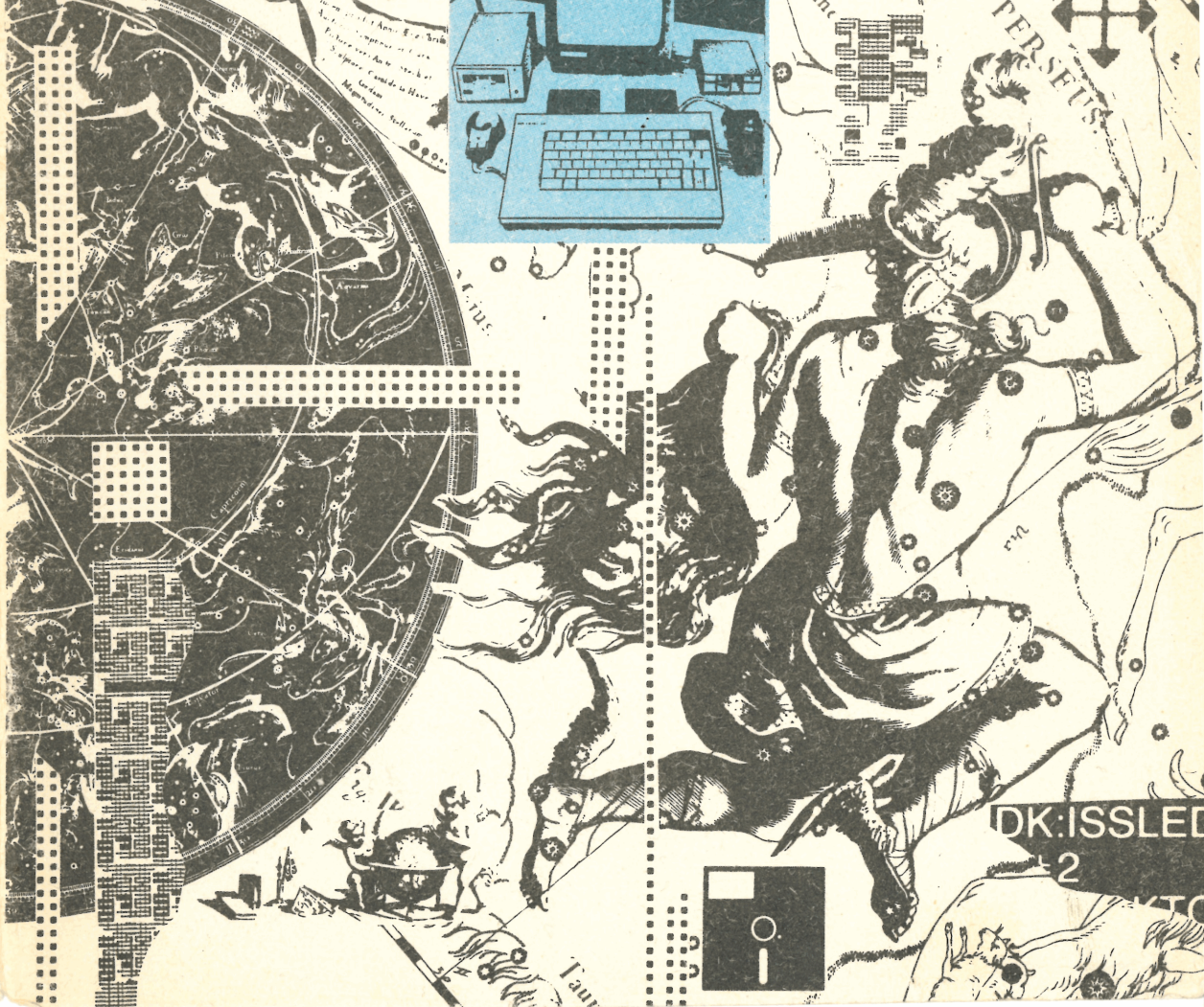
ISSN 0236-1809

ПЕРСОНАЛЬНЫЙ КОМПЬЮТЕР БК-0010, БК-0011м

Приложение
к журналу
«ИНФОРМАТИКА
И ОБРАЗОВАНИЕ»

Выпуск 1'94



УВАЖАЕМЫЕ ЧИТАТЕЛИ!

По вашему запросу мы готовы предоставить на дискетах (3,5" или 5,25"), через электронную почту или в виде распечаток на матричном или лазерном принтере любые статьи из опубликованных в журналах «Информатика и образование», «Персональный компьютер БК-0010 — БК-0011М», «Персональный компьютер УКНЦ». В редакции вы можете приобрести отдельные выпуски этих журналов, а также программное и аппаратное обеспечение для БК и УКНЦ.

БК:

Дисковая операционная система NORD.
Профессиональный просмотрщик (вьювер) копий экрана БК-0010 для переноса графики на IBM.
Конвертор листингов на вильнюсском Бейсике в текстовые файлы формата EDASP и обратно.
Резидентный драйвер ANIRAM-MIRIADA, дополняющий ОС ANDOS следующими возможностями:

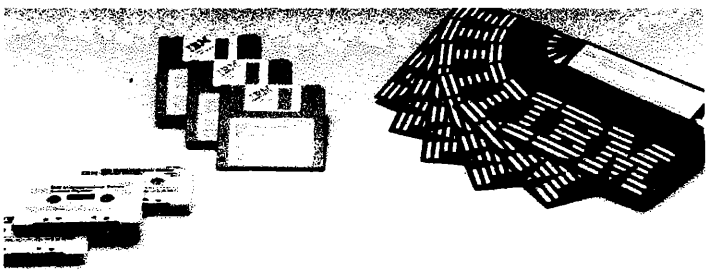
- копирование экрана в файлы;
- сопровождение диалога с БК на принтере;
- поддержка дополнительного устройства «P:» – «принтер».

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

УКНЦ:

Электронный диск на 512 Кб с программным обеспечением.
Универсальный архиватор FCU версии 1.00 с возможностью создания самораспаковывающихся архивов.
Многофункциональная программа FMZ (функции форматировщика и диск-доктора).

ЗАКЛЮЧАЕМ С АВТОРАМИ ДОГОВОРА НА ПРОДАЖУ ПРОГРАММНЫХ И АППАРАТНЫХ РАЗРАБОТОК.



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

ПЕРСОНАЛЬНЫЙ КОМПЬЮТЕР

Приложение
к журналу
«ИНФОРМАТИКА
И ОБРАЗОВАНИЕ»

БК-0010-
БК-0011м

Выпуск 1'94 (2)

Издается с 1993 г.

В НОМЕРЕ

Москва «Информатика и образование» 1994

Авторы выпуска

Аскеров Р.
Ерохин С. Л.
Зальцман А. Ю.
Иванов В. Е.
Константинов В. В.
Копосов А. Н.
Кузнецов А. В.
Лядвинский М. В.
Милюков А. В.
Новиков А. В.
Рахманкулов Р. А.
Рогов В. И.
Страхов А. Ю.
Усенков Д. Ю.
Юров В. В.
Юров В. П.

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

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

Если вы сделали что-то один раз — это случайность, если два — совпадение, если же вам удалось осуществить это трижды, значит, вы открыли закон.

Грейс Хоппер (разработчица первого компилятора)

ЕСТЬ «СОВПАДЕНИЕ» — ДАЕШЬ «ЗАКОН»!

После подписания этого выпуска в печать, победных реляций и взаимных рукотрясений следует оценить потери. Увы, инфляция всеядна, и наш очередной выпуск вырос по формату (в дальнейшем формат изменяться не будет), но потерял в объеме (в дальнейшем...).

Дальнейшее пока не просматривается и не прогнозируется. Одно стабильно — рост себестоимости издания, отчего настроение в редакции, по Кирсанову, унылое:

Хоть бы эту зиму выжить,
Пережить хотя бы год,
Под наркозом, что ли, выждать
Свист и вой непогод...

и, по Кирсанову же, боевое:

Знаем — скажут потомки в будущем:
«Эти жили совсем не буднично!»
Глянут в книжицы позабытые —
Нас увидят и позавидуют!

Еще и читатели приложения дарят нам надежду: писем много, письма хорошие. Читатели не дают нам проспать экономическую зиму, не позволяют окунуться в наркотический кайф легких заработков водочно-сигаретной коммерции. Следовательно, будем работать.

Нынешняя ситуация тяжела, но не безнадежна, попробуем выбраться из нее с помощью изрядной доли безрассудства: на второе полугодие 1994 г. запланировали увеличить количество выпусков приложения по БК.

С содержанием следующего выпуска вы познакомитесь на третьей странице обложки, там же вы узнаете об «очень смешных ценах» наших изданий, объявленных на второе полугодие 1994.

Обратим внимание на новое оформление обложки выпуска. Надеемся, что этот «костюмчик» нам к лицу. Отметим, что новый выпуск содержит много статей, посвященных рассмотрению различных сторон графической темы.

Начата публикация большого материала по программированию на языке ассемблера.

Традиционно сохраняем рубрику «Обмен опытом» и в заключение — «Потехе час».

А. Ю. Страхов,
г. Москва

ШАГ НАЗАД И ... ПРЫЖОК В БУДУЩЕЕ

Вероятнее всего, даже в нашей, не самой компьютеризированной державе едва ли найдется человек, никогда не слышавший слова «компьютер». При этом далеко не все хорошо представляют себе, что это такое. Однако это не мешает хотеть (что, как известно, не вредно) приобщиться к использованию современных информационных технологий. А для этого надо учиться. И учить.

На сегодняшний день взгляды большинства устремлены в прекрасное далеко: IBM, «Macintosh», «VAX» и т.д. Вести сегодня речь о БК — для них шаг назад, а не в будущее. И все же...

Замечательная программа компьютеризации учебных заведений по большому счету не подкреплена ничем, кроме с помпой преподнесенных «Пилотных школ». Последнее предусматривает поставки классов стоимостью «всего лишь» полтора—два десятка тысяч долларов на базе машин фирмы IBM типа PS/2. Правда, была альтернатива в виде зверинца из всевозможных «Агатов», «Корветов», УКНЦ, заморских «Ямах», эрзац-«Синклеров» всех мастей и, конечно же, БК. Нет необходимости говорить, что самыми распространенными стали последние. Причин этому более чем достаточно.

Во-первых, это самый первый отечественный компьютер данного класса, появившийся уже в 1984 г.

Во-вторых, до недавнего времени самый доступный (а в 1984 г., видимо, и единственный в мире) 16-разрядный бытовой компьютер.

В-третьих, самый массовый: «родили» в Павлово—Посаде, а затем наладили выпуск на семи заводах — от Казани до Минска и от Шяуляя до Еревана.

И наконец, в-четвертых, самый «персональный», потому что продавался в магазине и, что еще более существенно, стоил 650 рублей (не пугайтесь — это было в 1990 г.). Тогда это равнялось приблизительно двум-трем средним зарплатам или цене цветного телевизора. На сегодня эти пропорции выглядят еще более привлекательно.

Это напрямую повлияло и на ситуацию с программными средствами. Сегодня их насчитываются тысячи: системных, прикладных, учебных и, конечно же, игровых. Причем представлены все основные классы программ, а качество многих не уступает аналогам, написанным для более мощных машин. Это и многофункциональные текстовые процессоры, и системы управления базами данных, и электронные таблицы, и мощные графические пакеты. Встречается даже такая экзотика, как издательские системы, которые действительно позволяют создать на БК полноценный оригинал-макет!

Для организации учебного процесса наиболее удобной является локальная сеть (ЛС). Самой распространенной локальной сетью, используемой в системе образования, стал КУВТ-86 — комплект учебной вычислительной техники, представляющий из себя ЛС с центральной машиной типа ДВК-2(3) в качестве рабочего места преподавателя, оснащенной двумя дисководами емкостью по 720кб, печатающим устройством и двенадцатью машинами БК-0010. Только павлово-посадский завод «Экситон» выпустил их порядка десяти тысяч.

И что же? Восемьдесят процентов этих классов сегодня не работают. Причина этого «редкого» явления весьма тривиальна: неисправна ДВК, а без центральной машины, как известно, «и ни туды и ни сюды...».

В то же время можно с уверенностью утверждать, что надежнее БК сегодня машины нет. Правда, никто не сказал, что нет лучше.

А что же лучше?

Во-первых, необходимо определить, что такое «лучше» и по каким критериям производить оценку.

Самое, пожалуй, печальное, что это не очевидно для подавляющего большинства тех, кто на сегодняшний день занимается практическим решением вопросов в данной области, — вчерашних преподавателей математики, физики, а то и просто учителей, имевших наименьшую нагрузку и порой плохо представляющих себе разницу между «Cray» и «VAX», «Macintosh» и «Sinclair», ДВК и IBM.

Перечень же критериев, по которым необходимо оценивать систему, на мой взгляд, весьма очевиден: ВОЗМОЖНОСТИ—НАДЕЖНОСТЬ—УДОБСТВО—ЦЕНА. Причем именно в таком порядке и именно в комплексе.

Не витая в заоблачных высях полета фантазии и обещаний власть предержащих и опустившись на нашу грешную и не самую богатую землю, придется начинать с конца. Итак...

Цена

Цена среднего персонального компьютера типа IBM PC/AT стандартной конфигурации без печатающего устройства составляет порядка 700—800 долларов, а более мощного с процессором 386 — еще на несколько сотен «зелененьких» выше. О машинах же на процессоре 486 и вовсе говорить не стоит — дорого. Даже слишком. Аналогичный по классу «Macintosh» стоит еще в 1,5—2 раза дороже.

Пересчитав все это на рубли, получаем... Страшно сказать, какие деньги! И это за одно место. А на класс их надо хотя бы 10—12. Даже если взять экзотическую сегодня за рубежом XT (их выпуск практически прекращен), то все равно это 1,5—2 миллиона рублей. Это лишь стоимость «железа».

Теперь о программах, что является отдельной статьей расходов.

Даже если брать в расчет лишь базовое программное обеспечение, то только оно обойдется никак не меньше нескольких сотен долларов. А разработки «специального» назначения, т.е. учебные программы, да еще по всем необходимым предметам, да на русском (или любом другом, кроме английского) языке, да плюс прикладные программы, — это еще три раза по столько. Минимум. На одно рабочее место. А ведь все это еще необходимо объединить в сеть. Надо ли продолжать?

Надежность

О надежности «Агатов», «Корветов» и т.п. говорить можно много и долго. БК из этого ряда явно выбивается — она по меньшей мере на порядок надежнее (в особенности после устранения в 1991 г. основного недостатка — неудобной и ненадежной клавиатуры).

Говорить о зарубежной технике, думается, не стоит — что да, то да. Отечественные аналоги уступают, но все же...

Удобство

Говоря об удобстве, необходимо, на мой взгляд, прежде всего остановиться на внешней памяти, т.е. на используемых носителях информации.

На практике приходится иметь дело фактически лишь с двумя видами внешних запоминающих устройств — НГМД и НЖМД (накопители на гибких и жестких, или, как их еще называют, «винчестерских», магнитных дисках).

