4% TH CH$=CHR$(27%)
700 IF L%=8% TH CH$=CHR$(8%)
710 IF L%=2% TH CH$=CHR$(25%)
720 IF L%=32% TH CH$=CHR$(32%)
730 IF L%=64% TH CH$=CHR$(10%)
740 IF L%=96% TH CH$=CHR$(3%)
750 IF L%=0% TH CH$=""
760 RETURN
780 ? AT(XM%+6%,YM%+1%);" ";
790 GOSUB 850
800 RETURN
820 ? AT(I%+6%,J%+1%);CHR$(146%);
    CHR$(156%);" ";CHR$(156%);
    CHR$(145%);
830 RETURN
850 A%=17408%+(XM%+6%)*2%+
    (YM%+1%)*640%
860 POKE A%,(PEEK(A%) XOR 2016%)
870 POKE A%+64%,(PEEK(A%+64%)
    XOR 1632%)
880 POKE A%+128%,(PEEK(A%+128%)
    XOR 384%)
890 POKE A%+192%,(PEEK(A%+192%)
    XOR 2016%)
900 POKE A%+256%,(PEEK(A%+256%)
    XOR 6552%)
910 POKE A%+320%,(PEEK(A%+320%)
    XOR 384%)
920 POKE A%+384%,(PEEK(A%+384%)
    XOR 384%)
930 POKE A%+448%,(PEEK(A%+448%)
    XOR 384%)
940 POKE A%+512%,(PEEK(A%+512%)
    XOR 2016%)
970 RETURN
990 A%=PEEK(140%)
1000 C%=21845%
1010 IF SUM%2 TH C%=-21846%
1020 IF SUM%4 TH C%=-1%
1030 POKE 140%,C%
1040 ? AT(XM%+6%,YM%+1%);CH$;
1050 POKE 140%,A%
1060 GOSUB 850
1070 RETURN
1090 ?CHR$(157%);CHR$(7%);CHR$(157%);
1100 ? AT(1%,23%);CHR$(156%);
     " Вы подорвались на мине... ";
1110 ? CHR$(156%);
1120 KY%=1%
1130 GOTO 1170
```

```
1150 KY%=0%
1160 ? AT(2%,23%); " Game aborted
     by player. ";
1170 FOR I%=1%TO20%
1180 FOR J%=1%TO20%
1190 IF P%(I%,J%)<>1% TH GOS1740
     EL1210
1200 GOTO 1220
1210 IF KY%=0% OR NOT(I%=XM%
     AND J%=YM%) TH GOS1420
     EL GOS1450
1220 NEXT J%,I%
1230 ? AT(0%,0%);CHR$(153%);
     AT(2%,0%);CHR$(146%);
     "PRESS ANY KEY,PLEASE !";
     CHR$(145%)
1240 IF INKEY$="" TH1950
1250 GOTO 1310
1270 KY%=0%
1280 BEEP
1290 ? AT(2%,23%); "Поздравляю!
     Вы выиграли !";
1300 GOTO 1170
1310 ? AT(1%,23%);CHR$(153%);
     CHR$(146%);" Сыграем еще? ";
1320 ? CHR$(147%);"<D/N> ";CHR$(145%);
1330 ? AT(0%,0%);CHR$(153%);
1340 CH$=INKEY$
1350 IF CH$="" TH1340
1360 CH%=(ASC(CH$) AND
     (NOT(160%)))
1370 IF CH%=68% TH20
1380 CLS
1390 ? AT(2%,10%);"Good-by, my boy,
     good-by..."
1400 END
1420 ? AT(I%+6%,J%+1%);"$";
1430 RETURN
1450 ?AT(I%+6%,J%+1%);
     CHR$(156%);"";CHR$(156%);
1460 RETURN
1470 GOSUB 1630
1480 ? AT(0%,3%)
1490 ? " ----------------------------------------"
1500 ? " |                                      |#"
1510 ? " |      И Г Р А                        |#"
1520 ? " |                                      |#"
1530 ? " |САПЕР (МИННОЕ ПОЛЕ)                  |#"
1540 ? " |                                      |#"
1550 ? " |";CHR$(155%);"ПРОСТЕЙШАЯ
     АДАПТАЦИЯ С IBM";
     CHR$(155%);"                    |#"
1560 ? " |                                      |#"
1570 ? " | SCREW  МОСКВА 1992                   |#"
1580 ? " ----------------------------------------#"
1590 ? " ########################################"
```

Уважаемые читатели!

Подписаться на журнал «Персональный компьютер БК-0010 — БК-0011М» можно в любом отделении связи или непосредственно в редакции. В каталоге ЦРПА «Роспечать» данные о журнале следует искать на букву «Б» — «Библиотечка журнала «Информатика и образование».

Чтобы приобрести отдельные выпуски журнала через редакцию:

- частным лицам необходимо перечислить за каждый выпуск стоимость журнала (уточнить в редакции) плюс стоимость почтовых расходов (по пересылке бандероли из Москвы в пункт назначения) и 300 руб. (орграсходы);
- предприятиям и организациям необходимо перечислить на расчетный счет редакции за каждый выпуск стоимость журнала (уточнить в редакции) плюс стоимость почтовых расходов (по пересылке бандероли из Москвы в пункт назначения) и 300 руб. (орграсходы).

Расчетный счет для Москвы и Московской области: 609602 в ММКБ филиал «Интеллект», МФО 212199, уч.1Е

Расчетный счет для других городов России и ближнего зарубежья: 609602 в ММКБ филиал «Интеллект», корр. счет 216161800 в ЦРКЦ ГУ ЦБ РФ, уч.С3, МФО 211004.

Перечисление денег необходимо подтвердить письмом с вложенной в конверт заявкой (см. на обороте) по адресу:

103051, Москва, ул.Садовая Сухаревская, д.16, комн.9
Редакция журнала «Информатика и образование».
Справки по телефону: (095) 151-19-40.
E-MAIL: mail @ infoobr.msk.sv

