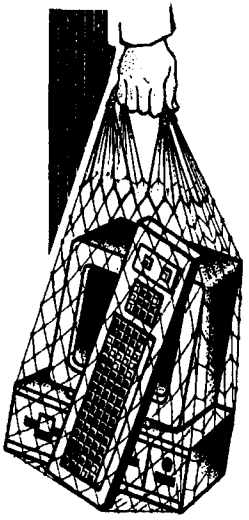
УВАЖАЕМЫЕ ЧИТАТЕЛИ!

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.

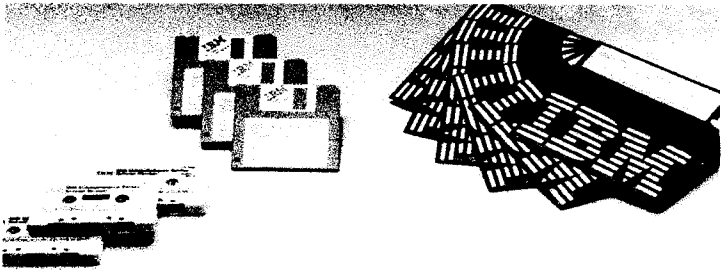
Система автоматической генерации спецификаций в соответствии с требованиями ГОСТ для ППП PCAD.

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.

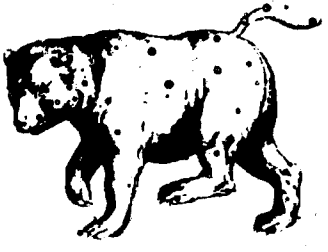


Телефон: 151-19-40
Факс: 208-67-37
E-Mail: mail@infoobr.msk.su

ПЕРСОНАЛЬНЫЙ КОМПЬЮТЕР БК-0010 БК-0011м

Приложение
к журналу
«ИНФОРМАТИКА
И ОБРАЗОВАНИЕ»
Выпуск 4'94 (5)

В НОМЕРЕ



Москва «Информатика и образование» 1994

Авторы выпуска

Аскеров Р.
Еремин Е. А.
Зальцман Ю. А.
Ивайлов П.
Котов Ю. В.
Лелейкин С. В.
Петров П. В.
Прудковский А. Г.
Рахманкулов Р. А.
Свилюк А.
Фролкин Б. Ф.
Юдин И.

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

«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— № 3 за 1994 г.

Ю. А. Зальцман,
г. Алма-Ата

МИКРОЭВМ БК-0010. АРХИТЕКТУРА И ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ АССЕМБЛЕРА

Операторы ассемблера

Операторы (команды) определяют, ЧТО программа должна делать и в какой последовательности. Операторов в составе ассемблера много, и их можно классифицировать по-разному. Самой простой и удобной для изложения, видимо, является классификация по функциям:

- вычислительные операторы (одно- и двухоперандные),
- операторы управления программой,
- операторы управления машиной,
- прочие.

При описании операторов мы будем приводить форму их записи в общем виде следующим образом. Если оператор может работать и с байтами, и со словами, то дополнение его имени для работы с байтами будет приводиться в скобках, например **MOV(B)**. Если в составе команды допустимы любые способы адресации, то операнды в общем виде будут обозначаться как **A,B** (A — ОПЕРАНД-ИСТОЧНИК, в процессе работы команды не меняющийся, а B — ОПЕРАНД-ПРИЕМНИК, куда заносится результат операции). Если один и тот же операнд является и источником, и приемником, то обозначим его как **AB**. Если в процессе выполнения команды содержимое операндов не меняется (оба они как бы являются источниками), будем обозначать их **A1,A2**. Если операнд в составе команды используется только как адрес перехода, обозначим его в общем виде как **N**. Если же описываемый оператор допускает не любые способы адресации, а лишь некоторые, это будет отмечаться особо, причем для обозначения этих способов будут использоваться уже знакомые нам символы **RN**, **X**, **MET**.

Перед тем как перейти к описанию конкретных операторов, кратко напомним, что кроме восьми регистров общего назначения в составе процессора имеется еще один служебный регистр PS или регистр СЛОВА СОСТОЯНИЯ ПРОЦЕССОРА (ССП), с которым вы уже знакомы. При описании операторов нас будут интересовать 4 младших разряда (бита) ССП — так называемые флаги УСЛОВИЙ. Вспомним, что они обозначаются буквами N, Z, V, C и значение каждого из них устанавливается равным нулю или единице в зависимости от результата выполнения очередной команды. Эти биты используются для

реализации условных переходов и для некоторых других целей. Их значение устанавливается равным единице в следующих случаях:

- N=1, если результат операции — отрицательное число;
- Z=1, если результат операции — ноль;
- V=1, если при операции произошло переполнение (т. е. перенос в знаковый разряд);
- C=1, если при операции произошел перенос (т. е. выход единицы за пределы слова или байта).

В противных случаях соответствующие биты ССП равны нулю.

1. Вычислительные операторы

С одним операндом

- CLR(B) B — ОЧИСТКА. Обнуляет слово (байт) операнда, записывая в него нуль, независимо от исходного содержимого. Используется для обнуления (сброса) программных счетчиков, признаков, сумматоров, подготовки ячеек и т. п.
- COM(B) AB — ПОРАЗРЯДНОЕ ИНВЕРТИРОВАНИЕ. Все биты операнда независимо друг от друга заменяются на противоположные (нуль на единицу, а единица — на нуль). Пусть в регистре R4 записано число 1 011 101 100 111 000 (восьмеричное 135470). После выполнения команды COM R4 в регистре R4 будет записано 0 100 010 011 000 111 (восьмеричное 042307).

Другое название данной операции — ДОПОЛНЕНИЕ ДО ЕДИНИЦЫ, так как сумма исходного значения и результата операции во всех разрядах содержит единицы: AB+COM(AB)=1 111 111 111 111 111.

- INC(B) AB — ИНКРЕМЕНТ. К операнду прибавляется 1.
- DEC(B) AB — ДЕКРЕМЕНТ. Из операнда вычитается 1.

При исполнении операторов INC и DEC, а также некоторых других (ASR, ROL, ADD, SUB и пр.) может произойти перенос за пределы слова или байта, например при выполнении последовательности команд:

```
MOV #177777,R0
INC R0
```

Однако флаг переноса (C-разряд) при этом НЕ УСТАНАВЛИВАЕТСЯ, что необходимо учитывать при использовании данных операторов вместе с операторами условного перехода.

- NEG(B) AB — ИНВЕРСИЯ ЗНАКА. Число заменяется на отрицательное, равное прежнему по абсолютной величине, но представленное в дополнительном коде.

Другое название данной операции — ДОПОЛНЕНИЕ ДО ДВУХ, так как сумма исходного числа и результата равна нулю плюс перенос «за пределы» слова (т. е. устанавливается флаг переноса): AB+NEG(AB)=0+ПЕРЕНОС.

(При использовании оператора NEG(B) значение 100000_8 заменяется самим собой, так как в записи с дополнением до двух для наибольшего отрицательного числа 100000_8 нет соответствующего положительного. — *Прим. ред.*)

- TST(B) A — ПРОВЕРКА СОДЕРЖИМОГО ОПЕРАНДА. Операнд при этом не изменяется, но биты условий ССП устанавливаются в соответствии с его содержимым. Например, если операнд равен нулю, то устанавливается в единицу бит Z, а если отрицателен, то N.

Кроме проверки операнда оператор TST(B) иногда применяется для инкремента (или декремента) содержимого регистра на 2. Например, команда TST (R2)+ увеличивает на 2 содержимое R2, причем делает это быстрее, чем две команды INC R2, и занимает в ОЗУ всего одно слово. Однако при таком использовании данного оператора в регистре должно быть записано число, соответствующее РЕАЛЬНО СУЩЕСТВУЮЩЕМУ АДРЕСУ ЭВМ. В противном случае, например если в регистре R2 записано число 177776 (не существующий в БК-0010 адрес), произойдет прерывание по зависанию. (Это замечание, конечно, относится к любой команде, выполняемой по косвенному адресу, независимо от того, с какой целью она применяется.) Необходимо также отметить, что на некоторых ЭВМ с аналогичной системой команд косвенное обращение к СЛОВУ по НЕЧЕТНОМУ содержимому регистра тоже ведет к зависанию. На БК-0010 вместо этого производится обращение по меньшему ЧЕТНОМУ адресу, т. е. адрес обращения корректируется. Содержимое же регистра после такой операции (при автоинкрементной или автодекрементной адресации), как обычно, изменяется на 2, т. е. остается нечетным.

- ASL(B) AB — АРИФМЕТИЧЕСКИЙ СДВИГ ВЛЕВО. Слово (или байт) операнда поразрядно сдвигается влево, старший разряд при этом переписывается в бит С ССП, а в младший разряд заносится ноль. Пусть, например, в регистре R2 записано слово 1 011 101 100 111 000 (восьмеричное 135470). После выполнения команды ASL R2 получим: 0 111 011 001 110 000 (восьмеричное 073160), разряд С=1.

Оператор ASL(B) применяется для умножения на 2 и других целей.

- ASR(B) AB — АРИФМЕТИЧЕСКИЙ СДВИГ ВПРАВО. Слово (или байт) операнда поразрядно сдвигается вправо, младший разряд при этом переписывается в бит С ССП, старший разряд не меняется (т. е. не меняется знак числа. — *Прим. ред.*). Пусть, например, в регистре R2 хранится слово 1 011 101 100 111 000 (восьмеричное 135470). После выполнения команды ASL R2 получим: 1 101 110 110 011 100 (восьмеричное 156634), разряд С=0.

Оператор ASR(B) применяется для деления на 2 и других целей.

- ROL(B) AB — ЦИКЛИЧЕСКИЙ СДВИГ ВЛЕВО. Слово (или байт) операнда поразрядно сдвигается влево, старший разряд при этом переписывается в бит С ССП, а предшествующее содержимое последнего переходит в младший разряд операнда. Слово или байт как бы замыкается при этом в кольцо (причем замыкающим звеном является бит С ССП) и происходит ВРАЩЕНИЕ этого кольца на один разряд влево.
- ROR(B) AB — ЦИКЛИЧЕСКИЙ СДВИГ ВПРАВО. Слово (или байт) операнда поразрядно сдвигается вправо, младший разряд при этом переписывается в бит С ССП, а предшествующее содержимое С переходит в старший разряд операнда (аналогично оператору ROL(B), но с вращением в другую сторону).

Два последних оператора могут также применяться для поразрядной перезаписи одного операнда в другой через С-разряд. Примеры:

```
ROL R2          ; старший бит содержимого R2
ROR R4          ; переносится в старший бит R4
```

```
ROL R2      ; старший бит R2
ROL R4      ; переносится в младший бит R4

ROR R2      ; младший бит R2
ROL R4      ; переносится в младший бит R4

ROR R2      ; младший бит R2
ROR R4      ; переносится в старший бит R4
```

Выполняя такие сочетания команд подряд несколько раз, можно «выворачивать наизнанку» слово или его часть, выделять нужное число разрядов в другое слово и т. д.

- ADC(B) AB — ПРИБАВЛЕНИЕ РАЗРЯДА ПЕРЕНОСА. К операнду прибавляется содержимое бита С ССП. Применяется для учета переноса при сложении, умножении, вычислении контрольных сумм и для других целей.
- SBC(B) AB — ВЫЧИТАНИЕ РАЗРЯДА ПЕРЕНОСА. Из операнда вычитается содержимое бита С ССП. Применяется для учета переноса при вычитании, делении и для других целей.

(Оператор ADC(B) может быть использован для обеспечения многократной точности при сложении чисел:

```
ADD A0,D0   ; сложение младших частей
ADC B1      ; прибавить перенос к старшей части
ADD A1,B1   ; сложение старших частей
```

SBC(B) аналогично используется для обеспечения многократной точности при вычитании:

```
SUB A0,D0
SBC B1
SUB A1,B1
```

В данном примере показано вычитание с двойной точностью. — *Прим. ред.*).

- SWAB AB — ПЕРЕСТАНОВКА БАЙТОВ. Старший и младший байты операнда меняются местами. Применяется для побайтного доступа к содержимому слова и для других целей.
- SXT AB — РАСШИРЕНИЕ ЗНАКА. Всем разрядам операнда (слова) присваивается значение знакового разряда. В результате операции слово может стать равным или 0 (если исходное число положительное), или 177777 (если отрицательное). Применяется для операций с плавающей запятой, учета знака и для других целей.
- MFPS B — ЧТЕНИЕ ССП. Содержимое ССП записывается в операнд. О структуре ССП подробно говорилось в разделе, посвященном описанию центрального процессора. Применяется этот оператор очень редко, в основном для сохранения текущего ССП с целью его последующего восстановления или анализа. Сохранение и анализ ССП при вызовах подпрограмм и прерываний, а также при условных переходах ЦП обычно осуществляет автоматически, поэтому данная команда требуется только тогда, когда обработка ССП отличается от стандартной.
- MTPS A — ЗАПИСЬ ССП. Слово-операнд записывается в ССП. Применяется чаще всего для задания необходимого уровня приоритета, реже для восстановления ССП после нестандартного преобразования. Нужно отметить, что команды MTPS и MFPS являются БАЙТОВЫМИ, при косвенной автоинкрементной

или автодекрементной адресации через регистр его содержимое будет меняться на 1, а не на 2, а при записи содержимого ССП в регистр его старший байт будет изменяться, как это описано ниже для команды MOVB.

С двумя операндами

- MOV(B) A,B — ПЕРЕСЫЛКА. Содержимое операнда A записывается (копируется) в операнд B. Содержимое A не меняется, исходное значение B теряется. Этот оператор, особенности его работы и примеры были подробно рассмотрены раньше. Напомним только, что при байтовой пересылке В РЕГИСТР в нем происходит РАСПРОСТРАНЕНИЕ ЗНАКА младшего байта на весь старший байт (подробно этот эффект рассмотрен в одном из предыдущих номеров журнала. — *Прим. ред.*). Оператор MOV — наиболее часто используемый из всех операторов ассемблера.
- BIC(B) A,B — ОЧИСТКА РАЗРЯДОВ ПО МАСКЕ. Во втором операнде обнуляются биты (разряды) по МАСКЕ ПЕРВОГО. Смысл этого действия следующий: во втором операнде (слове или байте) устанавливаются равными НУЛЮ те разряды, которым соответствуют ЕДИНИЦЫ в первом операнде. Пусть в регистре R2 хранится слово 0 100 001 111 010 100, а в регистре R4 — 0 110 100 000 100 111. Выполнив команду BIC R2,R4, получим в регистре R4 число 0 010 100 000 100 011 — биты R4, которым соответствуют единицы в R2, обнулены, остальные не изменились.

Еще один пример. После выполнения команд

MOV #177777,R4

BIC #177400,R4

содержимое R4 станет равно 377 (выделен младший байт R4).

Оператор BIC(B) широко применяется для выборочного стирания отдельных разрядов слова (байта) с целью выделения нужных разрядов и т. п.

- BIS(B) A,B — УСТАНОВКА РАЗРЯДОВ ПО МАСКЕ. В отличие от предыдущего оператора, во втором операнде устанавливаются в ЕДИНИЦУ те разряды, которым соответствуют ЕДИНИЦЫ в первом. Для тех же исходных данных, что в предыдущем примере, результатом выполнения команды BIS R2,R4 будет содержимое R4: 0 110 101 111 110 111, а последовательность команд

CLR R4

BIS #377,R4

даст содержимое R4, равное 377.

Оператор BIS(B) широко применяется для выборочной записи определенных разрядов, «наложения» операндов и т. п.

- BIT(B) A1,A2 — ПРОВЕРКА БИТОВ ПО МАСКЕ. Проверяются биты (разряды) второго операнда по МАСКЕ ПЕРВОГО. Оба операнда не меняются, но по результатам проверки устанавливаются биты условий ССП. Смысл операции следующий. Если ХОТЯ БЫ ОДНОМУ разряду первого операнда, равному ЕДИНИЦЕ, соответствует ЕДИНИЦА в том же разряде второго операнда, результат НЕ НУЛЕВОЙ. Если же ВСЕМ ЕДИНИЦАМ первого операнда соответствуют только НУЛИ второго, результат НУЛЕВОЙ. Чаще всего оператор используется для проверки какого-либо отдельного бита, для чего в первом операнде задается маска, в которой присутствует только одна единица в соответствии с проверяемым разрядом второго. Например, если

нажата любая клавиша, в системном регистре 177716 разряд 06 будет равен 0, в противном случае — единице. Тогда для выявления факта нажатия клавиши можно применить команду BIT #100,@#177716. Если клавиша нажата, результат исполнения команды равен нулю и наоборот.

- CMP(B) A1,A2 – СРАВНЕНИЕ. Операнды сравниваются между собой и по результатам сравнения устанавливаются биты условий ССП. Сами операнды при этом не меняются. Сравнение осуществляется псевдовычитанием ВТОРОГО операнда из ПЕРВОГО, т. е. как бы осуществляется действие (A1—A2). Например, если операнды равны, то результат операции — нуль (Z=1), если второй операнд больше первого, то результат отрицательный (N=1) и т. д. Оператор применяется очень широко как для сравнения операндов, так и для инкремента (или декремента) регистров общего назначения, причем он позволяет модифицировать содержимое регистра сразу на 4 (например CMP (R2)+,(R2)+), ничего при этом не меняя, кроме содержимого R2 и битов условий, и занимая всего одно слово. Но необходимо помнить, что биты условий при таком использовании не имеют никакого отношения к содержимому регистра и обнаружить после такой команды, например, произошло ли переполнение R2, невозможно. Кроме того, содержимое регистра должно указывать на реально существующий адрес, иначе произойдет зависание.
- ADD A,B — СЛОЖЕНИЕ. Содержимое операндов (слов) суммируется, результат записывается во ВТОРОЙ операнд, первый же при этом не меняется, т. е. выполняется присваивание B=B+A. Операция широко применяется для сложения и умножения.
- SUB A,B — ВЫЧИТАНИЕ. Из ВТОРОГО операнда вычитается ПЕРВЫЙ, результат записывается во ВТОРОЙ операнд, первый не меняется, т. е. выполияется присваивание B=B–A. Необходимо всегда помнить, что порядок вычитания в операторе SUB (из второго — первый) обратен псевдовычитанию в операторе CMP (из первого — второй), это часто путают. Оператор SUB широко применяется для вычитания и деления, иногда — для сравнения.
- XOR RN,B — ПОРАЗРЯДНОЕ «ИСКЛЮЧАЮЩЕЕ ИЛИ». Выполняется логическая функция «исключающее или» между соответствующими разрядами (битами) первого и второго операндов, результат заносится во второй операнд. Первый операнд может быть только регистром общего назначения, второй адресуется любым способом.

Что такое «исключающее или»? Это функция, фиксирующая несовпадение битов, так называемая функция НЕРАВНОЗНАЧНОСТИ. Поясним сказанное таблицей состояний.

1-й бит	2-й бит	Результат
0	0	0
0	1	1
1	0	1
1	1	0

Подобная операция выполняется между всеми разрядами операндов, в результате чего получается число, содержащее не разность, а как бы разницу между ними. Одинаковые операнды дают в результате ноль, взаимно-инверсные — число 177777. Такая функция дает возможность «наслаивать» одно число на другое, например формировать на экране спрайты на фоне другого изображения с сохранением последнего. Если

вывести то же самое изображение с помощью функции XOR на то же место вторично, то оно исчезнет, а фон восстановится. Аналогично можно «наслаивать» звуки в многоголосных мелодиях, воспроизводимых одним динамиком, и т. п. Пусть в R2 записано число 0 101 110 111 001 011, а в R4 — 1 001 100 111 101 011. Тогда после выполнения команды XOR R2,R4 в R4 будет записано 1 100 010 000 100 000.

2. Операторы управления программой

Эта группа операторов предназначена для управления ходом вычислительного процесса (организации ветвлений, подпрограмм и циклов).

- BR MET — БЕЗУСЛОВНОЕ ВЕТВЛЕНИЕ. Вызывает переход (передачу управления) по адресу MET независимо от битов условий. Метка, по которой выполняется переход, может быть как обычной, так и локальной.

Здесь, как и во всех других командах ВЕТВЛЕНИЯ, адрес передачи управления определяется суммой текущего значения счетчика команд PC и младшего байта команды, в который при трансляции заносится разность (#MET−PC) в дополнительном коде, деленная на 2, — СМЕЩЕНИЕ. При исполнении команды ветвления смещение умножается на 2 с учетом знака и прибавляется к содержимому PC. Таким образом обеспечивается передача управления на расстояние до 256д байт «вперед» или «назад», считая от текущего адреса.

Все остальные команды ветвления (кроме BR) являются УСЛОВНЫМИ, т. е. передача управления по адресу MET осуществляется, если биты условий, установившиеся после выполнения ПРЕДШЕСТВУЮЩЕЙ команды в разрядах 00...03 ССП, удовлетворяют условиям оператора ветвления. В противном случае программа выполняется дальше, а оператор ветвления игнорируется. Сами по себе операторы ветвления НЕ МЕНЯЮТ биты условий, поэтому можно располагать их друг за другом для последовательной проверки нескольких условий. Удобно привести все операторы ветвления в виде таблицы:

Команда	Условие ветвления	Определяющие биты условий
BNE MET	не равно нулю	Z=0
BEQ MET	равно нулю	Z=1
BPL MET	плюс	N=0
BMI MET	минус	N=1
BVC MET	нет переполнения	V=0
BVS MET	есть переполнение	V=1
BCC MET	нет переноса	C=0
BCS MET	есть перенос	C=1
...с учетом знака...		
BGT MET	больше нуля	N=V и Z=0
BLT MET	меньше нуля	N+V=1
BGE MET	больше или равно нулю	N=V
BLE MET	меньше или равно нулю	N+V=1 или Z=1

BGT MET	больше нуля	N=V и Z=0
BLT MET	меньше нуля	N+V=1
BGE MET	больше или равно нулю	N=V
BLE MET	меньше или равно нулю	N+V=1 или Z=1
...без учета знака...		
BHI MET	больше нуля	C=0 и Z=0
BLO MET	меньше нуля	C=1
BHIS MET	больше или равно нулю	C=0
BLOS MET	меньше или равно нулю	C=1 или Z=1

Первые восемь операторов в особых пояснениях не нуждаются, они проверяют отдельные биты условий. Операторы же BLO и BHIS, полностью совпадающие с BCS и BCC соответственно, введены исключительно для удобства программистов (дизассемблер обычно восстанавливает их только в виде BCS и BCC, не учитывая, в какой форме они были записаны в исходном тексте). А вот вопрос о различии в действии операторов переходов с учетом и без учета знака нуждается в отдельном рассмотрении.

Как известно, числа со знаком изображаются в системе команд нашей ЭВМ в дополнительном коде, т. е. старший бит числа является ЗНАКОВЫМ: ноль, если число положительное, и единица, если отрицательное. Диапазон абсолютных величин чисел со знаком, естественно, меньше за счет уменьшения разрядности на 1. Поэтому существует и второе представление — без знака, только для положительных чисел, но с большей разрядностью. В связи с этим и возникают парадоксы при сравнении чисел. Например, любое число без знака больше нуля. А со знаком — больше нуля только положительные числа, отрицательные меньше. Это утверждение кажется банальным, но при использовании операторов ветвления форму представления чисел обязательно надо учитывать. Как правило, при логических операциях (сравнение кодов, сортировка и т. п.) удобнее рассматривать числа без знака, а при арифметических вычислениях — со знаком. Вообще же правильный выбор команд ветвления — довольно сложная задача, и некоторые авторы не случайно рекомендуют сводить сравнение чисел к таким вариантам, когда становится возможным применение лишь двух операторов — BNE и BEQ, которые всегда действуют одиозначно. Относясь к таким рекомендациям сочувственно, нужно все-таки ясно отдавать себе отчет, что далеко не все задачи сводимы к частным вариантам, а плата за такое преобразование велика — потеря памяти и быстродействия. Поэтому следует применять все операторы ветвления, но тогда, когда это необходимо. А чтобы не делать ошибок, нужно точно представлять себе, какой результат возможен при той или иной операции, при тех или иных исходных данных, а не применять операторы бездумно, по шаблону.

Пример простейшей программы с использованием операторов ветвления:

```
        ;"мигание" экрана
0:      MOV #177777,R0      В R0 все биты единичные
1:      MOV #40000,R1       Начало экрана
2:      MOV R0,(R1)+        Запись очередного слова в ОЗУ
        CMP #100000,R1      Конец экрана?
        BNE 2               Нет — продолжать
        TST R0              R0 = 0 (цикл очистки)?
        BEQ 0               Да — записать единицы в R0
        CLR R0              Иначе обнулить R0 и войти
        BR  1               в цикл очистки экрана
        END
```

- JMP N — БЕЗУСЛОВНЫЙ ПЕРЕХОД. Эта команда, в отличие от безусловного ветвления BR, передает управление по любому адресу, а не только на 256д байт, и допускает задание адреса перехода любым способом (есть, конечно, и исключения. Например, невозможна передача управления на регистр — JMP RN). Локальные метки в составе адреса перехода не допускаются.
- JSR RN,N — ПЕРЕХОД К ПОДПРОГРАММЕ. По этой команде процессор прерывает обработку текущей последовательности операторов, записывает в стек содержимое указанного регистра RN (как бы автоматически выполняя команду MOV RN,-(SP)), переписывает в регистр RN содержимое PC (а там в этот момент записан адрес следующей команды программы), а затем заносит в PC адрес подпрограммы N (способ его задания может быть любым, с теми же ограничениями, что и для оператора JMP), вследствие чего следующей исполняется первая команда подпрограммы.

В конце подпрограммы должна стоять КОМАНДА ВОЗВРАТА — RTS RN, где RN — тот же самый регистр, что в операторе JSR. По этой команде ЦП переписывает содержимое RN в PC (т. е. восстанавливает адрес следующей за переходом к подпрограмме команды основной программы), затем восстанавливает из стека содержимое RN (как бы автоматически выполняя команду MOV (SP)+,RN), и выполнение основной программы продолжается. Из одной подпрограммы можно обратиться к другой, из нее — к третьей и т. д. Важно лишь, чтобы используемый в качестве буфера адреса возврата регистр (это может быть как один и тот же регистр для всех вложенных подпрограмм, так и разные) во время работы подпрограммы не менял свое содержимое (если, конечно, это не делается со специальными целями), иначе правильный возврат из подпрограммы станет невозможен. Количество последовательных обращений из одной подпрограммы в другую (так называемый УРОВЕНЬ ВЛОЖЕННОСТИ) ограничен допустимой глубиной стека.

Наиболее удобно использовать в качестве RN счетчик команд PC. Если при этом проанализировать работу операторов JSR и RTS, то окажется, что по команде JSR PC,N в стеке сохраняется, а по команде RTS PC восстанавливается непосредственно адрес возврата из подпрограммы! Этот способ обращения к подпрограммам очень удобен тем, что позволяет не занимать ни одного операционного регистра под адрес возврата, ведь их и так не очень много. Понятно, что данный способ обращения к подпрограммам используется чаще других, и для него в ассемблере «МИКРО.10К» даже предусмотрена специальная форма записи: вместо JSR PC,N — CALL N, вместо RTS PC — RET. Наличие такой формы отнюдь не запрещает в «МИКРО.10К» и стандартного обращения, просто такое — короче и удобнее.

Входов в подпрограмму (как и выходов) может быть несколько, в зависимости от разных условий, но всегда нужно помнить, что при входе в подпрограмму указатель стека SP уменьшается на 2 и при выходе из нее не оператором RTS (RET), а иначе (например BR, JMP и т. п.) стек необходимо восстанавливать. В противном случае он рано или поздно будет исчерпан и ЭВМ перестанет работать. В самой же подпрограмме со стеком следует работать осторожно и заботиться, чтобы при выходе из нее адрес возврата находился в вершине стека.

- RTS RN — ВОЗВРАТ ИЗ ПОДПРОГРАММЫ. Подробно рассмотрен выше. В «МИКРО.10К» вместо RTS PC может использоваться оператор RET.
- MARK X — ПОМЕТИТЬ СТЕК. Эта команда применяется крайне редко, может быть, потому, что ни в одном литературном источнике она не описана достаточно понятно. Попытаемся восполнить этот пробел.

Бывают случаи, когда при обращении к подпрограмме нужно передать ей ряд чисел — ПАРАМЕТРОВ. Эти числа при каждом обращении к одной и той же подпрограмме могут быть различными, и их может быть разное количество. В таких случаях довольно удобно перед вызовом заносить параметры в стек, извлечь же их в подпрограмме можно, например, с помощью индексной адресации или иным путем. Но так как параметров может быть разное количество, то каждый раз по выходе из подпрограммы стек надо восстанавливать. Команда MARK делает это автоматически. Как ее применять?

Сначала выберем один из регистров RN под буфер адреса возврата и запишем его содержимое в стек. Какой регистр выбрать? В отличие от общего случая применения команды JSR RN, при использовании команды MARK в качестве регистра возврата может быть использован ТОЛЬКО R5 — это заложено в ЦП аппаратно. Затем занесем в стек X параметров. После этого запишем в стек саму команду в форме MARK X, где X — количество параметров (восьмеричное число от 0 до 77). Наконец, запишем адрес вершины стека в R5 и обратимся к подпрограмме через другой регистр. Если в конце подпрограммы стоит команда RTS R5, то стек и регистры автоматически будут восстановлены. Не вдаваясь в дальнейшие подробности, просто приведем пример. Регистром возврата выбран R5 (напомним, что только он и может быть выбран!), а обращение к подпрограмме пусть производится через регистр PC. Занесем в стек три параметра:

```
        MOV     R5,-(SP)        Сохранить R5
        MOV     #101,-(SP)      Занести параметры -
        MOV     #102,-(SP)      коды символов
        MOV     #103,-(SP)      "A", "B" и "C"
        MOV     MR3,-(SP)       Занести код команды MARK 3
        MOV     SP,R5           Занести адрес возврата
        CALL    SBR             Перейти к подпрограмме SBR
        ;.....                  Дальнейший текст программы
SBR:    ;.....                  Текст подпрограммы SBR
        RTS     R5              Выход из подпрограммы
                                ;с восстановлением стека
MR3:    MARK    3               Код команды MARK 3
```

Читатель, программирующий на ассемблере, может, конечно, предложить массу других способов как для передачи параметров подпрограмме, так и для восстановления стека без команды MARK. Но поскольку данная команда существует, разобрать ее было необходимо хотя бы для того, чтобы избежать вопросов. А применять ее или нет — дело сугубо личное. Большинство программирующих на ассемблере решают этот вопрос отрицательно и, может быть, правильно делают — при программировании и так хватает забот, даже без использования столь сложной по структуре и малоэффективной команды.

- SOB RN,MET — ВЫЧИТАНИЕ ЕДИНИЦЫ И ВЕТВЛЕНИЕ. Этот оператор может быть использован для организации циклов. При его исполнении сначала происходит вычитание единицы из содержимого регистра RN, а затем, если результат НЕ НУЛЕВОЙ, — переход к метке MET. Если же содержимое RN стало равно нулю, то программа выполняется дальше. Метка MET может быть как локальной, так и обычной и должна быть расположена ДО оператора SOB, т. е. передача управления возможна только «назад» и не более чем на 64д слова. Это объясняется тем, что в операторе SOB для хранения смещения используются только младшие 6 бит, а при исполнении перехода смещение (без знака) умножается на 2 и вычитается из PC. Циклы могут быть и вложенными, но число вложений, если не принимать специаль-

ных мер, не может превышать количества операционных регистров, которые используются для хранения цикловых переменных. Помимо организации вычислений этот оператор часто используется для создания временных задержек при работе программ, причем «пустой» цикл SOB при исходном содержимом регистра, равном нулю, выполняется на БК-0010 примерно за 0.4 с. Если потребуются большие задержки, следует прибегнуть к вложенным циклам или ввести в цикл «пустые» команды, например CMP RN,RN. Приведем пример использования данного оператора, причем это будет уже знакомая программа «мигания» экрана, но теперь решенная иными средствами.

```
        ;"мигание" экрана
0:      MOV     #177777,R0      Все биты в R0 - единицы
1:      MOV     #40000,R1       Начало экрана
        MOV     #20000,R2       Цикл из 20000 повторов
2:      MOV     R0,(R1)+        Запись очередного слова в ОЗУ
        SOB     R2,2            Если не конец - продолжать
        BEQ     0               Если 0, записать все единицы
        CLR     R0              Иначе обнулить R0 и войти
        BR      1               в цикл очистки экрана
        END
```

Сравните эту программу с аналогичной, приведенной ранее (где цикл был организован с помощью оператора CMP). Какая из них короче и какая работает быстрее? Обратите внимание, что оператор SOB не меняет биты условий, поэтому нам удалось обойтись без проверки содержимого R0 оператором TST.