Первые характеризуются малыми объемами хранимой информации и небольшой скоростью обмена, а отечественные представители — еще и низкой надежностью. Вторые же обладают практически одними достоинствами: высокой скоростью доступа и обмена, большим объемом хранимой информации, значительной надежностью. Причем если учесть, что большинство выпускаемых сегодня отечественных ПЭВМ комплектуется зарубежными винчестерами, то эти качества в полной мере можно отнести и к ним. Но на ДВК,

БК, «Агатах», «Корветах», впрочем, как и на всех иных видах отечественных машин, объединенных в учебные классы, они отсутствуют. Выводы очевидны.

Возможности

Рассматривать возможности великолепных «Макинтошей» и IBM — Замечательно! Восхитительно! Прекрасно! — однако см. раздел «Цена».

Прежде всего необходимо понимать, что нельзя вести речь о возможностях собственно «железа» — оно-то как раз и ничего («Корвет», «Агат», «Синклер»), а то и вовсе «очень даже» (БК—0010, БК—0011М, ПК 11/16, УК НЦ, «Поиск»), а если учитывать возможности дальнейшего развития, то можно сказать и «о-го-го». Но ведь ВОЗМОЖНОСТИ — это «железо» + ИМЕЮЩЕЕСЯ программное обеспечение!

Уже ни для кого не секрет, что коэффициент использования компьютерной техники в нашей стране смехотворно мал — всего лишь 3—4%. А это означает, что при нашей катастрофической бедности мы позволяем себе порядка 95% затраченных средств попросту заморозить. Это говорит о том, что мы «палим из пушки по воробьям». Учитывая наличие высококачественного ПО для БК, становится очевидным тот факт, что по соотношению ВОЗМОЖНОСТИ—ЦЕНА у БК конкурентов на ближайшее время нет.

Правда, и IBM манит своими прелестями. Тем более что у некоторых школ нашлись богатые «дяди», купившие одну—две заморские диковинки. Вот бы что использовать в качестве центральной машины! Да вот незадача — процессор в БК имеет другую систему команд, со всеми вытекающими отсюда последствиями.

Однако, как известно, во всем мире никого уже не удивляет объединение в сети машин не только различных классов, но и имеющих процессоры с различными системами команд. Речь, естественно, идет об информационной совместимости. Возможно ли такое в отношении отечественной техники?

ДА!

Специалистами ИКЦ «ИНКОМСЕРВИС» осуществлена разработка такой разнородной сети, получившей название «MicroNet X!». Рассмотрим ее состав и работу. На рисунке приведена общая схема.

В качестве центральной машины используется любая IBM-совместимая ЭВМ (в том числе и отечественного производства), а в качестве периферийных — БК-0010.01.

Центральная ЭВМ через стандартный последовательный порт RS-232 (в машинах отечественного производства это СТЫК-2) соединяется с блоком мультиплексора. Он представляет собой коммутатор, последовательно опрашивающий каналы. Как только любая из периферийных машин выставит запрос, происходит активизация канала обмена, который обслуживается резидентной сетевой программой и производится по четырехпроводной линии.

Каждая машина оснащена индикатором, позволяющим различать три текущих состояния сети: «сеть работоспособна и готова к обслуживанию», «сеть занята другим пользователем» и «обмен по сети». Обеспечивается проверка правильности передачи информации с последующей выдачей звукового сигнала.

Следует отметить, что БК используются как интеллектуальные терминалы, т.е. как полноценные ЭВМ, лишь обменивающиеся информацией с выполняющей роль мощного файл-сервера центральной машиной.

Особенностью данного вида ЛС следует назвать то, что она не требует выделенного файл—сервера, так как задача обслуживания сети является фоновой. Это позволяет одновременно работать как на периферийных машинах, так и на централь-

ной, не теряя ее как полноценное рабочее место. При этом скорость работы центральной машины во время осуществления обмена снижается лишь на несколько процентов, что практически незаметно.

В учебном процессе этот фактор дает дополнительные преимущества: позволяет продемонстрировать современные программные продукты и тенденции их развития, а также организовать занятия с наиболее «продвинутыми» учащимися на IBM PC, не прерывая работу остальных. И все это при поразительно надежной работе сети и фантастически низкой цене: десяток рабочих мест, объединенных в локальную сеть, равны по стоимости одной AT-286, а при использовании отечественной XT стоимость полностью укомплектованного класса (центральная ЭВМ + сетевое оборудование + программная поддержка сети с набором утилит + периферийные машины) равна 1400—1500 долларов (в ценах на январь 1994 г.).

На практике имеется возможность быстрого переоборудования имеющихся КУВТ-86 (или просто группы разрозненных машин) и объединения их в локальную сеть «MicroNet X!».

Применение описанной конфигурации сети необычайно эффективно: ведь мы получаем полноценную локальную сеть с общими периферийными устройствами — винчестером, дисководами, принтером. Имея доступ к винчестеру центральной машины, мы можем перемещаться по дереву каталогов файл-сервера, где можно создать или удалить директорию, прочитать или записать файл, запустить исполняемую программу. Подготовив документ на периферийной машине, его можно сразу отправить для распечатки на принтер, подсоединенный к центральной ПЭВМ. В случае необходимости можно переслать служебное сообщение на экран машины преподавателя, а также многое другое.

Для удобства пользователя имеется оболочка, позволяющая перемещаться по дереву каталогов винчестера центральной машины, создавать и удалять директории и отдельные файлы. Причем имеется возможность защиты файлов от удаления с периферийной машины, при этом они остаются доступными для чтения. Для новой версии сети «MicroNet X!» v2.xx программистами ИКЦ «Инкомсервис» написана оболочка, практически полностью повторяющая пользовательский интерфейс и функции наиболее популярной у нас в стране программы «NORTON COMMANDER».

Тактико-технические данные сети «MicroNet X!»:

Топология	«звезда»
Количество объединяемых машин	8—16 и более
Тип центральной машины	IBM PC XT/AT
Тип периферийной машины	БК-0010.01
Коммуникационный порт	RS-232
Соединительная линия	4-проводная
Скорость обмена по линии	9600 бит/с
Операционная система	MS-DOS v3.31 и выше
Задача обслуживания сети	фоновая
Необходимое пространство ОЗУ	24 Кб

Ю. А. Зальцман,
г. Алма-Ата

МИКРО-ЭВМ БК-0010. АРХИТЕКТУРА И ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ АССЕМБЛЕРА

«В начале было слово...» Точнее, не слово, а множество вопросов пользователей БК-0010 к автору данной статьи, на самые различные темы, касающиеся программирования и архитектуры этой микро-ЭВМ. Сначала вопросы по поводу программирования на языке Фокал, затем — на языке Бейсик, а еще позже — по программированию в машинных кодах, или на языке ассемблера. Если число вопросов по Фокалу и Бейсику было относительно невелико и быстро пошло на убыль (это, видимо, объясняется тем, что эти языки достаточно полно описаны в прилагаемой к БК-0010 документации и различных периодических изданиях), то вопросы по программированию в машинных кодах не прекращаются и по сей день.

Введение

Что же представляет собой наш самый массовый компьютер? О БК-0010 писали много. Писали в специальной литературе и в газетах, хорошо и плохо, профессионалы и дилетанты. Среди лиц, избалованных мощной импортной техникой, распространено презрительное отношение к БК, как к «большому калькулятору», «уродцу в семействе ЭВМ» и т.п. Но создается впечатление, что подобные отзывы исходят в основном от людей, либо совсем не знакомых с БК, либо пытавшихся с ней работать, но не имеющих навыков программирования и вследствие этого просто не справившихся с этим «калькулятором». Такие люди привыкли к тому, что ЭВМ все делает за них сама, а на дискетах имеется полный набор первоклассных фирменных программ — графические и текстовые редакторы, трансляторы языков высокого уровня, базы данных, электронные таблицы... Пользователю надо лишь освоить работу в операционной системе, и ему становятся доступны все возможности компьютера. Можно составлять каталог своей личной библиотеки или готовить статью в «Литературную газету» о том, какая это плохая ЭВМ — БК-0010 и какие плохие люди ее делают...

Но прервем разбег фантазии. Все мы знаем, что хороший компьютер лучше, чем плохой. А БК-0010, что это все-таки такое? И так ли уж это плохо?

Уважаемый читатель, давайте чуть отвлечемся от темы. Вам нравятся микро-ЭВМ семейства ДВК? ДВК-1, ДВК-2, ДВК-3, наконец, ДВК-4? Если вам приходилось работать с этой техникой, ответ, скорее всего, будет либо положительный, либо нейтральный — дело вкуса. Но никто, конечно, не скажет, что ДВК-2М, например, — это большой калькулятор. А теперь внимание: БК-0010 — полный аналог ДВК, и не только по системе команд, но и по многим другим параметрам! БК-0010, как и ДВК, изготовлен на базе микропроцессорного комплекта К1801, имеет такой же центральный процессор (К1801ВМ1), практически такое же быстродействие и адресное пространство. Можно ли сказать что БК-0010 — это, с точки зрения пользователя, практически ДВК-1? Не совсем. БК-0010 лучше!

Да, в нем ОЗУ пользователя всего 16 Кбайт. Но зато экранное ОЗУ входит в общее адресное пространство и с экраном можно работать, как с ОЗУ пользователя. Вы удивлены и не понимаете, что это за преимущество? Достаточно отметить, что это позволяет очень легко и просто организовать работу ЭВМ в

графическом режиме. Но мало того, очень многие задачи (начиная с компьютерных игр и кончая «системами распознавания образов») позволяют использовать экранное ОЗУ БК-0010 как часть ОЗУ пользователя и решать задачи прямо на экране, не занимая основную память ЭВМ.

В БК-0010 реализована черно-белая графика довольно высокого разрешения — 512 х 256 точек, что в 2,5 раза выше, чем на столь популярной за рубежом ЭВМ «СИНКЛЕР», и даже чуть выше, чем на первых мониторах (CGA) IBM PC! Заметим также, что на ДВК-1 и ДВК-2 графического режима нет вообще! Кроме того, БК-0010 имеет цветной режим, позволяя выводить на экран цветного монитора четыре основных цвета — красный, зеленый, синий и черный, причем программным путем можно легко увеличить число цветов до 8—16.

БК-0010 не имеет дисковода. Это серьезный минус, но это и плюс! Отсюда его относительно низкая стоимость и высокая надежность. Монитор-драйверная система (МДС, или та же ОС) БК-0010 записана в постоянных запоминающих устройствах (ПЗУ) компьютера и поэтому не требует загрузки извне. Она не «зависает», не дает сбоев, нечувствительна к «вирусам». В состав ОС (МДС) БК-0010 помимо основных программ, обеспечивающих работу ЭВМ, ее связь с внешними устройствами, и т.д., входят два интерпретатора языков высокого уровня — вильнюсский Бейсик (по возможностям близкий к BASIC—MSX микро-ЭВМ «ЯМАХА-2») и Фокал — очень простой и гибкий язык, предназначенный для решения научно-технических задач, к сожалению до сих пор малознакомый программистам. Фокал БК-0010 имеет расширенный набор средств, в том числе графические операторы, и вполне может потягаться с Бейсиком по своим возможностям. Все эти средства доступны сразу после включения ЭВМ! Вам даже не нужно устанавливать в дисковод дискету и обращаться к помощи ОС.

Есть в БК-0010 и так называемая МОНИТОРНАЯ СИСТЕМА ДИАГНОСТИКИ — МСД. Она не только дает возможность проверить работоспособность и исправность ЭВМ, но и содержит средства, позволяющие писать, запускать и отлаживать программы непосредственно в машинных кодах. БК-0010 имеет довольно большой (для бытового компьютера) набор шрифтов и псевдографических символов, может выводить на экран дисплея два формата шрифта — 64 и 32 символа в строке...

Опытный пользователь персональных ЭВМ уже давно, наверное, с усмешкой читает этот панегирик БК-0010. У него готов вопрос: «А прикладное программное обеспечение? Ведь известно, что за рубежом его стоимость обычно в 5—10 раз превышает стоимость "железа ", т.е. самого компьютера! А чем может похвастать БК?» Может, уважаемый оппонент, и очень многим! На сегодня для БК-0010 разработано свыше 6000 прикладных, системных и инструментальных программ. На БК адаптированы языки: Бейсик (около 10 версий), ФОРТ (3—4 основных версии), Си (2 основные версии), Т-язык. Разработан совершенно новый специализированный язык-транслятор ALMIC. На БК имеется не менее 40 версий редакторов текста, столько же, если не больше, графических редакторов, очень удобные и мощные ассемблер-системы. И, конечно, игры. Любые — логические (в том числе Шахматы), динамические, игры-приключения, с графикой и текстовые, с объемными изображениями и звуковыми эффектами, черно-белые и цветные, полиэкранные, многосерийные и всякие другие, какие только можно вообразить. Ну а что касается математических программ (статистическая обработка данных, вычисления по формулам и т.п.), то их такое множество, что автор не решается даже перечислить их основные классы. Многие из них представляют результаты обработки данных и вычислений не только в цифровом виде, но и графически, в том числе в цвете. А как с базами данных? Имеется несколько простых версий, но есть и СУБД, аналогичная dBASE IBM PC. Есть специальные программы сортировки по алфавиту, драйверы для печати на принтерах разных систем, от КОНСУЛ-254 до EPSON и ThinkJet, и электронные таблицы. Есть программы упаковки и вывода изображений и текстов, уплотнения записей на магнитной ленте (МЛ) и даже синтезатор речи с неограниченным словарем русского языка!

Однако тут самое время остановиться, ведь даже простое перечисление имеющихся для БК программных средств займет не одну страницу, а может быть, даже не одну книгу! Пусть-ка лучше автор ответит на вопрос: где эти программные средства взять?

Что же, вопрос резонный. Почти все перечисленные средства написаны не профессиональными программистами, а любителями. Но в большинстве случаев это ничуть не снижает их качество. Ведь что такое профессиональный программист (если, конечно, это действительно программист, а не ремесленник)? Это просто любитель, получающий зарплату за свой труд. Будем смотреть на вещи трезво — в нашей стране почти никто из так называемых профессиональных программистов не имеет специального программистского образования... И почти никто из имеющих такое образование программистом не работает! Эта в высшей степени парадоксальная ситуация, видимо, объясняется тем, что программирование скорее призвание, талант, чем банальная профессия.