```
1600 GOSUB 1990
1610 ? AT(1%,20%);CHR$(155%);
     "ПОДОЖДИТЕ, ИДЕТ ИНИЦИАЛИ
     ЗАЦИЯ ИГРЫ...";CHR$(155%);
1620 RETURN
1630 ? CHR$(140%);CHR$(140%);
1640 POKE 138%,0%
1650 POKE 140%,-1%
1660 RETURN
1670 ? AT(0%,20%);CHR$(153%);
1680 ? AT(5%,15%);"--------------------"
1690 ? AT(5%,16%);"|PRESS ANY KEY! |#"
1700 ? AT(5%,17%);"--------------------#"
1710 ? AT(5%,18%);" ############"
1720 RETURN
1740 A%=17408%+(I%+6%)*2%+
     (J%+1%)*640%
1750 POKE A%,-21846%
1760 POKE A%+64%,-32766%
1770 POKE A%+128%,-32766%
1780 POKE A%+192%,-32766%
1790 POKE A%+256%,-32766%
1800 POKE A%+320%,-32766%
1810 POKE A%+384%,-32766%
1820 POKE A%+448%,-32766%
1830 POKE A%+512%,-32766%
1840 POKE A%+576%,-32766%
1850 POKE A%+640%,-21846%
1860 RETURN
1870 I%=XM%
1880 J%=YM%
1890 GOSUB 1740
1900 RETURN
1910 L%=PEEK(-52%)

1920 POKE -52%,0%
1930 POKE -52%,8%
1940 IF L%=64% OR L%=32% OR
     L%=96% TH170 EL160
1950 L%=PEEK(-52%)
1960 POKE -52%,0%
1970 POKE -52%,8%
1980 IF L%=64% OR L%=32% OR
     L%=96% TH1250 EL1240
1990 ? AT(5%,16%);CHR$(146%);
     "Сложность (0-10)=";CHR$(145%);
2000 INPUT SL%
2010 IF SL%>=0% AND SL%<11% TH RET
2020 ? AT(5%,16%);CHR$(153%);
2030 GOTO 1990
```

СОДЕРЖАНИЕ

ПЕРСОНАЛЬНЫЙ КОМПЬЮТЕР БК-0010 – БК-0011М

Главный редактор
Васильев Б. М.

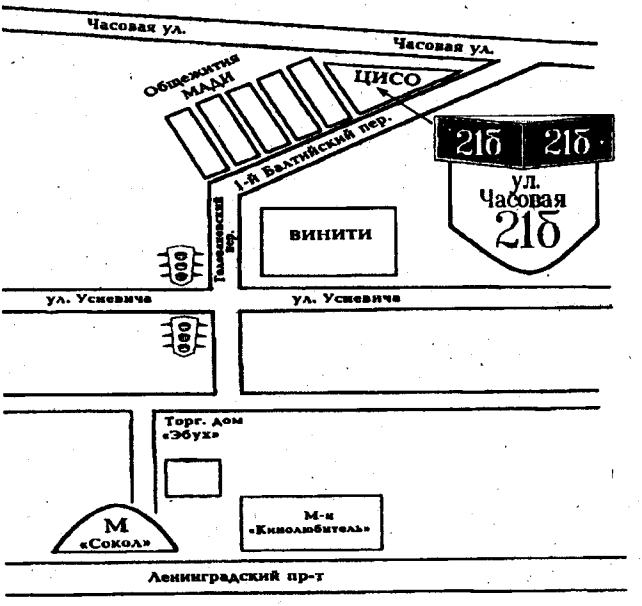
Редактор
Усенков Д. Ю.

Художник
Конюхов В. Н.

Корректор
Кириченко И. Б.

Компьютерная верстка
Лагунова Э. Е.

Наш адрес: Москва, ул. Часовая, 21Б, помещение Центра Интерактивных Средств Обучения (ЦИСО), комн.20:



Телефон редакции: 151-19-40

ПЕРСОНАЛЬНЫЙ КОМПЬЮТЕР БК-0010 - БК-0011М

Подписано в печать с оригинал-макета издательства «Информатика и образование» 09.09. 94 г.
Формат 70×100 1/16. Бумага офсетная.
Усл.печ.л. 6,5. Заказ №
Цена 4000 руб. (по подписке).
В розничной продаже цена договорная.

Ордена Трудового Красного Знамени Чеховский полиграфический комбинат Министерства печати и информации Российской Федерации. 142300, Чехов, Московской обл.

СОДЕРЖАНИЕ ВЫПУСКА 4'94