При работе с циклами нужно быть внимательным, чтобы не передать управление на оператор, задающий начальное значение переменной цикла, иначе цикл станет бесконечным. С этой же целью нужно заботиться о сохранении в цикле содержимого регистра-счетчика RN, если он должен использоваться для других целей.

3. Операторы прерывания

Эта важная группа операторов имеет общую черту: встретив такой оператор, ЦП прерывает выполнение текущей задачи. Что такое прерывание, было уже достаточно подробно описано в разделе, посвященном архитектуре ЭВМ (см. № 2 за 1994 г.), и возвращаться к этому вопросу мы не будем. Подробно разберем только КОМАНДНЫЕ ПРЕРЫВАНИЯ.

- EMT X — КОМАНДНОЕ ПРЕРЫВАНИЕ ПО ВЕКТОРУ 30.
- TRAP X — КОМАНДНОЕ ПРЕРЫВАНИЕ ПО ВЕКТОРУ 34.

Командные прерывания EMT и TRAP (иначе называемые КОМАНДНЫМИ ЗАПРОСАМИ) очень похожи друг на друга. Знак X здесь — АРГУМЕНТ EMT или TRAP, лежащий в пределах 0...377. Рассмотрим, как работают эти команды, на примере EMT.

Пусть процессор встретил в программе команду с кодом EMT X. Первым делом происходит прерывание по вектору 30 (как обычно сохраняя в стеке SP и PC). И вот тут-то появляется «неожиданность» — по адресу, указанному в ячейке @#30, находится не конкретная, единственная программа обработки прерывания, а целый пакет программ! Но ничего ужасного не происходит: «на пороге» нас встречает маленькая, но очень важная программа — EMT-ДИСПЕТЧЕР. Что он делает? Первым делом выделяет младший байт команды EMT, т. е. ее аргумент. По аргументу (или, как еще говорят, по

НОМЕРУ EMT) EMT-диспетчер находит в специальной таблице адрес программы, которая нам нужна, и передает ей управление. Таким образом, задавая номер EMT, мы можем с помощью этой единственной команды обращаться к множеству программ, выполняющих самые разные функции.

Но чем же такое обращение отличается от вызова подпрограмм, ведь их тоже можно написать множество и обращаться к каждой просто по имени метки? Это вроде бы проще и понятнее. Верно, только вот «закавыка» — команда обращения к подпрограмме занимает в памяти два слова, а EMT — только одно! Таких обращений в тексте программы может быть несколько сотен, и тогда EMT дает экономию памяти в сотни слов. Для БК с ее, мягко говоря, не слишком емкой памятью, это немаловажно.

А что же TRAP, какова роль этой команды? Практической разницы между EMT и TRAP нет, вернее, разница чисто условная. Для команды TRAP выполняется прерывание по вектору 34. В самой же команде указывается, как и в EMT, номер в диапазоне 0...377. Команды EMT принято использовать в системных программах ЭВМ, а TRAP — в пользовательских. Что же касается порядка их обработки, он одинаков — для вектора 34 программа обработки должна начинаться с TRAP-ДИСПЕТЧЕРА, аналогичного EMT-диспетчеру.

Программа EMT-диспетчера БК-0010 расположена по адресу 100112 и обслуживает множество команд EMT с различными номерами. Каждая из них делает что-то свое, иногда эти действия весьма сложны, причем ими может воспользоваться не только сама ЭВМ, но и программист. Тем самым иногда можно избежать написания длинных и очень сложных подпрограмм — достаточно вписать в программу команду EMT с нужным номером, а подпрограмма обработки уже готова — она «ждет» в ПЗУ. Тут необходимо отметить (спасибо разработчикам БК-0010, а особая благодарнось — автору МДС М. И. Дябину), что содержимое мониторно-драйверной системы нашей ЭВМ (а тем самым и система командных запросов EMT) осталось неизменным с 1985 г., когда компьютер БК-0010 «пошел в серию», и до сего дня, когда освоен выпуск последней модели БК-0010.01 с новой отличной бездребезговой клавиатурой и полностью переработанной печатной платой. Это очень важно — ведь изменись МДС хоть частично, все ранее разработанные прикладные программы оказались бы написанными впустую, они не работали бы на новых моделях компьютеров! К чему это приводит, видно на примере БК-0011. Не успела она появиться в продаже, как разработчикам что-то в ней не понравилось и они сочли своим долгом изменить ее мониторно-драйверную систему. А новая МДС БК-0011М оказалась программно несовместимой с прежней. Это, без всякого преувеличения, настоящая катастрофа для всех владельцев БК-0011 — старый компьютер уже не выпускают, программ для него нет и никогда не будет — слишком мало экземпляров БК-0011 было выпущено! Это трагедия и для разработчиков — они сразу и навсегда подорвали доверие пользователей к новой машине: кто гарантирует, что завтра снова не изменится МДС БК-0011М и ее владельцы вновь не окажутся «у разбитого корыта»... Это удар и для программистов — все, разработанное ими в ужасной спешке и зачастую почти бесплатно для БК-0011, уже не годится для БК-0011М и никогда не окупится.

К счастью, с БК-0010 ничего такого не происходило — все ее модели до настоящего времени полностью программно совместимы друг с другом на уровне машинных команд, в том числе и EMT-запросов.

На что же годны имеющиеся в системе БК-0010 командные прерывания EMT, или, другими словами, что они делают?

Командные прерывания EMT БК-0010

Начнем с того, что на БК-0010 для номеров EMT приняты только четные числа. Это чистая условность — можно было бы использовать и нечетные, но с четными EMT-диспетчер вышел чуть проще, а такого множества функций, какое было бы возможно с использованием всех чисел, просто не нужно.

Каждая команда EMT (имеющая свой номер) делает что-то свое. Но между ними есть и нечто общее — обычно они требуют задания каких-либо параметров (исходных данных). Данные эти задаются, как правило, в одном или нескольких регистрах общего назначения, до вызова команды EMT. После выполнения команды содержимое РОН может остаться тем же (сохраняться) или стать иным. Часто новое значение РОН несет ценную информацию, которая может быть использована (а иногда эта информация — единственный результат выполнения EMT). По ходу знакомства с различными EMT-функциями мы будем приводить сведения как о входной, так и о выходной информации (если она представляет интерес), и указывать, какие РОН не сохраняются при выполнении EMT (о их сохранении следует позаботиться до вызова EMT). Наиболее употребительным функциям EMT мы будем уделять больше внимания. Будем рассматривать их не в порядке номеров, а по функциональному назначению.

- EMT 4 — ИНИЦИАЛИЗАЦИЯ ДРАЙВЕРА КЛАВИАТУРЫ. Переписываются (устанавливаются в исходные значения) векторы прерывания от клавиатуры (60 и 274), сбрасывается маска прерываний от клавиатуры (разряд 06 регистра 177660), устанавливается режим передачи кодов клавиатуры по запросам рабочей программы и код 12 для клавиши «ВВОД». R0 не сохраняется. EMT 4 используется весьма редко, например если по ходу программы мы запретили прерывание от клавиатуры и нужно его разрешить.
- EMT 14 — ИНИЦИАЛИЗАЦИЯ ВСЕХ ДРАЙВЕРОВ. Применяется, напротив, очень часто для «начальной установки» всех параметров ЭВМ (а особенно режимов дисплея). Обеспечивает установку в исходное состояние всех ячеек системной области (кроме стека), всех векторов прерывания и системных регистров, очистку экрана, установку исходных режимов дисплея и ТЛГ-канала, очистку порта ввода-вывода. Таким образом, БК-0010 приводится практически в такое же состояние, как после перезапуска системы (или включения питания). Регистры, кроме R5, не сохраняются. Эту команду следует применять с некоторой осторожностью, так как из-за сохранения содержимого стека после выполнения EMT 14 иногда по окончании работы программы возможен неожиданный эффект — выход из МСД в ПМ или даже в ФОКАЛ со стиранием ОЗУ.
- EMT 6 — ВВОД КОДА СИМВОЛА С КЛАВИАТУРЫ. По данной команде работа программы прерывается, и ЭВМ ждет нажатия клавиши. Код нажатой клавиши заносится в младший байт R0, старший байт очищается. Применяется очень широко как для ввода кодов отдельных символов или команд, так и просто для приостанова программы и ожидания нажатия клавиши (организация паузы). Следует помнить, что, когда ЭВМ ждет нажатия клавиши по команде EMT 6, она находится в режиме прерывания, следовательно, вершина стека уже на 4 меньше исходной. Нажатие в этот момент клавиши «СТОП» — не одно и то же, что останов с ее помощью работающей программы, в смысле содержимого стека на момент останова

и положения его вершины (содержимого SP), иногда это может быть существенно.

- EMT 16 — ПЕРЕДАЧА КОДОВ ДРАЙВЕРУ ДИСПЛЕЯ. Это «EMT 6 наоборот» — вывод на экран символа, код которого содержится в младшем байте R0 (или исполнение команды управления режимами экрана, если в R0 ее код). Содержимое старшего байта R0 при исполнении EMT 16 значения не имеет и не меняется. Пожалуй, это наиболее часто применяемая команда EMT — с ее помощью можно выводить как отдельные символы, так и целые тексты. Все регистры при ее исполнении сохраняются. Приведем примеры использования этих команд.

```
1)      MOVB    #77,R0          Вывод на экран
        EMT     16              символа "?"
        END

2)      MOVB    #14,R0          Сброс экрана
        EMT     16              и вывод символа
        MOVB    'A',R0          "A"
        EMT     16
        END

3)      MOV     'АБ',R0         Вывод на экран
        EMT     16              двух символов
        SWAB    R0              "А" и "Б"
        EMT     16
        END

4)  0:  EMT 6                   Вывод на экран
        EMT     16              символов, вводимых
        CMPB    #40,R0          с клавиатуры; завершение -
        BNE     0               по клавише "ПРОБЕЛ"
        END

5)      MOV     #7,R0           Цикл
        MOV     #100,R1         из 64д звуковых
    0:  EMT     16              сигналов
        SOB     R1,0
        END

6)  0:  EMT     6               Пауза до нажатия клавиши
        MOV     #TEX,R1         Адрес начала текста
        MOVB    #14,(R1)+       Команда очистки экрана
        MOVB    'П',(R1)+       Ввод в ОЗУ
        MOV     'ри',(R1)+      строки символов -
        MOV     'ве',(R1)+      слова "Привет"
        MOVB    'т',(R1)+
        CLRB    (R1)+           Нулевой байт — конец текста
        MOV     #TEX,R1         Адрес начала текста
    1:  MOVB    (R1)+,R0        Вывод текста на экран
        BEQ     0               побайтно до нулевого байта,
```

```
        EMT     16          затем — повторение с начала
        BR      1
TEX:    END
```

В приведенных примерах листинги заканчиваются командой END, но это не обязательно — они могут представлять собой подпрограммы или фрагменты программ. Для полного использования возможностей команды EMT 16 необходимо знать не только коды символов БК-0010, но и командные коды. Эти сведения достаточно полно изложены в прилагаемых к БК руководствах, и подробно останавливаться на них нет необходимости.

Заметим, что приведенная в последнем примере строка текста предварительно записывается в память отнюдь не оптимальным способом, но другого варианта мы пока еще «не проходили».

Контрольные вопросы и задания

1. Чему будет равно содержимое регистра R0 в результате исполнения последовательно каждой из команд:

```
MOV     #77000,R0
INC     R0
COMB    R0
```

— 77000; 77001; 77376.

2. Чему будет равно содержимое регистра R5 в результате исполнения последовательно каждой из команд:

```
MOV     #177777,R5
ASL     R5
ASL     R5
ROL     R5
ROL     R5
```

— 177777; 177776; 177774; 177771; 177763.

3. Что получится в результате выполнения команды XOR #1000,@#2400 ?

— Ничего, так как первым операндом команды XOR может быть только RN — один из регистров общего назначения. Такая команда даже не будет оттранслирована ассемблером.

4. Как переписать младший байт из регистра R0 в регистр R4, не изменяя при этом старшие байты R0 и R4 (остальные регистры и ячейки памяти не использовать)? Напишите соответствующую программу.

```
—   CLRB    R4
    BISB    R0,R4
```

5. Как установятся биты условий ССП N, Z, V, C в результате исполнения последовательно из каждой команд:

```
MOV     #177777,R0
INC     R0
ADD     #77777,R0
ASL     R0
ASL     R0
```

— N=1; Z=0; V=0; C=0
N=0; Z=1; V=0; C=0
N=0; Z=0; V=0; C=0
N=1; Z=0; V=1; C=0
N=1; Z=0; V=0; C=1

6. Будет ли выполняться ветвление по оператору BGE в данной программе:

```
        MOV     #377,R0
        MOV     #120,R1
        CMPB    R0,R1
        BGE     MET
```

Что изменится, если вместо оператора CMPB использовать CMP?

— Не будет, так как число 377 отрицательное и меньше числа 120, а оператор BGE учитывает знак. При замене CMPB на CMP ветвление будет выполняться.

7. Сколько раз будет выполнен цикл SOB в данном случае:

```
        MOV     #10,R0
0:      DEC R0
        SOB     R0,0
```

— 4 раза, так как регистр-счетчик R0 дополнительно модифицируется в теле цикла.

8. Сколько символов «A» выведет на экран следующая программа:

```
        MOV     'A',R0
0:      MOV     #10,R1
        EMT     16
        SOB     R1,0
```

— Бесконечное множество, так как цикл организован неправильно — запись константы в R1 включена в цикл.

9. Напишите программу, которая по нажатию любой клавиши (кроме клавиши «СТОП») выводила бы на экран символ «?».

```
— 0:    EMT     6
        MOV     '?',R0
        EMT     16
        BR      0
```

10. Напишите программу, которая преобразовывала бы символы, вводимые с клавиатуры в любом регистре («РУС», «ЛАТ», «СТР», «ЗАГЛ»), в символы в регистре «ЛАТ-ЗАГЛ» соответственно нажатой клавише (например, «л» — в «L») и выводила на экран исходный символ, а через черточку — преобразованный, причем каждая следующая пара символов должна выводиться с новой строки.

Подсказка: для того чтобы преобразовать символы из одного регистра в другой, достаточно сбросить (или установить) определенные разряды (биты) кода символа. Определить, какие именно, легко, если выписать двоичные коды символов разных регистров друг под другом.

```
— 0:    MOV     #12,R0
        EMT     16
        EMT     6
        EMT     16
        MOV     R0,R1
        MOV     '-',R0
        EMT     16
        MOV     R1,R0
        BIC     #240,R0
        EMT     16
        BR 0
        END
```

(Продолжение следует)

Завершаем публикацию заводских инструкций, известных среди пользователей БК под названием «ремонтная документация». Вниманию читателей предлагается описание устройства и функционирования БК-0010 (начало см. в № 3 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

Плата клавиатуры

Клавиатура представляет собой совокупность клавишных переключателей, собранных в матрицу из 8 вертикальных шин («Y») и 10 горизонтальных («X»). Шины соединяются с входами БИС D4 (X_i), которые через резистивную сборку E2 и резистор R6 подключены к цепи питания, и с входами-выходами (Y_i), которые через резистивную сборку E3 подключены к общей шине. Делитель, образованный одной из пар этих резисторов при нажатии клавиши, обеспечивает ввод ее кода в БИС.

Исходное состояние: на входах X — уровень +5 В, на входах Y — уровень общей шины. При нажатии какой-либо клавиши напряжение $\approx 0.9\ V_{CC}$ поступает на вход Y_i и вызывает срабатывание входных Y-триггеров, после чего на выходе Y_i устанавливается уровень $U^0_{вых} \leq 0.4$ В. Напряжение $U^0_{вых}$ через нажатую клавишу поступает на вход X_i, вызывая срабатывание входных X-цепей и БИС D4. Последняя формирует код символа, выставляет запрос на прерывание текущей программы, сообщает процессору адрес вектора прерывания и передает код символа. После отпускания клавиши БИС переходит в исходное состояние.

Контроллер клавиатуры ставит в соответствие каждой нажатой клавише ее код и передает этот код в микроЭВМ. Схема контроллера включает в себя следующие узлы:

- БИС К1801ВП1-014,
- триггер «строчный-заглавный» на микросхеме D3.1,
- схема формирования кода «ПРОБЕЛ» на микросхемах D2.1, D2.2,
- схема формирования сигнала «СТОП» на элементах D1.1, VD1, R1, R5, C2,
- RC-цепочка для защиты от дребезга клавиш R3, R4, C3, C4.

Триггер «строчный-заглавный», управляемый клавишами «СТР» и «ЗАГЛ», устанавливает и удерживает уровень на входе EC1 БИС D4:

- $U_{EC1}=1$ — программно формируются строчные символы,
- $U_{EC1}=0$ — программно формируются прописные символы.

Схема формирования кода «ПРОБЕЛ» (40_8) срабатывает при нажатии на клавишу «ПРОБЕЛ» и имитирует одновременное нажатие клавиш «ПР» и «0». Постоянная времени R2•C1 обеспечивает опережение воздействия сигнала «ПР», VD2 служит для развязки цепи «ПР».

Схема формирования сигнала «СТОП» вырабатывает короткий отрицательный импульс длительностью t=R1•C2 для организации радиального прерывания процессора при нажатии клавиши «СТОП».

RC-цепочка R4,C3 обеспечивает задержку выработки запроса на прерывание на время дребезга клавиш при их нажатии (длительность задержки равна R4•C3). RC-цепочка R3,C4 обеспечивает задержку установления схемы в исходное состояние на время дребезга при отпускании клавиш.

Блок питания

Структурная схема блока питания показана на рис. 16*, электрическая принципиальная схема приведена в Приложении.

---

* В данной статье, начало которой было опубликовано во втором выпуске журнала за 1994 г., сохранена сквозная нумерация рисунков и таблиц.

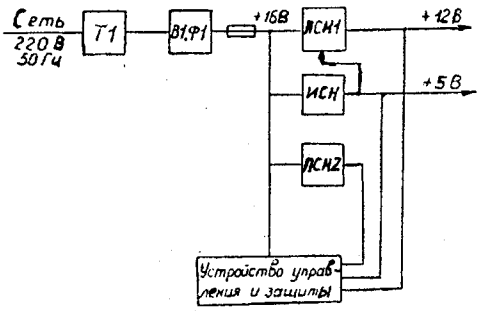


Рис. 16. Структурная схема блока питания

Напряжение сети (50 Гц, 220 В) поступает на *силовой трансформатор Т1*, вторичная обмотка которого нагружена на выпрямитель В1 с фильтром Ф1.

Выпрямитель В1 выполнен по мостовой схеме на диодах 1VD1—1VD4 и нагружен на емкостной фильтр Ф1, состоящий из конденсаторов 1С1, 1С2. Дроссель 1L1 в фильтре играет роль подавителя высокочастотных помех, возникающих при работе импульсного стабилизатора напряжения (ИСН) +5 В (для ограничения поступления помех в сеть). Аналогичное назначение имеет конденсатор С1.

Выпрямленное напряжение $+16^{+4}_{-2}$ В поступает на стабилизаторы напряжения.

Линейный стабилизатор напряжения (ЛСН1) +12 В/0.2 А выполнен на транзисторах 2VT6, 2VT7, 2VT9, 2VT11. Транзисторы 2VT7 и 2VT9 используются в схеме дифференциального усилителя постоянного тока (УПТ), а 2VT6, 2VT11 выполняют функцию регулирующего элемента (РЭ). Опорным напряжением ЛСН1 является выходное напряжение +5 В. Такое схемное решение обеспечивает появление напряжения +12 В позже +5 В, а длительность задержки определяется постоянной времени цепи 2R21, 2С8. Установка выходного напряжения +12 В производится переменным резистором 2R32 («+12 В»).

Импульсный стабилизатор напряжения (ИСН) выполнен по принципу релейного стабилизатора, представляющего собой автоматическую систему регулирования, в которой РЭ (транзистор 2VT5) переключается из открытого состояния в закрытое и обратно, когда изменяющееся во времени выходное напряжение достигает соответственно порога срабатывания или отпускания релейного элемента, управляющего РЭ. Частота переключения меняется в пределах 16—30 кГц.

Управление ИСН осуществляется микросхемой 2D2 (К142ЕП1), содержащей в своем составе источник опорного напряжения и схему порогового устройства.

В промежутки времени, когда транзистор 2VT5 закрыт, ток нагрузки поддерживается энергией, запасенной в дросселе 2L1, через коммутирующий диод 2VD1.

Установка выходного напряжения +5 В производится переменным резистором 2R19 («+5 В»).

Линейный стабилизатор напряжения (ЛСН2) выполнен по схеме эмиттерного повторителя с РЭ на транзисторе 2VT1. Опорным напряжением для ЛСН2 является напряжение, снимаемое с вывода 06 микросхемы 2D2. Установка вспомогательного напряжения +5 В осуществляется переменным резистором 2R9 («+5 В всп»).

Схема защиты выходных цепей +5 В и +12 В от перегрузки по току выполнена на элементах 2R2-2R5, 2С2, 2С3, 2D1, 2VT2, 2VT3. При превышении тока нагрузки более чем в 1.5 раза от номинального значения увеличившееся падение напряжения на датчиках тока 2R2, 2R3 открывает транзисторы 2VT2, 2VT3, и на вывод 05 микросхемы 2D1 поступает уровень «лог. 1», который переключает триггер (2D1.2, 2D1.3) и выключает источник +5 В путем подачи управляющего сигнала низкого уровня («лог. 0») на вывод 04 микросхемы 2D2. Выключение источника +5 В приводит к выключению источника +12 В.

Включение блока после снятия перегрузки может быть осуществлено путем выключения и повторного включения тумблера «СЕТЬ» с задержкой 5—10 с.

Защита выходных цепей +5 В, +12 В от перенапряжения осуществляется с помощью тиристора 2VD4. При включении тиристора (например, по причине пробоя транзистора 2VT5) выпрямитель В1 оказывается в режиме короткого замыкания, в

результате чего плавкий предохранитель перегорает и все стабилизаторы обесточиваются.

Порог срабатывания защиты от перенапряжения устанавливается делителями 2R22, 2R23, 2R25, 2R26. В качестве опорного напряжения используется +5 $В_{всп}$.

Защита срабатывает при превышении выходного напряжения +5 В или +12 В более чем на 20 ± 5% от номинального значения. При этом открывается соответствующий транзистор (2VT8 или 2VT10) схем сравнения и его коллекторный ток, усиленный 2VT12, включает тиристор.

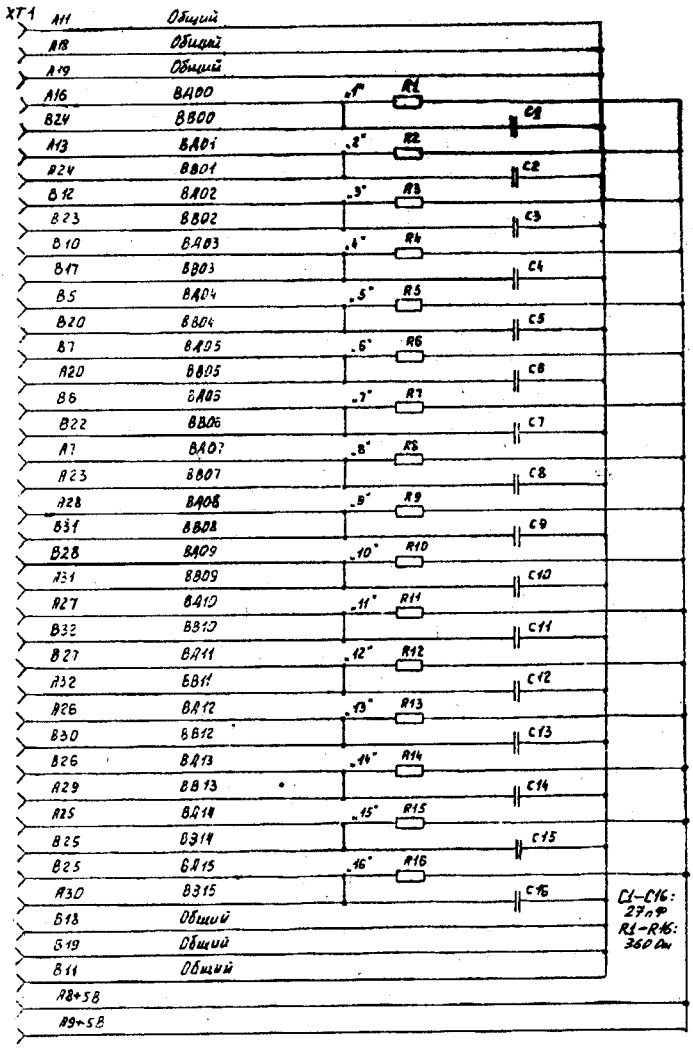


Рис. 17. Блок нагрузок

Блок нагрузок

Блок нагрузок обеспечивает резистивно-емкостную нагрузку выходных каскадов порта-источника и их соединение с соответствующими входами порта-приемника при прохождении тест-программ. Схема электрическая принципиальная блока нагрузок приведена на рис 17.

(Некоторые заводы поставляют в комплекте с БК последних лет выпуска блок нагрузок с максимально упрощенной схемой: выходы BD_i просто закорочены на входы BB_i отрезками монтажного провода. — *Прим. ред.*)

Системное программное обеспечение

Драйверы являются связующим звеном между программой—интерпретатором языка ФОКАЛ и аппаратурой микроЭВМ. Из интерпретатора ФОКАЛа обращение к драйверам осуществляется с помощью командных прерываний EMT (табл. 5) с заданным аргументом, определяющим требуемую функцию. (Исходный текст программы «монитор БК-0010(.01)», прошитой в ПЗУ по адресам $120000—137777_8$, и подробное описание к нему планируется опубликовать в одном из следующих номеров журнала. — *Прим. ред.*)

Работа драйверов. Отрабатывая требуемую функцию, драйвер обращается к конкретным ячейкам адресного пространства (аппаратным регистрам или ячейкам ОЗУ) и в определенной последовательности изменяет их состояние. (Например, запись «лог. 0» во все биты экранной памяти приводит к очистке экрана, а запись «лог. 1» в определенные биты используется для фор-

Таблица 5

Обозначение	Входные параметры	Выходные параметры	Назначение
Драйвер клавиатуры			
EMT 4	Нет	Нет	Инициализация драйвера клавиатуры
EMT 6	Нет	R0 (младший байт)	Чтение кода с клавиатуры
EMT 10	R1 — адрес буфера в ОЗУ, R2 — длина и символ-ограничитель	R1, R2	Чтение строки с клавиатуры
EMT 12	R0 — номер (1-10, 0 — отмена), R1 — адрес строки	Нет	Программирование ключа с заданным номером
Драйвер ТВ-приемника			
EMT 14	Нет	Нет	Инициализация драйверов системы
EMT 16	R0 — код символа (младший байт)		Вывод символа или переключение режимов вывода на экран
EMT 20	R1 — адрес строки, R2 — длина строки и символ-ограничитель	R1, R2	Вывод строки
EMT 22	R0 — код символа (0 — очистка), R1 — позиция по горизонтали	Нет	Вывод символа в служебную строку
EMT 24	R1, R2 - новые координаты курсора (X, Y)	Нет	Установка координат курсора
EMT 26	Нет	R1, R2 — координаты курсора (X, Y)	Определение координат курсора
EMT 30	R0 — рисование/стирание (1/0), R1, R2 — координаты (X, Y)	Нет	Рисование/стирание точки
EMT 32	R0 — рисование/стирание (1/0), R1, R2 — координаты конца (X, Y)	Нет	Рисование/стирание отрезка (начало — последняя нарисованная/стертая точка или конец последнего нарисованного/стертого отрезка)
EMT 34	Нет	R0 — режимы вывода на экран	Определение текущих режимов вывода на экран
Драйвер магнитофона			
EMT 36	R1 - адрес блока параметров	Нет	Все операции ввода-вывода на магнитную ленту
Драйвер последовательного канала			
EMT 40	R0 — номер скорости передачи	Нет	Инициализация драйвера и задание скорости передачи информации
EMT 42	R0 — код символа (младший байт)	Нет	Передача байта на линию

EMT 44	Нет	R0 — код символа (младший байт)	Прием байта с линии
EMT 46	R1 — адрес массива байтов в ОЗУ, R2 — длина массива	R1	Передача массива байтов на линию
EMT 50	R1 — адрес буфера в ОЗУ	R2 — длина принятого массива (байт)	Прием массива байтов с линии
EMT 52 — EMT 110			Резервные входы в подпрограммы по адресам $160000\text{-}160074_8$

Примечания

1. Более подробное описание EMT-функций приводится в книге: *Осетинский Л.Г., Осетинский М.Г., Писаревский А.Н.* ФОКАЛ для микро- и мини-компьютеров. Л.: Машиностроение, 1988.

2. О нестандартных EMT-функциях, не указанных в таблице и даже не предусмотренных создателями БК, рассказано в статье Д.Ю. Усенкова «Дополнительные возможности EMT-прерывания БК-0010». Вычислительная техника и ее применение. 1991. № 11.

мирования курсора, символов, точек или линий.) При вызове EMT-функций в драйверы через регистры процессора R0—R2 передается исходная информация. (Например, для установки курсора в верхний левый угол экрана нужно вызвать прерывание EMT 24, предварительно задав координаты местоположения курсора (X=0, Y=0): R1=0, R2=0.)

Драйвер отрабатывает функцию и возвращает управление ФОКАЛу.

Взаимодействие «оператор (пользователь) — драйверы». Код какой-либо клавиши может быть введен в микроЭВМ только в том случае, когда программа пользователя (или программа, зашитая в ПЗУ, например монитор или интерпретатор ФОКАЛа) использует оператор, обеспечивающий чтение символа с клавиатуры. При этом программа обращается к драйверу клавиатуры (команда EMT 6), который ждет нажатия клавиши пользователем, принимает код от аппаратной части микроЭВМ (считывает из регистра @#177662), анализирует его, обрабатывает (формирует из кода клавиши КОИ7 8-разрядный код) и передает обратившейся к драйверу программе. Каждой клавише с нанесенной на ней символикой соответствуют несколько кодов и один из двух адресов векторов прерывания, но каждому нажатию на данную клавишу соответствуют только один код и один адрес вектора (в зависимости от выбранного регистра: «НР», «ПР», «ЛАТ», «РУС», «ЗАГЛ», «СТР», «СУ»).

Некоторые коды в программу не передаются, а пересылаются ТВ-драйверу и служат для управления режимами вывода на экран.

Программа анализирует переданный драйвером код и проводит некоторые действия (например, отправляет код в ТВ-драйвер для индикации). Программа также может заменить код, полученный от драйвера клавиатуры, и послать на ТВ-драйвер другой код или совокупность кодов. Так, например, ФОКАЛ получает код 19 — «ВС», но передает в ТВ-драйвер последовательность кодов 8 — курсор влево, обеспечивая тем самым возврат в начало строки.

Программа может послать код в драйвер ТВ и вне связи с драйвером клавиатуры (например, можно включить режим «ИндСУ», передав в R0 при вызове EMT 16 код 130).

(Операция «чтение символа (кода) с клавиатуры», по сути, распадается на две отдель-

ные: «загрузка кода в микроЭВМ по прерыванию» и «чтение кода в программу». Когда пользователь нажимает какую-либо клавишу, БИС клавиатуры вырабатывает запрос на прерывание по вектору 60_8 или 274_8 (в зависимости от нажатия клавиши «НР» одновременно с выбранной знаковой). Если прерывание от клавиатуры разрешено, управление передается одному из двух драйверов клавиатуры (нижнего регистра, @#60, или верхнего, @#274). Драйвер считывает код КОИ7 из буфера клавиатуры @#177662 и программно обрабатывает его, формируя 8-разрядный код в зависимости от состояния флага «РУС»/«ЛАТ» (байт @#43) или полуграфику (драйвер @#274). «Заглавные» и «строчные» коды вырабатываются аппаратно.