Ассортимент разработанных для БК-0010 программ довольно широк, и многие из них сделаны на высоком профессиональном уровне. Почему же, несмотря на это, до сих пор наблюдается острый дефицит программных средств? Дело в том, что в СССР так и не были приняты законы, защищающие авторские права программиста, в связи с чем авторы НЕ ЗАИНТЕРЕСОВАНЫ были в распространении своих программ. (Закон Российской Федерации о правовой охране программ для ЭВМ и баз данных принят лишь 23 сентября 1992 г. Полный текст этого документа опубликован в журнале «Информатика и образование» №1 за 1993 г. — *Прим. ред.)* Нет до сих пор и единой системы распространения программных средств, единых расценок и гарантий оплаты. Многие государственные и кооперативные предприятия, торгующие программным обеспечением, делают это без всякого согласия авторов и не выплачивают им вознаграждения. Понятно, что при этом авторы даже препятствуют распространению программ, а те из них, которые тиражируются, зачастую представляют собой устаревшие версии или имеют ошибки, в том числе и намеренно внесенные. Однако уже сегодня есть целый ряд организаций, тиражирующих программы по договорам с авторами и предлагающих пользователям за весьма умеренную плату лучшие образцы творчества программистов для БК-0010. Пользователю нужно только следить за рекламой в журналах.

Теперь, после этого несколько затянувшегося, но, по-видимому, необходимого вступления, перейдем к вопросу: чего же все-таки не хватает БК-0010? Прежде всего, конечно, памяти. Хотя в настоящее время разработано уже несколько подходов для расширения ОЗУ и ПЗУ БК-0010, вряд ли можно ожидать, что их применение станет массовым, пока промышленность не наладит серийный выпуск таких «расширителей». Кроме малого объема ОЗУ пользователи БК-0010 также жалуются на ее низкое быстродействие. Но прошу минуту терпения! Обе эти жалобы относятся в основном к программированию на языках высокого уровня. Действительно, Фокал — весьма медленный язык, а вильнюсский Бейсик оставляет желать лучшего в распределении памяти и оптимальности написания программ. Однако есть способ, позволяющий как резко поднять скорость исполнения программ, так и в большинстве случаев существенно экономить память микро-ЭВМ. Это переход к программированию в машинных кодах, или, что по существу одно и то же, на языке ассемблера. Кроме общих преимуществ ассемблера на любой ЭВМ, система команд БК соответствует принятой для компьютеров фирмы Digital Equipment Corp. и имеет существенные преимущества перед стандартом для компьютеров фирмы IBM. Достаточно сказать, например, что, недавно реализованная на БК игра BLOCK OUT (объемный ТЕТРИС), практически не уступающая аналогичной игре на IBM PC, занимает менее 16 Кбайт (на IBM PC — около 200 Кбайт!).

Когда можно считать, что пользователь БК-0010 «созрел» для перехода на ассемблер? Прежде всего тогда, когда он ставит перед собой задачу, неразрешимую средствами Бейсика или Фокала. Затем он должен знать основы информатики и программирования и уметь программировать хотя бы на одном из языков, имеющихся на БК-0010 или другой ЭВМ. При невыполнении последнего условия осваивать программирование на языке ассемблера будет очень трудно. Далее, как уже было сказано выше, нужны некоторые способности (но не большие, чем для программирования на Бейсике), а главное — желание и терпение, необходимые для всякого дела, а для программирования на ассемблере в особенности, ибо первые практические результаты будут получены не так уж скоро.

В данной статье автор ставит перед собой задачу, во-первых, дать читателю возмож-

ность досконально изучить архитектуру БК-0010, а во-вторых, познакомить его с основами программирования на самом гибком и мощном из языков любой ЭВМ — языке ассемблера.

По возможности будут пояснены также некоторые общие вопросы информатики и устройства ЭВМ. Ограниченный объем публикации, к сожалению, не позволяет вдаваться в подробности. Приходится также учитывать различный уровень подготовки читателей: одним, например, вопросы схемотехники компьютера совершенно недоступны, другим скучно изложение основ информатики, с которыми они давно знакомы... Поэтому предполагается, что эти общие (или, наоборот, узкоспециальные) вопросы читатель может изучить самостоятельно, с помощью имеющейся обширной литературы, в зависимости от своего желания и возможностей. Автор старался сделать изложение материала как можно более живым и доступным пониманию, что иногда вынужденно приводило к некоторой нестрогости изложения, расплывчатости формулировок и прочим дефектам, избежать которых полностью, конечно, не удалось. Но при написании статьи и не ставилась целью подготовка профессиональных программистов, а излагаемый материал отнюдь не претендует на роль истины в последней инстанции.

Рекомендуемая литература:

1. Руководства и пособия, входящие в комплект БК-0010 (разные предприятия, выпускающие БК, комплектуют его различным набором литературы и под несколько отличающимися названиями, но примерно одного содержания).

2. *Вигдорчик Г.В., Воробьев А.Ю., Праченко В.Д.* Основы программирования на ассемблере для СМ ЭВМ. М.: Финансы и статистика, 1983.

3. *Сингер М.* Мини-ЭВМ PDP-11: программирование на языке ассемблера и организация машины. М.: Мир, 1984.

4. *Фролов Г.И., Гембицкий Р.А.* Микропроцессоры. Вып. 7. Автоматизированные системы контроля объектов. М.: Высшая школа, 1984.

5. *Брусенцов Н.П.* Микрокомпьютеры. М.: Наука, 1985.

6. *Фрэнк Т.С.* PDP-11. Архитектура и программирование. М.: Радио и связь, 1986.

7. *Брябрин В.М.* Программное обеспечение персональных ЭВМ. М.: Наука, 1988.

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

9. *Талов И.Л., Соловьев А.Н., Борисенков В.Д.* Микро-ЭВМ. В 8 кн. Кн. 1. Семейство ЭВМ «Электроника-60». М.: Высшая школа, 1988.

10. *Фролов Г.И., Шахнов В.А., Смирнов Н.А.* Микро-ЭВМ. В 8 кн. Кн. 8. Микро-ЭВМ в учебных заведениях. М.: Высшая школа, 1988.

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

12. *Напрасник М.В.* Микропроцессоры и микро-ЭВМ. М.: Высшая школа, 1989.

Данным списком вовсе не исчерпывается вся возможная литература. Просто ее перечень так велик, что привести все источники не представляется возможным.

Архитектура БК-0010

Что такое архитектура ЭВМ? По аналогии с архитектурой города можно догадаться, что это — общее построение аппаратных средств машины. Действительно, город, состоящий из улиц и домов, построенных из кирпичей, имеет АРХИТЕКТУРУ (эти кирпичи расположены определенным образом). С точки зрения жителей города, нас обычно не интересует, что это за кирпичи, их химический состав, размеры и т.п. Зато очень важны расположение районов, ширина улиц, удобство и размеры домов и прочее, с чем мы постоянно имеем дело, живя в городе. Немаловажным вопросом является и городской транспорт. Чтобы узнать, из каких материалов построен дом, его надо разобрать, а это уже дело не его жителей, а строителей или ремонтников. Жителя же интересует обычно только то, что он может увидеть, не прибегая к помощи отбойного молотка. Итак, АРХИТЕКТУРА ЭВМ — это ее аппаратные средства, ДОСТУПНЫЕ ПРОГРАММНЫМ ПУТЕМ. Если программист не может обратиться к каким-либо средствам ЭВМ, то они, по мнению автора, к архитектуре не относятся.

Любая ЭВМ предназначена для обработки информации, причем, как правило, она осуществляет эту обработку ОПОСРЕДОВАННО — представляя информацию в виде чисел. Для работы с числами машина имеет специальную, важнейшую часть — ЦЕНТ-

РАЛЬНЫЙ ПРОЦЕССОР (ЦП). Это универсальное логическое устройство, которое оперирует с двоичными числами, осуществляя простейшие логические и математические операции — сдвиг, сложение, сравнение и т.п., и делает это не просто как придется, а в соответствии с ПРОГРАММОЙ, т.е. в заданной последовательности.

Чтобы оперировать с данными, их нужно откуда-то брать, а значит, они должны предварительно где-то храниться. Также где-то должна храниться и необходимая последовательность действий ЦП — программа. Для этого служит ЗАПОМИНАЮЩЕЕ УСТРОЙСТВО — ЗУ. ЗУ бывают ПОСТОЯННЫЕ (ПЗУ), в которых информация хранится, не изменяясь, сколь угодно долго, и ОПЕРАТИВНЫЕ (ОЗУ), информация в которых может быть в любой момент изменена в соответствии с результатами ее обработки. Информация в ПЗУ записывается при их изготовлении, а откуда берется исходная информация в ОЗУ?

Для ввода информации в ЭВМ служат ВНЕШНИЕ УСТРОЙСТВА — КЛАВИАТУРА и НАКОПИТЕЛЬ НА МАГНИТНОЙ ЛЕНТЕ (НМЛ). Другое внешнее устройство — ДИСПЛЕЙ — служит для ВЫВОДА информации. Отметим, что вывод информации может осуществляться и на НМЛ, после чего она может храниться там неограниченно долго и вновь использоваться. На более мощных ЭВМ существует ряд других внешних устройств: печатающие устройства, или ПРИНТЕРЫ, накопители на гибких (НГМД) и жестких (НМД, «винчестер») магнитных дисках и т.п. Все эти возможности «открыты» и для БК, но рассказ о подключении принтеров и дисководов не входил в планы автора данной статьи. (О подключении дисководов читайте в журнале «Персональный компьютер БК-0010 — БК-0011М», №1 за 1993 г. — *Прим. ред.*)

Как же ЦП общается с ОЗУ, ПЗУ и внешними устройствами? Через так называемое АДРЕСНОЕ ПРОСТРАНСТВО ЭВМ, в котором каждое из внешних устройств (а также каждая ячейка памяти) имеет свой адрес, подобно тому как имеет номер каждый дом в городе. Но мы знаем, что в городах есть и многоквартирные дома, где, войдя в дом, мы еще не достигаем конечной цели — нужно найти квартиру, а возможно, и позвонить нужное число раз. Или представьте, что, попав в нужный дом, мы обнаружили, что там — почтовое отделение, а в нем своя система адресов и свои правила работы. Есть такие «дома» и в ЭВМ — это РЕГИСТРЫ ВНЕШНИХ УСТРОЙСТВ, или СИСТЕМНЫЕ РЕГИСТРЫ (СР). Обратившись к такому адресу, мы как бы выходим за пределы нашего города, перед нами открыт весь мир! Но чтобы общаться с ним, мы должны знать правила: как написать на конверте адрес, куда конверт опустить, а когда получим ответ — как его понять, вообще, куда он придет и каким будет.

Такова в общих чертах и архитектура БК-0010. Но мы рассмотрели ее очень поверхностно и схематично. Чтобы перейти к подробному рассмотрению, нам придется чуть отвлечься и разобраться, с какой же информацией и как ЭВМ работает.

Что такое системы счисления, вы, наверное, знаете. Привычная нам десятичная система очень неудобна для ЭВМ — ведь для обработки каждого числа нужно иметь 10 устройств, каждое из которых «знает» одну цифру из десяти — 0, 1, ..., 9. Однако любое число можно выразить и проще — в двоичной системе, где есть всего две цифры — 0 и 1. Правда, при этом запись числа становится очень объемной и неудобной для человека, а перевести двоичное число в десятичное весьма сложно. Есть компромисс — восьмеричная система, где запись чисел довольно компактна, а перевод в двоичную и обратно очень прост. В этой системе мы и будем общаться с ЭВМ, не забывая, однако, о том, что сама ЭВМ работает только с двоичными числами. Кратко напомним правила перевода двоичного числа в восьмеричное. Для этого нужно разбить двоичное число справа налево на тройки чисел (триады), а затем заменить каждую тройку на соответствующую восьмеричную цифру:

Двоичное число	Восьмеричная цифра
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

Пример:

```
1010100011111101          — исходное число
001 010 100 011 111 101   — разбиваем на триады
 1   2   4   3   7   5    — восьмеричное число 124375
```

Заметим, что мы дополнили двоичное число слева двумя незначащими нулями, чтобы получить полную триаду, но это не обязательно. Перевод из восьмеричной системы в двоичную делается в обратном порядке и также несложен.

Все цифровые данные в дальнейшем мы будем приводить в восьмеричном виде, а в тех случаях, когда необходимо использовать десятичное число, будем помечать его буквой «д». Заметим кстати, что восьмеричные числа от 0 до 7 равны десятичным и безразлично, в какой системе они приведены. Если же в составе числа встречаются цифры 8 и 9, то ясно, что число десятичное, и в пометке «д» оно не нуждается. Если нужно будет привести двоичное число, то это будет специально оговорено. Другие системы счисления мы использовать не будем.

Теперь несколько слов о нашем «городском транспорте». Существуют две основные, как их называют, КОНФИГУРАЦИИ ЭВМ — с раздельными шинами адрес—данные и с общей. По названию фирм, разработавших то или иное направление, они носят название INTEL и DEC, соответственно. Не останавливаясь в подробностях на преимуществах и недостатках каждой системы (а они есть и у той, и у другой), скажем лишь, что БК-0010 имеет конфигурацию DEC, или ОБЩАЯ ШИНА АДРЕСА—ДАННЫХ. Это означает, что и адрес, и данные передаются в ЭВМ по одним и тем же линиям, связывающим ее блоки. Но ЭВМ построена так, что программист не ощущает особенностей ее конфигурации, и ему не так уж важно, как она работает, а поэтому он может по-прежнему манипулировать адресом и данными отдельно и независимо, не путая их друг с другом. Одним словом, конфигурация ЭВМ (определяемая, кстати, в основном типом ЦП) к архитектуре ЭВМ (в нашем определении) не относится и поэтому нас мало интересует. Продолжая сравнение с городом, можно сказать, что транспорт у нас — только такси. Довольно сесть и назвать адрес, и нас по нему доставят. О маршруте мы можем не беспокоиться, это не наша забота. Однако в конце «поездки» придется платить, и на ЭВМ наша «плата» — это время. Если мы укажем адрес не оптимально («в центр — через пригород!»), «счетчик» может «нащелкать» слишком много, нам не расплатиться — ЭВМ будет слишком долго обрабатывать информацию. Косвенно мы все-таки учитываем конфигурацию ЭВМ через СПОСОБЫ АДРЕСАЦИИ, а как раз они-то в в системе команд, рассчитанной на конфигурацию DEC, чрезвычайно гибки и разнообразны. Это и есть основное преимущество «дековской» системы команд. Но обо всем — в свое время.

Контрольные вопросы и задания

1. Что такое архитектура ЭВМ?

— Это аппаратные средства ЭВМ, доступные программным путем.

2. Какова важнейшая часть ЭВМ, ее сокращенное наименование?

— Центральный процессор (ЦП).

3. Как сокращенно обозначаются оперативное и постоянное запоминающие устройства?

— ОЗУ, ПЗУ.

4. Какие внешние устройства ЭВМ БК-0010 вы знаете?

— Клавиатура, накопитель на магнитной ленте, дисплей.

5. Через какие регистры БК обращается к своим внешним устройствам, их сокращенное наименование?

— Регистры внешних устройств, или системные регистры (СР).

6. Переведите восьмеричное число 3747 в двоичное (можно пользоваться таблицей, приведенной выше).

Порядок перевода:

```
 3   7   4   7
011 111 100 111
```

Ответ: 11111100111.

7. Какова конфигурация БК-0010? Как она называется по наименованию фирмы, разработавшей данное направление?

— Общая шина адрес—данные (конфигурация DEC).

8. Какие конкретные преимущества и недостатки конфигураций DEC и INTEL приводит автор?

— Конфигурация DEC имеет меньшее число линий связи между элементами системы, что позволяет наращивать разрядность с меньшими затратами. Связь организована по принципу «запрос—ответ», т.е. асинхронно, что позволяет наращивать систему, не заботясь об удаленности ее частей друг от друга и об общей синхронизации. Все это позволяет с большей легкостью РАСПАРАЛЛЕЛИВАТЬ систему и тем наращивать ее мощность и возможности.

— Конфигурация INTEL имеет более высокое быстродействие вследствие того, что адрес выставляется по отдельным линиям связи и не требуется дожидаться ответа вызываемого устройства. Но это достигается только тогда, когда вся система размещена на одной плате или в одном корпусе, и время задержки сигнала в проводниках можно не учитывать. Элементы системы конфигурации INTEL заметно проще, чем конфигурации DEC (но с тем же ограничением). Шире ассортимент микросхем, особенно микросхем памяти, рассчитанных на подключение к этой шине.

В общем же обе конфигурации жизнеспособны и продолжают развиваться параллельно. В нашей стране почему-то бытует мнение, что конфигурация DEC не перспективна, — все журналы, посвященные компьютерной тематике, упорно публикуют материалы только по вопросам, связанным с IBM PC (конфигурация INTEL). Но фирма Digital Equipment Corp. это мнение явно не разделяет, оставаясь вторым в мире после IBM производителем компьютеров, хотя и в основном не персональных.

Адресное пространство БК-0010

Войдем в наш «город» — БК-0010. Мы сразу заметим его особенность — в нем всего одна «улица», зато какая! В 65536 «домов»! Это и есть АДРЕСНОЕ ПРОСТРАНСТВО нашей ЭВМ — 65536д БАЙТ. А что такое байт? Это блок информации, состоящий из 8 двоичных разрядов, или БИТ. Самое маленькое число, которое может быть записано в одном байте, — это, конечно, ноль (000), самое большое — 255д (377). Переведем последнее число в двоичный формат. Что получилось? Правильно, восемь единиц — 11111111.

Но в нашем «городе» нумерация «домов» немного сложнее. Они как бы стоят в два ряда. Не правда ли, вы и в своем городе такое встречали? Дом 56 и 56а, 78 и 78/1. Только у нас удобнее — каждый «дом» имеет свой номер, причем в «первой линии» стоят только четные «дома». Если каждый «дом» — это байт, то два байта называются МАШИННЫМ СЛОВОМ или просто СЛОВОМ. Четные байты имеют, конечно, и четные номера — 0, 2, ..., 177776, причем номера четного байта и слова, которому он принадлежит, совпадают. А нечетные? Они — «на задворках», и их номер на 1 больше, чем номер слова. Четный байт еще называют МЛАДШИМ, а нечетный — СТАРШИМ байтом слова, в соответствии с их номерами, ведь номер нечетного байта больше. Зачем так сделано? Дело в том, что во многих случаях мы обращаемся сразу к двум соседним байтам, т.е. к слову, и для этого байты объединены в слова. Но бывает, что нужен и более точный адрес — байт. К нечетному обратиться легко — его адрес отличается от адреса любого слова, а как быть с четным, адрес которого совпадает с адресом слова? Весьма просто — ЭВМ для обращения к байту (в отличие от обращения к слову) имеет специальные команды, они-то и определяют, что нам нужно не слово, а четный байт. И еще особенность — видели ли вы где-нибудь номер дома или квартиры «0»? А у нас в ЭВМ везде и всюду ноль — такое же равноправное число, как любое другое, и даже больше — нумерация всегда начинается с него!

Заметим, что каждый из разрядов (битов) байта или слова (в слове их, понятно, вдвое больше, чем в байте, — 16д) тоже имеет свой номер. Нумерация их идет справа налево и, как принято, начинается с 0. Правый, или МЛАДШИЙ, бит слова имеет номер 00, левый, или СТАРШИЙ, — номер 15 (для битов, как ни странно, принята десятичная нумерация: традиция, а кроме того, по определенным соображениям это удобно). Можно сказать, что номера битов — это номера «квартир» в наших «домах» — байтах, причем в «домах 2-й линии» (нечетных байтах) они имеют двойную нумерацию — сквозную для слова (8...15) и собственную для байта (0...7).

Назовем и еще одну единицу измерения информации, с которой нам придется встретиться, — КИЛОБАЙТ (Кб). Приставка «кило» говорит о том, что килобайт в 1000 раз больше байта, но это только приблизительно! В действительности же 1 Кб = 1024 байта, что удобно в двоичной системе счисления, так как именно 1024, а не 1000 является степенью числа 2.

Итак, наш «город» из одной «улицы» имеет 100000 слов, или 200000 байт, от 0 до 177777. А есть ли в этом «городе» районы? А как же!

Распределение адресного пространства

Когда-то, давным-давно, в незапамятные времена (а точнее 30—40 лет назад), когда ЭВМ только-только появились, а «динозавры» — механические счетные машины — еще не «вымерли», у первых ЭВМ память была раздельной: в одних определенных ее ячейках хранилась программа, а в других — данные, т.е. обрабатываемая информация. Позже от такой организации в большинстве систем отказались, и теперь почти во всех универсальных ЭВМ программа и данные могут занимать любые ячейки памяти во всем адресном пространстве. И все же отдельные адреса адресного пространства (или, говоря проще, но менее строго, АДРЕСА ПАМЯТИ) на любой ЭВМ, как правило, имеют особое назначение. Начнем с нулевого адреса нашего БК и посмотрим, каково распределение его адресного пространства.

Адреса 0...777 — СИСТЕМНАЯ ОБЛАСТЬ и СТЕК. Это специально выделенные зоны ОЗУ, используемые чаще всего для нужд самой ЭВМ. Более подробно об этом мы поговорим дальше.

Адреса 1000...37777 — ОЗУ ПОЛЬЗОВАТЕЛЯ. В этой зоне ОЗУ можно размещать программу и данные — что хотите и как хотите, это исключительное право программиста. Общая длина этой зоны памяти — 15,5 Кб.

Адреса 40000...77777 — ЭКРАННОЕ ОЗУ. Мы уже говорили, что оно входит в общее адресное пространство и никак от него не отличается. Никак? Но ведь это экран дисплея, и записанная туда информация может быть в любой момент разрушена. Стоит только нажать любую клавишу — на экране появится символ, т.е. информация в зоне экранного ОЗУ изменится, а если нажать клавишу «СБР» — она будет мгновенно уничтожена (экран очищен). Поэтому, используя экранное ОЗУ, мы всегда должны иметь это в виду. Экранное ОЗУ занимает 16 Кб.

Но посмотрим еще раз на уже известные зоны ОЗУ. Экран занимает так много! Нельзя ли все-таки использовать хоть часть этой памяти? Можно, и в БК-0010 это предусмотрено. Если нажать клавиши АР2 + «СБР», то экран «сожмется» до размера 4 строки текста, а программист получит в свое распоряжение дополнительную память, «отрезанную» от экрана. При этом распределение адресного пространства меняется: 1000...67777 — ОЗУ пользователя, а 70000...77777 — экранное ОЗУ. Заметим, что при этом переходе в режим РАСШИРЕНИЯ ПАМЯТИ (РП) та часть экранного ОЗУ, которая «отсекается» в пользу ОЗУ пользователя, сохраняет старую информацию, хотя на экране ее уже не видно. Напротив, при обратном переходе к большому экрану он очищается и информация, которая была до этого записана по адресам 40000...67777, безвозвратно теряется. Размер экранного ОЗУ в режиме РП — 4 Кб, а ОЗУ пользователя увеличивается на 12 Кб. (Путем использования служебных ячеек системной области БК-0010 можно произвольным образом изменять распределение памяти между ОЗУ пользователя и экранным ОЗУ, однако при этом программа и данные, помещаемые в область, «отрезанную» от экрана, воспроизводятся на нем в виде своеобразной «пестроты». Примером является программа VorteX!. — *Прим. ред.*)

Следующий большой «район» нашего «города» начинается с адреса 100000 и тянется почти до конца «улицы» — по адрес 177577. Это ОБЛАСТЬ ПЗУ. Каковы ее особенности? Если в область ОЗУ вы можете записать любую информацию по любому адресу (к чему это приведет — другой вопрос), то из области ПЗУ вы можете информацию только ЧИТАТЬ, на запись по этим адресам наложен запрет. Но тогда откуда же взялась та информация, которую мы оттуда читаем? По этим адресам «расположены» специальные микросхемы, в которые информация записана (или, как говорят, «зашита») на заводе-изготовителе, и изменить ее нельзя никакими средствами, доступными программисту. Это как бы «наследственная память» нашей ЭВМ, то, что она «знает от рождения». Что же это за знания и как они размещены?

100000...117777 — МОНИТОР-ДРАЙВЕРНАЯ СИСТЕМА (МДС). Это, без преувеличения, важнейшая часть ЭВМ, без которой она превращается в мертвое «железо». Здесь записаны все «основные инстинкты» ЭВМ. Ни одна операция, включая даже ввод символа с клавиатуры и вывод его на экран, не обходится без МДС. Эта часть ПЗУ так тесно связана в обеспечении работоспособности ЭВМ с процессором и содержит столь важные программы функционирования всех систем машины, что иногда ее называют ВИРТУАЛЬ-

НЫМ ПРОЦЕССОРОМ (здесь мы не станем расшифровывать этот термин).

Далее идет уже, так сказать, «кора головного мозга». Если без МДС машина мертва, то только с ней она тоже немногое может — загрузить программу, запустить ее... Но ни десятичной арифметики, ни чего-либо иного машина «не знает». Она даже не может дать нам возможность «просмотреть» свою собственную память. Вся «надежда» — на программы, загружаемые в ОЗУ! Но у машины есть и другие микросхемы ПЗУ, причем они отличаются для различных типов БК-0010, а в последних моделях БК-0010.01 есть все их виды:

- 120000...137777 — ИНТЕРПРЕТАТОР ЯЗЫКА БЕЙСИК или ЯЗЫКА ФОКАЛ.
- 140000...157777 — ИНТЕРПРЕТАТОР ЯЗЫКА БЕЙСИК или РЕЗЕРВНОЕ АДРЕСНОЕ ПРОСТРАНСТВО («пустое место»).
- 160000...177577 — ИНТЕРПРЕТАТОР ЯЗЫКА БЕЙСИК или МОНИТОРНАЯ СИСТЕМА ТЕСТОВ И ДИАГНОСТИКИ (МСД, МСТД).

Уточним, что по техническим причинам одна микросхема ПЗУ БК-0010 включает не более 20000 байт (или 8 Кб) информации, поэтому все ранее перечисленные «части» ПЗУ, кроме последней, имеют эту длину.

Теперь вам ясно, что интерпретатор языка Фокал занимает одну микросхему ПЗУ, а языка Бейсик — три. Ясно также, что Фокал не может работать одновременно с Бейсиком, а Бейсик — с МСД, так как они занимают одни и те же адреса.

Контрольные вопросы и задания

1. Каков размер (в байтах) адресного пространства БК-0010?

— 65536 байт.

2. Сколько бит в байте? В слове?

— 8 и 16, соответственно.

3. Номер какого байта, старшего или младшего, равен номеру слова?

— Младшего.

4. Старший байт слова имеет четный номер?

— Нет, четный номер имеет младший байт.

5. Как машина различает обращение к младшему байту или к слову, имеющим один и тот же адрес?

— В зависимости от команды обращения по этому адресу.

6. В каком направлении нумеруются биты в слове — слева направо или справа налево? Какой номер имеет младший бит?

— Справа налево. Младший бит имеет номер 0.

7. Чему равен килобайт? Его сокращенное обозначение?

— 1024д байта; «К».

8. Могут ли программа и данные располагаться в памяти произвольно? От чего это зависит?

— Да, по желанию программиста.

9. Нарисуйте распределение адресного пространства БК-0010, изображая отдельные зоны прямоугольниками и наращивая адреса сверху вниз. Подпишите их наименования. Укажите адреса начала и конца каждой зоны и ее длину в Кб (при выполнении последнего задания можно воспользоваться текстом статьи). Сделайте такой же рисунок для режима РП. Выучите распределение адресного пространства наизусть — это очень важно.

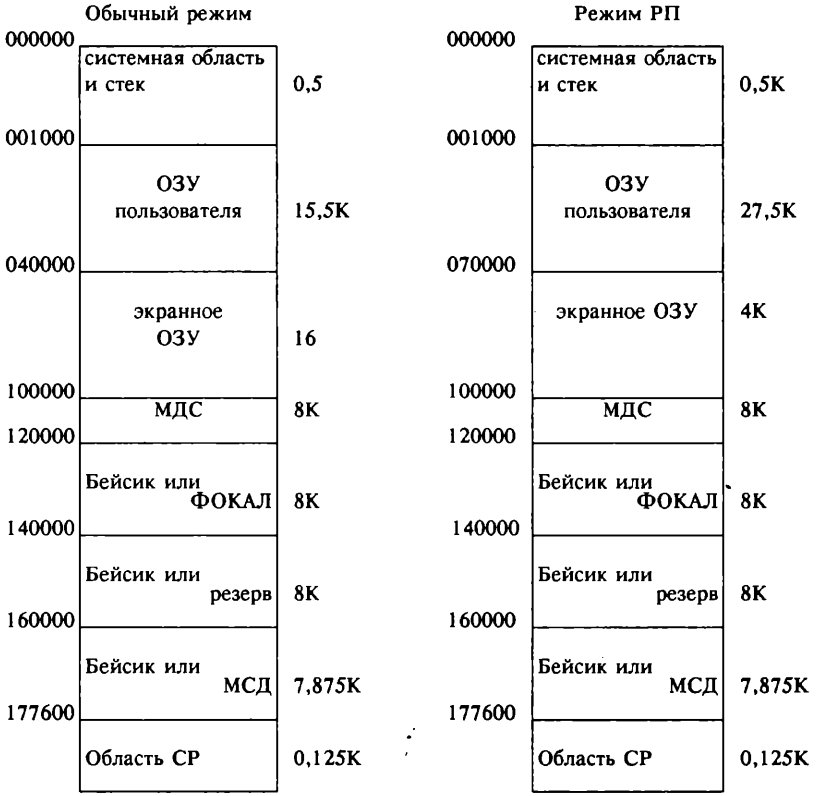
Последнюю область, «Область СР», вы имели полное право не изображать, так как мы с ней еще не знакомы. Но на рисунке приведено все, «как положено», чтобы к этому вопросу больше не возвращаться.

10. Какова информационная емкость одной микросхемы ПЗУ БК-0010? Может ли быть изменена записанная туда информация?