Сформированный код драйвер записывает в буферную ячейку-байт ОЗУ (@#104) и в флаговую ячейку-байт (@#105), после чего заканчивает работу (производится возврат из прерывания). Некоторые же коды сразу передаются в подпрограммы ТВ-драйвера, не попадая в ячейку @#104.

Вторая часть операции чтения начинается, когда процессор встречает в программе на ассемблере (это может быть программа пользователя или реализующая оператор «чтение кода с клавиатуры» подпрограмма интерпретатора ФОКАЛа) команду чтения кода ЕМТ 6. Подпрограмма реализации ЕМТ 6 вначале проверяет наличие еще не считанного кода в ячейке @#104 и, если он есть, возвращает его вызвавшей программе. Иначе подпрограмма ЕМТ 6 ждет, пока будет нажата какая-либо клавиша, произойдет прерывание, сформируется и появится в ячейке @#104 «свежий» код. И после этого код возвращается в программу, вызвавшую ЕМТ-функцию. Правда, следует учитывать, что реальный механизм работы ЕМТ 6 несколько сложнее, нежели рассмотренный выше, и обеспечивает реализацию табуляции и программируемых ключей. — Прим.ред.)

Драйвер клавиатуры предназначен для управления работой клавиатуры и выполняет следующие функции:

- установка режима работы клавиатурной БИС;
- чтение кодов символов из регистра данных клавиатуры;
- перекодировка из стандарта КОИ-7 во внутренние 8-разрядные коды;
- передача кодов драйверу телевизионного приемника или в основную программу.

Драйвер телевизионного приемника обеспечивает формирование и отображение на экране алфавитно-цифровой и графической информации. Растр 512•256 точек позволяет формировать 25 информационных (текстовых) строк, одна из которых является служебной и предназначена для индикации текущих режимов работы клавиатуры и вывода на экран, а также для отображения прочей служебной информации, заданной пользователем. В каждой строке в зависимости от заданного режима может размещаться 32 (матрица 16•10 точек) или 64 символа (8•10 точек). Символы программно формируются на экране на черном или белом фоне, в прямом или инверсном виде, с подчеркиванием или без него. (Режимы формирования символов задаются клавишами или путем передачи соответствующих кодов из программы в ТВ-драйвер.)

В драйвере реализован режим автоматического сдвига информации вверх или вниз при достижении курсором границ экрана (режим рулона).

Драйвер кассетного магнитофона (КМ) предназначен для работы с бытовым кассетным магнитофоном типа «Электроника-302».

Для записи информации на ленту используется разновидность метода широтно-импульсной модуляции, который обеспечивает скорость записи и чтения информации 1200 бод. Плотность записи при такой скорости составляет 25 бит на 1 мм. Информация на ленте (емкость кассеты МК60-1 $\approx$ 0.5 Мб) размещается в виде файлов произвольной длины, имеющих следующую структуру:

- настроечная последовательность в начале записи — ряд импульсов с длительностью, соответствующей «лог. 1», который заканчивается маркером;
- служебная информация: адрес (2 байта), длина (2 байта) и имя файла (16 байт);
- массив байтов указанной длины;
- контрольная сумма (2 байта).

Каждый бит информации записывается двумя импульсами, первый из которых определяет его тип («1»/«0»), а второй является синхронизирующим.

Настроечная последовательность предназначена для задержки на время стабилизации переходных процессов, возникающих в начале записи, а при чтении используется для настройки на скорость, с которой был записан данный файл. Таким образом, драйвер позволяет работать с файлами, записанными на ленту с различной скоростью.

Стандартный драйвер магнитофона, зашитый в ПЗУ, позволяет производить запись на ленту содержимого указанной области памяти (с заданием имени и длины), чтение файла с указанным именем с ленты в заданную область памяти и фиктивное чтение, при котором происходит только поиск конца указанного файла.

При записи и чтении двигатель магнитофона автоматически запускается перед началом операции и останавливается по ее завершении. Кроме того, драйвер позволяет осуществлять запуск двигателя и его останов по соответствующей команде.

Запись производится с того места на ленте, которое в данный момент оказалось под головкой магнитофона. При чтении осуществляется поиск файла на ленте по заданному имени. Если имя встреченного файла не совпадает с заданным, то формируется соответствующий ответ и управление возвращается вызвавшей программе. Для продолжения поиска требуемого файла управление нужно вновь передать драйверу. Для прекращения выполнения любой операции надо нажать клавишу «СТОП».

Обращение к драйверу магнитофона производится по команде EMT 36.

Работа драйверов при включении микроЭВМ. При начальном запуске процессора на микропрограммном уровне запрещаются прерывания с низким приоритетом (в слово РСП заносится константа 340_8). Процессор считывает из регистра 177716_8 (SEL1) адрес начала системной программы 100000_8 и запускает ее (адрес соответствует модулю 1 ПЗУ). Программа начинается с безусловного перехода на адрес 100260_8. Далее указатель стека устанавливается на адрес 1000_8.

Ниже перечислена последовательность обращения к регистрам и ячейкам ОЗУ микроЭВМ:

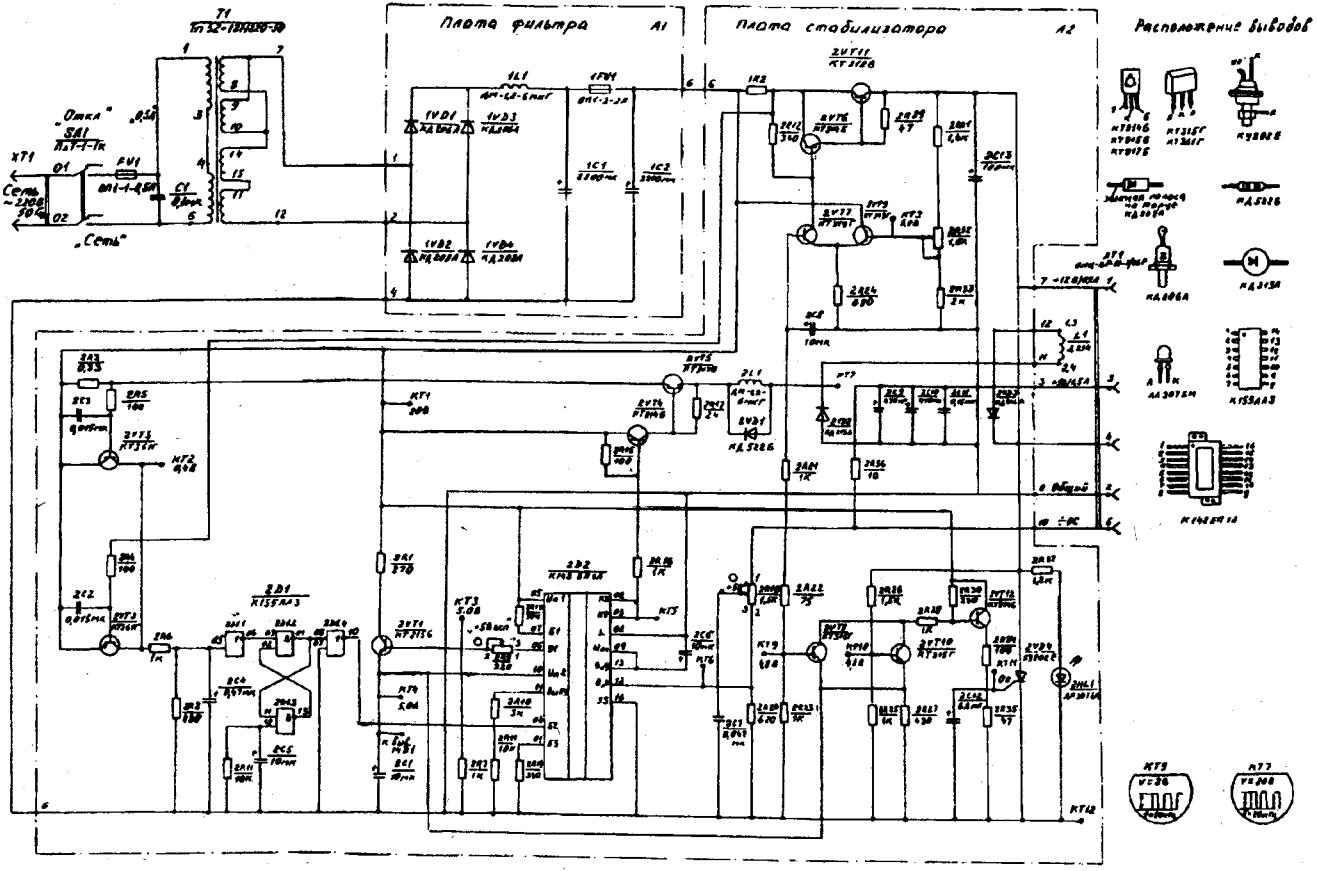
- установка (копирование констант из ПЗУ) векторов прерывания по зависанию, от клавиатуры и по командам EMT;
- установка маски в регистре режима БИС D4;
- установка некоторых рабочих ячеек в диапазоне адресов 0—400_8;
- запись числа 1330_8 в регистр смещения БИС D19;
- очистка экрана, формирование курсора и служебной строки;
- обнуление порта пользователя (@#177714);
- запись константы 220_8 (останов магнитофона) в системный порт (@#177716);
- разрешение прерываний (РСП=0);
- передача управления интерпретатору ФОКАЛа (переход на подпрограмму по адресу 120000_8).

Если модуль 2 (ФОКАЛ) отсутствует, то происходит прерывание по зависанию; при этом осуществляется передача управления на пусковой монитор (модуль 1 ПЗУ), который нормализует указатель стека (устанавливает на адрес 1000_8), заносит в системный порт константу 220_8 и печатает на экране "?", после чего ожидает ввод директив пускового монитора. Иначе запускается интерпретатор ФОКАЛа, устанавливающий ячейку с адресом 262_8 в состояние 177777 (не нулевое), что при дальнейшей работе драйвера клавиатуры является признаком наличия в системе языка ФОКАЛ.

Пусковой монитор является средством начальной загрузки микроЭВМ. Например, можно считать с магнитной ленты какую-либо программу в машинных кодах и запустить ее. При этом установка второго и четвертого модулей ПЗУ не требуется.

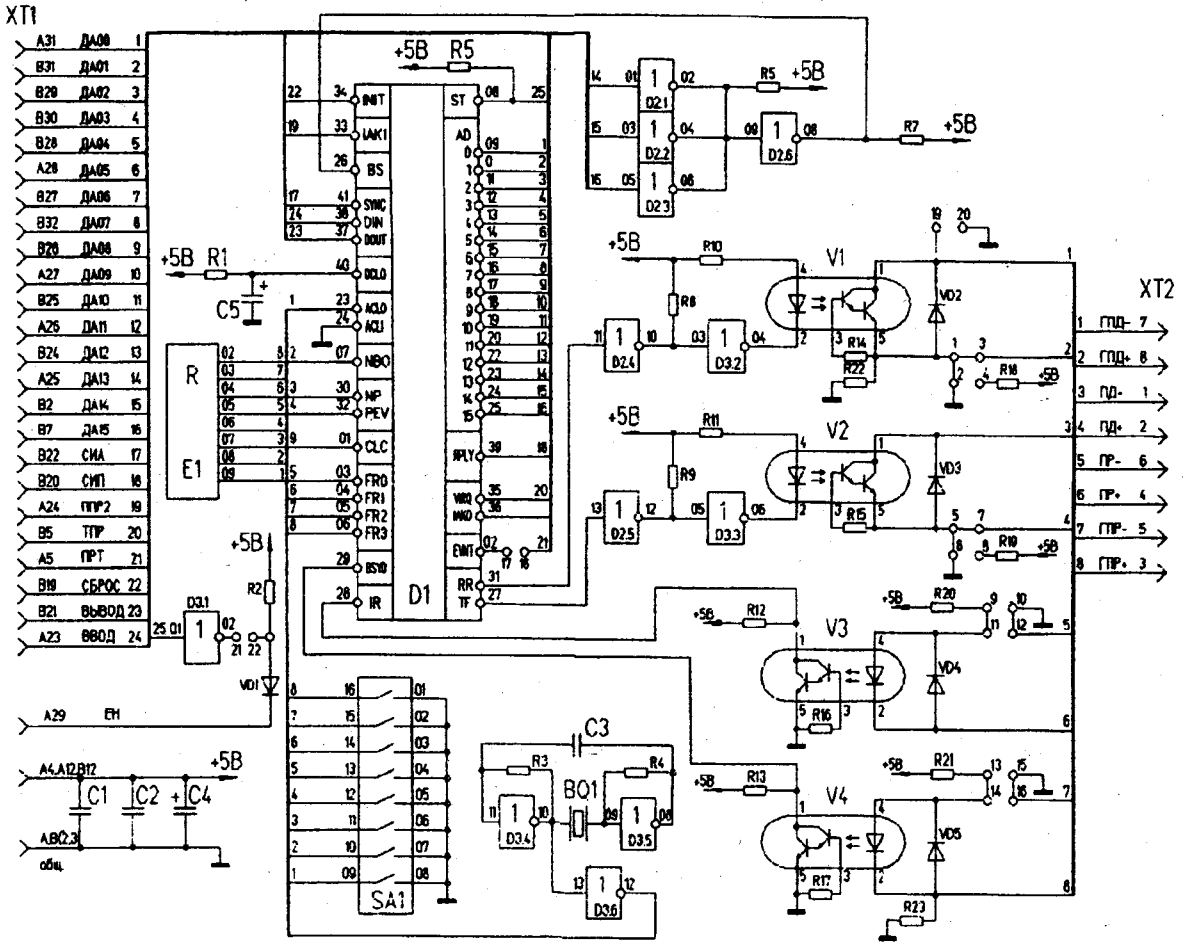
Программы должны быть написаны в системе команд «Электроника 60» с учетом адресного пространства «Электроника БК-0010».

Электрическая схема блока питания



Примечание. К сожалению, редакция оригиналом лучшего качества не располагает.

Схема электрическая принципиальная блока ИРПС



СПРАВОЧНЫЙ ЛИСТОК

ВНУТРЕННИЕ РАЗЪЕМЫ КЛАВИАТУРЫ БК-0010

Разъем	Номера выводов разъема															
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
XT1	X0	X1	X2	X3	X4	X5	X6	Y8	X7	X8	X9	не исп.	ЗАГЛ	СТР	ПРОБ	+5 В
XT2	Y1	Y2	Y3	Y4	Y5	Y6	Y7	Y9	СУ	ПР	НР	ИНД	не исп.	ОСТ1	ОСТ2	+5 В

Примечание: Y8 — общий провод, ИНД — цепь звуковой индикации, ОСТ1 и ОСТ2 — выводы клавиши СТОП.

ФУНКЦИИ, ОТРАБАТЫВАЕМЫЕ БК-0010 ПРИ ЗАМЫКАНИИ СООТВЕТСТВУЮЩИХ ЦЕПЕЙ X-Y

	X0	X1	X2	X3	X4	X5	X6	X7	X8	X9
Y1	ПОВТ	ТАБ	КРАСНЫЙ	ВПРАВО	1	9	A	I	Q	Y
Y2	ИНД.СУ	ВВОД	КУРСОР В НАЧАЛО ЭКРАНА	ВВЕРХ	2	:	B	J	R	Z
Y3	КТ	СБРОС КОНЦА СТРОКИ	ВС	ВНИЗ	3	;	C	K	S	[
Y4	БЛ.РЕД	СБР/РП	ГТ	ВЛЕВО-ВВЕРХ	4	,	D	L	T	\
Y5	ГРАФ	УСТ.ТАБ	ПЕРЕВОД СТРОКИ	ВПРАВО-ВВЕРХ	5	-	E	M	U	]
Y6	ЗАП	РУС	СДВИЖКА	ВПРАВО-ВНИЗ	6	.	F	N	V	↑
Y7	СТИР	ЛАТ	РАЗДВИЖКА	ВЛЕВО-ВНИЗ	7	/	G	O	W	_(Ъ)
Y8	ШАГ	ВЛЕВО	СБР.ТАБ	ЗАБОЙ	0	8	@	H	P	X

Примечание. Функции указаны для регистра ЛАТ ЗАГЛ, для отработки СУ, НР, АР2, ПРОБЕЛ, СТР, ЗАГЛ необходимо замкнуть соответствующую шину разъемов XT1-XT2 с Y8 (общим проводом).

(Вышеприведенные таблицы печатаются по материалам журнала «Информатика и образование» № 4 за 1991 г. с незначительными исправлениями. — Прим.ред.)

МОСКОВСКИЙ КЛУБ БК

П. В. Петров,
г. Москва

ДОПОЛНИТЕЛЬНОЕ ОЗУ ДЛЯ БК-0010(.01)

В журналах уже не раз публиковалась информация о дополнительном ОЗУ для БК-0010, но, к сожалению, эти статьи были основаны на устаревших идеологиях и схемных решениях, рожденных 3—4 года назад. Эти ДОЗУ были рассчитаны на работу только с магнитофоном. При этом основным желанием было «ну хоть немного добавить памяти...» при работе вне БЕЙСИКа, жалко ведь терять целых 24 Кб. Заводские же разработки ДОЗУ (например, завода «Эридан»), по-видимому, проектировались по принципу «ОЗУ вообще»: не было учтено ни удобство работы с ними, ни мнения программистов и, что самое важное, не предполагалась их работа совместно с дисководом и с использованием режима «турбо».

Попытки расширения памяти БК-0010 за счет установки ППЗУ с зашитыми в них операционными системами и другими вспомогательными и обслуживающими программами были популярны некоторое время за счет их относительно малой стоимости и простоты установки на ЭВМ. Основными недостатками такого способа являются жесткая привязка к одной из операционных систем (может быть, не к лучшей из них), невозможность перераспределения памяти ЭВМ и замены старой версии ОС на более современную и т. д. Таким образом, сегодня расширение с помощью установки в БК-0010 ППЗУ неперспективно.

Одно время предлагалась существенная переделка основной платы БК-0010 и установка на нее 128 Кб ОЗУ, в результате чего получалась и уже не БК-0010, и далеко не БК-0011М. Описываемая идея благополучно почила, не получив поддержки пользователей, на основании того, что машину после такой доработки никто уже не брался ремонтировать.

Сегодня, наверное, даже пользователи с дальней периферии уже знают, что БК-0010 прекрасно работает с дисководом (кто попробовал — уже никогда не вернется к магнитофону), но при этом ей необходимо дать как минимум 16 Кб ДОЗУ (на статических микросхемах памяти, что дает возможность сохранения информации при отключении питания ЭВМ). И не где-нибудь, а в области адресов от 120000 до 157776 с подавлением ПЗУ «штатного» БЕЙСИКа. Читатель возразит: а как же БЕЙСИК, тоже ведь жалко! Но и эта задача уже решена, и у разработчиков и изготовителей ДОЗУ уже считается «признаком хорошего тона» снабжать свои изделия возможностью работы со штатным ПЗУ-БЕЙСИКом. А так как ДОЗУ уже не мыслится без дисковода, то и размещают его в корпусе КНГМД (контроллера дисковода), благо место есть. Подобная «родственность»

Организация ДОЗУ 16 Кб (старый стандарт)

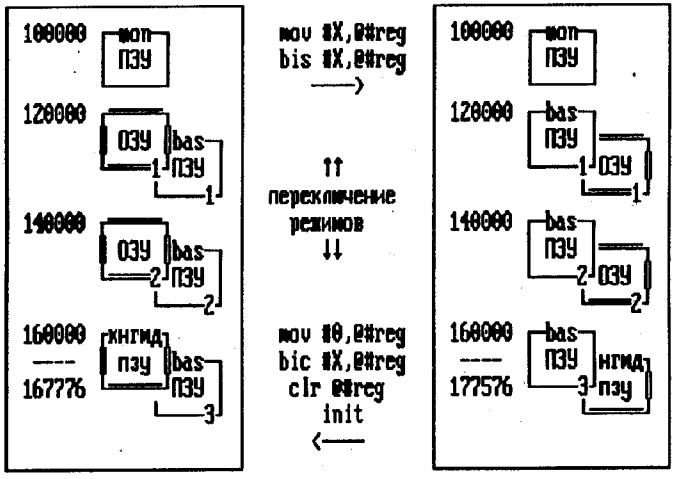


Рис. 1. Конфигурация части ОЗУ ЭВМ при включении питания

Рис. 2. Конфигурация части ОЗУ ЭВМ при инициализации БЕЙСИКа

Организация ДОЗУ 32 Кб (новый стандарт)

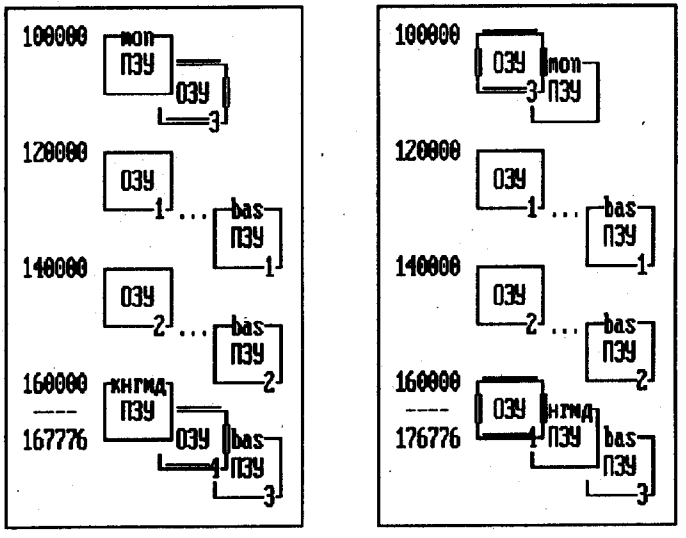


Рис. 3. Конфигурация части ОЗУ ЭВМ при включении питания

Рис. 4. Конфигурация части ОЗУ ЭВМ при инициализации режима 1

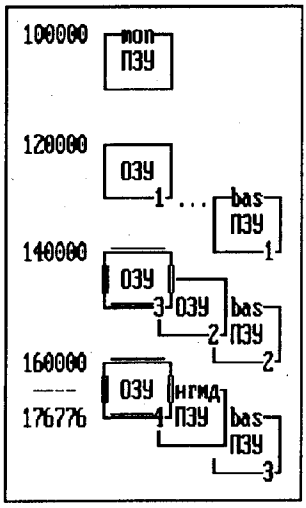


Рис. 5. Конфигурация части ОЗУ ЭВМ при инициализации режима 2

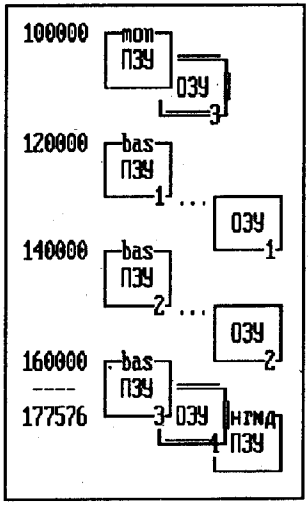


Рис. 6. Конфигурация части ОЗУ ЭВМ при инициализации БЕЙСИКа

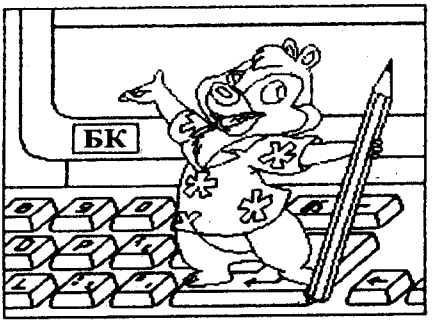
этих двух устройств позволила довольно легко схемно реализовать программное переключение с ДОЗУ на ПЗУ и обратно, в результате чего БЕЙСИК получил потенциальную возможность работы и с дисководом, и с магнитофоном.

Учитывая недостатки вышеописанных устройств, был разработан новый тип ДОЗУ для БК-0010, ставший сегодня своего рода стандартом.

Система дополнительного ОЗУ 32 Кб позволяет иметь на БК-0010 кроме уже описанных выше 16 Кб ОЗУ еще две страницы памяти, причем местоположение новых страниц можно варьировать, исходя из потребностей. Изначально конфигурация ДОЗУ 32, представленная на рис. 3, имеет две страницы ОЗУ по 8 Кб с начальными адресами 120000_8 и 140000_8. Помимо этого существуют еще две такие же страницы, находящиеся в неактивном состоянии. Система ДОЗУ 32 позволяет подключать изначально неактивные страницы двумя способами, первый из которых показан на рис. 4. Здесь третья и четвертая страницы ОЗУ одновременно подменяют две микросхемы ПЗУ с начальными адресами 100000_8 и 160000_8 (ПЗУ при этом отключаются). Второй способ представлен на рис. 5: страница ОЗУ по адресу 140000_8 отключается, и на ее место подключается первая из ранее неактивных станиц ОЗУ. Область ПЗУ по адресу 160000_8 подменяется второй страницей ОЗУ аналогично предыдущему случаю.

Включение трех ПЗУ БЕЙСИКа полностью аналогично варианту ОЗУ 16 Кб, а получившаяся конфигурация представлена на рис. 6. (Обратите внимание на объем старшей страницы при включении по вариантам 1 и 2.)

При установке ДОЗУ на БК-0010 вам уже никогда не понадобится включать блок МСТД, так как ФОКАЛ, например, загружается из операционной системы и, аналогично БЕЙСИКу, тоже умеет работать с дисководом. Переход из одного языка в другой, выход в операционную систему и обратно (в любой из языков) упрощается практически до выбора пункта в меню. Собственно говоря, расширение ДОЗУ 32 превращает БК-0010(.01) в ЭВМ класса ДВК с ОЗУ 63,5 Кб.



БРЯНСКИЙ КЛУБ БК

Р. Аскеров,
г. Брянск

ПОДКЛЮЧЕНИЕ ПЕРИФЕРИЙНЫХ УСТРОЙСТВ К ПОРТУ БК

В нижеприведенной таблице указана распайка некоторых периферийных устройств при их подключении к порту ввода-вывода БК (разъем «УП»).

Распайка мыши, плоттера и музыкального блока «Менестрель» стандартизирована. Распайка принтера соответствует используемой в большинстве случаев (для большинства драйверов). Что же касается джойстиков, здесь представлен лишь один из вариантов их подключения (два джойстика одновременно).

Порт		Мышь	Джойстик	Плоттер	Принтер	Менестрель
разряд	контакт					
ВВ00	В24		1			
ВВ01	А24		1			
ВВ02	В23		1			
ВВ03	В17		1			
ВВ04	В20		1КН0			
ВВ05	А20	КН1	1КН1			
ВВ06	В22	КН2	1КН2			
ВВ07	А23			READY		
ВВ08	В31		2		BUSY	
ВВ09	А31		2			
ВВ10	В32		2			
ВВ11	А32		2			
ВВ12	В30		2КН0			
ВВ13	А29		2КН1			
ВВ14	В29		2КН2			
ВВ15	А30					READY
ВД00	А16			PEN	DATA0	DATA0
ВД01	А13			MOVE	DATA1	DATA1
ВД02	В12			X/Y	DATA2	DATA2
ВД03	В10	СБР		+/−	DATA3	DATA3
ВД04	В5				DATA4	DATA4

ВД05	В7				DATA5	DATA5
ВД06	В6				DATA6	DATA6
ВД07	А7			TEST	DATA7	DATA7
ВД08	А28				STROBE	ADDR0
ВД09	В28					ADDR1
ВД10	А27					SEL1
ВД11	В27					SEL2
ВД12	А26					WRITE
ВД13	В26					
ВД14	А25					
ВД15	В25					STROBE
ПРТ	В1					TIMER
+5 В GND	АВ8 АВ11	+5 В GND	ОБЩИЙ	GND	GND	+5 В GND

П. Ивайлов

ПОДКЛЮЧЕНИЕ ТЕРМОПРИНТЕРА МС6302 К БК-0010.01

Термопечатающее устройство (ТПУ) МС6302 предназначено для вывода на специальную термобумагу только символьной информации в кодах стандарта ASCII. Данное ТПУ по своим техническим возможностям и эксплуатационным показателям можно отнести к разряду примитивных печатающих устройств, не устраивающих вкус большинства пользователей. Оно имеет только две кнопки: «СЕТЬ» и «ПРОГОН». (Тем не менее, за отсутствием лучшего МС6302 вполне можно использовать на БК.) Связь ТПУ с компьютером осуществляется с помощью параллельного интерфейса ИРПР-М, основой которого является программируемый порт ввода-вывода (разъем ХТ5 УП, регистр с адресом &О177714). Принтер соединяется с БК при помощи 14-жильного ленточного кабеля, распаянного на разъемы ГРПМШ-1 (прилагаемый к ПУ) и СНП 58-64/94х9В-23-В (входящий в комплект БК) соответственно приведенной ниже таблице. Как и любой другой принтер, ТПУ нуждается в драйвере — программе, обслуживающей порт ввода-вывода БК. Он имеется, например, в ПЗУ БЕЙСИКа по адресу &О134626 (команда LLIST). При печати из монитора или из программ в кодах необходимо использовать драйвер, аналогичный реализованному в БЕЙСИКе. Для ТПУ также подойдут драйверы для принтеров УВВП4 30-004, МС6312, МС6313 (кроме печати графической копии экрана). Данное ТПУ может работать и с оболочкой Disk Master (в режимах User, View, Menu), входящей в состав ANDOS v2.50$.

Таблица соответствия разъемов БК И ПУ

Номер контакта УП БК	Номер контакта МС6302
А16	Б9
А13	Б10
В12	Б11
В10	Б12
В5	Б13
В7	Б14
В6	Б15
А28	Б6
В31	Б7
А23	Б5
(АВ) 11, 18	Б4, Б23, А22

Р. Аскеров,
г. Брянск

ЗАМЕНА ОПЕРАТОРА IF В БЕЙСИКе БК-0010

Как и многие другие языки высокого уровня, БЕЙСИК-БК позволяет присваивать переменным значения логических выражений. Но в этом языке нет специальных логических переменных — их роль выполняют обычные (числовые). Если значение переменной равно 0, то в составе логического выражения оно трактуется как «ложь», а в любом другом случае — как «истина». (Следует отметить, что результатом логического выражения может быть только 0 — «ложь» или -1 — «истина».) Из всего вышеизложенного следует способ замены цепочки операторов IF вида

IF <условие A> THEN M=A
IF <условие B> THEN M=B
IF <условие N> THEN M=N

одним арифметическим выражением

M=-(условие A)*A+(-(условие B)*B)...+(-(условие N)*N), откуда следует, что

M=-(условие A)*A-(условие B)*B...-(условие N)*N, или

M=-((условие A)*A+(условие B)*B...+(условие N)*N).

Пусть истинным окажется <условие A>, тогда (**условие A**) примет значение -1, а (**условие B**), (**условие C**) и др. — 0. Отсюда

M=-(-1)*A-0*B-0*C... = -(-1)*A = 1*A = A,

что равносильно выполнению всей ранее упомянутой цепочки IF.

Логические выражения в строке желательно взять в скобки, иначе из-за несоответствия приоритетов логических операций могут возникнуть ошибки. Так, если вместо **-(XY)*A** записать **-XY*A**, результат будет далеко не один и тот же.

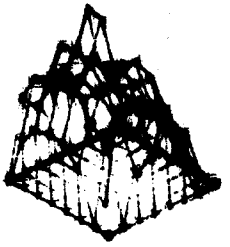
Аналогичным образом можно добиться эффекта выполнения оператора CASE (выбор). Вот фрагмент подпрограммы реакции на нажатия клавиш, написанной новым способом:

```
10 A%=-(KEY%=8%)*1%-(KEY%=25%)*2%-(KEY%=26%)*3%-
       (KEY%=27%)*4%-(KEY%=32%)*5%-(KEY%=10%)*6%
20 ' 8 - влево, 25 - вправо, 26 - вверх, 27 - вниз,
       32 - пробел, 10 - ввод
30 ON A% GOTO 100,200,300,400,1000,2000
40 ' 100 - влево, 200 - вправо, 300 - вверх, 400 - вниз,
     1000 - выстрел, 2000 - оружие
```

Вместо шести строк здесь используются две (если не считать комментариев) и не тратится время на выполнение операторов IF. Конечно, такой метод программирования применим на любой машине, но именно на БК он особенно актуален в силу значительной экономии памяти.