— 8 Кб. Не может. Заметим, что микросхемы ПЗУ БК-0010 — так называемые МАСОЧНЫЕ, типа К1801РЕ2, в них информация заносится в процессе изготовления кристалла микросхемы. Существует «легенда», что можно эту информацию переписать заново с помощью специального ПРОГРАММАТОРА. Не верьте, дети, сказкам — это неправда. Зато существуют особые перепрограммируемые ПЗУ (ППЗУ), например, типа К573РФ3, КМ1801РР1. В них действительно можно записать и переписать информацию, но в БК-0010 их нет и никогда не было, они используются в основном при разработке новых ЭВМ. (Однако ППЗУ некоторых типов являются аналогами использованных в БК масочных ПЗУ и могут быть установлены взамен последних без каких-либо изменений в схеме. В одном из выпусков редакция планирует вернуться к этой теме разговора. — *Прим. ред.*)

Мониторная система диагностики (МСД)

Сразу оговоримся, что в состав МСД входит также система тестов нашей микро-ЭВМ.



ЭВМ — настолько сложное устройство, что проверить ее может только она сама! Поэтому и созданы специальные тест-программы, с которыми, конечно, вы знакомы, ведь ваш БК-0010, наверное, проверялся согласно инструкции, как при покупке, так и в дальнейшем. Напомним, как это делается. После включения нажмите клавиши ЛАТ и ЗАГЛ (если у вас БК-0010.01, то перед включением должен быть подключен блок МСТД), а затем нажимайте последовательно: Р, ПРОБЕЛ, Т, ВВОД. На экране появится приглашение тест-системы: «+». Теперь, в зависимости от того, какой из тестов вы хотите провести, нажмите одну из цифровых клавиш 1...5 и выполняйте указания БК. Набор из пяти тестов проверяет все системы микро-ЭВМ исчерпывающим образом (правда, с оговоркой, что алгоритмы некоторых тестов довольно примитивны).

А как перейти в МСД? Мы не зря вспомнили про тесты — выход в МСД производится через тест-систему. Перейдя в тесты, нажмите клавиши: РУС, Т, С. На экране появится приглашение МСД: — « ¤ ». Это так называемый «знак денежной единицы», или, на программистском жаргоне, КОЛЕСО (иногда его называют также «черепашка», «солнышко»). МСД, как и пусковой монитор (выход в который из Фокала — Р, ПРОБЕЛ, М, ВВОД; из Бейсика — MON, ВВОД; приглашение «?»), также имеет ряд команд, или директив, но количество их и, соответственно, возможности МСД, намного больше, чем в мониторе. Отметим, что МСД еще называют СИСТЕМОЙ ОТЛАДКИ программ в кодах, что само по себе уже кое о чем говорит. Что же может МСД? Удобнее всего отвечать на этот вопрос, изучая ее директивы.

Директивы МСД

МСД «понимает» директивы, состоящие из символов (русские заглавные буквы и латинская G) и цифр, причем если в составе директивы присутствуют цифры, то число, следующее ДО буквы, задает какой-либо параметр. Буква без предшествующих цифр соответствующий параметр не задает, а индицирует, т.е. выводит на экран какое-либо зна-

чение. Помимо директив для работы с памятью ЭВМ в набор команд МСД входят и команды работы с магнитофоном. Все цифровые данные МСД принимает и индицирует только в восьмеричной системе счисления. Эти данные мы будем обозначать как (...). Они могут состоять максимум из шести восьмеричных цифр, причем если их введено больше, то действительны только последние шесть. Незначащие нули слева можно не набирать. Ошибку можно исправить с помощью клавиши «ЗАБОЙ» (перемещение курсора влево со стиранием символа, код клавиши — 30). Итак, директивы:

- (...)А — установить значение текущего адреса. Если мы введем, например, 1000А, то текущий адрес будет равен 1000.
- А — проконтролировать значение текущего адреса. МСД выдает ответ «=(...)», в нашем случае: «А=1000».
- (...)Д — установить длину массива (в байтах).
- Д — проконтролировать длину массива. Выполняется аналогично директиве А.
- (...)Р — размножить число (...) в заданном диапазоне адресов, т.е. записать число (...) во все слова с адреса А до А+Д (не включая последний!). Например, если нам нужно «обнулить» (попросту говоря, стереть, или очистить) участок памяти, начиная с адреса 1000 до 4000, мы набираем: 1000А 3000Д0Р. Так как запись производится по словам, а А и Д задаются в байтах, то следует задавать лишь четные А и Д. Иначе используется ближайшее четное значение, меньшее заданного. (Это замечание относится ко всем случаям, когда параметры задаются в байтах, а работа производится со словом.)
- Х — подсчитать контрольную сумму массива, размещенного в памяти, начиная с адреса А до А+Д. МСД выдает ответ: «=(...)». Например, подсчитаем контрольную сумму ПЗУ МДС: 100000А20000ДХ. Ответ: «=177777», т.е. то же самое значение, которое БК выдает для первого ПЗУ, когда мы проводим ТЕСТ 1. Тут, может быть, уместно пояснить, что это такое. ЦИКЛИЧЕСКАЯ КОНТРОЛЬНАЯ СУММА вычисляется сложением всех слов (в других случаях — байт) контролируемого массива информации с прибавлением бита переноса за пределы слова (байта). Что такое, в свою очередь, перенос, станет ясно в дальнейшем (правда, еще не так скоро, но не будем забегать вперед). Контрольная сумма позволяет убедиться в идентичности массивов, не сравнивая их с эталоном, так как вероятность ее случайного совпадения в общем случае не превышает 1/65536 (около 0,0015%), т.е. практически нулевая. Но есть частные случаи, когда контрольная сумма теряет свое ключевое значение. Достаточно привести такой пример: контрольная сумма любого пустого (состоящего из одних нулей) массива всегда равна нулю независимо от его длины. (Иногда по некоторым причинам при чтении файлов с магнитофона БК воспринимает все считываемые биты как нулевые, включая и записанное вместе с файлом значение его контрольной суммы. В этом случае, хотя ошибка чтения очевидна, компьютер считает, что операция выполнена правильно. — *Прим. ред.*) Разумеется, используемый в МСД способ вычисления контрольных сумм не единственно возможный и не самый лучший.
- (...)П — переслать по адресу (...) массив, записанный, начиная с адреса А до А+Д. Отметим, что «переслать» — не совсем удачный термин, так как исходный массив тоже сохраняется в памяти. Правильнее было бы сказать: «скопировать». Попробуем переслать содержимое какого-либо ПЗУ в экранную область памяти, это выглядит довольно эффектно. Набираем: 100000А20000Д40000П. Содержимое ПЗУ почти мгновенно появляется на экране, что также дает некоторое представление об «истинном», т.е. не «связанном» языками высокого уровня, быстродействии ЭВМ — ведь ей пришлось при этом переместить в памяти более 4000д чисел и проделать еще кое-какие операции! Пересылка возможна и с «пересечением» зон адресов пересылаемых массивов, например: 2000А10000Д1000П. Но пересылка «в обратную сторону» (1000А10000Д2000П) не даст ожидаемого результата — оба массива будут испорчены! Если вы вспомните, что

пересылка производится по словам, начиная с первого байта массива, то легко догадаетесь, почему так происходит.

- (...)С — сравнить в памяти два массива: эталонный, расположенный с адреса А до адреса А+Д, и контролируемый, с адреса (...). Сравнение производится по словам.
- В процессе проверки все расхождения информации в массивах выдаются на экран в следующем виде:

	(эталонный массив)		(контролируемый массив)	
(1)	адрес	данные	адрес	данные
(2)	адрес	данные	адрес	данные
	. . . . .	. . . . .	. . . . .	. . . . .
(N)	адрес	данные	адрес	данные

Если массивы полностью совпадают, на экран ничего не выдается. Например, попробуем переслать в память и сравнить с исходным содержимое одного и того же ПЗУ. 100000А20000Д1000П — мы выполнили пересылку по адресу 1000. Теперь намеренно «испортим» массив в ОЗУ уже известной нам директивой: 2000А10Д0Р — мы записали в массив 8 байт (т. е. 4 слова) нулей. Теперь введем: 100000А20000Д1000С. На экран будет выдано четыре строки указанного выше вида — специально «сделанная» нами ошибка в составе контролируемого массива выявлена!

- (...)Л — листать (выдать на экран) массив, начиная с адреса А, длиной (...) байт. После этого А принимает новое значение, равное А+(...). Массив выдается на экран в виде машинных слов (а не байт). Например, выполним: 100000А10Л — на экран будет выдано содержимое первых четырех слов ПЗУ, после чего если мы дадим директиву А, то получим: А=100010.
- И — индикация содержимого слова по текущему адресу.
- (...)И — запись числа (...) в слово по текущему адресу.
- Б — индикация содержимого байта по текущему адресу.
- (...)Б — запись числа (...) в байт по текущему адресу.

Необходимо отметить, что как слово, так и байт выдаются в виде шести цифр. При выдаче байта, конечно, три старшие цифры всегда нули. При вводе байта с клавиатуры может быть набрано любое количество цифр, но запись производится только по модулю 256д (т.е. остатка от целочисленного деления на 256) шести цифр, набранных последними. При байтовых операциях параметр А может быть и нечетным, при этом он определяет старший байт слова с четным адресом, меньшим на единицу.

- Ц — циклическое (т.е. непрерывное) чтение слова по адресу А. Команда может быть полезна для исследования ячеек памяти с изменяющейся во времени информацией. Выход из цикла — клавиша «СТОП». Например, дайте директиву: 177662АЦ, затем нажимайте различные клавиши и наблюдайте за результатами циклического чтения данной ячейки на экране.
- Щ — снять защиту системной области. Мы уже отмечали, что область адресов 0...777 называется СИСТЕМНОЙ и используется преимущественно для собственных нужд ЭВМ. Запись туда случайной информации может грозить неприятностями, вплоть до полного отказа ЭВМ в работе (конечно, временного, до отключения питания или перезапуска ЭВМ, но информация, хранящаяся в ОЗУ, при этом будет утрачена). Чтобы избежать неприятностей, системная область защищена в МСД от случайной записи, и при попытке сделать это выдается сообщение «ЗЩ» — ЗАЩИТА. После подачи директивы «Щ» защита снимается и возможна запись в системную область. Восстановление защиты — клавиша «СТОП».
- , (запятая) — чтение слова с инкрементом (ИНКРЕМЕНТ — увеличение). По данной директиве текущий адрес А увеличивается на 2, а затем содержимое ячейки по этому адресу выводится на экран.
- (...), — запись слова по текущему адресу с инкрементом: по текущему адресу записывается слово (...), а затем адрес увеличивается на 2 и выводится слово по этому адресу. Эти директивы используются преимущественно для чтения или записи последовательно расположенных ячеек памяти.

- – (минус) — чтение слова по текущему адресу с декрементом.
- (...)– — запись слова (...) по текущему адресу с декрементом (ДЕКРЕМЕНТ — уменьшение). Данная директива аналогична предыдущей, с той лишь разницей, что значение текущего адреса А не увеличивается, а уменьшается на 2.
- .(точка) — чтение байта с инкрементом.
- (...). — запись байта с инкрементом.
- :(двоеточие) — чтение байта с декрементом.
- (...): — запись байта с декрементом.

Пара последних директив полностью аналогична двум предыдущим, только работают они не со словом, а с байтом, и увеличение или, соответственно, уменьшение текущего адреса при их исполнении происходит не на 2, а на 1. Для этих команд действительны и нечетные адреса.

- МП — пуск мотора магнитофона при наличии дистанционного управления (ДУ) от БК-0010. Останов мотора — любая символьная клавиша или «СТОП». Отметим, что описанная в «Руководстве системного программиста БК-0010» директива «МС» (останов магнитофона) бессмысленна, так как по нажатию первой же клавиши (М) магнитофон остановится и нажимать вторую уже как-то ни к чему, разве что для порядка.
- МФ — фиктивное чтение файлов с магнитной ленты. На запрос «ИМЯ=» нужно ввести имя того файла, ПОСЛЕ которого магнитофон должен быть остановлен. Загрузка файла в ОЗУ и проверка контрольной суммы по данной директиве не производится, а имена, не совпадающие с заданным, выводятся на экран. (Обычно эта директива используется для просмотра содержимого кассеты при поиске нужного файла. — *Прим. ред.)*
- МЧ — загрузка файла с МЛ. На запрос «АДРЕС=» необходимо ввести адрес в ОЗУ, начиная с которого должен быть загружен файл, а на «ИМЯ=» — имя загружаемого файла. После набора очередных цифровых данных или имени файла необходимо нажимать клавишу «ВВОД», исправлять ошибки можно клавишей «ЗАБОЙ». Если вместо адреса задать число «0», то файл будет загружен по адресу, указанному в его ОГЛАВЛЕНИИ, т.е. в специальном блоке, записанном на МЛ в начале файла. Фактически файл при этом будет загружен в ту область памяти, из которой он был записан на МЛ, а чаще всего именно это и требуется.
- МЗ — запись хранящегося в ОЗУ или ПЗУ массива данных в виде файла на МЛ. На запрос «АДРЕС=» вводится адрес начала массива, «ДЛИНА=» — его длина в байтах, «ИМЯ=» — имя файла, под которым он должен быть записан (имя файла в МСД может состоять не более чем из 16д ЛЮБЫХ символов; если его длина меньше, оно автоматически дополняется пробелами). Все указанные параметры заносятся в оглавление файла при его записи на МЛ.

Заметим, что при исполнении трех последних директив магнитофон с помощью ДУ автоматически включается, а по окончании чтения или записи (или при нажатии клавиши «СТОП») — отключается. Нужно сказать, что ДУ магнитофона при работе с БК — громадное удобство, оценить которое могут лишь те, кто с ним работал. К сожалению, немногие бытовые магнитофоны имеют вход ДУ, совместимый с БК-0010, поэтому большинство пользователей никогда не работали с ним и даже не знают, чего они лишены! (Многие пользователи, имея в своем магнитофоне дистанционное управление, и не подозревают об этом! Так, в некоторых модификациях магнитофона «Электроника-302» ДУ «встроено» в цепь внешнего микрофона и может быть использовано при работе БК, если включить в гнездо микрофонного входа штекер  стандартного магнитофонного кабеля БК. — *Прим. ред.*)

- К — выход из МСД. При этом управление передается Фокалу, т.е. по адресу 120000. Содержимое ОЗУ пользователя стирается (кроме зоны адресов 1000...1377).
- ТК — выход в тесты. После выполнения данной директивы появляется приглашение тест-системы: «+». Если не проводить тесты 1 и 5, то содержимое ОЗУ сохраняется. Можно вернуться в МСД, как обычно: РУС, Т, С.
- ТД — переход к тестам внешней диагностики. Применяется при нали-

чии дополнительной диагностической аппаратуры, подключаемой к разъему системной магистрали (разъем МПИ) БК-0010.