Все материалы, опубликованные в рубрике «Брянский клуб пользователей БК», любезно предоставлены редакции руководителем клуба Р. Аскеровым. Адрес для переписки: 241047, Брянск-47, а/я 109, Брянский клуб пользователей БК.

Редакция приглашает к сотрудничеству все клубы и прочие объединения любителей БК. Ваши статьи — советы, рекомендации, описания новых программных и аппаратных разработок, справочные сведения и другие материалы о БК — благодаря публикации на страницах журнала «Персональный компьютер БК-0010 — БК-0011М» смогут стать бесценной копилкой опыта для вас и ваших коллег-БКманов, позволят различным клубам БК обменяться полезной информацией и послужат приглашением к объединению в составе клуба индивидуальных пользователей БК.



Описывается схема и конструкция многоканального усилителя (точнее преобразователя сигналов) и управляемого от него простейшего графопостроителя, которые могут служить примерами типовых устройств для изготовления в школьных кружках, в научно-исследовательской аботе студентов, а также всеми другими пользователями БК.

Ю. В. Котов,
г. Москва

ГРАФОПОСТРОИТЕЛЬ ДЛЯ БК-0010 — СВОИМИ РУКАМИ

Хотя малые ПЭВМ с ограниченным объемом памяти, например БК-0010, в наше время нельзя считать универсальными, они вполне пригодны для решения различных специальных задач, в профессиональной среде, в учебном процессе, при управлении различными приборами и установками. Для БК-0010 задачу сопряжения с разнообразными внешними устройствами облегчает наличие несложно программируемого 16-разрядного параллельного канала, тогда как для ряда других моделей их разработчики ограничиваются 8 разрядами. При небольшой доработке БК возможно использование и последовательного интерфейса.

В практике использования БК-0010 в учебном процессе проводились работы по их сопряжению с большими графопостроителями: ЕС-7052 (рулонный, входящий в комплект машин серии ЕС, отечественного производства) и «Дигиграф» (профессиональный планшетный построитель чехословацкого производства). При этом подключение ПЭВМ возможно как непосредственно к построителю, так и путем переноса данных на магнитной ленте. В последнем случае БК-0010 соединяется с устройством подготовки данных на магнитной ленте ЕС-9004; это же устройство или аналогичное соединяется (как это реализовано во многих организациях) с графопостроителем и позволяет, таким образом, осуществлять его работу с лентой, записанной как на ЕС ЭВМ, так и на малой ПЭВМ.

Сегодня БК-0010 нередко является основой для построения учебных, экспериментальных или игровых автоматизированных систем и установок. Создание таких систем и работа с ними открывают широкое поле деятельности для технического творчества, производственного обучения.

Усилитель в рассматриваемом варианте имеет девять одинаковых каналов, подключается к параллельному порту вывода БК-0010 и является, вообще говоря, многоцелевым, но в первую очередь он рассчитан на управление шаговыми двигателями графопостроителя. Выходное напряжение усилителя — 24 В, допустимый диапазон изменения силы тока в выходной цепи (в зависимости от сопротивления последней) — 0.5...1 А. Каждый канал построен на двух транзисторах — КТ315В и П605А (вариант: КТ339А, П214 и др.), первый каскад выполняет функции инвертора сигнала с порта БК, второй — ключ, управляемый поступающим через инвертор от БК сигналом 5 В и манипулирующий напряжением питания 24 В. Питание осуществляется от трансформатора ТАН104, с обмоток которого (в том числе соединенных параллельно) снимаются напряжения 5 В и 24 В. Для большей наглядности в схему включены контрольные индикаторы (в данном варианте использованы миниатюрные лампочки на 6...9 В, поэтому они подключаются через гасящие резисторы, мощность и сопротивление ко-

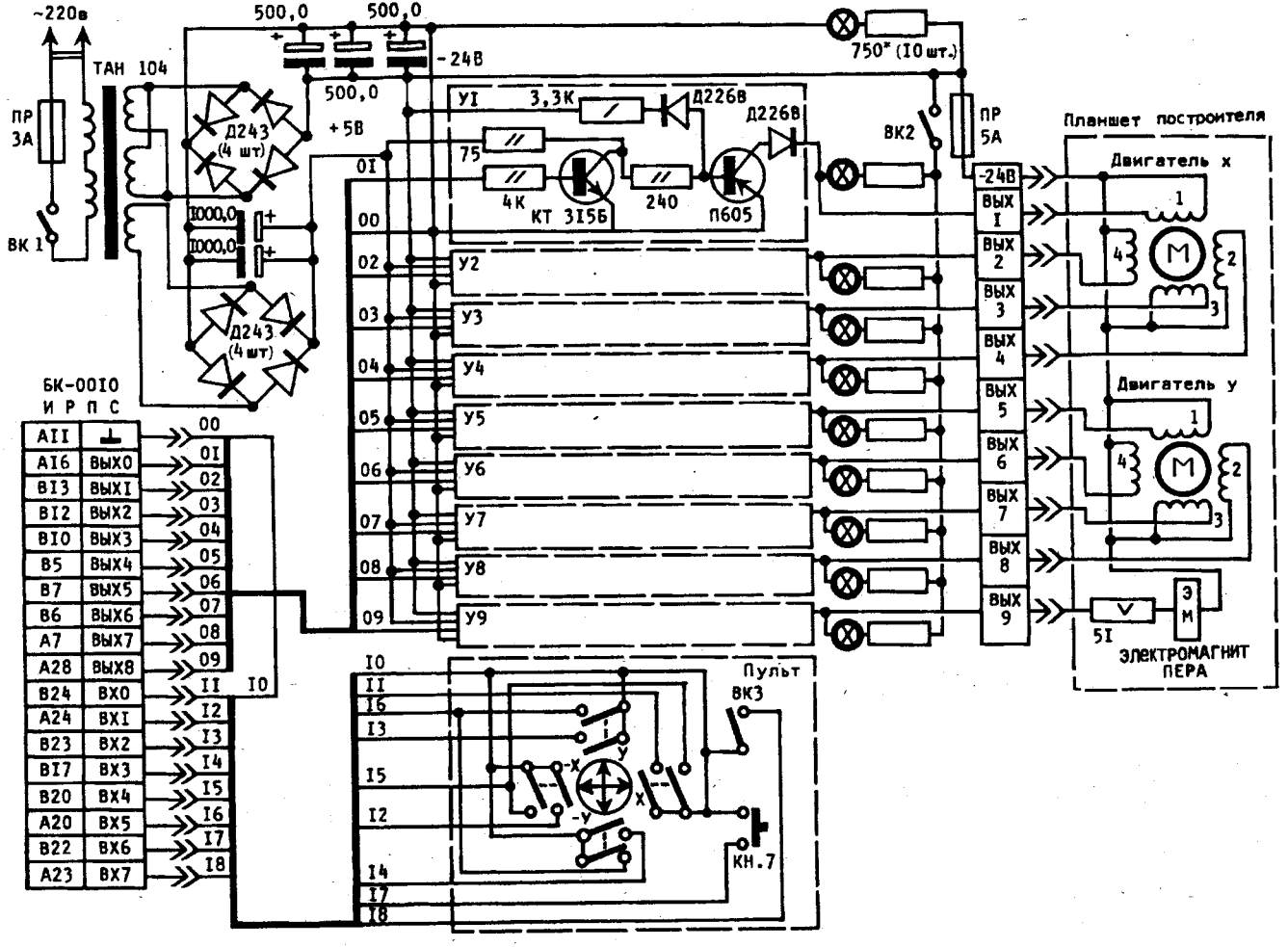

~220в
ТАН 104
ПР 3А
ВК 1
Д243 (4 шт)
Д243 (4 шт)
1000,0
1000,0
500,0
500,0
500,0
-24В
+5В
750* (10 шт.)
ВК2
ПР 5А
У1
3,3К
Д226В
Д226В
75
4К
КТ 315Б
240
П605
01
00
02
03
04
05
06
07
08
09
У2
У3
У4
У5
У6
У7
У8
У9
-24В
ВЫХ 1
ВЫХ 2
ВЫХ 3
ВЫХ 4
ВЫХ 5
ВЫХ 6
ВЫХ 7
ВЫХ 8
ВЫХ 9
Планшет построителя
Двигатель х
Двигатель у
1
2
3
4
М
5I
ЭЛЕКТРОМАГНИТ ПЕРА
БК-0010
И Р П С
АII
АI6
ВI3
ВI2
ВI0
В5
В7
В6
А7
А28
В24
А24
В23
ВI7
В20
А20
В22
А23
ВЫХ0
ВЫХI
ВЫХ2
ВЫХ3
ВЫХ4
ВЫХ5
ВЫХ6
ВЫХ7
ВЫХ8
ВХ0
ВХI
ВХ2
ВХ3
ВХ4
ВХ5
ВХ6
ВХ7
00
01
02
03
04
05
06
07
08
09
II
I2
I3
I4
I5
I6
I7
I8
I0
Пульт
ВК3
КН.7
X
-X
Y
-Y

Рис. 1а

торых необходимо подобрать в зависимости от типа используемых ламп). Схема усилителя приводится на рис.1а.

К усилителю можно дополнительно подсоединить линейку переключателей, что позволит управлять одними каналами от ПЭВМ, а другими, если требуется, — напрямую, контактными выключателями или датчиками.

Сигналы с выходов каналов усилителя можно использовать для управления электромеханическими реле распространенных 24-вольтовых типов, подключаая через них к БК любые нестандартные внешние устройства. Однако их основное назначение — питание обмоток шаговых двигателей и катушки электромагнита подъема пера графопостроителя. Применяются двигатели типа ШДА-3Ф, номинально они рассчитаны на 27 В, но здесь питаются несколько пониженным напряжением. Такой двигатель диаметром 42 мм (по фланцу 60 мм) и высотой 60 мм имеет четыре обмотки, соединяемые «звездой». Его магнитный ротор проворачивается шагами по 22.5^{o}. Обмотки при этом включаются попарно, для каждого шага переключается одна пара, например: 1-2, 2-3, 3-4, 4-1, 1-2... Таким образом, для управления двумя двигателями требуется восемь каналов. (Надо сказать, что управление двигателями от ПЭВМ только через усилитель — несколько неэкономный способ в отношении количества используемых разрядов выходного порта, но это максимально упрощает схему. При инкрементном же способе управления для каждого двигателя достаточно двух разрядов, через которые посылаются импульсы для шагов в прямом и обратном направлениях. В схему в этом случае надо включить так называемые кольцевые коммутаторы и промежуточные регистры, хранящие напряжения для нужных обмоток. Для упомянутых 4-обмоточных двигателей (а бывают еще 6-, 12-обмоточные и др.) такие коммутаторы достаточно просты и могут быть собраны на микросхемах К155ТМ2 или им подобных.)

Промежуточное решение (если учесть, что у каждого двигателя при работе всегда включена одна из обмоток 1 и 3 и одна из обмоток 2 и 4) получается, если в приведенной на рис. 1а схеме подать сигнал на вход усилителя У3 через дополнительный

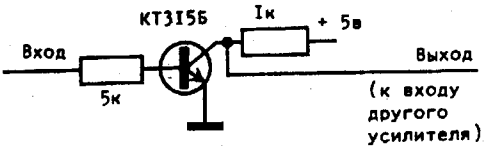


Рис. 1б

инвертор от входа усилителя У1 (рис. 1б) и на вход усилителя У4 через еще один инвертор от входа У2 (аналогично — для другого двигателя). Таким образом, для двух двигателей будет задействовано четыре разряда. Правда, при этом, когда процесс черчения закончится, не удастся программным путем отключать питание всех обмоток двигателей.

Один из каналов используется для включения-выключения (т. е. подъема-опускания) пера построителя (для нескольких перьев можно добавить дополнительные каналы усиления). Электромагнит включения использован в данном случае от заводского построителя АП-7251 или ЕС-7052. Гнездо, куда ввинчивается трубчатое перо (рапидограф), установлено на ползунке, снабженном для легкости хода «линейным» шариковым подшипником и пружинками. Катушка рассчитана на напряжение 12 В, поэтому она питается через балластный резистор.

При одном переключении двигателей величина перемещения пера (элементарный шаг) составляет около 0.1 мм. Скорость черчения при этом равна примерно 50 мм/с и ограничивается в данном случае частотными характеристиками двигателей. Указанная скорость достигается при ее трехступенчатом регулировании (программой ПЭВМ), т. е. в начале черчения каждого отрезка, если он не очень малой длины, имеются две ступени «разгона», а в конце черчения — две ступени «торможения», короткие же отрезки чертятся на уменьшенных скоростях.

Сам построитель (рис. 2) собран из имеющихся под рукой и специально изготовленных деталей. Привод рейки-каретки и рабочей головки на ней (с держателем пера) осуществляется через тросики (в экс-

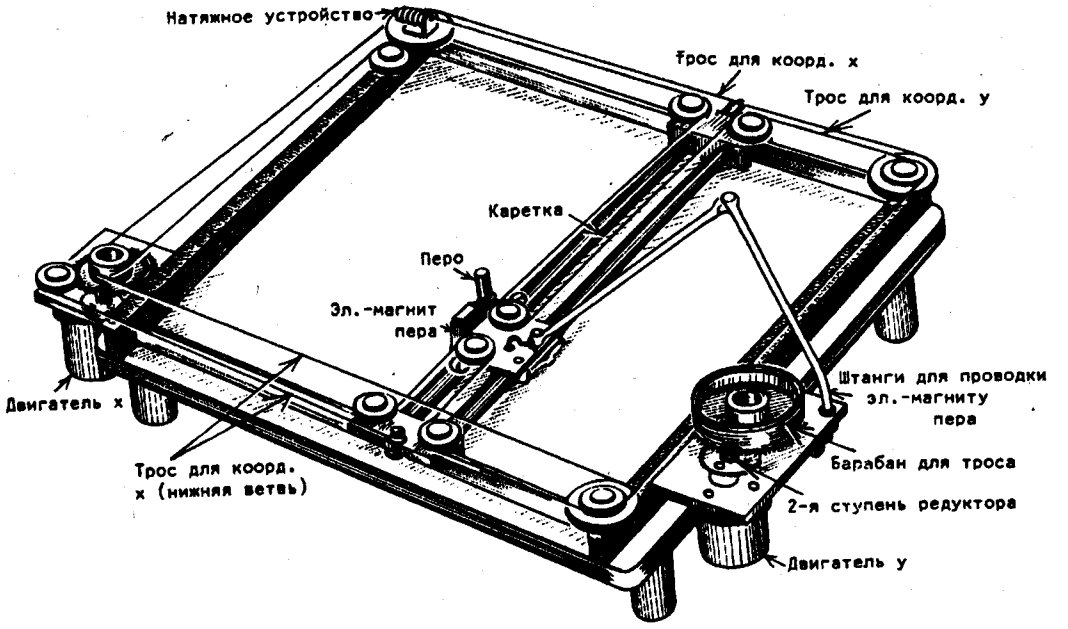


Рис. 2

периментальном варианте — через стальную струну типа корда для авиамоделей). Трос (струна) намотан на барабанчики, соединенные с валами двигателей через двухступенчатые шестеренные редукторы. (В конструкции автора зубчатые колеса и один из барабанов взяты от механической бритвы.) Каретка и рабочая головка перекатываются по трубчатым направляющим на миниатюрных шариковых подшипниках. Не считая самого планшета, рамы с направляющими и кронштейнов, для изготовления построителя потребовались следующие детали:

- двигатели — 2 шт,
- электромагнит включения пера — 1,
- перья трубчатые — от 1 до 10 (набор разной толщины),
- зубчатые колеса — 8,
- барабанчики для троса — 2,
- ролики для троса — 13,
- подшипники для каретки, головки, роликов, редукторов, трос (струна), проводка, крепеж, втулки.

Примечание. Электромагнит включения пера можно заменить на самостоятельно изготовленный, а вместо подшипников для перемещения каретки и головки по направляющим применить пластиковые втулки.

Возможны различные усовершенствования конструкции. Бумага в экспериментальном образце крепится к планшету липкой лентой, но возможно монтирование электростатического прижима (с питанием высоким напряжением через катушку типа индукционной от автомобильной системы зажигания). Можно использовать концевые выключатели, ограничивающие ход каретки и рабочей головки и подключаемые к разрядам входного порта ПЭВМ (в том числе перемещаемые выключатели, положение которых можно регулировать в зависимости от размера и положения листа бумаги на планшете). В данной конструкции размер бумаги, устанавливаемой на планшет, — до 54x45 см, размер чертежа — до 44x35 см, однако по широкой стороне планшета бумага может выходить за пределы построителя, что позволяет разделить чертеж большого размера на части и прочертить их поочередно, перемещая при этом бумагу (по возможности точно, используя специальные метки). Однако размеры планшета могут быть и большими.

В экспериментальной конструкции построителя движущиеся части, элементы привода, двигатели специально оставлены открытыми (в принципе дополнительно можно сделать кожух), так как в известной мере он служит наглядным пособием.

Шаговый принцип работы двигателей сам по себе прост для реализации, обеспечивает (если не происходит потери шагов) отсутствие накопленной ошибки и показателен в учебном плане, однако возникающая вибрация приводит к шуму, особенно в зубчатых зацеплениях. В порядке усовершенствования конструкции возможно включение упругих демпферов (в данном простейшем случае «младшая» шестеренка крепится прямо на вал двигателя) и использование беззазорного зацепления.

Двигатели ДШР-48 хотя и значительно менее мощные, но по частотным характеристикам лучше других (по паспорту — примерно до 700 Гц) и имеют меньший шаг на валу — 7.5°, что позволяет упростить всю конструкцию за счет применения одноступенчатого редуктора.

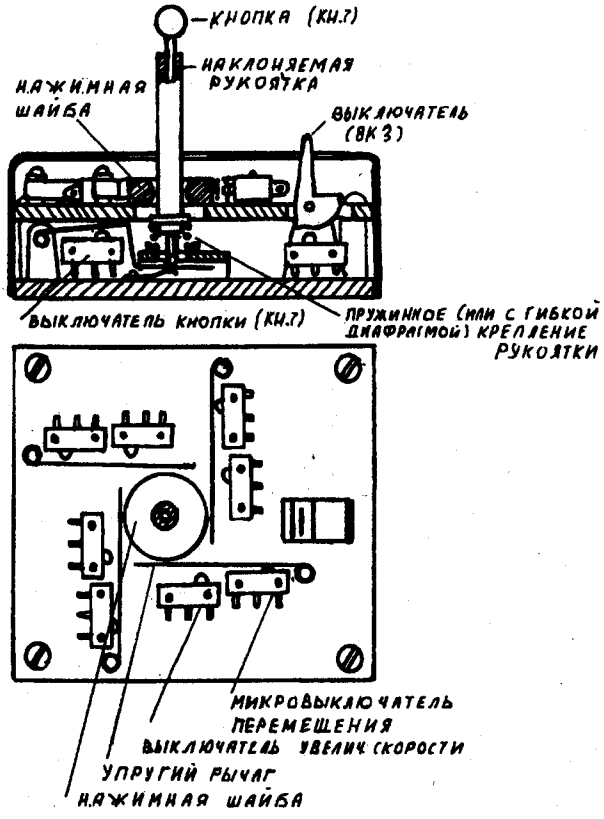


Рис. 3

Для установки пера в начальное положение при отключенном питании (или нулевом состоянии всех каналов усилителя) перемещать головку от руки нельзя, так как двигатели, обладая магнитными свойствами, тормозят перемещение. Довольно легко, однако, вращать сами валы двигателей, если на них кроме приводной шестерни надеть дополнительные маховички. Правда, тогда перемещение не будет быстрым (это удобно лишь для корректировки положения), поэтому на практике автономное перемещение пера (головки) производится через ПЭВМ с помощью обслуживающей программы и управляющей рукоятки (или кнопок). В простейшем случае для этого можно задействовать некоторые кнопки клавиатуры ПЭВМ (например, со стрелками). Но удобнее использовать переносной пульт, который можно положить рядом с планшетом или даже на сам планшет. В этом случае контакты кнопок или наклоняемой рукоятки (она крепится к основанию посредством резиновой втулки, пружины либо на шаровом шарнире) соединяются с определенными разрядами входного порта ПЭВМ. Управляющую рукоятку можно использовать как для передвижения головки построителя, так и, при необходимости, для перемещения маркера на экране ПЭВМ и для других целей. Пульт имеет восемь микровыключателей, фиксирующих наклон рукоятки, и два дополнительных (ВК. 3 и КН. 7). Как показано на схеме (рис. 3), четыре микровыключателя фиксируют направление наклона рукоятки (т. е. требуемое направление перемещения), при этом, если рука наклонена диагонально, одновременно замыкаются два выключателя. Еще четыре выключателя, попарно соединенные в параллель с первыми, срабатывают при увеличении отклонения рукоятки и регулируют таким способом скорость перемещения. Действительно, для точной подводки пера нужна малая скорость, а для больших перемещений —

увеличенная, и это относится не только к графопостроителю, но и к маркеру на экране. При этом регулировка скорости по координатам X и Y действует раздельно, что в некоторых случаях позволяет реализовать не 8, а 16 возможных направлений перемещения пера или маркера (как это происходит, думаем, читатели догадаются сами). Дополнительная кнопка на верху рукоятки и переключатель на корпусе пульта могут использоваться для разных целей, например для включения пера (на построителе можно «автономно» рисовать несложные фигуры), для пуска программы черчения, для временной остановки в процессе черчения и др. Всего от пульта задействовано восемь разрядов входного порта.

(Если в электронной схеме реализованы кольцевые коммутаторы, к ней можно добавить и генератор импульсов, подаваемых на тот или иной канал переключением кнопок пульта или микровыключателей рукоятки. В этом случае ручное управление движением рабочей головки будет действительно автономным (без участия компьютера), хотя большого смысла в этом вроде бы и нет — вряд ли здесь стоит экономить машинное время.)

Возможно, подобный простейший построитель покажется кому-то наивной безделушкой. Но, сравнивая его с заводским, даже малогабаритным, рассчитанным на подключение к малой ПЭВМ, надо уж сравнивать весь набор (построитель, ПЭВМ, рукоятка и усилитель). В конструкцию заводского построителя обычно входят микропроцессор, ПЗУ и ОЗУ, коммутаторы, усилители и т. д. Фактически там уже содержится своя ПЭВМ, хотя и специализированная. Как правило, в заводских построителях входная информация задается на языке HP-GL (или подобном ему) в виде команд, содержащих координаты конечных точек последовательно вычерчиваемых отрезков (причем с учетом заданного или принятого по умолчанию масштаба), параметры для вычерчивания окружностей и их дуг и т. д. В память некоторых построителей даже закладывается начертание шрифтов для вывода надписей. Но всегда ли такое совершенство оправдано, если с тем же (или даже с большим) успехом эти же функции можно реализовать с помощью ПЭВМ, к которой подключается построитель? Ведь стоимость построителя, в котором оставлено только самое необходимое (но не в ущерб качеству его работы!), может быть значительно снижена. Пока же маленький заводской планшетный построитель стоит в несколько раз дороже БК-0010!

Используя же ПЭВМ типа БК-0010 с простейшим построителем, можно и запрограммировать язык HP-GL, и создать диалоговый графический редактор, позволяющий выводить чертеж на экран и построитель, и осуществить считывание графической информации, когда на рабочую головку построителя вместо пера устанавливается визир с увеличительным стеклом и с помощью пульта управления наводится на отдельные точки или проводится по нужным линиям, и т. д. Ограниченный объем памяти БК-0010 не позволяет, конечно, создавать особо сложные программные системы, но возможна последовательная работа, когда в каком-то промежуточном виде (в том числе на языке, подобном HP-GL) генерируемый чертеж записывается на магнитную ленту (как обычный файл данных для БК), а затем, после смены программы, читается с ленты (сразу или поблочно) и вычерчивается на построителе. В этом случае сложность чертежа принципиально не ограничена. Сама же программа-драйвер управления построителем может быть добавлена к программе пользователя или быть автономной, занимающей всю память машины. В последнем случае ее можно расширить дополнительными функциями: контрольной прорисовкой изображения на экране ПЭВМ, его редактированием, делением на части, изменением масштаба и т. д. Известно, что программы, написанные в машинных кодах, занимают меньший объем памяти и работают быстрее, поэтому для регулярного использования имеет смысл создать или адаптировать программы именно такого типа. Из программы пользователя на языке БЕЙСИК машинные программы можно вызывать через функцию USR.

Далее рассмотрим один из возможных вариантов программы управления построителем, написанный на языке БЕЙСИК и предусматривающий чтение информации о чертеже с магнитной ленты.

После пуска программы в верхней строке экрана выводится список команд:

1 — вывод на экран;
2 — вывод на графопостроитель;
3 — ручное управление;
4 — масштаб;
5 — адрес;

6 — очистка экрана;

7 — дополнительные операции;

0 — выход из программы.

При нажатии соответствующей цифровой клавиши программа переходит к нужному действию. Так, команда «1» осуществляет вывод изображения на экран (предполагается, что оно уже загружено в оперативную память) в заданном масштабе. Аналогично «2» начинает вывод изображения на построитель, который в этот момент должен находиться в состоянии готовности.

Команда «3» переключает построитель на работу от управляющей рукоятки, при этом кнопка, расположенная на ее вершине, вызывает, пока удерживается нажатой, включение пера.

При начальном пуске текущее положение пера принимается за начало координатной системы чертежа (заметим, что в данной программе перемещение пера в режиме ручного управления не смещает начальную точку: если затем задать команду «2», изображение начертится на прежнем месте). Команда «4» позволяет изменять масштаб и осуществлять перенос изображения (смещать его вправо-влево и вверх-вниз относительно координатной системы на построителе или экране). При активизации этой команды появляется дополнительный запрос: «Для экрана (0) или графопостроителя (1)?» (таким образом, смещение и масштаб задаются отдельно для построителя и экрана).

Если чертежи не слишком сложные, в память можно ввести описание сразу нескольких из них. По некоторым причинам описание чертежей вводится в машинную область памяти, высвобождаемую оператором CLEAR, а сам ввод осуществляется с помощью BLOAD. По умолчанию чертеж загружается в ОЗУ с адреса &O30000. Если описание чертежа введено с другого адреса, команда «5» позволяет указать этот адрес программе вывода (после соответствующего запроса).

Команда «6» очищает экран и снова выводит на него список основных команд (например, после того как чертеж для контроля был выведен на экран).

Команда «7» выводит дополнительное меню:

1 — начальная точка;

2 — установка по координатам;

3 — выход в основное меню.

Генерация начальной точки («1») для графопостроителя обычно производится после перемещения пера ручным управлением, и тогда эта точка становится началом координатной системы (изображение может быть начерчено в новом месте). Команда «2» запрашивает рабочие параметры (X и Y — координаты точки, куда должно быть направлено перо построителя, P% — индекс положения пера: 0/1 — выключено/включено). Этот режим, в частности, можно использовать для полуавтоматического вывода чертежа.

Расширенная программа такого рода может включать в себя также возможности упрощенного графического редактора, т. е. создания по геометрическим элементам описания чертежа или его изменения (редактирования). Из геометрических элементов, обрабатываемых такой программой, используются отрезки, составленные из них ломаные или (если отрезки короткие) кривые линии, окружности и дуги окружностей. Задавая масштабные множители разными по различным осям, можно превратить окружности в эллипсы частного положения (с горизонтальной и вертикальной осями). При необходимости могут быть добавлены возможности генерации штрихов для вывода штриховых или штрихпунктирных линий и специальных символов, повторения типовых фрагментов чертежа, переключения используемых перьев (если их на построителе несколько) и т. д. Понятно, что расширять возможности подобных программ можно в разных направлениях, но в данном случае это не является основной целью статьи. Поэтому сосредоточим внимание на основных блоках алгоритма.

Описание чертежа имеет вид численно-кодового массива (ЧКМ) целого типа. Координаты и линейные размеры задаются в десятых долях миллиметра, углы — в десятых долях градуса (что соответствует единичным масштабам по осям). Числа — коды управления имеют величину 10000 и больше: 10000 — конец описания черте-

жа, 10001 — выключение пера (холостой ход, после достижения следующей точки перо будет включено), 10002 задает режим кодирования отрезков, 10003 — кодирования окружностей, 10004 — дуг. При кодировании отрезков (включая холостые ходы) указываются координаты конечных точек и, таким образом, каждый из отдельно расположенных отрезков задается пятью числами (код подъема пера, холостой ход и собственно отрезок; отдельная команда опускания пера по окончании холостого хода не требуется, так как эта операция выполняется автоматически). Окружности задаются каждая тремя числами — координатами центра и радиусом, дуга окружности — координатами центра, радиусом, начальным углом дуги и ее угловой величиной, знак которой определяет направление обхода. При черчении окружности перо автоматически включается в ее начальной точке и выключается в конечной, для дуги же перо по окончании работы самостоятельно не выключается (для этого нужно дополнительно использовать код 10001). Пример описания простой фигуры показан на рис. 4, а общая блок-схема программы — на рис. 5 (значки ⬭ на блок-схеме содержат номера строк БЕЙСИК-программы).

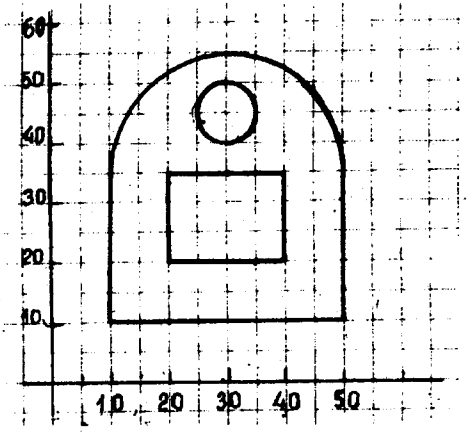


Рис. 4. Описание простой фигуры в ЧКМ.
Текст описания: 200, 200, 400, 200, 400, 350, 200, 350, 200, 200, 10001, 100, 100, 500, 100, 10004, 300, 350, 0, 1800, 10002, 100, 100, 10001, 10003, 300, 450, 50, 10000

Текст всей программы здесь, к сожалению, привести невозможно, поэтому остановимся только на ее основных блоках — на подпрограмме линейной интерполяции, определяющей последовательность переключения обмоток двигателей при черчении отрезка или отработке холостого хода, и на вызываемой для этого подпрограмме отработки одного шага перемещения пера. Предварительно (в блоке инициализации) в два рабочих массива — MA% и MB% — засылаются числа, которые, при сложении двух нужных элементов из этих массивов и представлении результата в качестве двоичных чисел, указывают разряды выходного порта, куда нужно подать в данный момент напряжение, чтобы включить соответствующие обмотки двигателей. Для массива MA% это числа 3%, 9%, 12% и 6%, для MB% — 48%, 144%, 192% и 96%. При сложении первых элементов обоих массивов имеем число 51% (в двоичном виде — 00110011), т. е. для каждого двигателя будут нагружены обмотки 1 и 2.

Для последовательного переключения обмоток каждого двигателя некоторый параметр-счетчик должен изменяться от 1 до 4 (затем опять сбрасываясь в 1), для вращения в противоположном направлении — в обратном порядке. В программе для этого использованы переменные KX% и KY%, получающие начальные единичные значения в блоке инициализации. Кроме того, требуются счетчики физических координат устройства (в его «шагах») — переменные X% и Y% (при инициализации и при установке начальной точки они приобретают нулевые значения).

Исходные данные для подпрограммы интерполяции — координаты следующей опорной точки X и Y (в данном варианте — вещественного типа). Сначала эти координаты подвергаются преобразованию (перенос и масштабирование; в коэффициентах WX!, WY!, WM! и WN! учитывается, если надо, масштаб, в котором задается геометрическая информация). Если физические шаги по координатам X и Y несколько отличаются от 0.1 мм, коэффициенты WM! и WN! должны также отличаться от единицы. По полученным физическим координатам новой точки XN% и YN% подсчитываются (с нужными знаками) количества шагов, которые должен отработать тот и другой двигатель (DX%, DY%).

Далее, перед началом интерполяционного цикла определяется, нужно ли сменить состояние пера (включенное или выключенное), для чего сравниваются старое и новое значения индекса пера P% и PP%. Если состояние пера следует изменить, в порт вывода засылается значение, отлича-

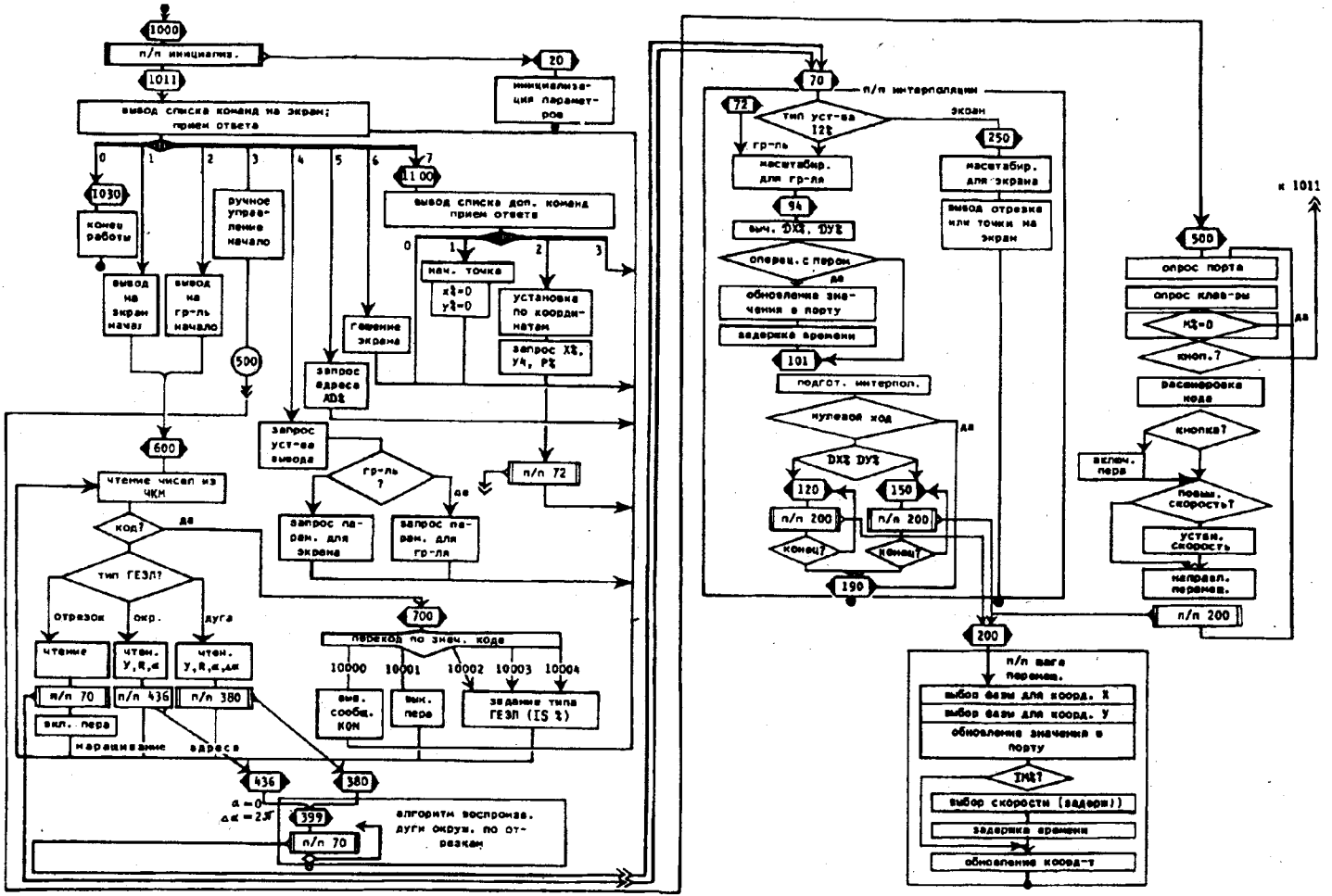

1000
п/п инициализ.
1011
вывод списка команд на экран; прием ответа
20
инициализация параметров
0
1
2
3
4
5
6
7
1100
1030
конец работы
вывод на экран начал
вывод на гр-ль начало
ручное управление начало
500
вывод списка доп. команд прием ответа
0
1
2
3
нач. точка
x%=0
y%=0
установка по координатам
запрос X%, Y%, P%
гашение экрана
запрос адреса AD%
600
чтение чисел из ЧКМ
код?
да
тип ГЕЗЛ?
отрезок
окр.
дуга
чтение
чтен. Y,R,α
чтен. Y,R,α,Δα
п/п 70
п/п 436
п/п 380
вкл. пера
нарезание
адреса
436
α=0
Δα=2π
399
п/п 70
380
запрос уст-ва вывода
гр-ль ?
да
п/п 72
запрос парам. для экрана
запрос парам. для гр-ла
700
переход по знач. коду
10000
10001
10002
10003
10004
выв. сообщ. КОН
выкл. пера
задание типа ГЕЗЛ (IS %)
алгоритм воспроизв. дуги окруж. по отрезкам
70
п/п интерполяции
72
тип уст-ва I2%
гр-пь
экран
масштабир. для гр-ла
масштабир. для экрана
94
выч. DX%, DY%
операц. с перем.
да
обновление значения в порту
задержка времени
101
подгот. интерпол.
нулевой ход
да
DX% DY%
120
150
п/п 200
п/п 200
конец?
конец?
190
200
п/п вала перемещ.
выбор фазы для коорд. X
выбор фазы для коорд. Y
обновление значения в порту
TM%?
выбор скорости (задерж.)
задержка времени
обновление коорд-т
к 1011
500
опрос порта
опрос клав-ры
K%=0
да
кноп.?
расшифровка кода
кнопка?
включ. перо
новая скорость?
устан. скорость
направл. перемещ.
п/п 200

Рис. 5

ющееся от имеющегося восьмым разрядом (связанным с электромагнитом пера), а затем устраивается цикл, производящий задержку времени, чтобы электромагнит успел сработать.

После этого производится анализ на нулевые перемещения, которые время от времени могут попадаться, и сравниваются абсолютные величины чисел шагов по двум координатам. В зависимости от результата программа переходит на одну из двух параллельных ветвей алгоритма, в каждой из которых «ведущей» считается координата с большим числом шагов. (Алгоритм интерполяции составлен так, чтобы в цикле было по возможности меньше арифметических действий, не было умножений и делений, которые замедляют процесс.) Должен ли быть следующий шаг осевым либо диагональным, определяется по значению текущей переменной LY% (имеется в виду ветвь для ведущей координаты X%), которая при осевом шаге получает приращение на AY% (число шагов по Y), а при диагональном шаге, кроме того, — на AX% (число шагов по X).

В зависимости от вида и направлений вращения двигателей (или одного из них) устанавливаются значения индексов JX% и JY%, после чего вызывается подпрограмма отработки шага, начинающаяся со строки 200. Ясно, что каждый из этих индексов может иметь значения 0, 1 или -1.

В цикл интерполяции в каждой ветви включен анализ нажатия клавиш ПЭВМ, что позволяет при необходимости прервать процесс. Вместе с тем можно анализировать и кнопку на переносном пульте.

Подпрограмма отработки шага наращивает или уменьшает значение программных счетчиков KX% и KY% на величины JX% и JY%, с помощью условных операторов реализуя их кольцевое изменение (можно также использовать операцию MOD — «деление по модулю»). Нужные разряды, указывающие обмотки двигателей, как уже было сказано, выбираются из массивов MA% и MB% с помощью этих счетчиков, добавляется также разряд, управляющий состоянием пера, и полученный таким образом код посылается в порт вывода (ячейка 177714_8). Следует заметить, что при этом сигнал инвертируется, поэтому посылается код, дополнительный к полученному (строка 210).

В строке 212 определяется текущая ступень скорости (путем сравнения обработанного числа шагов L% с нулем и с максимальным для данного хода числом шагов LM%), после чего выполняется цикл задержки времени. Содержимое переменных скорости V1, V2 и V3 задается в блоке инициализации и может быть там изменено. Константы в операторе 212 — пробег (в «шагах») на пониженных скоростях в начале и перед окончанием интерполяции, их значения тоже можно менять.

Описываемая в данной статье конструкция изготовлена автором и успешно им эксплуатируется начиная с 1990 г. По просьбам заинтересовавшихся читателей автор готов выслать принципиальные схемы и необходимую документацию, а также помочь консультациями. Для этого необходимо переслать письмо с вопросами или заказом в адрес редакции журнала с вложением чистого конверта с наклеенными марками, это письмо будет передано автору разработки.

Тексты рассмотренных подпрограмм с краткими комментариями

```
.....
72 XN%=WX!+X*WM!                          'преобразование
73 YN%=WY!+Y*WN!
.....
94 DX%=XN%-X%                             'определение приращений (числа шагов)
95 DY%=YN%-Y%
96 IF P%=PP% GOTO 101                     'изменять положение пера?
97 POKE &0177714, &077777-MA%(KX%)-MB%(KY%) - P%*&O400 'посылка в порт
98 FOR I=1 TO 70                          'задержка (константа регулируется)
99 NEXT
100 PP%=P%
101 AX%=ABS(DX%)                          'абс. значения числа шагов
102 AY%=ABS(DY%)
```

```
103 P2%=P%*&O400
104 IF (AX%+AY%)=0% GOTO 190          'выделение нулевых ходов
110 IX%=SGN(DX%)                      'направления перемещений
111 IY%=SGN(DY%)
112 L%=0%
116 IF AX%<AY% GOTO 150               'определение ведущей координаты
120 LM%=AX%                           'начало ветви для координаты X
121 LY%=0%
122 BX%=AX%\2%                        'половина ведущ. шага для сравнения
124 L%=L%+1%
125 IF INKEY$<>"" GOTO 194            'нажата ли клавиша?
126 LY%=LY%+AY%
128 IF LY%>BX% GOTO 140               'определение типа шага
130 JY%=0%                            'осевой шаг
131 JX%=IX%
136 GOSUB 200          'вызов п/п отработки шага (для обоих типов шагов)
138 IF L%<LM% GOTO 124                'проверка на конец интерполяции
139 GOTO 190                          'переход к окончанию
140 LY%=LY%-AX%                       'диагональный шаг
142 JY%=IY%
143 GOTO 131
150 LM%=AY%                           'ветвь для координаты Y
151 LX%=0%
152 BY%=AY%\2%
154 L%=L%+1%
155 LF INKEY$<>"" GOTO 194
156 LX%=LX%+AX%
158 IF LX%>AX% GOTO 170
160 JX%=0%
161 JY%=IY%
162 GOSUB 200
168 IF L%<LM% GOTO 154
169 GOTO 190
170 LX%=LX%-AY%
171 JX%=IX%
172 GOTO 161
190 RETURN                            'выход из п/п
194 WZ%=99%
195 RETURN                            'выход при нажатии клавиши
199 REM                               'п/п отработки шага
200 KX%=KX%+JX%                       'инкремент кольц. счетчиков
203 IF KX%<1% THEN KX%=4% ELSE IF KX%>4% THEN KX%=1%
204 KY%=KY%+JY%
206 IF KY%<1% THEN KY%=4% ELSE IF KY%>4% THEN KY%=1%
210 POKE &0177714, &077777-MA%(KX%)-MB%(KY%)-P2%     'засылка в порт
211 IF IM%=1% GOTO 214                'обход для режима ручного управления
212 IF L%<6% OR (LM%-L%)<6% THEN VI=V1 ELSE IF L%<12% OR (LM%-L%)<16%
THEN VI=V2 ELSE VI=V3                 'выбор скорости
214 FOR J=1 TO VI                     'задержка
216 NEXT
218 X%=X%+JX%                         'наращивание координат
219 Y%=Y%+JY%
220 RETURN
```

Борис Ф. Фролкин,
г. Москва

ПСЕВДОГРАФИКА БЕЙСИКА-БК И EPSON-СОВМЕСТИМЫЙ ПРИНТЕР

Способы подключения EPSON-совместимых принтеров обсуждались в литературе неоднократно. Однако все опубликованные варианты страдают одним недостатком — без дополнительных усилий невозможно воспользоваться штатными операторами LLIST и LPRINT БЕЙСИКа, прошитого в ПЗУ.

Происходит это по трем причинам. Первый, наиболее очевидный и легко преодолимый барьер — БЕЙСИК-транслятор, как и подавляющее большинство других программ для БК, передает байты на принтер по стандарту ИРПР, отличающемуся от общеупотребительного в БК CENTRONICS. Различие заключается в инверсном и, соответственно, неинверсном способах передачи байтов данных, а также тем, что в CENTRONICS строб должен появляться не ранее, чем спустя 1 мкс после данных (и уж во всяком случае не одновременно с ними), и заканчиваться по крайней мере за 1 мкс до их снятия. Различия можно проследить по диаграммам, приведенным на рис. 1.

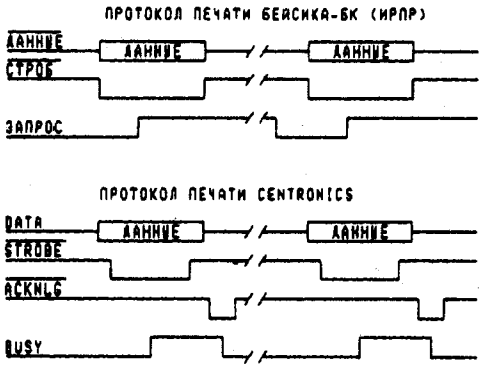


Рис. 1

Второй причиной неработоспособности является то, что драйвер принтера БЕЙСИКа-ПЗУ перед посылкой очередной строки на печать (а иногда и перед передачей каждого отдельного символа) посылает код 13H. Но любой EPSON-совместимый принтер понимает его как «DC3 — невыбор печатающего устройства» и замирает в ожидании кода 11H («DC1 — выбор печатающего устройства»), которого он вряд ли дождется. Помогает при этом лишь «волшебная» клавиша POWER.

Наконец, третья причина — различие в кодировке символов. На БК принят стандарт КОИ-8, а все русифицированные EPSON-совместимые принтеры содержат так называемую альтернативную кодировку символов, часто — основную и лишь иногда — КОИ-8 (причем без псевдографики).

Предлагаемое схемное решение (рис. 2) позволяет обойти все эти проблемы. Устройство готово к работе сразу же после включения питания, как и БЕЙСИК-ПЗУ. Прошивка микросхемы К573РФ5 приведена ниже. Она состоит из восьми страниц, причем четыре старшие — инверсная копия четырех младших. Осуществляется передача кодов символов как без изменения (для распечатки графической информации), так и с переводом из стандарта БК в альтернативную кодировку IBM PC для правильной распечатки текстов. В младшей половине кодовой таблицы все символы передаются без изменения, за исключением 13H (заменяется на 11H), 24H и 7FH (заменяются, соответственно, на FDH и DBH, имеющими требуемое начертание). В старшей половине кодовой таблицы все символы подвергаются транслитерации, причем знак «℔», не имеющий эквивалента по начертанию, заменяется на «$», а карточные символы «♥», «♠», «♣» и «♦» — соответственно на «ё», «Ё», «±» и «№». Коды 80H...9FH, являющиеся на БК служебными, печатаются в виде небольшого черного квадратика (код FEH).

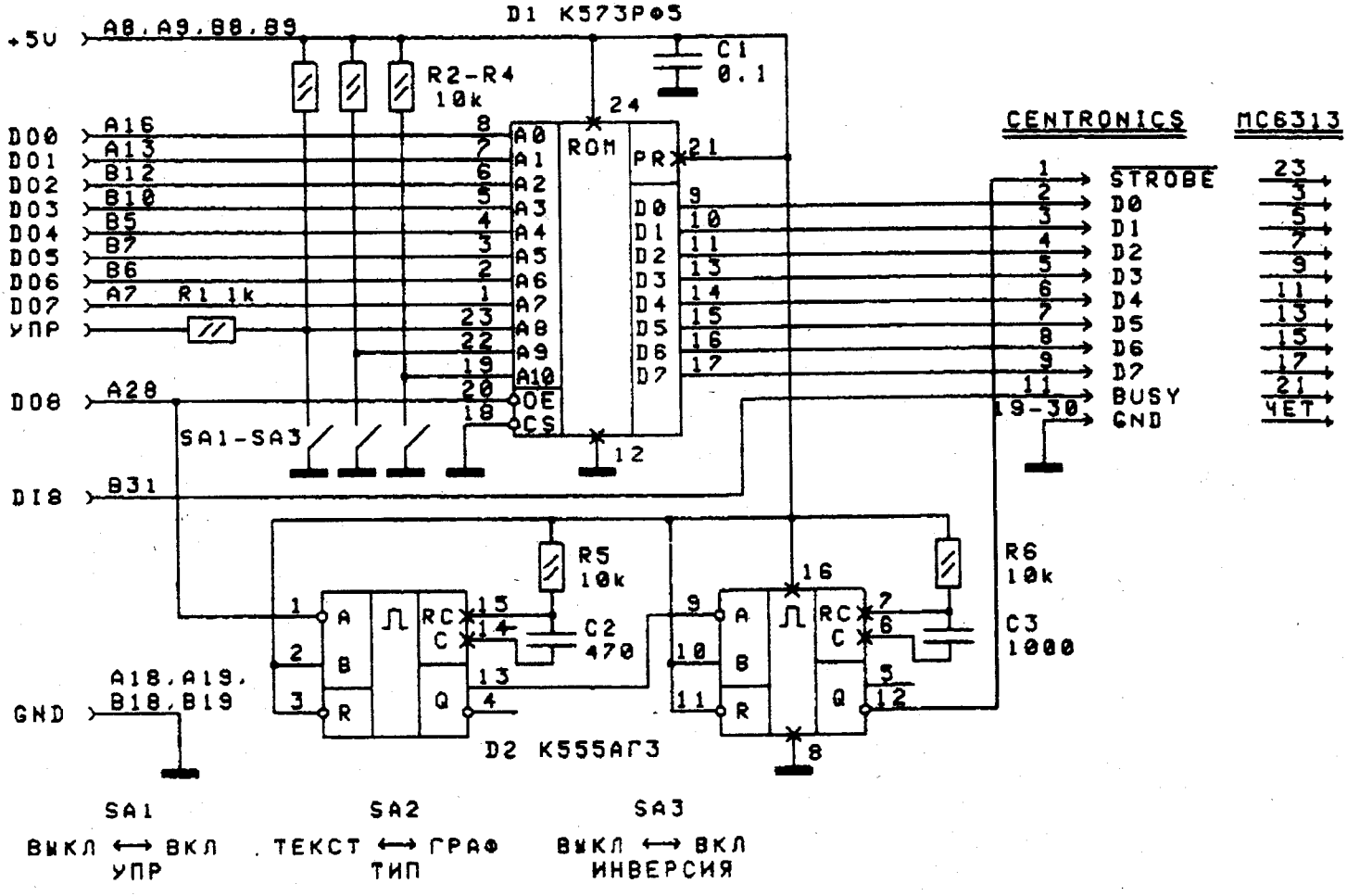

D1 К573РФ5
+5U
A8,A9,B8,B9
R2-R4
10k
C1
0.1
24
D00
D01
D02
D03
D04
D05
D06
D07
УПР
A16
A13
B12
B10
B5
B7
B6
A7
R1 1k
A0
A1
A2
A3
A4
A5
A6
A7
A8
A9
A10
OE
CS
ROM
PR
21
D0
D1
D2
D3
D4
D5
D6
D7
12
D08
A28
SA1-SA3
DI8
B31
CENTRONICS
MC6313
1 STROBE 23
2 D0 3
3 D1 5
4 D2 7
5 D3 9
6 D4 11
7 D5 13
8 D6 15
9 D7 17
11 BUSY 21
19-30 GND ЧЕТ
R5
10k
R6
10k
C2
470
C3
1000
A
B
R
RC
C
Q
GND
A18,A19,
B18,B19
D2 К555АГ3
SA1
ВЫКЛ ⟷ ВКЛ
УПР
SA2
ТЕКСТ ⟷ ГРАФ
ТИП
SA3
ВЫКЛ ⟷ ВКЛ
ИНВЕРСИЯ

Рис. 2

Проверить собранное устройство можно, набрав в БЕЙСИКе ряд строк-комментариев (оператор REM), используя строчные и прописные буквы русского и латинского алфавитов, а также символы псевдографики, и распечатав их командой LLIST. При этом DIP-переключатели принтера должны настраивать его на работу с альтернативной кодировкой для IBM PC. Для примера на рис. 3 приведены их установки для трех распространенных типов принтеров.

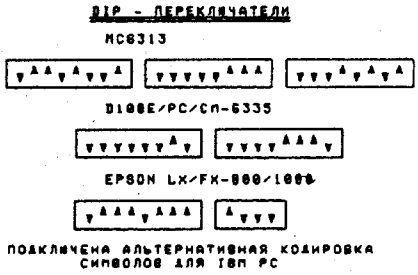


Рис. 3

Прошивка ПЗУ D1 К573РФ5

```
0000 FF FE FD FC FB FA F9 F8 F7 F6 F5 F4 F3 F2 F1 F0
0010 EF EE ED EC EB EA E9 E8 E7 E6 E5 E4 E3 E2 E1 E0
0020 DF DE DD DC DB DA D9 D8 D7 D6 D5 D4 D3 D2 D1 D0
0030 CF CE CD CC CB CA C9 C8 C7 C6 C5 C4 C3 C2 C1 C0
0040 BF BE BD BC BB BA B9 B8 B7 B6 B5 B4 B3 B2 B1 B0
0050 AF AE AD AC AB AA A9 A8 A7 A6 A5 A4 A3 A2 A1 A0
0060 9F 9E 9D 9C 9B 9A 99 98 97 96 95 94 93 92 91 90
0070 8F 8E 8D 8C 8B 8A 89 88 87 86 85 84 83 82 81 80
0080 7F 7E 7D 7C 7B 7A 79 78 77 76 75 74 73 72 71 70
0090 6F 6E 6D 6C 6B 6A 69 68 67 66 65 64 63 62 61 60
00A0 5F 5E 5D 5C 5B 5A 59 58 57 56 55 54 53 52 51 50
00B0 4F 4E 4D 4C 4B 4A 49 48 47 46 45 44 43 42 41 40
00C0 3F 3E 3D 3C 3B 3A 39 38 37 36 35 34 33 32 31 30
00D0 2F 2E 2D 2C 2B 2A 29 28 27 26 25 24 23 22 21 20
00E0 1F 1E 1D 1C 1B 1A 19 18 17 16 15 14 13 12 11 10
00F0 0F 0E 0D 0C 0B 0A 09 08 07 06 05 04 03 02 01 00

0100 9A 97 99 9D 98 87 9B 9C 82 86 93 92 91 90 9F 8F
0110 8E 8D 8C 8B 8A 89 88 95 83 94 85 84 96 81 80 9E
0120 EA E7 E9 ED E8 A7 EB EC A2 A6 E3 E2 E1 E0 EF AF
0130 AE AD AC AB AA A9 A8 E5 A3 E4 A5 A4 E6 A1 A0 EE
0140 B1 F6 C6 CF D2 D8 D9 FC B3 D7 C4 FB F8 CE F7 B4
0150 BA C5 F9 D0 C2 DA F0 D1 CD C0 C3 B5 BF F1 C1 24
0160 FE FE FE FE FE FE FE FE FE FE FE FE FE FE FE FE
0170 FE FE FE FE FE FE FE FE FE FE FE FE FE FE FE FE
0180 DB 7E 7D 7C 7B 7A 79 78 77 76 75 74 73 72 71 70
0190 6F 6E 6D 6C 6B 6A 69 68 67 66 65 64 63 62 61 60
01A0 5F 5E 5D 5C 5B 5A 59 58 57 56 55 54 53 52 51 50
01B0 4F 4E 4D 4C 4B 4A 49 48 47 46 45 44 43 42 41 40
01C0 3F 3E 3D 3C 3B 3A 39 38 37 36 35 34 33 32 31 30
01D0 2F 2E 2D 2C 2B 2A 29 28 27 26 25 FD 23 22 21 20
01E0 1F 1E 1D 1C 1B 1A 19 18 17 16 15 14 11 12 11 10
01F0 0F 0E 0D 0C 0B 0A 09 08 07 06 05 04 03 02 01 00
```

```
0200 9A 97 99 9D 98 87 9B 9C 82 86 93 92 91 90 9F 8F
0210 8E 8D 8C 8B 8A 89 88 95 83 94 85 84 96 81 80 9E
0220 EA E7 E9 ED E8 A7 EB EC A2 A6 E3 E2 E1 E0 EF AF
0230 AE AD AC AB AA A9 A8 E5 A3 E4 A5 A4 E6 A1 A0 EE
0240 B1 F6 C6 CF D2 D8 D9 FC B3 D7 C4 FB F8 CE F7 B4
0250 BA C5 F9 D0 C2 DA F0 D1 CD C0 C3 B5 BF F1 C1 24
0260 FE FE FE FE FE FE FE FE FE FE FE FE FE FE FE FE
0270 FE FE FE FE FE FE FE FE FE FE FE FE FE FE FE FE
0280 DB 7E 7D 7C 7B 7A 79 78 77 76 75 74 73 72 71 70
0290 6F 6E 6D 6C 6B 6A 69 68 67 66 65 64 63 62 61 60
02A0 5F 5E 5D 5C 5B 5A 59 58 57 56 55 54 53 52 51 50
02B0 4F 4E 4D 4C 4B 4A 49 48 47 46 45 44 43 42 41 40
02C0 3F 3E 3D 3C 3B 3A 39 38 37 36 35 34 33 32 31 30
02D0 2F 2E 2D 2C 2B 2A 29 28 27 26 25 FD 23 22 21 20
02E0 1F 1E 1D 1C 1B 1A 19 18 17 16 15 14 11 12 11 10
02F0 0F 0E 0D 0C 0B 0A 09 08 07 06 05 04 03 02 01 00

0300 FF FE FD FC FB FA F9 F8 F7 F6 F5 F4 F3 F2 F1 F0
0310 EF EE ED EC EB EA E9 E8 E7 E6 E5 E4 E3 E2 E1 E0
0320 DF DE DD DC DB DA D9 D8 D7 D6 D5 D4 D3 D2 D1 D0
0330 CF CE CD CC CB CA C9 C8 C7 C6 C5 C4 C3 C2 C1 C0
0340 BF BE BD BC BB BA B9 B8 B7 B6 B5 B4 B3 B2 B1 B0
0350 AF AE AD AC AB AA A9 A8 A7 A6 A5 A4 A3 A2 A1 A0
0360 9F 9E 9D 9C 9B 9A 99 98 97 96 95 94 93 92 91 90
0370 8F 8E 8D 8C 8B 8A 89 88 87 86 85 84 83 82 81 80
0380 7F 7E 7D 7C 7B 7A 79 78 77 76 75 74 73 72 71 70
0390 6F 6E 6D 6C 6B 6A 69 68 67 66 65 64 63 62 61 60
03A0 5F 5E 5D 5C 5B 5A 59 58 57 56 55 54 53 52 51 50
03B0 4F 4E 4D 4C 4B 4A 49 48 47 46 45 44 43 42 41 40
03C0 3F 3E 3D 3C 3B 3A 39 38 37 36 35 34 33 32 31 30
03D0 2F 2E 2D 2C 2B 2A 29 28 27 26 25 24 23 22 21 20
03E0 1F 1E 1D 1C 1B 1A 19 18 17 16 15 14 13 12 11 10
03F0 0F 0E 0D 0C 0B 0A 09 08 07 06 05 04 03 02 01 00

0400 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
0410 10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F
0420 20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F
0430 30 31 32 33 34 35 36 37 38 39 3A 3B 3C 3D 3E 3F
0440 40 41 42 43 44 45 46 47 48 49 4A 4B 4C 4D 4E 4F
0450 50 51 52 53 54 55 56 57 58 59 5A 5B 5C 5D 5E 5F
0460 60 61 62 63 64 65 66 67 68 69 6A 6B 6C 6D 6E 6F
0470 70 71 72 73 74 75 76 77 78 79 7A 7B 7C 7D 7E 7F
0480 80 81 82 83 84 85 86 87 88 89 8A 8B 8C 8D 8E 8F
0490 90 91 92 93 94 95 96 97 98 99 9A 9B 9C 9D 9E 9F
04A0 A0 A1 A2 A3 A4 A5 A6 A7 A8 A9 AA AB AC AD AE AF
04B0 B0 B1 B2 B3 B4 B5 B6 B7 B8 B9 BA BB BC BD BE BF
04C0 C0 C1 C2 C3 C4 C5 C6 C7 C8 C9 CA CB CC CD CE CF
04D0 D0 D1 D2 D3 D4 D5 D6 D7 D8 D9 DA DB DC DD DE DF
04E0 E0 E1 E2 E3 E4 E5 E6 E7 E8 E9 EA EB EC ED EE EF
04F0 F0 F1 F2 F3 F4 F5 F6 F7 F8 F9 FA FB FC FD FE FF
```

```
0500 65 68 66 62 67 78 64 63 7D 79 6C 6D 6E 6F 60 70
0510 71 72 73 74 75 76 77 6A 7C 6B 7A 7B 69 7E 7F 61
0520 15 18 16 12 17 58 14 13 5D 59 1C 1D 1E 1F 10 50
0530 51 52 53 54 55 56 57 1A 5C 1B 5A 5B 19 5E 5F 11
0540 4E 09 39 30 2D 27 26 03 4C 28 3B 04 07 31 08 4B
0550 45 3A 06 2F 3D 25 0F 2E 32 3F 3C 4A 40 0E 3E DB
0560 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01
0570 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01
0580 24 81 82 83 84 85 86 87 88 89 8A 8B 8C 8D 8E 8F
0590 90 91 92 93 94 95 96 97 98 99 9A 9B 9C 9D 9E 9F
05A0 A0 A1 A2 A3 A4 A5 A6 A7 A8 A9 AA AB AC AD AE AF
05B0 B0 B1 B2 B3 B4 B5 B6 B7 B8 B9 BA BB BC BD BE BF
05C0 C0 C1 C2 C3 C4 C5 C6 C7 C8 C9 CA CB CC CD CE CF
05D0 D0 D1 D2 D3 D4 D5 D6 D7 D8 D9 DA 02 DC DD DE DF
05E0 E0 E1 E2 E3 E4 E5 E6 E7 E8 E9 EA EB EE ED EE EF
05F0 F0 F1 F2 F3 F4 F5 F6 F7 F8 F9 FA FB FC FD FE FF

0600 65 68 66 62 67 78 64 63 7D 79 6C 6D 6E 6F 60 70
0610 71 72 73 74 75 76 77 6A 7C 6B 7A 7B 69 7E 7F 61
0620 15 18 16 12 17 58 14 13 5D 59 1C 1D 1E 1F 10 50
0630 51 52 53 54 55 56 57 1A 5C 1B 5A 5B 19 5E 5F 11
0640 4E 09 39 30 2D 27 26 03 4C 28 3B 04 07 31 08 4B
0650 45 3A 06 2F 3D 25 0F 2E 32 3F 3C 4A 40 0E 3E DB
0660 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01
0670 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01
0680 24 81 82 83 84 85 86 87 88 89 8A 8B 8C 8D 8E 8F
0690 90 91 92 93 94 95 96 97 98 99 9A 9B 9C 9D 9E 9F
06A0 A0 A1 A2 A3 A4 A5 A6 A7 A8 A9 AA AB AC AD AE AF
06B0 B0 B1 B2 B3 B4 B5 B6 B7 B8 B9 BA BB BC BD BE BF
06C0 C0 C1 C2 C3 C4 C5 C6 C7 C8 C9 CA CB CC CD CE CF
06D0 D0 D1 D2 D3 D4 D5 D6 D7 D8 D9 DA 02 DC DD DE DF
06E0 E0 E1 E2 E3 E4 E5 E6 E7 E8 E9 EA EB EE ED EE EF
06F0 F0 F1 F2 F3 F4 F5 F6 F7 F8 F9 FA FB FC FD FE FF

0700 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
0710 10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F
0720 20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F
0730 30 31 32 33 34 35 36 37 38 39 3A 3B 3C 3D 3E 3F
0740 40 41 42 43 44 45 46 47 48 49 4A 4B 4C 4D 4E 4F
0750 50 51 52 53 54 55 56 57 58 59 5A 5B 5C 5D 5E 5F
0760 60 61 62 63 64 65 66 67 68 69 6A 6B 6C 6D 6E 6F
0770 70 71 72 73 74 75 76 77 78 79 7A 7B 7C 7D 7E 7F
0780 80 81 82 83 84 85 86 87 88 89 8A 8B 8C 8D 8E 8F
0790 90 91 92 93 94 95 96 97 98 99 9A 9B 9C 9D 9E 9F
07A0 A0 A1 A2 A3 A4 A5 A6 A7 A8 A9 AA AB AC AD AE AF
07B0 B0 B1 B2 B3 B4 B5 B6 B7 B8 B9 BA BB BC BD BE BF
07C0 C0 C1 C2 C3 C4 C5 C6 C7 C8 C9 CA CB CC CD CE CF
07D0 D0 D1 D2 D3 D4 D5 D6 D7 D8 D9 DA DB DC DD DE DF
07E0 E0 E1 E2 E3 E4 E5 E6 E7 E8 E9 EA EB EC ED EE EF
07F0 F0 F1 F2 F3 F4 F5 F6 F7 F8 F9 FA FB FC FD FE FF
```

С. В. Лелейкин,
г. Москва

БК-0010(.01) + «МАЯК-231-СТЕРЕО»

Всем известно, что разработчиками персонального компьютера БК-0010(.01) предусмотрена возможность управления двигателем магнитофона. Однако подключить непосредственно к управляющему выходу БК можно только магнитофон «Электроника-302». Использование других моделей, обладающих более высокими качествами, требует их доработки. Один из вариантов такой доработки для магнитофона «Маяк-231-стерео» предлагается в этой статье.