- T0, T1...T5 — запуск тестов МСД. Набор из этих шести тестов несколько отличается от тестов 1...5 тест-системы, но в целом они аналогичны. Отметим, что весьма полезным является тест 2 МСД, а тест 0 может выполняться только при работе ЭВМ в составе локальной сети (в классе КУВТ).
- (...)G — запуск программы по адресу (...). Это единственное исключение, когда директива МСД подается в регистре ЛАТ—ЗАГЛ.

Теперь, когда мы ознакомились с директивами МСД, ответим на вопрос, для чего может быть полезна система отладки. Ясно, что в МСД можно как просматривать память ЭВМ, так и изменять ее содержимое, т. е. заносить в ячейки памяти информацию. Зная язык МАШИННЫХ КОДОВ (или КОМАНД), в МСД можно писать программы в кодах, вводить массивы данных, записывать их в виде файлов на МЛ, запускать, отлаживать.

Хотя язык машинных кодов не так сложен, как принято обычно считать, программировать непосредственно в кодах неудобно, это требует большого напряжения и отличной памяти (имеется в виду, конечно, память программиста, а не БК). Поэтому разработан специальный вспомогательный ЯЗЫК АССЕМБЛЕРА, он позволяет достичь тех же самых результатов гораздо меньшей ценой. Зато, имея уже готовую распечатку программы в кодах, можно относительно легко ввести ее в память, пользуясь директивами МСД. Это позволяет публиковать тексты сравнительно небольших программ в печатных изданиях, избавляя от необходимости тиражировать их непосредственно на МЛ. Если вам требуется ввести такую программу, то, чтобы избежать ошибок, рекомендуется сделать это дважды по разным адресам, а затем провести сверку этих массивов — маловероятно, что в обоих случаях вы сделаете одинаковые ошибки. Попробуйте ввести в память и запустить простейшую программу в кодах, для чего выполните команды:

1000A
12700, 14, 104016, 12700, 101, 104016, 0,
1000G

На выводимые по директиве «запятая» числа не обращайте внимания и набирайте новые. После запуска эта программа выполняет сброс экрана и выводит символ «А». Как видите, программирование в кодах — не такая уж сложная штука! (Если перед входом в режим МСД из тест-системы выполнить тест 1 (ОЗУ, ПЗУ), то при последующем вводе программы в кодах после нажатия на клавишу «запятая» будет каждый раз выводиться адрес очередной ячейки памяти. — *Прим. ред.)*

Директивы МСД для работы с магнитофоном, как легко догадаться, позволяют не только загружать программы с МЛ, но и копировать их, если после загрузки записать на МЛ загруженный массив, причем безразлично, на каком языке написана эта программа, нужно лишь знать ее адрес и длину. (Кроме некоторых программ с автозапуском или загружаемых в ОЗУ экрана. — *Прим. ред.)*

В заключение приведем некоторые полезные сведения, которыми можно воспользоваться при работе в МСД.

- Определить параметры успешно загруженного файла можно, если знать, что в ячейке 264 хранится адрес его загрузки, а в ячейке 266 — длина (достаточно дать директиву 264А4Л). Контрольная сумма файла хранится в ячейке 312 и выводится директивой 312АИ. Но можно определить параметры файла (кроме контрольной суммы) и не загружая его, для чего необходимо в режиме фиктивного чтения (МФ) прочитать его имя (независимо от того, совпадает ли оно с заданным), после чего адрес файла содержится в ячейке 346, а длина — в ячейке 350, причем адрес при этом будет «истинный», т.е. указанный в оглавлении на МЛ, а не заданный в процессе загрузки, как это имеет место при чтении ячейки 264. Ячейки 346 и 350 могут быть прочитаны и после загрузки файла с тем же результатом. Отметим, что контрольная сумма в ячейке 312 вычисляется не так, как описанная выше, и поэтому обычно не совпадает с контрольной суммой массива, полученной в МСД по директиве «Х».
- По директиве 100274G происходит переход в ПУСКОВОЙ МОНИТОР (ПМ, символ диалога — «?») без стирания ОЗУ (ПМ мы не описывали, так как он практически всем хорошо

известен и значительно уступает МСД по набору директив и своим возможностям). Перейти же из ПМ в МСД можно через тесты, нажав клавиши: Т, ВВОД, РУС, Т, С (также без стирания ОЗУ). Перейти в ПМ без стирания экрана и выхода из режима РП (если он установлен) можно, выполнив директивы: 4АЩ100442И, после чего нужно нажать клавишу «СТОП». Смысл того, что при этом делается, вы поймете в дальнейшем.

- Перейти из МСД в Фокал без стирания ОЗУ можно, выполнив последовательность директив:

  Щ
  120020А26Д0П
  262А177777И
  «СТОП»

После этого на экране появляется сообщение Фокала «ОСТАНОВ ПО КЛАВИШЕ СТОП». Чтобы такой переход был успешным, выход из Фокала перед этим должен быть произведен непосредственно в МСД, а не через ПМ, и в МСД не должны загружаться (а тем более запускаться!) программы в кодах. При соблюдении этих условий после выхода в Фокал содержимое ОЗУ полностью сохраняется. Ранее загруженные в Фокале программы могут даже запускаться для дальнейшей работы, возможен просмотр их листинга и т.п. Переход Фокал—МСД—Фокал может делаться таким способом неоднократно, например, с целью модификации в МСД Фокал-программ или для изучения их формата. К сожалению, более подробно останавливаться на этом вопросе нет возможности.

- Выйти из пускового монитора в Бейсик без стирания ОЗУ (с сохранением ранее загруженной Бейсик-программы) можно командой С120234 ВВОД. Правда, ключи клавиатуры (клавиши 1—9 и 0 по регистру АР2) при этом не сохраняются.

Диагностические сообщения МСД

В процессе работы в МСД система ведет развернутый диалог с пользователем, выдавая различные указания и сообщая результаты исполнения директив. Обычно сообщения МСД выдаются в достаточно понятной форме и не требуют особых комментариев, особенно если вы хорошо знаете директивы. Но есть несколько специальных ДИАГНОСТИЧЕСКИХ СООБЩЕНИЙ, которые весьма кратки и нуждаются в пояснении. С одним из них мы уже познакомились:

- ЗЩ — защита системной области. Выдача этого сообщения означает, что произведена попытка выполнить с помощью одной из директив МСД запись в область адресов 0...777. Напомним, что на запись в системную область наложен запрет (до снятия защиты) исключительно для директив МСД, но не для машинных команд, поэтому от программы пользователя системная область никак не защищена, это забота программиста.

Еще два сообщения МСД нам пока не встречались, и их следует рассмотреть более подробно.

- ЗВ(...) — зависание по адресу (...). Это сообщение выдается в двух основных случаях: при попытке записи по адресу ПЗУ и попытке чтения или записи по несуществующему адресу.

Относительно записи информации в область адресов, занимаемую ПЗУ, все ясно — запись по адресам 100000...177577 запрещена аппаратно и невозможна по техническим причинам. А что такое несуществующий адрес? Это адрес, где нет реального аппаратного устройства, которое могло бы ответить на вызов процессора. При выполнении подобных некорректных операций процессор, не получив ответа от устройства (например, ОЗУ), выполняет так называемое ПРЕРЫВАНИЕ ПО ЗАВИСАНИЮ и производит переход по ВЕКТОРУ 4 (что это такое, станет ясно в дальнейшем), результатом чего и является выдача сообщения «ЗВ» — ЗАВИСАНИЕ и значения текущего адреса А. Понятно, что если зависание возникло в результате исполнения директивы МСД (например, 100000А377И), то выданное значение обычно и есть тот адрес, по которому произошло зависание. Но в прочих случаях, например при исполнении программы, выданное значение и адрес, по которому произведено некорректное обращение, как правило, не имеют ничего общего. В этом случае информацию несет только само сообщение «ЗВ».

- НК — неправильная команда. В набор команд процессора входит строго фиксированный перечень кодов. В случае, если код очередной команды,

встретившейся в программе, не входит в этот перечень, процессор выполняет ПРЕРЫВАНИЕ ПО РЕЗЕРВНОМУ КОДУ и производит переход по ВЕКТОРУ 10 (что также будет в дальнейшем разъяснено). В результате и выдается сообщение «НК». Следует особо отметить, что далеко не весь набор команд процессора К1801ВМ1 использован в БК-0010 (или, как говорят, реализован аппаратно). Вследствие этого есть такие коды (не перечисленные, в частности, в «Руководстве системного программиста», прилагаемом к БК), которые вместо ожидаемого сообщения «НК» вызывают сообщение «ЗВ». Примером является код 12. По нему процессор должен был бы произвести запись некоторой информации в область системных регистров. Но нужных регистров в адресном пространстве БК-0010 просто нет, поэтому и происходит прерывание по зависанию. Примером же действительно несуществующей команды является код 30, вызывающий, как и положено, сообщение «НК». Знание этих особенностей поведения процессора позволяет иногда разобраться в загадочных, казалось бы, сообщениях при отладке программы.

Необходимо отметить, что описанные сообщения выдаются только в том случае, если выполняются программы пользователя, загруженные и запущенные в МСД (либо при выполнении директив самой МСД). Если же загрузка и запуск программ производятся из ПМ либо программа пользователя изменяет значение ВЕКТОРОВ ПРЕРЫВАНИЯ 4 и 10, диагностические сообщения МСД, разумеется, выдаваться не будут.

Итак, мы вкратце познакомились с мониторной системой диагностики БК-0010, хотя нам и пришлось ради этого надолго прервать описание архитектуры ЭВМ. Но затраченное время в дальнейшем окупится — теперь вы сможете не просто знакомиться с материалом, предлагаемым вашему вниманию, но во многих случаях активно проверять его на своем БК, экспериментировать. Никогда не упускайте случая потренироваться в работе с МСД, а заодно и проверить автора — он ведь тоже может ошибаться!

Контрольные вопросы и задания

1. Для чего служит МСД?

— Для просмотра и изменения содержимого ОЗУ, записи и чтения файлов, написания и отладки программ в кодах.

2. Как называют знак диалога МСД?

— Знак денежной единицы, «колесо».

3. В каком регистре (РУС, ЛАТ, СТР, ЗАГЛ) подаются буквенные директивы МСД? Какое исключение из этого правила?

— В регистре РУС—ЗАГЛ, за исключением директивы G.

4. Выполните следующее задание:

- запишите на МЛ содержимое монитор-драйверной системы БК-0010 (вы помните адрес и длину этого массива?);
- загрузите записанный файл в экранное ОЗУ (с адреса 40000);
- вызовите из ячеек 264, 266, 312, 346, 350 его параметры и поясните, «что есть что».

— Порядок действий (сообщения МСД частично опущены):

- МЗ; АДРЕС=100000; ДЛИНА=20000; ИМЯ=МДС;
- МЧ; АДРЕС=40000; ИМЯ=МДС;
- 264А4Л; ответ: 040000 020000 — адрес загрузки и длина файла МДС;
- 346А4Л; ответ: 100000 020000 — «истинные» адрес и длина файла МДС;
- 312АИ; ответ: 017341 — контрольная сумма файла МДС (не совпадающая, как видите, с контрольной суммой в МСД, равной 177777).

5. Как перейти из МСД в ПМ без стирания экранного ОЗУ?

— 4АЩ100442 «СТОП».

6. Предположим, вы загрузили и запустили из режима МСД некоторую программу, например, ассемблер-систему. При работе с ней (или при запуске написанной в ней программы) получено сообщение: ЗВ003474. О чем это сообщение говорит и почему вы так думаете?

— Сообщение говорит только о том, что произошло зависание; число 003474 не несет никакой информации, так как работала программа пользователя, а не директивы МСД.

7. Постарайтесь выучить директивы МСД и порядок работы с ними наизусть — вам часто придется ими пользоваться.

(Продолжение следует)

В. П. Юров,
г. Москва

ГРАФИЧЕСКИЕ РЕДАКТОРЫ ДЛЯ БК-0010(.01)

Первым графическим редактором стал **GRED0**, разработанный О.Туйкиным в начале 1987 г. и предназначенный для создания спрайтов. Цветные рисунки размером 8x10 точек создаются в режиме «лупы»(увеличенное в несколько раз изображение). Сервисные возможности GRED0, особенно при сохранении рисунка, минимальны. Сейчас этот редактор уже морально устарел, но он во многом послужил стандартом для последующих разработок.

В настоящее время для БК существует более 20 графических редакторов, различающихся по назначению, разрешению (цветной или черно-белый режимы), способу хранения рисунка, уровню сервиса, качеству разработки и т.д. Они позволяют создавать полноэкранные изображения при художественном конструировании, в том числе для оформления документов и создания рисунков и заставок, чертежи и схемы, спрайты для игр, мультфильмов и оформления документации с иллюстрацией фаз движений или развития объекта.

Большинство редакторов написаны авторами для собственных нужд при создании заставок и спрайтов для игровых программ. Такие редакторы часто плохо оформлены, имеют минимальный сервис, работают неустойчиво, не учитывают достигнутый не только на других компьютерах, но и на БК уровень качества программ. Разобраться в работе такого редактора бывает сложно не только начинающему пользователю, но и опытному программисту. Ниже рассматриваются только те редакторы, которые могут быть рекомендованы для практического применения и которые достаточно совершенны по своему сервису.

Растровые экранные графические редакторы

Одной из наиболее известных разработок этого типа является графический редактор **БК-PAINT v1.0-89** (А. В. Бакерин). По принципам своего построения редактор соответствует стандартам, принятым для подобных программ на компьютерах типа IBM PC. На рис.1 представлен общий вид экрана рассматриваемого редактора: оконное меню для основных функций и режимов работы расположено в верхней части экрана (одно из этих меню «раскрыто»), меню из пиктограмм команд рисования — слева, меню толщины линий пера — в левом нижнем углу и меню фактур закрашивания — внизу.

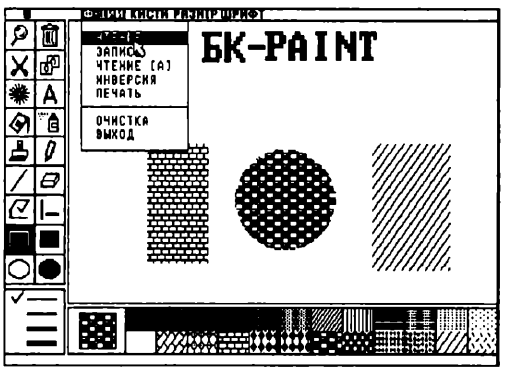


Рис. 1

Управление «пером» может осуществляться от клавиатуры (версия 1K), от джойстика (эта версия автору статьи неизвестна, но отмечена в документации к редактору) или от мышки (версия 1M). Редактор обеспечивает рисование точками, прямыми и ломаными линиями. Предусмотрены режимы закрашенных и незакрашенных кругов и прямоугольников, закрашивания замкнутых областей,

«разбрызгивания», стирания, размножения и переноса фрагментов рисунка, режим лупы, режим ввода текста и некоторые другие. Для ввода текста может использоваться стандартный шрифт БК и специальный. Размеры шрифта можно менять. Этот редактор с успехом может использоваться для создания чертежей и схем средней сложности, а также для детского рисования. Под рабочую область выделено 3/4 экрана, а рисунок выполняется при черно-белом разрешении. Редактор прост в освоении и доступен для неподготовленных пользователей.

К сожалению, меню редактора не продублированы клавишами, что значительно снижает оперативность работы с ними. Для того чтобы изменить режим, необходимо перевести перо из текущей точки к пиктограммам меню и снова вернуться на место. Для придания выразительности рисунку такие переключения осуществляются довольно часто, а это весьма утомительно. Снижают ценность редактора также возможность работы исключительно в черно-белом режиме и использование только 3/4 экрана. Наибольшую неприятность доставляет принятый в редакторе специальный формат записи рисунка: он делится на пять частей, каждая из которых определенным образом упаковывается и по отдельности (в виде 5 файлов) записывается на магнитофон. Записанный таким образом рисунок не может быть загружен обычным способом в экранную область ОЗУ БК или в другой редактор, а также распечатан существующими средствами. В некоторых вариантах редактора не предусмотрен вывод рисунка на принтер, а там, где он есть, рисунок выводится на принтер повернутым на 90° и только одного размера. Отмеченные недостатки значительно ограничивают область применения редактора.

В полноэкранном цветном редакторе **SCREDIT** (Марков) предусмотрено рисование точками и линиями, построение закрашенных и незакрашенных прямоугольников, кругов и эллипсов, перенос фрагментов с переворотом, зеркальным отображением, с изменением цветов, с увеличением и без, закрашивание замкнутых областей, ввод текста и псевдографики, режим лупы и некоторые другие, а также создание спрайтов. Размер пера регулируется. Управление им осуществляется от клавиатуры или джойстика. Запись рисунков может производиться в двух режимах — в обычном и уплотненном. Режимы работы редактора выбираются через небольшое меню, выполненное в виде пиктограмм, и детализируются с помощью клавиатуры. На рис.2 показан верхний правый угол экрана редактора, в котором расположено меню.

Рис. 2. Графический редактор SCREDIT. Фрагмент экрана

Замкнутые области рисунка или весь рисунок закрашиваются только одним из четырех основных цветов компьютера, и этот режим работает неустойчиво. Не совсем удобно, с недостаточными пояснениями осуществляются переключения режимов редактора, что затрудняет создание сложных рисунков.(Последнее, однако, сглаживается использованием линзы.) Недостаточен сервис для режима рисования спрайтов. Из-за слишком ограниченного объема памяти компьютера едва ли целесообразно разрабатывать универсальные графические редакторы, предназначенные для создания полноэкранных изображений и для спрайтов. В итоге неплохой редактор SCREDIT значительно уступает в сервисе гораздо менее удачным редакторам.

Существует еще несколько редакторов, аналогичных SCREDIT'у, например **PGM** и **PAINTER,** которые лучше него оформлены, имеют более развитое меню и предусматривают различные фактуры для закрашивания, но, к сожалению, они еще менее удобны в работе из-за неустойчивости реализации функций и частых зависаний.

Одна из версий популярного многоцветного графического редактора **GRAF36**, в котором были сделаны заставки для многих игр, была описана в журнале «Байтик», № 1 за 1992 г. На рис.3 представлен общий вид экрана редактора. В рабочей части экрана показана заставка, выполненная этим же редактором. На рис. 4 приведены примеры заставок для игр, сделанные в этом редакторе.

В новой версии редактора сохранены прежние возможности: изменение размеров

Рис. 3. Графический редактор GRAF36. Общий вид экрана

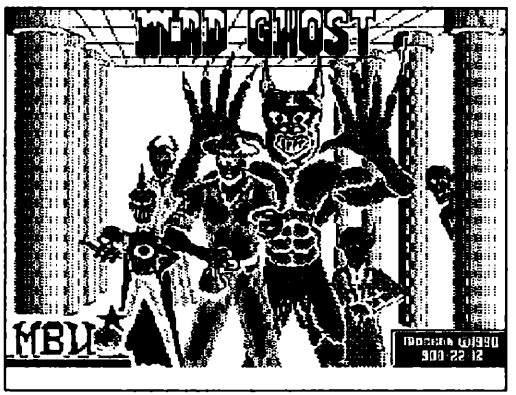


Рис. 4. Заставки к играм, выполненные в редакторе GRAF 36

и шага пера, рисование точками, прямыми и ломаными линиями, прямоугольниками, окружностями, закрашенными многоугольниками, закрашивание замкнутых областей рисунка и всего экрана, перенос с точностью до одной точки фрагментов изображения (как всех цветов фрагмента, так и их части), поворот фрагментов относительно оси X, Y или обеих осей и их размножение. Рисование и закрашивание осуществляются одним из 36 цветов (4 основных цвета компьютера и 32 дополнительных). В редакторе использовано более 70 команд для изменения режимов его работы, в том числе имеется возможность расширения палитры цветов. Все команды выполняются нажатием соответствующих клавиш. Управление пером может осуществляться от клавиатуры, джойстика или мышки. Максимальный размер рисунка составляет 256x240 точек, что соответствует величине телевизионного экрана без служебной строки. Запись рисунка осуществляется в обычном формате. Вывод его на принтер может производиться с помощью входящей в комплект редактора утилиты или обычными средствами, а также с помощью специальных пакетов обработки и вывода графики на принтер. Редактор предназначен для создания заставок к компьютерным играм, оформления документации и художественного конструирования. При использовании джойстика или мышки с редактором вполне могут работать дети, постепенно осваивая его команды. Однако особо высококачественные рисунки лучше выполнять при управлении пером от клавиатуры.

В последней версии редактора обеспечена его работа с различными дисковыми операционными системами, а также возможность ввода текста. Улучшены некоторые команды и ускорены некоторые операции.

Определенные неудобства при использовании редактора возникают при его освоении. Так, в нем отсутствует меню команд, и, прежде чем начать работать с ним, необходимо эти команды выучить. Отсутствие режима лупы также усложняет работу с редактором. Так как GRAF36 реализован частично в ассемблере, а частично в Бейсике, то для его работы с дисководом необходимо, чтобы контроллер обеспечивал подключение штатного ПЗУ с Бейсиком, что возможно не во всех контроллерах.

Векторные графические редакторы

От ранее рассмотренных программ принципиально отличаются графические редакторы серии **GRAF**. Существует несколько версий этого редактора. В отличие от рассмотренных растровых редакторов, в которых производится прямое необратимое изменение экранного ОЗУ, в векторных запоминаются команды, с помощью которых был получен рисунок, а также координаты опорных точек. При сохранении такого рисунка в виде файла последний содержит перечень команд и при его «отработке» изображение рисуется заново. Этот процесс можно сравнить с рисованием картинки при запуске Бейсик-программы, содержащей графические операторы.

В векторном графическом редакторе **GRAF4** (Н.М. Саттаров) предусмотрена возможность рисования точками, линиями, окружностями, дугами и прямоугольниками. Предусмотрено изменение размеров и шага пера, а также выбор шести размеров шрифтов, в том числе с вертикальной ориентацией текста. В редакторе возможно запоминание и запись в виде файла (библиотеки) до 45 простых рисунков, например элементов схем, с последующим вложением этих рисунков в другие. Рисунки из библиотеки при вложении могут быть повернуты на угол, кратный 90°, а их масштаб — изменен. Управление пером может осуществляться от клавиатуры, джойстика или мышки. Рисунок можно записать в файл в виде команд или в виде копии экрана (при подгрузке соответствующего драйвера экрана), а также вывести на печать (при подгрузке драйвера принтера). Редактор может работать как в цветном, так и в черно-белом режиме. Максимальный размер рисунка составляет 256x240 точек. В редакторе также имеется возможность создания небольших мультфильмов и их просмотра вне редактора.

Рис. 5

На рис.5 показан общий вид экрана редактора GRAF4. Этот редактор может быть рекомендован для создания чертежей средней сложности, особенно схем. Библиотеки с типовыми элементами для таких схем, в зависимости от сферы применения, могут быть единожды созданы в редакторе и затем записаны в виде файла-библиотеки для последующего использования. Например, можно создать библиотеку радиоэлементов и на ее основе вычерчивать радиосхемы. Для этого достаточно курсором указывать место, куда необходимо вывести тот или иной элемент. При этом, как отмечено выше, возможен поворот элемента и изменение его размеров. Кроме того, после внесения элемента в схему он специальной операцией может быть перемещен в другое место.

Редактор GRAF4 проигрывает из-за недостаточно продуманного сервиса и оформления. Отчасти причиной этого было желание совместить в одном редакторе два режима: экранный и спрайтовый. В то же время создание хороших спрайтов вне режима лупы (в редакторе GRAF4 этого режима нет) практически невозможно, да и сервис для создания спрайтов требуется совершенно иной, чем для экранного редактора. Тем не менее при некотором изменении сервиса и введении ряда функций данный редактор вполне бы мог стать лучшим для создания чертежей средней сложности, схем и печатных плат.

Графические редакторы спрайтов

После разработки первого графического спрайтового редактора GRED0 было предпринято много попыток создать более удобный редактор спрайтов, имеющий более широкие возможности. И хотя редакторы, превосходящие GRED0, были разработаны, они использовались в основном только самими авторами для создания спрайтов к разрабатываемым ими играм, и достаточного распространения и признания не получили в силу их малой надежности в работе и низкого сервиса. Среди таких редакторов более или менее стабильны в работе **MULTIPAINT, V003, GRAFRED.SW.**

В настоящее время лидером в рассматриваемой области является мультипликационный редактор спрайтов **Animatic** (В.В. Юров), способный удовлетворить наиболее взыскательного пользователя. Редактор поставляется в комплекте с документацией и примерами спрайтов и мультфильмов. Он предназначен прежде всего для облегчения труда программистов при создании компьютерных игр, но в силу своего сервиса и возможностей доступен и неподготовленным пользователям при оформлении документации с иллюстрацией фаз движений или развития объекта, а также всем, кто увлекается мультипликацией, в том числе детям от 5 лет и старше. В последней версии этого редактора кроме текущих улучшений расширен перечень операционных систем, в которых он может работать, и увеличен объем памяти, отводимый под спрайты.

Animatic отличает высокий уровень художественного оформления (художник М. Бабешко) и дружественный интерфейс. Редактор позволяет создавать, редактировать и просматривать в движении одновременно до 128 спрайтов. Размеры спрайтов составляют от 4x1 до 64x64 цветных точек. Четыре режима работы выбираются курсором из основного меню: работа со списком и банком спрайтов (рис.6а), создание и редактирование спрайтов (рис.6б), мультипликатор и работа с магнитофоном или дисководом. В каждом режиме предусмотрены подсказки. Пункты меню продублированы клавиатурой.

Рис. 6а. Графический редактор Animatic. Общий вид экрана в режиме работы со списком и банком спрайтов

На рис.6б видно, как создание и редактирование спрайта осуществляется в увеличенном масштабе. Под редактируемый спрайт отведена большая часть экрана. Спрайты с максимальным размером не умещаются в отведенной области экрана, но при подводе курсора к границам этой области их изображение перемещается в видимую часть. Одновременно в левом верхнем углу редактируемый спрайт выводится в натуральную величину. В режиме редактирования спрайты могут

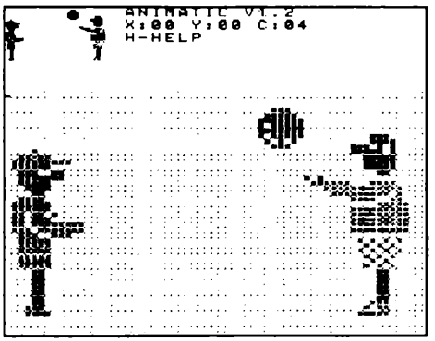


Рис. 6б. Графический редактор Animatic. Общий вид экрана в режиме создания и редактирования спрайтов

быть смещены относительно центра в любом направлении или повернуты. После редактирования предлагается выбрать способ «упаковки» спрайта: по горизонтали (строчка за строчкой) или по вертикали (столбец за столбцом), что дает возможность пользователю без дополнительных манипуляций выбрать наиболее удобный для его программы способ. Для тех же, кто использует Animatic для создания мультфильмов или иных целей, не имеет значения способ размещения, и поэтому на запрос компьютера можно выбрать любой ответ.

В режиме работы со списком и банком спрайтов (рис.6а) для программистов указываются адреса их размещения. В этом же режиме можно предварительно оценить спрайты в движении или их размножить. Последнее прежде всего необходимо для упрощения создания фаз движений. Для получения новой фазы достаточно занести соответствующей клавишей нужный спрайт в список дважды и отредактировать меняющиеся части рисунка. После того как созданы по крайней мере два спрайта, их можно предварительно оценить в движении, перемещая курсор по списку спрайтов. Строка, на которой находится курсор, инвертируется, а соответствующий спрайт индицируется в нижнем правом углу экрана. При перемещении курсора по списку спрайтов они поочередно друг за другом, со скоростью перемещения курсора, появляются на экране, создавая эффект движения.

После того как все необходимые спрайты созданы, их можно занести в любой последовательности с любым количеством повторов в банк спрайтов. Делается это также наглядно и просто: курсор устанавливается на нужный номер спрайта и нажимается клавиша «ПРОБЕЛ». При этом происходит как бы последовательное заполнение киноленты кадрами. Порядок кадров и количество их повторов устанавливаются такими, как это предполагается сделать в программе или в мультфильме. Просмотр мультфильма осуществляется в режиме мультипликатора. В этом режиме экспериментально подбираются и оцениваются необходимые скорости вывода и перемещения спрайтов в любом направлении по экрану. Полученные значения скоростей без дальнейших экспериментов могут быть использованы в программе, для которой предназначаются спрайты.

Созданные спрайты могут быть записаны в виде файла с блоком данных, если они предназначены для мультфильма, или без блока данных, если их предполагается использовать в программе или для иллюстрации фаз движений и изменений объекта в документации.

Вывод графики на принтер

Большинство графических редакторов предусматривают вывод графики на принтер, но в силу ряда причин, в том числе и из-за малой оперативной памяти компьютера, печать осуществляется только в одном из режимов принтера, с одним разрешением и одним размером. При этом печать допускается только на каком-то одном принтере. Такое решение, конечно, не может устроить большинство пользователей, поскольку перед ними стоят самые разнообразные задачи, и используют они различные принтеры, включенные по разным схемам. Безусловно, работая в редакторе, удобно тут же распечатать полученный рисунок или чертеж и сразу же его подправить, если это необходимо. Но даже на более мощных компьютерах для вывода графики на принтер часто используют специальные дополнительные программы, позволяющие учесть большинство потребностей пользователей.