К выполнению подобной доработки побудило желание максимально облегчить работу пользователя с БК и магнитофоном. Однако решена эта проблема несколько иначе, чем, например, в [1]. (При всех достоинствах описанной в [1] конструкции она не лишена и недостатков, не позволяющих воспроизвести ее широкому кругу читателей. Во-первых, изготовить такую «систему» под силу только квалифицированному радиолюбителю, — чтобы понять это, достаточно взглянуть на ее принципиальную схему. Во-вторых, того, кто ее сделает, ждет перспектива работы только в среде БЕЙСИКа и только с файлами данных. К тому же уже имеющиеся БЕЙСИК-программы придется переделывать. Загрузка же программ на ФОКАЛе, в машинных кодах или для какой-либо другой прикладной системы представляется весьма проблематичной, даже если удастся написать соответствующее программное обеспечение, которое неминуемо съест и без того ограниченный объем памяти БК.)

Все это, а также крайне низкое качество и неудобство при работе с магнитофоном «Электроника-302» привели к мысли состыковать БК с магнитофоном «Маяк-231-стерео», обладающим лучшими характеристиками (однако принцип, положенный в основу схемы, может быть применен к любому магнитофону с электронно-логическим управлением). При этом учитывались следующие обязательные условия: полная имитация «Маяком» работы «Электроники-302» совместно с БК-0010(.01) без каких-либо доработок последнего, минимальные материальные и трудовые затраты, отсутствие необходимости вносить какие-либо изменения в механическую конструкцию магнитофона (дорабатывается лишь электронная схема), сохранение всех эксплуатационных характеристик «Маяка-231» с возможностью быстрого переключения из режима работы с компьютером в «штатный», применение недефицитных радиодеталей и, наконец, возмож-

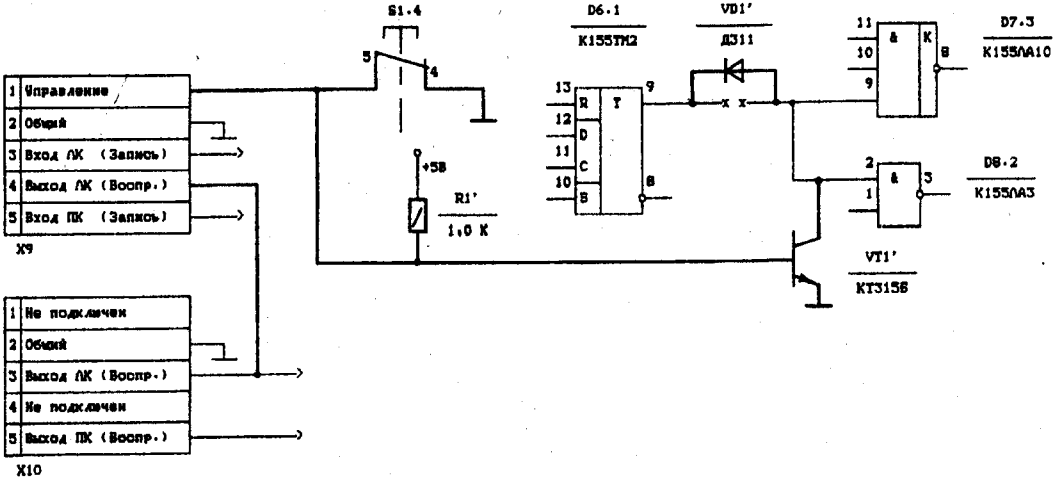


Рис. 1

ность повторения доработки любым читателем, умеющим пользоваться паяльником. Выполнение всех этих требований позволило бы обойтись без разработки какого то ни было специального программного обеспечения и, как следствие этого, дало бы возможность использовать любые ранее написанные для БК программы без каких-либо переделок или дополнений.

Анализ схемы «Маяка-231-стерео» и алгоритма управления магнитофоном с компьютера БК-0010(.01) позволили сделать вывод, что для реализации всех предъявленных требований достаточно установить на выводе 9 микросхемы D7.3 и на выводе 2 микросхемы D8.2 (здесь и далее обозначения соответствуют принципиальной схеме, прилагаемой к магнитофону) уровень логической единицы для включения на «воспроизведение» или «запись» (в зависимости от режима, предварительно установленного пользователем) и логического нуля — для остановки магнитофона (необходимые доработки схемы магнитофона показаны на рис. 1).

Сущность доработки заключается в подаче на вышеупомянутые сигнальные линии уровня логического нуля при разомкнутых контактах реле БК и логической единицы — при замкнутых. Для этого в схему вводятся ключевой транзистор VT1' (КТ315Б) и резистор R1' (1,0 кОм 0,25 Вт). Для развязки с цепью включения «временной остановки» в разрыв проводника, идущего от вывода 9 микросхемы D6.1, включен диод VD1' (Д311). Его катод подключается к выводу 9 D6.1, что обеспечивает нормальную работу кнопки «временная остановка».

Для переключения режимов «ПРОГРАММНОЕ УПРАВЛЕНИЕ»—«УПРАВЛЕНИЕ ОТ КОМПЬЮТЕРА» задействованы обычно не используемые нормально замкнутые контакты 4 и 5 переключателя S1.4 («отключение автостопа») на плате программирования A13 магнитофона. Они соединяются отрезками монтажного провода с контактами 1 и 2 разъема X9 «ВХОД» (на задней стенке корпуса). Теперь при нажатии на кнопку S1.4 все режимы «программного управления» магнитофона выключаются, и он будет управляться от БК.

Следует заметить, что данная доработка магнитофона абсолютно «прозрачна» для пользователя, т. е., даже если и включен режим «УПРАВЛЕНИЕ ОТ КОМПЬЮТЕРА», «Маяк» полностью подчиняется всем предусмотренным для него командам в полном соответствии с прилагаемой к нему инструкцией (за исключением «программного управления», «руководство» которым в это время передается компьютеру). Выключение режима управления магнитофоном от БК-0010(.01) вернет пользователю и все функции «программного управления».

В принципе, можно сохранить при работе с БК и все режимы «программного управления», но для этого вместо переключателя S1.4 придется использовать дополнительный выключатель, который должен замыкать базу транзистора VT1' на контакт «Общий» разъема X9.

Транзистор VT1' припаивается непосредственно к проводящим дорожкам на печатной плате автоматики A11 магнитофона. Наиболее подходящее место для этого находится между микросхемой D7 и резистором R13. Эмиттер VT1' следует припаять к точке подключения проволочной перемычки к дорожке, идущей к выводу 7 микросхемы D7 (общий провод). Коллектор — к расположенной поблизости дорожке, идущей к выводу 2 микросхемы D8.2. База транзистора соединяется отрезком монтажного провода с неиспользуемой точкой на печатной плате, расположенной рядом с контактом 1 разъема X5, и к ней же припаивается вывод резистора R1' и провод, идущий к контакту 1 («управление») разъема X9. Другой вывод резистора R1' подключается к расположенной рядом дорожке с напряжением питания +5В. Проводник, идущий от вывода 9 микросхемы D6.1, следует перерезать и подпаять в этот разрыв диод VD1' (Д311), катодом к выводу 9 D6.1. Контакт 3 разъема X10 («ВЫХОД») на задней стенке магнитофона следует соединить с контактом 4 разъема X9.

Вместо вышеуказанных радиодеталей можно использовать любой транзистор из серий КТ315, КТ3102, КТ503 и диод Д310, Д311, Д220, КД521, КД522.

Кабель магнитофона для подключения к БК-0010(.01) следует изготовить новый

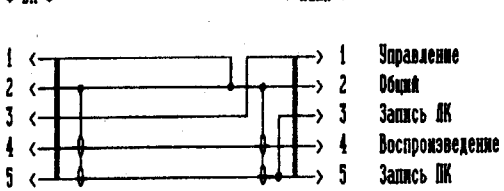


Рис. 2

по схеме, приведенной на рис. 2 (или перепаять имеющийся в комплекте БК). Если все сделано правильно, соедините БК-0010(.01) (разъем «МГ») и «Маяк» (разъем X9) новым кабелем и нажмите на кнопку «откл. АС» магнитофона. (ВНИМАНИЕ! Соблюдайте правильность подключения — кабель «несимметричный».) Дальнейшие ваши действия ничем не отличаются от приведенных в «Руководстве по эксплуатации БК-0010(.01)», т. е. если вы хотите считать программу или данные с магнитофона — нажмите предварительно на кнопку «воспроизведение», для записи на магнитофон — одновременно «запись» и «воспроизведение». Далее БК все сделает сам.

В заключение можно порекомендовать ввести в ваш «Маяк» (поскольку для описанной выше доработки все равно придется его разбирать) некоторые дополнительные усовершенствования: установить выключатель двигателя, описанный в [2], доработать магнитофон согласно рекомеидациям [3] и, если возникают проблемы при считывании программ и данных, записанных на другом магнитофоне, установить переключатель, соединяющий контакт 4 разъема X9 с контактом 3 или 5 (на выбор) разъема X10 и тем самым обеспечивающий возможность считывания с «хорошей» дорожки на магнитофонной ленте (рис. 3).

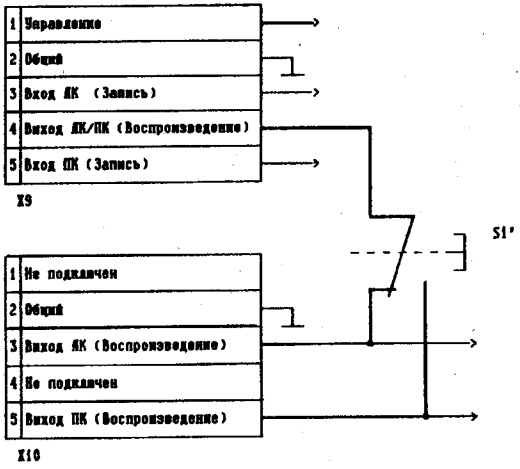


Рис. 3

Литература

1. *Сапогов В. В.* Система автоматического управления накопителем на магнитной ленте (бытовым магнитофоном) на базе БК-0010 // Вычислительная техника и ее применение. 1990. № 8. С. 23—33.

2. *Поспелов В.* Выключение электродвигателя в «Маяке-231-стерео» // Радио. 1986. № 9. С. 45.

3. *Медведев Ю., Кучерак С.* Повышение качества записи магнитофона «Маяк-231» // Радио. 1986. № 6. С. 46.

А. Свилюк,
«ИнтерСервер»

ИСПОЛНЯЮЩАЯ СИСТЕМА EXE10 PLUS ДЛЯ БК-0011М

Назначение и условия применения

Исполняющая система EXE10 Plus обеспечивает запуск на ПЭВМ БК-0011М программ для БК-0010 в машинных кодах (расширение .BIN), на ФОКАЛе (.FOC) и БЕЙСИКе (.ASC и .COD). Для ее нормальной работы требуется 34.5 Кб ОЗУ (верхний адрес свободной памяти 105000В). Кроме того, дополнительные страницы ОЗУ (3—6) не должны использоваться операционной системой. При работе с ОС версии 2.3 и старше эти страницы следует освободить командой **.Setswp Disk.**

В состав исполняющей системы входят следующие файлы:

EXE10P.SAV — головной модуль системы;

DISK.E10 — модуль драйвера диска и возврата в операционную систему;

BK0010.E10 — копия монитора и ФОКАЛа БК-0010;

BASIC.E10 — БЕЙСИК BASIC/10;

BINIO .SAV — утилита чтения/записи файлов БК-0010;

TR10 .SAV — утилита преобразования текстовых файлов.

Характеристики системы

Исполняющая система позволяет использовать на ПЭВМ БК-0011М практически все программное обеспечение, разработанное для БК-0010. Распределение адресного пространства при ее работе показано на рис.1.

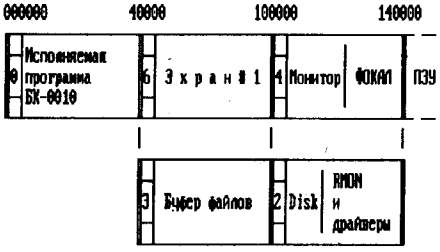


Рис. 1

Для исполняемых программ создается среда, полностью идентичная БК-0010. Для этого по адресам 100000—120000 размещается копия монитора БК-0010 (содержимого ПЗУ 1801РЕ2-017). По адресам 120000—140000 может располагаться как интерпретатор языка ФОКАЛ, так и любая другая программа, считываемая с диска. Кроме того, по адресам 100000—120000 могут быть подключены ПЗУ, установленные в отсеке пользователя (страница #12).

Исполняющая система перехватывает запрос обмена с магнитной лентой EMT 36, заменяя его вводом-выводом на диск. При работе в режиме БК-0010 имена файлов преобразуются в формат RT-11 по следующему алгоритму:

- если в имени есть двоеточие, то первые три символа до него воспринимаются как имя устройства, иначе подразумевается рабочий диск DK: ;
- следующие шесть символов до возможной точки воспринимаются как имя файла;
- если в имени есть точка, то следующие три символа определяют расширение файла, иначе берется расширение по умолчанию (.DAT или иное заданное в ключе /E:ext). Например:

LEXIC.BIN — DK :LEXIC .BIN
BY0:ENGL — BY0:ENGL .DAT
ABC:DEFGHIJ.KLM — ABC:DEFGHI.KLM

При использовании EXE10 Plus существует несколько ограничений аппаратного характера. Так, невозможно использовать программы, содержащие обращения к ПЗУ БЕЙСИКа БК-0010 и ПЗУ тестов (1801РЕ2-019), а также работающие с сетью КУВТ-86.

Запуск системы

Перед запуском исполняющей системы следует назначить логическое имя E10 устройству с ее модулями (команда **.Assign DEV E10**). Запуск осуществляется командой **.EXE10P filespecs[/options]**, если исполняющая система находится на системном диске, или

.Run [DEV:]EXE10P
Command: filespecs[/options],

для другого носителя. Здесь **filespecs** — спецификации файлов (до шести в одной команде), устройство по умолчанию — DK:, расширения имен файлов — .BIN; **options** — ключи командной строки; **DEV** — устройство, на котором расположена исполняющая система.

БЕЙСИК BASIC/10

БЕЙСИК BASIC/10 представляет собой адаптированный БЕЙСИК БК-0010.01 и имеет следующие отличия от прототипа:

- операции чтения/записи при наличии в имени файла символов «ТТ:» производят обмен с накопителем на магнитных дисках;
- реализован оператор MERGE;
- реализовано вычерчивание закрашенных прямоугольников (оператор LINE ..., BF);
- устранена ошибка в операторе DRAW;

- установлен драйвер принтера с интерфейсом Centronics;
- команда MONIT (а также функциональная клавиша АР2/0) вызывает возврат в операционную систему.

Распределение адресного пространства при работе БЕЙСИКа BASIC/10 представлено на рис.2.

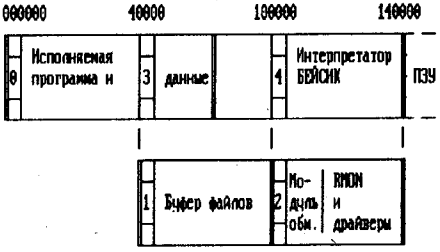


Рис. 2

Утилиты BINIO и TR10

Утилита BINIO предназначена для переписывания файлов БК-0010 с магнитной ленты на дискеты БК-0011М и обратно. Запуск BINIO осуществляется командой **.RUN DEV:BINIO** (или **.BINIO**, если она расположена на системном диске). После запуска на запрос «Command:» следует ввести режим работы: чтение (R), запись (W) или выход в операционную систему (E).

Таблица

Ключи исполняющей системы

Ключ	Назначение
/B	Запуск интерпретатора языка БЕЙСИК БК11/10
/E:ext	Установка расширения файлов данных в исполняющей системе, по умолчанию — .DAT
/F	Запуск интерпретатора языка ФОКАЛ, при этом расширения файлов по умолчанию — .FOC
/L:n	Адрес загрузки файла (400...77776)
/M	Обмен данными с магнитофоном (по умолчанию — с дисководом)
/N	Файл не формата .BIN
/P:n	Установка палитры n
/R	Запуск ПЗУ (БЕЙСИК или ФОКАЛ — в зависимости от наличия ключа B)
/S[:n]	При наличии параметра для ключа S — установка адреса старта, при отсутствии — только загрузка файлов с выходом в монитор
/T	Загрузка файла по адресу 120000

В режиме чтения требуется ввести имя файла на ленте (пустое имя соответствует первому встреченному файлу) и включить магнитофон в режим воспроизведения. В случае удачного считывания программа запрашивает имя файла для записи на диск и (при отсутствии ошибок) выводит начальный адрес, длину файла и признак нормального завершения операции «OK». При возникновении ошибок выдается соответствующая диагностика и повторяется запрос «Command:».

При записи файлов на ленту аналогично производится запрос имени файла на диске, затем имени для записи на ленту, при удачном окончании операции на экран выводятся параметры записанного файла и признак нормального завершения «OK». В случае возникновения ошибок операция прекращается, выдается диагностическое сообщение и происходит возврат на запрос «Command».

Нажатие клавиши КТ при вводе имени файла производит возврат к запросу «Command».

Утилита TR10 предназначена для преобразования текстовых файлов, полученных на БК-0010 с помощью редакторов Lexic, EDASP и др. Запуск TR10 осуществляется командой **.RUN DEV:TR10** (или **.TR10**, если она расположена на системном диске).

Параметрами программы являются имена входных и выходного файлов, записанные по стандарту CSI. По умолчанию входные файлы имеют расширение .DAT, выходной — расширение .TXT . Имя выходного файла может быть не указано, при этом оно будет тем же, что и у входного, но с расширением .TXT . Если указано

несколько входных файлов, то будет получен объединенный выходной файл с именем первого входного.

Программа распознает два ключа:

/L — по окончании преобразования выводится длина полученного файла;

/8 — формат КОИ-8 (без ключа — КОИ-7).

Примеры:

```
.R TR10
Files... TEXT.TXT=TEXT.DAT
Files... ^C

.TR10 TEXT10 OUTTXT/8
.

.TR10 TEXT1,TEXT2,TEXT3 TEXTC/8/L
Total xxxxxx. bytes in xx.block(s)
.
```

Диагностические сообщения

При запуске исполняющей системы:

Use the 'Setswp disk' command — для операционных систем V2.3 и старше требуется освободить дополнительные страницы ОЗУ командой Setswp disk.

Insufficient memory — недостаточно памяти, следует выгрузить не используемые в настоящий момент драйверы.

No device E10 — «нет устройства E10», следует назначить устройству с модулями системы логическое имя E10.

Invalid option — неверный ключ.

Invalid value specified with option — неверное значение ключа.

Conflicting options — конфликтующие ключи (недопустимо совместное использование ключей /T и /R, /T и /F).

... requires a value — для данного ключа требуется значение.

ROM error — отсутствуют или неисправны ПЗУ в страницах 10—12.

Channel in use — «канал занят», ошибка операционной системы.

File not found — на устройстве E10 не найден модуль исполняющей системы.

Input error — ошибка чтения модуля исполняющей системы.

При работе программы BINIO:

File not found — не найден файл с данным именем.

Input error — ошибка чтения файла с диска.

Output device full — диск или его каталог полностью заполнен.

Write error — ошибка записи на диск.

xxxxx/xxxxx — найден файл с именем <FILE>, не совпадающим с заданным (xxxxx/xxxxx — адрес загрузки и длина найденного файла).

Checksum error — ошибка контрольной суммы при считывании файла с ленты.

Address — недопустимый диапазон адресов.

Trap to 4 — прерывание по вектору 4 или по клавише СТОП.

В процессе работы программы TR10:

Bad fetch — ошибка загрузки драйвера внешнего устройства.

Channel in use — канал занят, системная ошибка.

Directory output error — ошибка в каталоге выходного устройства.

File not found — входной файл не найден.

Input error — ошибка чтения входного файла.

Insufficient memory — недостаточно памяти для работы программы.

Invalid command — неверная команда.

Invalid option — в командной строке указан неверный ключ.

No input file — в командной строке не указан входной файл.

Output device full — выходное устройство заполнено.

Output file error — ошибка записи выходного файла.

Output file full — выходной файл полон, недостаточно места на диске.

Protected file already exists — на выходном устройстве существует защищенный файл с тем же именем.

КРАТКИЙ СПРАВОЧНИК ПО СИСТЕМЕ EXE10 PLUS

Исполняющая система EXE10 Plus предоставляет пользователю возможность эксплуатации на персональной ЭВМ БК-0011М программ для БК-0010 в машинных кодах, на языках ФОКАЛ и БЕЙСИК. Ниже приведены наиболее часто используемые операции при работе с исполняющей системой.

1. Установка системы.

Нужно разместить исполняющую систему на системном диске и выполнить команду **.Assign SY E10** (на системных дискетах, поставляемых «ИнтерСервером», эта команда включена в стартовый командный файл и выполняется автоматически). Если операционная система версии 2.3 и выше находится в виртуальном режиме работы, то следует перейти к стандартному режиму командой **.Setswp Disk .**

2. Исполнение программ в машинных кодах.

Запуск программ БК-0010 производится командой **.EXE10P FILE,** где **FILE** — имя файла, имеющего расширение .BIN и расположенного на рабочем диске.

3. Работа с ФОКАЛом.

Для работы с интерпретатором языка ФОКАЛ выполните команду **.EXE10P /F** . Происходит запуск ФОКАЛа. Файлы программ считываются и записываются командами Library Get и Library Save аналогично работе на БК-0010:

***L G FOCTST**

***L S PROG**

Имена файлов с текстами программ на ФОКАЛе имеют расширение .FOC.

4. Работа на БЕЙСИКе BASIC/10.

Запуск БЕЙСИКа осуществляется командой **.EXE10P /B** . В операционных системах, поставляемых ИнтерСервером, в UCL задана команда, осуществляющая все необходимые операции для запуска БЕЙСИКа и загружающая жирный шрифт на время работы с ним: .BASIC10 .

Непосредственный запуск программ на БЕЙСИКе из операционной системы может быть осуществлен командой **.EXE10P FILE.ASC/B** или **.EXE10P FILE.COD/B** . При этом БЕЙСИК-программа обязательно должна располагаться на рабочем диске (DK:).

5. Возврат в операционную систему.

В большинстве программ выход в операционную систему осуществляется штатно или по клавише СТОП. Если управление при этом передано монитору БК-0010 (приглашение «?»), то следует дать команду **? P** .

Возврат из ФОКАЛа:

*** P M**

? P

Возврат из БЕЙСИКа:

Ok

MONIT (или комбинация клавиш AP2/0) .

Если программа на БЕЙСИКе запущена непосредственно из операционной системы, то возврат в ОС происходит автоматически при ее завершении (например, по клавише КТ).

6. По окончании работы с системой.

Для ОС версии 2.3 и выше можно вернуться в виртуальный режим работы командой **.Setswp Memory** .

7. Система Exe10 Plus и InterCommander.

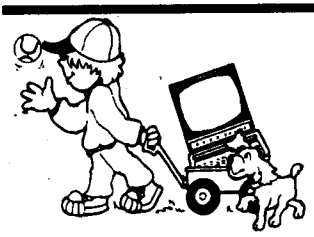
При работе с операционной оболочкой InterCommander в файл IC.EXT можно добавить следующие команды:

```
BIN: Exe10P !:!
ASC,COD: Exe10P !.!/B
```

Для версии оболочки 0.8 и старше желательно создать также меню под названием «БК-0010», куда следует включить основные функции исполняющей системы (например, запуск интерпретатора ФОКАЛ, загрузку программы по адресу 120000 или преобразование текстового файла БК-0010 в формат RT-11):

```
B: Режимы БК-0010
#4
. . . . . .
^L#4:Режимы БК-0010
M: Монитор
 Exe10P /S
F: ФОКАЛ
 Exe10P /F
B: БЕЙСИК
 PCfont_Exe10P /B_STfont
H: 120000
 Exe10P !:!/T
T: Преобразование текстов
 TR10 !:!.!/L/8
```

НАЧИНАЮЩИМ ПОЛЬЗОВАТЕЛЯМ



Казалось бы, нет ничего проще, чем дать программе имя при записи или ввести его при чтении файла с магнитофона. Однако начинающие пользователи и здесь могут столкнуться с различными трудностями. Постараемся помочь им.

От редакции

ПОГОВОРИМ ОБ ИМЕНАХ

Имя программы в БК-0010 представляет собой цепочку из 16 символов, записываемую в заголовке файла вместе с его начальным адресом и длиной. При чтении или записи имя, введенное пользователем, помещается в буфер функции EMT36, начальный адрес которого заносится в регистр R1. Этот адрес может быть любым, но в качестве стандартного значения в БК-0010 выбрано 320_8. Оно используется в системных подпрограммах обработки команд «М» монитора, «МЧ», «МЗ» и «МФ» режима ТС, а также во всех операторах обмена с магнитофоном в ФОКАЛе и БЕЙСИКе. При этом символы имени записываются в область 326_8—345_8 (16 байт), а названия встреченных при чтении или поиске программ, если они не совпадают с именем, указанным пользователем, попадают в область 352_8—371_8. Если введенное пользователем имя имеет длину менее 16 символов, компьютер автоматически дополняет его незначащими пробелами справа.

Имя программы весьма часто позволяет сразу определить не только ее назначение, но и язык, на котором она написана. Наиболее широкие возможности при этом предоставляют пользователю монитор и режим ТС. Имя программы в кодах может состоять из любых символов, имеющихся в БК: букв (русских, латинских, заглавных и строчных), знаков, полуграфики, пробелов и даже кодов, не имеющих в БК графического представления (движение курсора, ГРАФ, ЗАП, СТИР и т. д.). Такой свободный подход позволяет не только давать программам в кодах «недвусмысленные» имена, но и при необходимости считывать в монитор и ТС любые файлы в формате ФОКАЛа и вильнюсского БЕЙСИКа.

ФОКАЛ не позволяет таких «вольностей» — в нем допускаются только русские и латинские буквы. Так что наличие в имени программы «запрещенных» символов — верный признак того, что она написана в машинных кодах. Однако, за исключением «нелюбви» ФОКАЛа к «небуквенным» символам, структура имени для его программ ничем не отличается от таковой для программ в кодах: та же цепочка символов, дополненная при необходимости справа пробелами до 16 байт. Из-за этого нередко случается путаница: пользователь по ошибке считывает ФОКАЛ-программу в монитор и пытается запустить ее командой «S» (после чего БК преспокойно «вылетает» в ФОКАЛ или БЕЙСИК) или загружает программу в кодах в ФОКАЛ. После этого, как правило, следует искреннее удивление тем, что считанная без сообщений об ошибке программа неработоспособна, или (во втором случае) — «хулиганским проделкам» компьютера при попытке получения листинга с помощью команды «W».

При считывании программы в копировщики типа HELP, HELP3, КОПИР, HELP7 и т.п., выводящие на экран начальный адрес, длину и имя копируемого файла, основным признаком ФОКАЛ-программы является начальный адрес 1752_8, но при чтении в монитор или режим ТС этот адрес не выдается (хотя в ТС его можно просмотреть командой «346АИ»), так что остается только посоветовать пользователям заранее давать программам на ФОКАЛе «особенные» имена, например начинающиеся с буквы F.

Создатели вильнюсской версии БЕЙСИКа «любезно избавили» пользователя от подобного рода ошибок. Наличие у имени программы расширения .ASC, .COD, .BIN, .DAT

и других однозначно свидетельствует: это БЕЙСИК. Рассмотрим структуру имен с расширениями .ASC, .COD и .BIN, а также способы чтения таких программ, например, в режиме ТС.

Программы с расширением .ASC

Имя имеет формат:

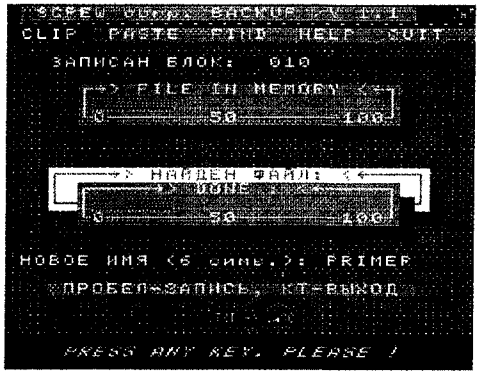
<******>.ASC␣#<номер><↑>,

где <******> — текст из 6 символов, <номер> — номер блока из трех цифр, знаком <␣> обозначен пробел, а <↑> — видимый только при включенном режиме БЛОК РЕД и работе команд чтения/поиска код «курсор вверх», который, собственно, и обеспечивает эффект «смены» номеров блоков в одной и той же строке. Имя последнего блока «укорочено» и имеет вид:

<******> .ASC␣␣<0000>,

где <0000> — четыре завершающих «символа» со «странным» кодом 0, отображаемых на экране при включенных режимах ИНД СУ и БЛОК РЕД в виде инверсных знаков «@». Для чтения текста БЕЙСИК-программы в ТС нужно, задавая имена с последовательно увеличивающимися номерами, считывать командой «МЧ» блок за блоком. (Следите за начальными адресами! Для каждого последующего блока адрес нужно вычислять по правилам восьмеричной арифметики, прибавляя к значению для предыдущего 400_8.) Последний блок (без номера) имеет длину 2 байта и служит всего лишь признаком конца программы, считывать его необязательно. Можно также читать программу в формате .ASC в текстовый редактор типа TED6, также блок за блоком (в этом случае о начальных адресах позаботится сам компьютер). В результате в редактор будет загружен текст программы, полностью аналогичный распечатке операторами LIST и LLIST. Аналогично можно преобразовывать содержащие только текст (!) файлы типа .DAT. (Следует заметить, что, в отличие от общепринятого в текстовых редакторах на БК правила обозначать «конец строки» кодом 0, в загруженных с БЕЙСИКа листингах используется код 12_8, поэтому после чтения, например, в TED6 нужно дать команду «FO».)

В редакции можно приобрести программу SCREW BACKUP SYSTEM, которая позволяет свободно преобразовывать листинги БЕЙСИК-программ из формата .ASC в EDASP и обратно, в том числе для переноса листингов на IBM.



Программы с расширением .COD

Содержимое файлов с таким расширением может понадобиться разве только для исследовательских целей. Это внутренний формат записи текста программы, в котором стандартные операторы БЕЙСИКа заменены их условными кодами. Тем не менее стоит поговорить и об этих файлах.

Имя в .COD-формате представляет собой конструкцию:

<******> .COD,

где <000000> — шесть «символов» с кодом 0 (инверсное «@»). Сам файл выглядит как единый блок и может быть считан как обычная программа в кодах, однако для ввода в имя нулевых байтов необходимо написать специальную кодовую программку, так как сделать это с клавиатуры невозможно.

Программы с расширением .BIN

Чаще всего это подпрограммы в машинных кодах (USR) или копии экрана. Имя с таким расширением — наиболее простое и привычное по структуре из всех «стандартных» имен БЕЙСИКа. К шести символам, введенным пользователем в операторе BSAVE, добавляются «.BIN» и пробелы справа.

Часто требуется, наоборот, загрузить в БЕЙСИК какую-либо подпрограмму в кодах, имеющую «нестандартное» имя (более шести символов и без расширения .BIN). Самый простой способ — считать файл в режиме ТС и записать на магнитофон заново, переименовав в <******> .BIN, а затем грузить в БЕЙСИК оператором BLOAD. Но на это требуются время, расход магнитной ленты и определенные усилия. Можно загрузить такую программу и непосредственно, воспользовавшись короткой подпрограммой в кодах (она перемещаема):

Длина: 50_8. Контрольная сумма: 125033_8.

```
020327  000377  001401  000207
010046  010146  010246  010346
```

```
010446  010546  011505  004737
100536  012605  012604  012603
012602  012601  012600  000207
```

Подпрограмма вызывается оператором A%=USR(A%), где A% — целое число, указывающее начальный адрес загрузки. Если в качестве аргумента USR-функции задано не целое число, происходит выход в БЕЙСИК без чтения, иначе появляется запрос «ИМЯ?» и БК загружает файл с проверкой контрольной суммы.

Отдельно поговорим об именах программ при использовании на БК дисковода. Так как изначально практически все программное обеспечение было ориентировано на работу с магнитофоном, создатели дисковых операционных систем стараются максимально приблизить используемый формат имен к «старым стандартам». Соответственно, для операций обмена с диском используется модифицированная подпрограмма реализации EMT36, вызов которой (и работа с буфером для EMT36) осуществляется точно так же, как и с магнитофоном. В качестве примера рассмотрим наиболее распространенные ОС NORD и ANDOS.

Имя файла в системе NORD (как и в большинстве ОС, построенных «по ДВК-шной идеологии») может состоять только из 14 символов. Соответственно, при переписываний файла с магнитофона на диск два последних символа «срезаются». Кроме того, имя в ОС NORD не должно начинаться с знака «\» (это признак имени подкаталога). Не разрешается использовать в составе имени и сочетания DIR, SWAP и BACK, а также одиночный символ «/», являющиеся служебными командами. И наконец, если предполагается запускать программу из Нортон-подобной оболочки, «внутри» ее имени не должно быть пробелов.

В ОС ANDOS, ориентированной на IBM-подобный стандарт записи информации на диск, допускаются имена файлов, состоящие из восьми символов и записанного через точку трехсимвольного расширения (т.е. как и в IBM — см. [1]). А чтобы оставалась возможность работы с программами для магнитофона, имеющими 16 символов, используется весьма хитрый алгоритм преобразования записанного в буфере EMT36 «длинного» имени в допустимое в ANDOSe (см. [2]). Что же касается действительно существенных ограничений, не допускается использование заглавного русского Е в качестве первого символа (это признак удаленного файла) и имен из одних пробелов или одной только точки (в ответ на это выводится каталог диска).

Следует отметить, что при работе с БК, оснащенной дисководами, имеется еще одна особенность. Если ранее существовало только одно устройство хранения информации (магнитофон), то теперь кроме него имеется один или два дисковода. Чтобы иметь возможность «адресоваться» к конкретному устройству, в начале имени можно указывать пары символов «М:» — магнитофон, «А:» и «В:» — первый и второй дисководы соответственно.

И последнее замечание. Нередко пользователи БК дают своим программам «секретные» имена, чтобы защитить их от несанкционированного доступа. Пример такого имени — «АААААА», где часть букв «А» — русские, а часть — латинские (в знакогенераторе БК эти и некоторые другие пары русских и латинских букв практически неразличимы). Позже пользователь забывает, какое имя он дал программе, и потом сам мучительно пытается ее загрузить. Несложная программа позволит легко расшифровать коды символов имени. После ее запуска (для чего нужно записать ее начальный адрес в ячейку 30_8 командой «Щ30АИ<нач.адрес><запятая>») символы любых сообщений (текстов и чисел), поступающих на экран, выводятся один за другим на отдельных строках с указанием их восьмеричных кодов. Теперь достаточно ввести команду «МФ», и вы сможете узнать коды символов, составляющих любое встреченное компьютером имя.

Длина: 140_8. Контрольная сумма: 035753_8.

```
010700  062700  000012  010037
000030  000000  010546  016605
000002  014505  020527  104016
001403  012605  000137  100112
010046  010146  010246  010346
010446  013746  000030  012737
100112  000030  010004  104016
012700  000040  104016  104016
042700  177400  004737  163220
012700  000012  104016  012637
000030  012604  012603  012602
012601  012600  012605  000002
```

Эта программа может работать только в режиме ТС, так как в ней используется обращение к подпрограмме, прошитой в ПЗУ МСТД (@#163220).

Литература:

1. *Брябрин В. М.* Программное обеспечение персональных ЭВМ. М.: Наука, 1989. С.22.

2. *Юров В. П.* БК-0010(.01) с дисководом //Персональный компьютер БК-0010—БК-0011М. 1993. № 1. С.41.

А. Г. Прудковский,
г. Москва

НЕСКОЛЬКО СЛОВ О ПЕРЕМЕЩАЕМЫХ ПРОГРАММАХ

Чтобы запустить ту или иную программу на выполиение, нужно предварительно загрузить ее в оперативную память компьютера. Большинство программ рассчитано на загрузку только с одного заранее оговоренного адреса, и поэтому они называются неперемещаемыми. Некоторые же программы, называемые перемещаемыми (другое название — «позиционно-независимые»), можно загружать в любую область памяти, с любого адреса. Такая универсальность может потребоваться, когда речь идет о каком-либо автономном драйвере, подгружаемом к различным программам. (Например, один и тот же драйвер печати копии экрана может использоваться в БЕЙСИКе, в ФОКАЛе, в графических редакторах серии GRAF и т. д., причем адрес его загрузки в каждом случае может быть разным.) Другой пример — программы, динамически распределяющие в ОЗУ отдельные рабочие модули (тот же GRAF постоянно «перетасовывает» в памяти драйвер печати экрана вместе с рисунками, а текстовый редактор VorteX!, определяя при загрузке конфигурацию БК, либо оставляет все свои рабочие модули в основном ОЗУ, либо переносит часть из них в расширенное), при этом конкретные адреса их размещения заранее просто неизвестны.

Как же обеспечить перемещаемость конкретной программы? Для этого нужно при ее написании использовать «перемещаемые» команды ассемблера.

Система команд БК (и других компьютеров, относящихся к серии DEC) предполагает восемь способов адресации с использованием регистров общего назначения (n — номер регистра, от 0 до 7):

Мнемоника	Код	Название	Пример
Rn	0n	Регистровая	TST R1
@Rn или (Rn)	1n	Косвенная	TST @R1
(Rn)+	2n	С автоувеличением	TST (R1)+
@(Rn)+	3n	Косвенная с автоувеличением	TST @(R1)+
-(Rn)	4n	С автоуменьшением	TST -(R1)
@-(Rn)	5n	Косвенная с автоуменьшением	TST @-(R1)
E(Rn)	6n	С индексацией	TST 6(R1)
@E(Rn)	7n	Косвенная с индексацией	TST @2(R1)

Примечание. В качестве примера выбрана одноперандная команда тестирования содержимого регистра или ячейки памяти. В двухоперандных командах могут комбинироваться различные способы адресации, например **MOV -(R2), R1**.

Первая из этих команд (с кодом 0n) работает только с регистрами, остальные же обращаются к каким-либо ячейкам памяти, и, следовательно, регистры, входящие в эти команды, должны содержать адреса требуемых ячеек.

При переписывании чисел из ячеек памяти в регистры и обратно или при записи в них констант применяются команды, использующие регистр R7. (Напомним, что в ассемблере он имеет специальное обозначение PC, так же как регистр R6 обозначается SP.) Эти команды, содержащие в себе явное указание констант, адресов ячеек памяти или имен меток, используют специальную символику в записи способа адресации (как это показано в примерах):

	Общепринятая запись	Эквивалентная запись через PC	Выполняемое действие
1)	MOV #5,R1	MOV (PC)+,R1 .#5	Запись в регистр R1 константы 5
2)	MOV #A,R1	MOV (PC)+,R1 .#A	Запись в регистр R1 абсолютного значения адреса метки A
3)	MOV @#A,R1	MOV @(PC)+,R1 .#A	Запись в регистр R1 содержимого ячейки памяти с меткой A
4)	MOV A,R1	MOV a(PC),R1	Запись в регистр R1 содержимого ячейки памяти с меткой A
5)	MOV @A,R1	MOV @a(PC),R1	Запись в регистр R1 содержимого ячейки памяти, адрес которой записан в ячейке с меткой A

6)	Отсутствует	MOV PC,R1
		ADD (PC)+,R1
		.@A+2
			Запись в регистр R1 абсолютного значения адреса метки A

Примечание. В командах под номерами (4) и (5) через **a** обозначена величина **.@A**, вычисляемая при трансляции с ассемблера как смещение от текущего адреса до метки A. В отличие от нее, обозначение **.#A** подразумевает указание конкретного адреса (или имени метки, взамен которого ассемблер-транслятор подставляет ее вычисленный адрес). Следует также заметить, что для случая (6) не существует отдельной записи в виде одной команды ассемблера. (Обозначения типа **.#A** и **.@A** соответствуют стандарту ассемблер-транслятора М18, для других трансляторов запись может быть иной. — *Прим.ред.*)

Легко видеть, что по назначению команды (2) и (3) эквивалентны командам (6) и (4) соответственно. Но не менее очевидна и разница между ними. Команды (2) и (3) могут быть использованы только в неперемещаемых программах, так как в них предполагается обращение к ячейкам с конкретными адресами (при написании программы нужно указать либо сам адрес, либо имя метки, которое транслятор автоматически заменяет на значение адреса, вычисляя его по заданному пользователем в момент компоновки начальному адресу программы). В отличие от них, команды (4) и (6) используют только смещение от текущего адреса (т. е. некий «относительный» адрес, «система отсчета» для которого связана с самой программой). Конкретные же адреса ячеек при этом знать не обязательно, а следовательно, эти команды обеспечивают перемещаемость программы.

Что же касается двух оставшихся случаев, то команда (1), очевидно, является перемещаемой, так как в ней не содержатся никакие адреса ячеек памяти (есть только число-константа). А вот достаточно редко применяемая команда (5) хотя и использует «позиционно-независимую» величину — смещение **.@A** до метки A, но содержимое ячейки с этой меткой рассматривается как абсолютный адрес операнда, т. е. эта команда неперемещаемая (перемещаемого аналога для нее не существует).

В некоторых новых, более расширенных по сравнению с М18, ассемблер-трансляторах имеется еще один способ обращения к ячейкам памяти через регистр, по сути представляющий собой вариант адресации с индексацией, например **TST #<имя метки>(Rn)**. Эта команда весьма удобна для реализации обращения к одномерным массивам (метка — начало массива (нулевой элемент), регистр содержит значение индекса), к таблицам и т. д., но, очевидно, является абсолютно неперемещаемой (так как значение адреса метки вычисляется в момент трансляции в соответствии с заданным начальным адресом программы). Для перемещаемых программ эту команду можно заменить на следующие ассемблерные фрагменты:

```
а) MOV PC, Rm
   ADD (PC)+, Rm
   .@<имя метки>+2
   ADD Rn, Rm
        ; адрес операнда находится
   ; в дополнительном регистре Rm
   TST @Rm    ; пример обращения
```

```
б) MOV PC, КОМ+2
   ADD (PC)+, КОМ+2
   .@<имя метки>+2
   ADD Rn, КОМ+2
        ; заносится адреса вычисляется
        ; и заносится непосредственно
        ; в тело команды обращения
   ; к операнду (использование Rm
   ; не требуется)
КОМ: TST @#0        ; "шаблон" команды
```

(В случае (б) **КОМ+2** соответствует использованию при обращении к операнду одноопернадной команды или двухоперандной, когда обсуждаемый способ адресации применяется для первого операнда, например аналога записи типа **MOV <#имя метки>(Rn),Rm**. Если же этот способ адресации задействован во втором операнде, скажем, **MOV Rm,<#имя метки>(Rn)**, то в ассемблерном фрагменте, подобном (б), нужно писать **КОМ+4.**)

Следует обратить внимание на следующий важный факт. Все ранее сказанное касалось только адресации к ячейкам памяти, находящимся в теле перемещаемой программы. Если же необходимо обратиться к внешним ячейкам, расположение которых не зависит от данной программы и поэтому их адреса можно считать «фиксированными» (это ячейки системной области и стека БК, регистры устройств, содержимое ПЗУ или ОЗУ экрана при работе, например, со спрайтами, а также ячейки «неподвижной» части основной программы, состоящей из нескольких фрагментов), то здесь все обстоит «с точностью до наоборот» — нужно использовать именно команды (2) и (3). Как в математике «минус на минус дает плюс», так и при обращении к «неподвижным» ячейкам «неперемещаемые» команды обеспечивают перемещаемость программы.

Итак, чтобы программа была перемещаемой, при ее написании необходимо помнить следующее:

- допускается использование любых адресаций между регистрами и команд типа (1) для записи в регистры констант;
- при обращении к ячейкам памяти внутри программы и при вызове подпрограмм нужно использовать команды типа (4), например **MOV NOM,R1** или **JSR PC,NAME** ;
- для записи в регистр абсолютного значения адреса нужно использовать комбинацию команд типа (6);
- при обращении к векторам прерываний, к служебным ячейкам и к системным регистрам, при вызове подпрограмм из ПЗУ,— словом, при любых обращениях к ячейкам памяти, не входящим в тело перемещаемой программы, нужно использовать команды типа (2) и (3) вместо (4) и (6).

В качестве примера рассмотрим несложную программу, управляющую движением «жука» (буква Ж) по экрану. Программа, приведенная в левой колонке, неперемещаема. Справа показаны варианты записи команд ассемблера, превращающие ее в перемещаемую.

```
;       НЕПЕРЕМЕЩАЕМАЯ      КОММЕНТАРИИ          ВАРИАНТ ДЛЯ
;       ПРОГРАММА                                ПЕРЕМЕЩАЕМОЙ
                                                 ПРОГРАММЫ
        EMT 14              ; инициализация
        MOV #232,R0         ; отключение
        EMT 16              ; курсора
        MOV #233,R0         ; включение
        EMT 16              ;  цветного режима
X:      MOV #10,R1          ; координата X и
Y:      MOV #10,R2          ;  координата Y
        EMT 24              ;  буквы Ж
        MOV 'Ж',R0          ; выдача на экран
        EMT 16              ;  буквы Ж
        JSR PC,KLAV         ; обращение к клавиатуре (код — в R4)
```

```
        MOV #30,R0      ; стирание
        EMT 16          ;  буквы Ж
V:      MOV #K,R1       ; занесение                    V: MOV PC,R1
                        ;  адреса метки K:                ADD (PC)+,R1
                        ;  в регистр R1                   .@K+2
W:      MOV #A,R2       ; занесение                    W: MOV PC,R2
                        ;  адреса метки A:                 ADD (PC)+,R2
                        ;  в регистр R2                   .@A+2
2:      CMPB (R1)+,R4   ; сравнение кода R4 с байтами массива K:,
        BEQ 3           ;  если код совпал, то выход на метку 3:
        TST (R2)+       ;  прокрутка массива адресов A: на 2 байта
        TSTB (R1)       ; проверка на конец массива K: (в конце—0)
        BNE 2           ; если не 0, то продолжение сравнения
3:      JSR PC,@(R2)+   ; выполнение программы,        3: ADD (R2),R2
                        ;  соответствующей коду           JSR PC,(R2)
        BR X            ; возврат на печать буквы Ж
K:      .B:10.B:31      ; список кодов, требующих обработки
        .B:32.B:33
        .E              ; конец списка — 0
A:      .#A10.#A31      ; список адресов (смещений)    A: .@A10.@A31
        .#A32.#A33      ;  программ, обрабатывающих       .@A32.@A33
        .#ERR           ;  коды, последняя — обраба-      .@ERR
                        ;  тывает коды, не включенные
                        ;  в список K:
A10:    TST X+2         ; программа, сдвигающая букву
        BEQ 1           ;  влево
        DEC X+2
1:      RTS PC
A31:    CMP X+2,#36     ; программа, сдвигающая букву
        BEQ 1           ;  вправо
        INC X+2
1:      RTS PC
A32:    TST Y+2         ; программа, сдвигающая букву
        BEQ 1           ;  вверх
        DEC Y+2
1:      RTS PC
A33:    CMP Y+2,#27     ; программа, сдвигающая букву
        BEQ 1           ;  вниз
        INC Y+2
1:      RTS PC
ERR:    CLR R1          ; программа, распечатывающая
        MOV 'E',R0      ;  слово ERR в служебной строке
        EMT 22
        INC R1
        MOV 'R',R0
        EMT 22
        INC R1
        EMT 22
        JSR PC,KLAV     ; задержка до нажатия клавиши
        MOV #236,R0     ; очистка служебной строки
        EMT 16
        RTS PC
```

```
                ; программа обращения к клавиатуре
KLAV:  MOV #20000,R0        ; задание «скорости» движения буквы по
                            ; экрану (чем меньше число — тем быстрее),
3:     TST @#177662         ; очистка клавиатурного регистра от прежних
       SOB R0,3             ; нажатий клавиш и задержка в соответствии
                            ; с содержимым регистра R0
1:     BIT #100,@#177716    ; ожидание нажатия клавиши
       BNE 1
       MOV @#177662,R4      ; занесение кода клавиши в регистр R4
       RTS PC
       END
```

При трансляции обоих вариантов программы-примера (неперемещаемого и перемещаемого) следует обратить внимание на одну особенность работы ассемблер-транслятора (дальнейшее относится конкретно к системе типа М18, но подобная функция предусмотрена и в большинстве других трансляторов). После компиляции (по команде СО) на экран выводятся имена использованных в программе глобальных меток и соответствующие им адреса. При этом некоторые имена меток выводятся инверсно и эта инверсия снимается только после указания конкретного начального адреса при компоновке (команды LA, LI и т. п.). И если наличие «инверсных» имен меток после компоновки является признаком ошибки (ссылка на отсутствующую метку или неправильная запись ее имени), то «инверсные» имена после компиляции, по сути, указывают на использование «неперемещаемого» типа адресации. Отсюда следует простой способ контроля перемещаемости создаваемой программы: если после компиляции (но перед компоновкой!) хотя бы одно имя метки выводится на экран инверсно, программа неперемещаема. Если же все метки «неинверсные», программа перемещаема, а последующее указание с помощью команды транслятора LA начального адреса представляет собой простую формальность. (Вариант компоновки перемещаемой программы из нескольких объектных модулей здесь не рассматривается.)

Напоследок отметим и еще одну особенность. Как это можно видеть из приведенного выше примера, перемещаемая программа на 4 байта длиннее, чем неперемещаемая. Это немного, если учесть, что общая ее длина составляет 272 (восьм.) байта. Однако для более длинных программ это различие может быть и большим, а на БК с ее извечным дефицитом памяти даже несколько байт могут оказаться решающими, если программу нужно «втиснуть» в область ОЗУ с заданными размерами. В подобном случае для сокращения длины программы приходится жертвовать ее перемещаемостью. Если же конкретное расположение программы заранее неизвестно, иногда можно использовать модификатор адреса, настраивающий ее на заданный начальный адрес. Для вышеприведенного примера этот модификатор может выглядеть следующим образом:

```
       MOV PC,R1            ; вычисление истинного адреса метки A
       ADD (PC)+,R1
       .@A+2
       MOV R1,R0            ; вычисление смещения от условного адреса
       SUB #A,R0            ; метки #A до истинного адреса
       MOV R1,W+2           ; модификация адреса W+2
       ADD R0,V+2           ; модификация адреса V+2
       MOV #5,R3            ; модификация
1:     ADD R0,(R1)+         ; массива адресов
       SOB R3,1             ; по метке A
```

На первый взгляд может показаться, что использование подобного модификатора абсолютно неоправданно, так как, помимо повышенного внимания при написании, его длина значительно больше сэкономленной разницы между перемещаемой и неперемещаемой программой. Однако заметим, что после срабатывания он становится уже не нужным и занимаемую им область памяти можно использовать для других целей.

От редакции

АВТОЗАПУСК ДЛЯ ПРОГРАММ В МАШИННЫХ КОДАХ

В большинстве случаев пользователи БК-0010(.01) запускают программы в кодах из монитора. Требуемые для считывания программы с магнитной ленты и ее запуска действия очень просты:

? M
Имя=**<имя>**
<имя>
? S <адрес>

БК читает записи программ на ленте, выдавая на экран имена встреченных файлов. После окончания чтения (если не было ошибки) БК печатает имя считанного файла. Команда «S <адрес>» запускает программу с заданного адреса или, если он не указан, — с адреса загрузки.

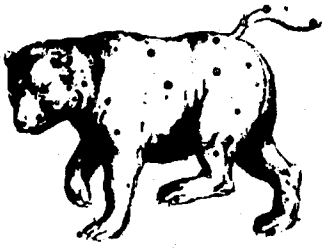
Такой способ используется почти для всех программ в кодах. Однако некоторые из них запускаются сами после окончания считывания с магнитофона, не дожидаясь ввода пользователем команды «S». Это их свойство повсеместно называется автозапуском. Кроме внешней эффектности автозапуск имеет и еще одно преимущество: если магнитофон оснащен электронным управлением и БК сам выключает его после окончания считывания, то ваши заботы исчерпываются вводом команды «M» и имени файла.

При работе с копировщиками легко заметить важный отличительный признак программ с автозапуском: все они имеют начальный адрес меньше 1000_8 (например, 760_8). Вот здесь-то собака и зарыта! Ведь область ОЗУ с адресами, меньшими 1000_8, в БК-0010(.01) отводится под стек. При пуске монитора значение указателя стека (регистр R6) приравнивается исходному 1000_8. Когда пользователь вводит команду «M» (считывание файла с магнитной ленты), монитор вызывает выполняющую чтение подпрограмму с помощью оператора ассемблера JSR R7,<адрес>. При этом процессор заносит в стек адрес возврата, т. е. адрес команды, следующей после оператора JSR. При считывании программы с магнитофона в ОЗУ ее начало (первые 20 байт в случае начального адреса, равного 760_8) попадает в стек, вытесняя содержавшиеся там предыдущие значения. Эти первые байты содержат адрес запуска (для гарантии «попадания» адреса запуска в нужное место стека он повторен несколько раз).

После окончания работы подпрограммы чтения по команде RTS R7 процессор должен извлечь из стека записанный туда адрес возврата, чтобы продолжить выполнение основной программы. Однако вместо адреса возврата в монитор на этом месте уже находится наш адрес запуска, и ничего не подозревающий процессор вместо команд монитора начинает выполнять команды только что считанной программы. Автозапуск произведен!

Кстати, из всего сказанного выше следует объяснение разнообразных ошибок, встречающихся при запуске таких программ, когда БК либо сразу же после считывания «вылетает» в БЕЙСИК или ФОКАЛ, стирая введенную только что программу, либо начинает гудеть, щелкать, мигать курсором или просто «повисает», либо уже сама программа хотя и запускается, но работает неправильно. Причина этого — всего-навсего ошибка при считывании. В обычных условиях сразу же после возврата из подпрограммы чтения с магнитофона производится подсчет контрольной суммы введенного файла и сравнение ее с записанным в файле числом. Если эти два значения не совпадают, то БК выдает сообщение об ошибке магнитофона. В нашем же случае БК до сравнения контрольных сумм просто не успевает дойти, программа запускается на выполнение сразу же после чтения. И если оно произведено с ошибкой, то процессор, не замечая этого и выполняя ошибочную программу, рано или поздно наталкивается на эту ошибку, и от нее зависит, как дальше поведет себя машина и какие фокусы она выкинет.

ОБМЕН ОПЫТОМ



Е. А. Еремин,
г. Пермь

РЕАЛИЗАЦИЯ ЦИКЛОВ С ПЕРЕМЕННЫМ СОДЕРЖИМЫМ ПРИ ПОМОЩИ СОПОДПРОГРАММ

Этот материал появился после прочтения в журнале «Информатика и образование» статьи Ю. Гиматова [1]. Последняя содержит целый ряд интересных идей и примеров, хотя некоторые из них применимы не всегда. В частности, самомодификация кода программ действительно встречается на практике редко, но отнюдь не по причине недогадливости программистов. Можно привести ряд серьезных возражений против такого приема: это принципиальная невозможность работы формирующей себя описанным способом программы в ПЗУ, нарушение процесса проверки с помощью контрольной суммы, получение при определенных условиях неидентичных копий на магнитных носителях для одной и той же программы и, наконец, значительное ухудшение читаемости листинга, что в большинстве случаев нежелательно. Впрочем, все эти аргументы направлены не столько на критику идей, изложенных в [1], сколько против их абсолютизации.

Легко получить компромиссное решение, при сохранении идеи самомодифицирования свободное от перечисленных недостатков. Для этого следует формировать изменяющуюся часть кода программы не внутри нее самой, а в некоторой рабочей области ОЗУ в виде подпрограммы, которая в дальнейшем вызывается обычным образом. Эта идея достаточно проста и в более детальном обсуждении не нуждается.

Существует и другой интересный способ создания самомодифицирующихся программ. Для реализации цикла, в теле которого при различных проходах выполняются разные действия, могут быть использованы соподпрограммы. Подчеркну, что подобная программная конструкция существует только для процессоров с системой команд типа PDP-11 с их развитой системой адресации; в процессорах INTEL и тем более в языках высокого уровня такая возможность не предусмотрена!

Что такое соподпрограмма? Это особый тип организации программы, когда две ее части по очереди вызывают одна другую (см., например, [2]). (Как известно, в классическом случае основная программа вызывает подпрограмму, но не наоборот.) Поскольку обе соподпрограммы функционально равноправны, то инструкция возврата RET в них не используется — вместо этого одна соподпрограмма просто вызывает другую и обратно.

Инструкция вызова соподпрограммы имеет вид **JSR PC, @(SP)+** и кодируется восьмеричным словом 004736. При выполнении этой команды процессор сначала извлекает из стека ранее сохраненный адрес входа в вызываемую соподпрограмму (о чем говорит запись операнда в виде @(SP)+) и сохраняет его в своей внутренней памяти. Затем, как и при обычном вызове подпрограммы, в стек аппаратно заносится адрес следующей за JSR PC инструкции и осуществляется переход по считанному ранее адресу. Нетрудно заметить,

что рассматриваемая модификация команды отличается от классического применения инструкции JSR только тем, что адрес начала вызываемой подпрограммы берется из стека. Такой прием как раз и позволяет передать управление альтернативной соподпрограмме и сохранить в стеке адрес для последующего возврата из нее. Для обратного перехода в исходную соподпрограмму достаточно проделать те же самые действия, т. е. снова выполнить ту же инструкцию процессора JSR PC, @(SP)+.

Для обеспечения нормального взаимодействия соподпрограмм необходимо перед их использованием занести в стек входной адрес второй соподпрограммы. После окончания работы с соподпрограммами ненужный «этаж» стека следует освободить.

Описанная программная конструкция применяется довольно редко и на первый взгляд кажется достаточно экзотической. Тем не менее на ее основе удается очень легко реализовать цикл с модифицируемыми командами. Тело цикла в этом случае служит одной соподпрограммой, а в роли другой выступают подставляемые в него по очереди команды, разделенные инструкциями JSR PC, @(SP)+.

Для иллюстрации рассмотрим следующую наглядную задачу. Пусть необходимо обойти все точки внутри расположенного на экране прямоугольника и закрасить их. Сделаем это не традиционным построчным сканированием, а более эффектным обходом по «закручивающейся спирали» (т. е. сначала нарисуем внешнюю границу прямоугольника, затем границу прямоугольника, оставшегося незакрашенным, и т. д.).

Обозначим координаты левого верхнего угла прямоугольника X1 и Y1, а правого нижнего — X2 и Y2. Тогда X2=X1+DX, Y2=Y1+DY, где DX и DY — длины сторон прямоугольника. Входные параметры X1 и X2 поместим в R1 и R2, а DX и DY — в R4 и R5 соответственно. При работе программы в R3 будем хранить адрес небольшого массива из четырех чисел: X1, Y1, X2, Y2. Чтобы не использовать конкретные ячейки памяти, массив разместим в стековой области (правда, задавать его при этом придется в обратном порядке!). Регистр R4 будет служить счетчиком точек.

Текст программы на ассемблере БК-0010 и коды команд с подробными комментариями приведены ниже. Основная часть программы состоит из подготовительных вычислений и короткого основного цикла, начинающегося после метки L2. Для каждой точки вызывается стандартная процедура закраски EMT 30 (подробности ее работы описаны, например, в [3]) и происходит обращение к соподпрограмме NEXTXY, возвращающей в R1 и R2 координаты следующей точки спирали. Листинг NEXTXY является, пожалуй, наиболее интересной частью программы. Первоначально вход в эту соподпрограмму соответствует метке M2, адрес которой был заранее занесен в стек. Пока текущее значение X не совпадет с X2, новая координата точки получается увеличением на единицу абсциссы, хранящейся в регистре R1. Когда X=X2, вход в соподпрограмму перезадается на метку M4, что при последующих обращениях приводит к сохранению прежнего значения X и инкрементированию Y вплоть до точки Y=Y2. Аналогично происходит движение и по двум оставшимся сторонам прямоугольника с той лишь разницей, что координаты при этом уменьшаются. После того как нарисован полный прямоугольник, текущие вершины сдвигаются «внутрь» и процедура повторяется.

Корректность окончания процесса существенно зависит от соотношения сторон прямоугольника. Если DY>DX, то при нечетном количестве точек по координате X остается незакрашенная вертикальная линия. Чтобы исправить положение, в конце соподпрограммы NEXTXY ставится специальная проверка, которая в случае X1=X2 гарантирует нормальное завершение.

Соподпрограмма NEXTXY может быть использована для смещения по спирали и в более сложных случаях, причем ее вызов можно помещать в нескольких местах основной программы. С ее помощью, например, легко определить координаты точки, расположенной на середине спирали, и т. п.

Приведенная ниже программа полностью перемещаема. Для заданных в ее листинге параметров на экране рисуется прямоугольник с вершинами в точках (8,8) и (64,128). Заменив константы в первых четырех командах, его размеры и местоположение легко изменить.

```
; Задать параметры
        MOV #10,R1      ; 012701    10      X=X1
        MOV #10,R2      ; 012702    10      Y=Y1
        MOV #100,R4     ; 012704    100     DX
        MOV #200,R5     ; 012705    200     DY

; Вычислить координаты вершин и поместить в стек
        MOV R2,-(SP)    ; 010246
        ADD R5,(SP)     ; 060516            Y2=Y1+DY
        MOV R1,-(SP)    ; 010146
        ADD R4,(SP)     ; 060416            X2=X1+DX
        MOV R2,-(SP)    ; 010246            Y1
        MOV R1,-(SP)    ; 010146            X1
        MOV SP,R3       ; 010603            адрес таблицы в R3

; Вычислить количество точек и поместить в R4
        INC R4          ; 005204            (на отрезке K единиц
        INC R5          ; 005205            находится K+1 точка)
        MOV R4,R0       ; 010400
        CLR R4          ; 005004
L1:     ADD R5,R4       ; 060504            R5*R4 => R4
        SOB R0,L1       ; 077002

; Определить адрес входа в соподпрограмму NEXTXY и записать в стек
        MOV PC,R0       ; 010700            адрес метки M2
        ADD #30,R0      ; 062700    30      (вход в соп/п NEXTXY)
        MOV R0,-(SP)    ; 010046            в стек

; Цикл вывода точек на экран
L2:     MOV #1,R0       ; 012700    1       зажечь точку экрана с
        EMT 30          ; 104030            координатами в R1 и R2
        JSR PC,@(SP)+   ; 004736            вызов соп/п NEXTXY
        SOB R4,L2       ; 077405
        ADD #12,SP      ; 062706    12      восстановить SP
        HALT            ; 000000            останов

;       Соподпрограмма NEXTXY
; Выдает координаты следующей точки на спирали

M1:     JSR PC,@(SP)+   ; 004736            вызов соп/п осн. цикла
                                            первый вход
M2:     INC R1          ; 005201            X=X+1
        CMP R1,4(R3)    ; 020163    4       сравнить X и X2
        BNE M1          ; 001373            цикл до X2

M3:     JSR PC,@(SP)+   ; 004736
M4:     INC R2          ; 005202            движение от Y1 к Y2
        CMP R2,6(R3)    ; 020263    6
```

```
        BNE M3          ; 001373

M5:     JSR PC,@(SP)+   ; 004736
        DEC R1          ; 005301        движение от X2 к X1
        CMP R1,(R3)     ; 020113
        BNE M5          ; 001374

M6:     JSR PC,@(SP)+   ; 004736
        DEC R2          ; 005302        движение от Y2 к Y1
        CMP R2,2(R3)    ; 020263    2
        BNE M6          ; 001373

        MOV R3,R0       ; 010300        коррекция вершин
        INC (R0)+       ; 005220
        INC (R0)+       ; 005220        прямоугольника
        DEC (R0)+       ; 005320
        DEC (R0)+       ; 005320        и переход к левой
        INC R1          ; 005201        вершине следующего
        INC R2          ; 005202        прямоугольника
        CMP (R3),4(R3)  ; 021363    4   X1=X2: осталась
        BEQ M4          ; 001751        незакрашенная
        BR M1           ; 000742        вертикальная линия
; ---------------------------------------------------------------
```

Литература

1. *Гиматов Ю.* Самомодифицирующиеся программы // Информатика и образование. 1992. № 2. С. 91—94.

2. *Лин В.* PDP-11 и VAX-11. Архитектура ЭВМ и протраммирование на языке ассемблера. М.: Радио и связь, 1989. С. 101—102.

3. *Зальцман Ю.* Архитектура и ассемблер БК // Информатика и образование. 1991. № 4. С. 53—58.

Примечание редактора

Предложенная Е. А. Ереминым программа хорошо структурирована и легко читаема. Следует лишь отметить, что в ней не предусмотрена предварительная очистка экрана и установка режима 32 или 64 символа в строке. При желании можно дополнить программу необходимыми операторами, разместив их в начале листинга (в режиме 32 символа в строке можно реализовать цветную закраску).

Что же касается самой идеи использования соподпрограмм, она, похоже, настолько же привлекательна, насколько непривычна. Такой способ на практике используется, пожалуй, еще реже, чем самомодификация. Но, как и в большинстве подобных случаев, неизвестное явно таит в себе немало интересного. Так что данный материал заслуживает самого пристального внимания со стороны программистов БК-0010.

Не волнуйся и не спеши — дней в году много ...

Старинная восточная пословица

Р. А. Рахманкулов,
г. Ташкент

ИГРА «КАЛАХ»

Калах (известный также под другими названиями: манкала, вари, овари) — старинная африканская игра. По ее теории написано очень мало, хотя в течение столетий в калах играют представители самых разных культур. Эта игра — чистая борьба умов, не содержащая какого-либо случайного элемента. Благодаря нехитрому инвентарю и простым правилам калах великолепно подходит для компьютерной реализации.

Правила игры следующие. Каждому игроку принадлежит шесть малых лунок вдоль длинной стороны поля и одна лунка большего размера по правую руку, называемая КАЛАХОМ. В начале игры в каждую малую лунку помещается некоторое количество камней (у африканцев обычно шесть). Ход игрока заключается в том, что он забирает все камешки в одной из малых лунок на своей стороне и раскладывает их по одному в остальные лунки, двигаясь против часовой стрелки. Первый камень кладется в лунку справа от той, из которой взяты камни, затем в следующие, включая свой калах и малые лунки противника (но не в калах противника!). Может случиться (и это допускается правилами), что, раскладывая камни, мы обойдем всю доску и вернемся в исходную лунку или даже пойдем дальше.

Есть два дополнения к правилу выполнения хода:

ПОВТОР. Если последний камень попал в одну из непустых малых лунок игрока, делавшего ход, причем камни клались и в лунки противника, то делается повторный ход из лунки, в которую попал последний камень. При известном умении (и везении) игрок может сделать сколь угодно длинную серию повторных ходов.

ВЗЯТИЕ. Если последний камень попал в одну из малых лунок противника и в этой лунке стало два или три камня, то эти камни «берутся в плен» и помещаются в калах игрока, сделавшего ход. Если при пленении в предыдущей лунке также оказалось два или три камня, то и они «попадают в плен». Теоретически игрок может за один ход полностью очистить сторону противника.

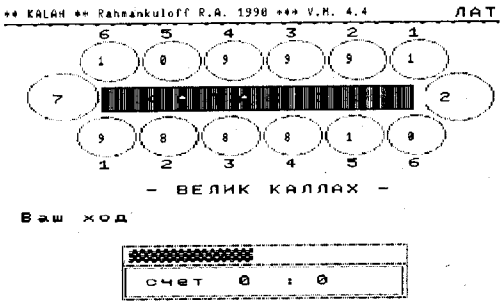
В данном компьютерном варианте игры время для обдумывания хода ограничено. Если вы не уложились в отведенные секунды, компьютер сделает ход за вас. (При желании можно «взять тайм-аут» для

обдумывания хода, нажав СУ+@. Продолжение игры — по нажатию любой клавиши. — *Прим. ред.*)

Если у игрока, получившего очередь хода, не осталось ни одного камня в малых лунках, то игра прекращается и все камни противника попадают в калах последнего. Выигрывает тот, у кого в калахе оказалось больше камней.

(**Примечание.** В нашей стране получила распространение игра «калах» с правилами, сильно отличающимися от описанных выше. Здесь же приводится листинг программы для «африканского» варианта.)

```
10 CLEAR 400
20 UK%=1%
30 CLS
40 DIM L%(14%), I1%(70%)
50 B$(0%)="Ну-ну"
60 B$(1%)="Сейчас я Вам задам"
70 B$(2%)= "Хе-хе"
80 B$(3%)="Сдавайтесь"
90 B$(4%)="Больше так не ходите"
100 B$(5%)="Ой-ой-ой,как испугали"
110 B$(6%)="Сейчас Вам станет нехорошо"
120 B$(7%)="Тоже мне Каспаров"
130 B$(8%)="Покой Вам только снится"
140 B$(9%)="Вы меня утомляете"
150 C$=" ** KALAH ** Rahmankuloff R.A. 1990
    *** V.M. 4.4 "
160 GOSUB 2750
170 CLS
180 ? CHR$(140%);CHR$(140%)
190 IF PEEK(&O40)=0%TH?CHR$(155%)
200 ? AT(5%,5%)"Звук уменьшить (Y/N)";
220 INPUT ZV$
230 GOSUB 3070
240 ? AT(5%,9%)"Уровень БК (1-5) ";
260 INPUT LE%
270 IF LE%<1%ORLE%>5%GOTO140
280 GOSUB 3070
290 ? AT(5%,13%)"Ваш уровень (1-5) ";
300 INPUT LW%
310 IF LW%<1%ORLW%>5%GOTO290
320 ? AT(5%,17%)"Вы ходите первым (Y/N)";
340 INPUT A$
```




```
350 GOSUB 3140
360 IF A$="Y" THEN GOTO 400
370 GOSUB 560
375 GOSUB 770
380 GOSUB 1420
390 GOTO 490
400 GOSUB 560
410 GOSUB 770
420 GOSUB 890
430 IF XL%=1% GOTO 1910
440 GOSUB 1200
450 GOSUB 770
460 GOSUB 1830
470 IF S2%=0%OR S1%=0% THEN GOTO 140
480 GOSUB 1420
490 GOSUB 1540
500 GOSUB 770
510 GOSUB 1830
520 IF S2%=0%ORS1%=0% THEN GOTO 140
530 GOTO 410
540 GOTO 400
550 GOTO 400
560 FOR I%=1% TO 6%
570 L%(I%)=6%
580 L%(I%+7%)=6%
590 NEXT
600 L%(7%)=0%
610 L%(14%)=0%
620 FOR K%=1% TO 6%
630 CIRCLE (22%+32%*K%,25%),15%,3%
640 CIRCLE (20%+32%*K%,83%),15%,2%
650 NEXT
660 CIRCLE (30%,52%),18%,3%
670 CIRCLE (234%,54%),18%,2%
680 LINE (50%,45%)-(210%,63%),1%,B
690 ? AT(6%,10%)"1  2  3  4  5  6";
700 ? AT(6%,0%)"6  5  4  3  2  1";
710 ? AT(8%,5%)" K A Л A X ";LE%;
720 ? AT(9%,12%)"- ВЕЛИК КАЛЛАХ -";
730 PAINT (65%,46%),2%,1%
740 LINE (62%,184%)-(205%,206%),3%,B
750 LINE (60%,182%)-(207%,208%),2%,B
755 LINE (60%,167%)-(207%,182%),2%,B
760 RETURN
770 IF P%=1% THEN RET
780 ? CHR$(155%)
```

```
790 FOR T%=1%TO6%
800 ? AT(3%+8%*T%,8%)STR$(L%(T%));
810 ? AT(3%+8%*T%,2%)STR$(L%(14%-T%));
820 IF ZV$="N" THBEEP
830 NEXT
840 ? CHR$(155%)
850 ? AT(2%,5%)L%(14%);
860 ? AT(27%,5%)L%(7%);
870 ? AT(9%,19%)"счет ";SK;" :";SM;
880 RETURN
890 REM
900 XL%=0%
910 V%=0%
920 A$="Ваш ход"
930 GOSUB 3110
940 GOSUB 3180
950 V%=V%+1%
955? AT(V%/(4%*(6%-LW%))+8%,17%);
    CHR$(191%);
960 IF V%>(6%-LW%)*60% GOTO 1070
970 A$=INKEY$
980 IF A$="" GOTO 940
990 J%=ASC(A$)-48%
1000 IF J%=0% GOTO 1040
1010 IF J%>64% GOTO 890
1020 IF L%(J%)=0% GOTO 890
1025 LINE (0%,230%)-(240%,230%),4%
1030 GOTO 1060
1040 XL%=1%
1045 LINE (0%,230%)-(240%,230%),4%
1050 GOSUB 3070
1060 ? AT(8%,17%)"                    ";
1065 RETURN
1070 A$="Долго думаете! Я пойду за Вас"
1080 GOSUB 3110
1090 FOR J%=1%TO6%
1100 IF L%(J%)<>0% THEN J1%=J%
1110 NEXT
1120 J%=J1%
1130 GOSUB 2720
1140 GOTO 1025
1150 IF P%=1%GOTO 1200
1160 A$="Повтор"
1170 GOSUB 3110
1180 GOSUB 770
1190 GOSUB 2720
1200 C%=L%(J%)
1210 L%(J%)=0%
1220 FOR V%=1%TOC%
1230 J%=J%+1%
1240 IF J%>13%THEN J%=1%
1250 L%(J%)=L%(J%)+1%
1260 NEXT
1270 KL%=0%
1280 PL%=0%
1290 IF J%<7%ANDL%(J%)>1%ANDC%>7%
     THEN KL%=1%
1300 IF J%<14% ANDJ%>7% AND (L%(J%)=2%
     OR L%(J%)=3%) THEN PL%=1%
1310 IF KL%=1% GOTO 1150
1320 IF PL%=0% THEN RETURN
1330 L%(7%)=L%(7%)+L%(J%)
1340 IF P%=1% THEN GOTO 1390
1350 A$="Вы меня обижаете!"
1360 GOSUB 3110
1370 GOSUB 2720
1380 GOSUB 3070
1390 L%(J%)=0%
1400 J%=J%-1%
1410 GOTO 1270
1420 P%=1%
1430 Z%=RND(1)*30%
1440 IF Z%=<9%THA$=B$(Z%)ELA$="Компьютер
     думает"
1450 GOSUB 3110
1460 GOSUB 2170
1470 P%=0%
1480 RETURN
1490 IF P%=1% THEN GOTO 1540
1500 A$="Повтор"
1510 GOSUB 770
1520 GOSUB 3110
1530 GOSUB 2720
1540 C%=L%(J%)
1550 IF P%=1% THEN GOTO 1600
1560 A$="Я хожу из лунки "
1570 GOSUB 3110
1580 ? AT(LEN(A$),14%)J%-7%
1590 GOSUB 2720
1600 L%(J%)=0%
1610 FOR V%=1%TO C%
1620 J%=J%+1%
1630 IF J%=15% THEN J%=1%
1640 IF J%=7% THEN J%=8%
1650 L%(J%)=L%(J%)+1%
1660 NEXT
1670 KL%=0%
1680 PL%=0%
1690 IF J%>7%ANDJ%<14%ANDC%>7% THEN
     KL%=1%
1700 IF J%<7%AND(L%(J%)=2% OR
     L%(J%)=3%) THEN PL%=1%
1710 IF KL%=1% GOTO 1490
1720 IF PL%=0% THEN RETURN
1730 IF P%=1% THEN GOTO 1780
1740 A$="Люблю кушать Ваши камешки"
1750 GOSUB 3110
1760 GOSUB 770
1770 GOSUB 2720
1780 L%(14%)=L%(14%)+L%(J%)
```

```
1790 L%(J%)=0%
1800 J%=J%-1%
1810 GOTO 1670
1820 REM
1830 S2%=0%
1840 S1%=0%
1850 FOR I%=1%TO6%
1860 S2%=S2%+L%(I%)
1870 S1%=S1%+L%(I%+7%)
1880 NEXT
1890 IF S2%=0%OR S1%=0% THEN GOTO 1910
1900 RETURN
1910 FOR I%=1%TO6%
1920 L%(14%)=L%(14%)+L%(I%+7%)
1930 L%(7%)=L%(7%)+L%(I%)
1940 NEXT
1950 IF L%(7%)>L%(14%) GOTO 1980
1960 IF L%(7%)<L%(14%) GOTO 2040
1970 IF L%(7%)=L%(14%) GOTO 2100
1980 A$="Поздравляю! Вы выиграли"
1990 GOSUB 3110
2000 GOSUB 2720
2010 SM=SM+1%
2020 GOSUB 3140
2030 GOTO 170
2040 A$="БК выиграл ! С Вас 3 рубля"
2050 GOSUB 3110
2060 GOSUB 2720
2070 GOSUB 3140
2080 SK=SK+1%
2090 GOTO 170
2100 A$="Ничья. Редкий случай"
2110 GOSUB 3110
2120 GOSUB 2720
2130 GOSUB 3140
2140 SM=SM+.5
2150 SK=SK+.5
2160 GOTO 170
2170 S%=0%
2180 S%=S%+1%
2190 IF S%<=3%THGOSUB 3180
2200 GOSUB 2620
2210 O%(S%)=1000%*(-1%)^S%
2220 SS%(S%)=S%MOD2%
2230 IX%(S%)=1%+7%*SS%(S%)
2240 I%(S%)=0%
2250 I%(S%)=I%(S%)+1%
2260 GOSUB 2670
2270 IF I%(S%)>6% GOTO 2440
2280 J%=I%(S%)+7%*SS%(S%)
2290 SX%(S%)=J%
2300 IF L%(J%)<>0% GOTO 2320
2310 GOTO 2250
2320 GOSUB 2530
2330 IF S%>=LE% THEN GOTO 2510
2340 IF S%<LE% GOTO 2180
2350 IF SS%(S%)=1% THEN 2360 ELSE 2400
2360 IF O%(S%)>O%(S%+1%) GOTO 2250
2370 O%(S%)=O%(S%+1%)
2380 IX%(S%)=SX%(S%)
2390 GOTO 2250
2400 IF O%(S%)=O%(S%+1%) GOTO 2250
2410 O%(S%)=O%(S%+1%)
2420 IX%(S%)=SX%(S%)
2430 GOTO 2250
2440 IF O%(S%)=1000%*(-1%)^S% THEN
       O%(S%)=-O%(S%)
2450 GOSUB 2670
2460 S%=S%-1%
2470 IF S%=0% THEN GOTO 2490
2480 GOTO 2350
2490 J%=IX%(1%)
2500 RETURN
2510 O%(S%+1%)=10%*L%(14%)+L%(13%)+L%
       (12%)+L%(11%)+L%(10%)+L%(9%)+L%
       (8%)-10%*L%(7%)-L%(6%)-L%(5%)-
       L%(4%)-L%(3%)-L%(2%)-L%(1%)
2520 GOTO 2350
2530 IF J%<7% THEN GOSUB 1200
2540 IF J%>7% THEN GOSUB 1540
2550 RETURN
2560 IF S%=1% THEN RETURN
2570 FOR K%=1% TO S%-1%
2580 J%=SX%(K%)
2590 GOSUB 2530
2600 NEXT
2610 RETURN
2620 T%=14%*(S%-1%)
2630 FOR K%=1% TO 14%
2640 L1%(K%+T%)=L%(K%)
2650 NEXT
2660 RETURN
2670 T%=14%*(S%-1%)
2680 FOR K%=1% TO 14%
2690 L%(K%)=L1%(K%+T%)
2700 NEXT
2710 RETURN
2720 FOR V%=1%TO10000%
2730 NEXT
2740 RETURN
2750 IF PEEK(&O40)<>0% THEN ?CHR$(155%)
2760 ? AT(12%,23%)"ОТЛИЧИЯ ОТ
       РАСПРОСТРАНЕННОГО
       КАЛАХА",CHR$(10%)
2770 ? AT(5%,23%)" ПОВТОР — производится в
       том случае,когда Вы"
2780 ? AT(15%,23%)"при своем ходе вернулись
       в свою лунку"
```

```
2790 ? AT(15%,23%)"и в ней еще есть камешки,
     и при этом"
2800 ? AT(15%,23%)"Вы клали камешки и в лунки
     противника."
2810 ? AT(15%,23%)"Повтор при этом производится
     из этой лунки."CHR$(10%)
2820 ? AT(5%,23%)" ВЗЯТИЕ — камешков противника
     производится в том"
2830 ? AT(15%,23%)"случае,если Ваш последний
     камень попал"
2840 ? AT(15%,23%)"в лунку противника и в ней
     оказалось 2 или 3"
2850 ? AT(15%,23%)"камешка. Если в предыдущей
     лунке тоже 2 или"
2860 ? AT(15%,23%)"3 камешка,то они также берутся
     и т. д."CHR$(10%)
2870 ? AT(15%,23%)" ВАШИ ЛУНКИ - НИЖНИЙ
     РЯД. ВАШ КАЛАХ - СПРАВА"CHR$(10%)
2880 ? AT(7%,23%)"Подробности см. Уэзерелл
     'Этюды для программистов'"CHR$(10%)
2900 LINE (2%,2%)-(510%,235%),2%,B
2990 A$=INKEY$
3000 IF A$="" GOTO 2990
3010 GOSUB 3140
3020 RETURN
3030 FOR I%=1% TO LEN(A$)
3040 ? AT(0%,14%)MID$(A$,LEN(A$)-I%+1%,1%);
     AT(0%,14%)CHR$(23%);
     AT(LEN(A$),14%);
3050 NEXT
3060 RETURN
3070 FOR I%=1 TO 31%
3080 ? AT(0%,14%)CHR$(23%)
3090 NEXT
3100 RETURN
3110 GOSUB 3070
3120 GOSUB 3030
3130 RETURN
3140 FOR I%=1% TO 24%
3150 ? AT(0%,0%);CHR$(20%)
3160 NEXT
3170 RETURN
3180 UK%=UK%+1%
3190 ? CHR$(155%)
3200 IF UK%=50%THUK%=1%
3210  D$=MID$(C$,UK%,50%-UK%)+MID$
     (C$,1%,UK%-1%)
3220 POKE &O160,&O40000
3230 ? D$
3240 ? CHR$(155%)
3250 LOCATE 0%,0%
3260  RET
```

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 4000 руб., почтовые расходы (по пересылке бандероли из Москвы в пункт назначения) и 300 руб. (орграсходы);
- предприятиям и организациям необходимо перечислить на расчетный счет редакции за каждый выпуск 7300 руб. (включая почтовые расходы 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», кор. 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».

Авторские программы для БК-0010(.01) и БК-0011(М), в том числе новую дисковую версию мультипликационного редактора спрайтов Animatic V2.1 со значительно увеличенным объемом памяти под спрайты, а также обновленные подборки «В помощь изучающим БЕЙСИК» и «В помощь начинающему системному программисту» с текстами программ соответственно на БЕЙСИКе и ассемблере с подробными комментариями на кассетах или дискетах Вы можете приобрести по адресу:

127349, г. Москва, а/я 9, Юров Вячеслав Петрович.

Условия поставок можно уточнить по телефону:

(095) 908-22-12 с 10 до 21 часа ежедневно.

По указанному адресу можно также приобрести программы для компьютеров ZX-Sinclair Spectrum и совместимых с ним, Atari XE/XL, Atari ST, Commodore 64/128, AMIGA 500, 600, 1200 и IBM PC и совместимых с ним.

При заказе необходимо указать тип и конфигурацию (комплектность) компьютера.

ЗАЯВКА

на журнал «Персональный компьютер БК-0010 — БК-0011М»

_____

(адрес подписчика с указанием почтового индекса)

_____

(фамилия, имя, отчество полностью)

_____

(номер выпуска и год издания)

_____

(общее количество экземпляров)

Перечислено на расчетный счет _____

_____ руб.

(общее количество экземпляров, стоимость одного экземпляра)

_____

Платежное поручение № _____ от _____ 199 ___ г.

Говоря о творчестве любителей БК, нельзя обойти вниманием и так называемые «компьютерные газеты» — текстовые файлы, объединенные с автономным «вьювером» для их просмотра. Конечно, не все они заслуживают внимания, но среди них попадаются и довольно интересные вещи — советы для начинающих, полезные программные фрагменты и т. д.

В одной из подобных «газет» нами и было обнаружено публикуемая ниже «классификация БКманов», которая, как мы надеемся, подарит нашим читателям несколько веселых минут. Автора же этого опуса (его координаты, к сожалению, не были указаны в «газете»), как и всех других, кто хотел бы поделиться своим опытом с пользователями БК, редакция приглашает к сотрудничеству.

И. Юдин

ВИДЫ БКманов
(ПОЧТИ НАУЧНАЯ КЛАССИФИКАЦИЯ)

Для начала оговоримся, что дальнейшее не будет касаться начинающих пользователей, еще только осваивающих компьютер и в силу своего исключительного разнообразия не поддающихся классификации. Поэтому речь пойдет о БКманах более искушенных, для кого БК — не только игрушка, но и главный рабочий инструмент.

Программисты по роду занятий имеют статусы: CRACKER, DIGGER, ISWRITER, HACKER, GAMEWRITER, SYSWRITER, GLUCKSEEKER, PROTECTER. Существуют также люди, не разрабатывающие программное обеспечение, но являющиеся его распространителями: ДОСТАВАЛЫ и ДИСТРИБЬЮТЕРЫ. (Конечно, можно было перевести эти два слова на английский, но тогда их было труднее читать и понять смысл.)

CRACKER. Вы спросите: откуда взялись надписи типа «Cracked by ...»? Вы еще спрашиваете!? Кто не знает выражение, обычно пишущееся на стенах и заборах: «Здесь был Вася»? Так вот, «Cracked by....» — это тоже самое, только переведенное на иностранный язык.

CRACKERы живут взламыванием защит. Это их призвание. CRACKERов очень не любят авторы программ, каждый из которых сам CRACKER (каждый уважающий себя программист должен быть немножко CRACKERом). Истинных CRACKERов очень мало. Их можно пересчитать по пальцам. Их знают в лицо. В крайнем случае, где-нибудь в записной книжке на особой странице, но не очень далеко, записаны их телефоны. На всякий случай...

DIGGER. В дословном переводе — «копатель». DIGGER копается в чужих программах для того, чтобы найти там что-нибудь интересное: адрес запуска, чтобы программа сразу показала призовую картинку или проиграла музон, ячейку, в которой хранятся жизни, расстояние до цели путешествия, номер начального лабиринта и др.

ISWRITER. ISWRITER (не путать с извращенцами!) — DIGGER, зашедший немного дальше, чем ему следовало бы. ISWRITERы обычно проявляются тем, что меняют в программе имя и телефон автора или номер версии программы без разрешения ее создателя.

Рано или поздно, став ISWRITERом, человек осознает, что его стремление прославить свое имя таким способом ни к чему хорошему не приведет, или же, наоборот, его желание прославиться в школе (по большей части ISWRITERы являются школьниками) достигает нужного эффекта, после чего это занятие перестает ISWRITERа интересовать и он перестает им быть. Отсюда следует, что ISWRITER — переходная стадия.

HACKER. Так же как и ISWRITER, HACKER является временной фазой длительного цикла эволюции программиста. HACKERом (как, впрочем, и ISWRITERом) человек бывает один раз в жизни. Устойчивых HACKERов в природе не существует и нельзя стать HACKERом, не пройдя фазу DIGGERа.

В чем же состоит отличие HACKERa от DIGGERa? У HACKERов есть одна особая черта: человека можно считать HACKERом, если, проведя один день без компьютера, он не находит себе места и стремится как можно скорее сесть за клавиатуру.

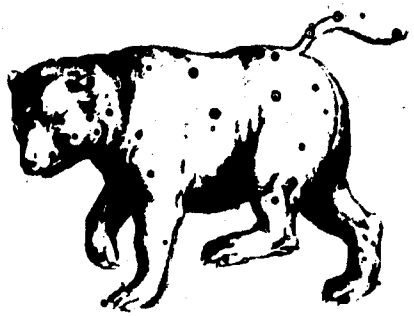
GAMEWRITER и SYSWRITER. Как это видно из названий, GAMEWRITERы пишут игры, а SYSWRITERы — системные и прикладные программы. Среди SYSWRITERов есть индивиды, которые пишут исключительно системные программы. Это — Системные Программисты, очень ценные люди.

GLUCKSEEKER (буквально: глюкоискатель). Человек, ищущий ошибки в чужих программах. GLUCKSEEKERов не так много, как может показаться. Если вы написали программу — найдите одного из GLUCKSEEKERов и отдайте ему эту программу, чтобы потом вам не звонили разъяренные пользователи. GLUCKSEEKERы позволяют сэкономить время и нервы.

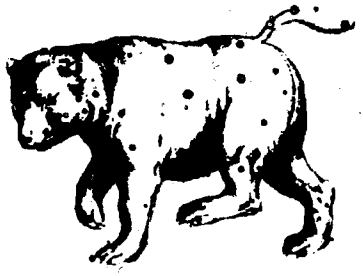
PROTECTER. PROTECTER — человек, ставящий защиты, которые, в свою очередь, нужны CRACKERам как воздух. Ставить защиты легче, чем снимать, но чтобы сделать действительно хорошую защиту, нужно сначала все тщательно обдумать. Используя некоторые неочевидные команды, PROTECTERы устанавливают защиты, красивые как снаружи, так и изнутри.

ДОСТАВАЛЫ. Это менялы ПО. Обычно они достают новые программы так: берут лучшее из того, что у них есть, и идут куда-нибудь, где меняют все это на более новые программы. ДОСТАВАЛЫ имеют огромные связи, по которым все ПО, которое бродит в окрестностях, в конечном счете приходит к ним.

ДИСТРИБЬЮТЕРЫ. Это поставщики дистрибутивов. Они тоже имеют связи, но более ценные, чем у ДОСТАВАЛ. Они лично знают разработчиков программ (системных и игровых), у которых берут самые последние версии и по некоторой, в основном устной, договоренности распространяют их.



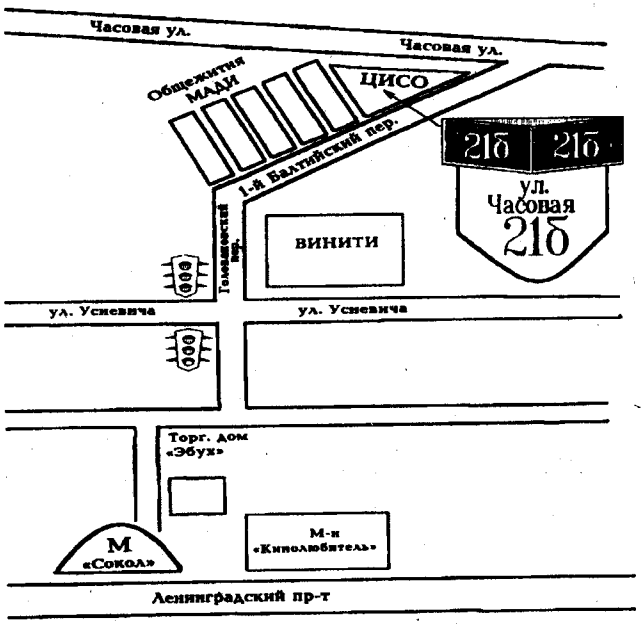
СОДЕРЖАНИЕ



ПЕРСОНАЛЬНЫЙ КОМПЬЮТЕР БК-0010 – БК-0011М

Наш адрес: Москва, ул. Часовая, 21Б, помещение Центра Интерактивных Средств Обучения (ЦИСО), комн.20
Телефон: (095) 151-19-40

Как к нам добраться:



Адрес для переписки: 103051, Москва, ул. Садовая Сухаревская, д. 16, комн. 9.

ПЕРСОНАЛЬНЫЙ КОМПЬЮТЕР БК-0010 - БК-0011М

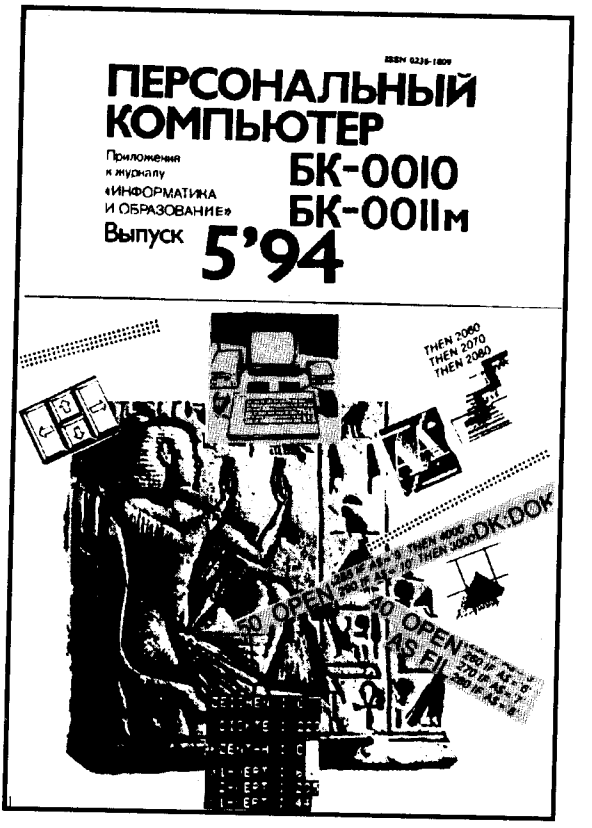
Главный редактор
Васильев Б. М.

Редактор
Усенков Д. Ю.

Корректор
Антонова В. С.

Компьютерная верстка
Панченко О. Н.

Подписано в печать с оригинал-макета издательства «Информатика и образование» 24.11.94. Формат 70×100 1/16. Бумага офсетная. Усл.печ.л. 6,5. Заказ №3636. Цена 4000 руб. (по подписке). В розничной продаже цена договорная.
Ордена Трудового Красного Знамени Чеховский полиграфический комбинат Комитета Российской Федерации по печати.
142300, Чехов Московской обл.



Планируемое содержание выпуска 5'94:

- Архитектура и программирование на языке ассемблера *(продолжение)*
- Описание 326-й прошивки ПЗУ КНГМД
- Использование арифметики чисел с плавающей запятой в ассемблерных программах
- Справочный листок
- БК плюс магнитофон: *доработки конструкции, контроль и повышение надежности работы*
- Дополнительное ОЗУ/ПЗУ
- База данных для БК-0011(М)
- «ЕСТЬ ИДЕЯ»: гезотайп — сверхбыстрая клавиатура
- Обмен опытом
- «ГЛЮКАДЕМИЯ»: «Тетрис» и клавиатура БК-0010.01
- Игра «Реверси»
- Содержание предыдущих выпусков