На БК тоже имеются специальные пакеты и программы для обработки и вывода на принтер графической информации. Почти одновременно появились пакет **HCOPY6** (В.В. Юров) и программа **GPS** (Д. Сотченко). Эти программы во многом схожи, но по разному оформлены, имеют разные сервис и диапазон используемых принтеров.

В HCOPY6 (Hard Copy — «твердая копия») на различные принтеры, выбираемые из меню, могут выводиться полноэкранные изображения или любые фрагменты из них, а также спрайты, разработанные в Animatic'е

или других редакторах и даже непосредственно из игр, если они не имеют специальной упаковки. В HCOPY6 предусмотрен специальный режим, позволяющий найти в памяти место расположения спрайта и вывести его на экран, а затем на принтер. В качестве полноэкранных изображений могут использоваться рисунки, полученные в редакторах GRAF36, SCREDIT, GRAF4 и т.д., а также непосредственные копии экрана, полученные любым доступным способом и записанные в виде файла. Для выбора фрагментов рисунка (точность выбора — до одной точки)предусмотрена специальная рамка с изменяющимися размерами, перемещаемая во всех направлениях по рисунку. Выбранный фрагмент или весь рисунок, а также отдельные спрайты или группа спрайтов при выводе их на принтер могут быть отпечатаны в любой указанной пользователем части листа, с любым отступом от правого края или строго по центру, если применен режим автоцентровки. Размеры рисунка при выводе на принтер могут быть увеличены или уменьшены по горизонтали и вертикали (по отдельности или одновременно), а также, при необходимости, рисунок может быть инвертирован.

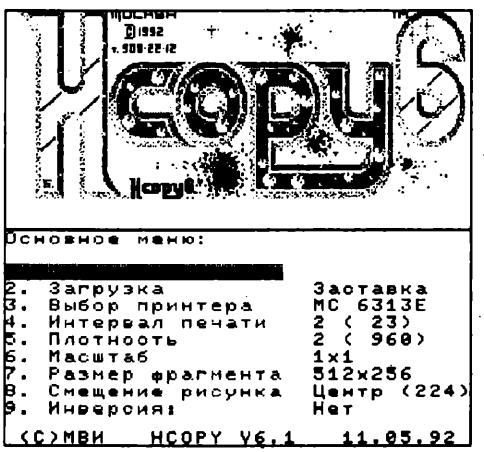


Рис. 7. Утилита HCOPY6. Общий вид экрана в режиме основного меню

На рис.7 показан экран с основным меню. В верхней части видна загруженная в HCOPY6 заставка от инструкции. Все режимы работы, кроме обработки спрайтов, выбираются курсором через меню, продублированное клавиатурой. При выборе принтера предусмотрена возможность небольшой коррекции его режимов. При этом принтер должен быть подсоединен по стандарту встроенного в компьютер драйвера, т.е. выполнять команды штатного Бейсика, что является критерием правильного подсоединения и установки его переключателей. HCOPY6 поставляется в комплекте с подробной документацией и примерами различных рисунков и спрайтов. Красочное художественное оформление документации и приложений выполнено М.Бабешко.

В GPS4 (Graphic Print System — «система графической печати») применен универсальный драйвер печати, работающий с рядом принтеров, и предложена оптимальная, по мнению автора программы, схема подключения принтера. Эта программа позволяет распечатывать с изменением масштаба полноэкранные рисунки или их фрагменты, а также спрайты, нарисованные в спрайтовом редакторе MULTIPAINT. При печати может быть задан отступ от правого края листа.

В программу также встроен небольшой графический редактор, позволяющий корректировать рисунки путем рисования точек, линий, закрашенных и незакрашенных прямоугольников, вывода текста двумя типами шрифтов, а также путем переворота рисунка на 90° и его инвертирования. В программе предусмотрена также возможность загрузки графических изображений из программ для ZX Spectrum. Кроме того, при использовании БК-0011М можно одновременно загрузить два рисунка и переносить фрагменты из одного в другой. Режимы работы программы выбираются путем набора команд с клавиатуры. Для основных режимов имеется подсказка, вызываемая по команде «Help».

Несмотря на «универсальность» драйвера, программа не работает, например, с достаточно распространенным и удобным принтером D100M, включенным по стандарту. Программа рассчитана для работы на БК-0011М, а на БК-0010(.01) она работает только в случае турбирования компьютера (повышения частоты).

На БК имеются также резидентные утилиты, позволяющие выводить копию экрана сразу на принтер. Такие утилиты прерывают исполнение программы и «застывший» экран выводят на принтер. Сервис в них самый минимальный, они чаще всего рассчитаны только на какой-то один принтер и один размер печати. Потребность в распечатке экрана возникает редко, и едва ли стоит ожидать

появления хорошего сервиса в рассматриваемых утилитах. Поэтому рекомендуется в случае необходимости вывода экрана на принтер вначале записать его в виде файла, а уже потом с помощью рассмотренных выше программ вывести на принтер. Для записи экрана в виде файла также существуют соответствующие утилиты. Например, представленные в статье копии экранов редакторов сделаны с помощью **SVSC.MVI** (В. В. Юров).

Использование графики с других компьютеров

Многие помнят, что в первых играх на БК (Lode Runner, Goat, Race и др.) были чернобелые заставки, но мало кто знает, что они были скопированы с игр для компьютера ZX Spectrum. В настоящее время многие игровые программы используют заставки и спрайты из программ ZX Spectrum. Существует несколько разработок, позволяющих не только загружать картинки с ZX Spectrum и записывать в формате БК, но и раскрашивать их в цветах БК. Достаточно устойчиво работают программы **SPECTRUM2, RDCLOADER** и **TRD-COPY.**

Программы SPECTRUM2 и RDCLOADER позволяют считать с магнитофона файл, содержащий заставку игровой программы ZX SPECTRUM, подобрать наиболее выразительные цвета и записать ее на магнитофон или дисковод в виде файла копии экрана БК. В дальнейшем этот файл может быть загружен в один из экранных редакторов для последующей обработки, а также распечатан на принтере. В рассмотренной выше программе GPS4 использован загрузчик от RDCLOADER (обе программы одного и того же автора), который позволяет сразу загрузить заставку от программы ZX Spectrum в GPS и вывести ее на принтер.

Программа TRD-COPY предназначена для работы только с дисководом и предпочтительно на БК-0011М. Программы от ZX SPECTRUM должны быть записаны на дискете в формате TRD-DOS. С помощью этой разработки можно не только считать заставки игр ZX-SPECTRUM и записать их в формате БК, но и скопировать спрайты из них для последующего использования в Animatic'e. Возможна также запись спрайтов в формате других редакторов.

Существуют и программы для перевода графических изображений с компьютера IBM PC в формат графики БК, например **KOI4.COM** и **IBMBK.EXE**. Эти программы работают только на IBM PC. Они позволяют переводить в формат БК графические изображения формата PCX (это один из наиболее распространенных форматов, принятых для графических изображений на IBM PC). При этом также возможна перекодировка цветов рисунка. Из-за разницы в разрешающей способности по горизонтали на IBM PC и БК графические изображения получаются искаженными, поэтому желательна их предварительная специальная обработка в различных графических редакторах IBM PC. Чтение полученных графических файлов для последующего использования может быть осуществлено на БК в среде ANDOS.

Заключение

В настоящее время на БК имеется достаточное количество графических редакторов, способных удовлетворить первоочередные потребности пользователей. Однако еще существуют значительные резервы для улучшения их сервиса и возможностей, особенно для редакторов, предназначенных для создания полноэкранных изображений. При этом целесообразна большая специализация графических редакторов по назначению: художественное конструирование, чертежи для машиностроения, различные схемы, печатные платы, спрайты.

Рассмотренные редакторы (кроме GRAF36) могут также работать на БК-0011М в режиме эмуляции БК-0010.

По всем вопросам (кроме ремонта), связанным с эксплуатацией компьютеров (БК, УКНЦ, ZX Spectrum, Atari XL/XE, Atari ST, Commodore 64/128, Amiga, IBM PC и совместимых с ним) и их программным обеспечением, обращаться по адресу: 127349, Москва, а/я 9, Юрову Вячеславу Петровичу. Условия можно узнать по телефону: (095) 908-22-12 с 10 до 21 часа ежедневно.

ГРАФИЧЕСКИЕ СРЕДСТВА ДЛЯ БК

(дополнение редактора)

Спектр графических средств для компьютеров серии БК настолько велик, что неудивительно, если в статье В.П. Юрова читатели не найдут названий тех или иных файлов (не говоря уже о том, что одни и те же программы могут иметься у разных пользователей под различными именами). Но хотя бесполезно пытаться описать все, что создано на сегодня

усилиями БКманов, все же следует рассказать о некоторых разработках, не упомянутых в рассмотренной статье. А для удобства построим изложение по тому же принципу, что и у В. П. Юрова.

Растровые экранные графические редакторы

Крупными недостатками программы **БК-PAINT** А.В. Бакерина являются, как и отмечено в статье В.П. Юрова, невозможность работы с диском (в некоторых дисковых системах) и нестандартный формат хранения рисунков. Однако оба эти недостатка могут быть устранены.

Так, работа с диском может оказаться невозможной по той причине, что порядковые номера частей рисунка записываются в именах соответствующих файлов в последней, 16-й, позиции. Поэтому в тех дисковых системах, где не предусмотрена «логическая обработка» имен (такая, как, например, в ANDOS), все пять частей рисунка записываются на диск под одним и тем же именем (последние символы просто-напросто «срезаются») и впоследствии не могут быть считаны программой в требуемом порядке. Силами редакции произведена доработка «клавиатурной» и «мышиной» версий программы БК-PAINT, позволяющая работать в любой дисковой системе, оснащенной перехватом EMT36 (можно также работать, как и прежде, с магнитофоном). В системе же ANDOS, обеспечивающей сохранение последнего символа в длинных именах, нормально работают как прежние, так и доработанные версии БК-PAINT. (Примечание. На диск, как и ранее на магнитную ленту, для каждого рисунка записываются пять коротких файлов. Так как на диске любой файл занимает только целое число секторов (в ANDOS — кластеров по 4 сектора), подобный способ хранения является несколько неэкономичным.)

Что же касается нестандартности формата записи рисунка, то и здесь все обстоит не так сложно. Автором разработана программа **UNPACK.V1**, обеспечивающая загрузку пятичастевого рисунка в формате БК-PAINT (имеются две версии UNPACK для «стандартных» и доработанных под диск вариантов этого графического редактора), его распаковку и запись рисунка на диск под тем же именем, но без номера части, в обычном для БК формате копии экранного ОЗУ. Затем этот рисунок можно загружать в другие графические редакторы, переносить на IBM, печатать на принтере с помощью HCOPY6, GPS4 и других подобных систем.

Векторные графические редакторы

Значительным преимуществом программ серии **GRAF** является возможность подгрузки драйверов печати копии экрана на принтер или сохранения ее на магнитной ленте либо диске. При редакции разработан большой набор подобных драйверов:

— комплект для печати на струйном принтере MC6312 с различными плотностями (разной шириной получаемых копий);

— драйвер-«Макси», позволяющий печатать копию экрана на MC6312 во весь лист формата А4;

— драйвер для печати на принтере D100 (исправленный драйвер из программы ANFOCAL8.D);

— драйвер копирования экрана на магнитную ленту или диск EKRAN_S, отличающийся от входящего в комплект к GRAF1 «старого» EKRANa тем немаловажным преимуществом, что при последовательном сохранении нескольких разных копий экрана они записываются в файлы с отличающимися (порядковыми номерами) именами. «Старый» же драйвер записывает все копии под одним и те же именем EKR, что приводит к путанице, а при работе в ANDOS и других дисковых системах, в которых не допускаются одинаковые имена файлов, и вовсе не позволяет сохранять более одной копии подряд;

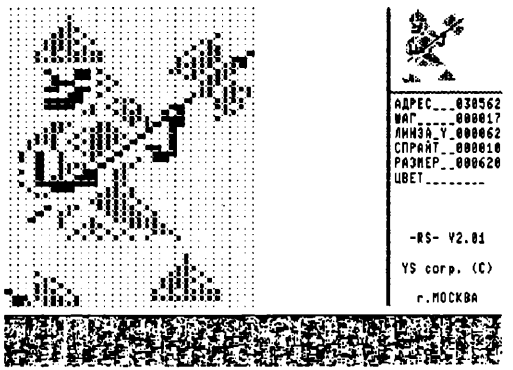
— драйвер GALine.DRV для записи части нарисованного с помощью GRAF изображения в виде спрайта.

Кроме того, имеется разработанный В. Кобяковым универсальный драйвер печати копии экрана на принтер MC6313, в котором предоставляется возможность выбора печатаемой части экрана с помощью «рамки» изменяемого размера и задания графического режима (горизонтальной плотности печати).

Графические редакторы спрайтов

Кроме описанных В.П. Юровым разработок, существует еще одна весьма любопытная программа — **RS.V2** (YS cogp). Она загружается в старшие адреса экранного ОЗУ (внизу экрана) и позволяет просматривать и редактировать по точкам в режиме линзы спрайты, содержащиеся в «теле» считанной в ОЗУ

пользователя программы. Соответственно, при этом становятся излишними такие операции, как извлечение («вырезка») спрайта из программы, работа с ним в любом из обычных спрайтовых редакторов и последующая «вставка» на место.



Просмотр графики на экране и ее вывод на принтер

К сожалению, в большинстве дисковых систем для БК отсутствуют «встроенные» средства для просмотра записанных в файлы копий экрана, подобные команде VIEW для просмотра текстов. Если же просто запустить такой файл «на выполнение», рисунок выводится на экран и тут же затирается дисковой системой или ее «Нортон-подобной» оболочкой так быстро, что рассмотреть его нет никакой возможности. Силами редакции разработана программа **GRAPVIEW**, позволяющая кроме обычных копий экрана просматривать картинки, упакованные ранее с помощью BKPACK17, а при необходимости — распаковывать их снова в обычную копию экрана. Последнее позволяет более экономно хранить копии экрана упакованными (объем таких файлов иногда можно уменьшить вдвоевтрое!), а при необходимости редактирования, печати на принтер или переноса на IBM вновь получить обычную копию экранного ОЗУ. Другая программа, **GRAPRINT**, имея те же функции просмотра и распаковки, позволяет распечатывать копию экрана на принтере MC6312 (из расчета две копии экрана на лист А4). Имеется также вариант GRAPRINT для печати на принтере D100.

Кроме того, в комплекте ANDOS есть программы «дискового Фокала» **ANFOCAL8** и **ANFOCAL8.D**, снабженные встроенным драйвером графической печати и «сопровождения» вывода текста. Версия с расширением «D» позволяет печатать графику на D100, однако графический драйвер в этой программе содержит ошибку, из-за которой нормальная печать копии экрана невозможна. (Силами редакции эта ошибка исправлена.)

И наконец, хотелось бы заметить, что замечание В. П. Юрова относительно невозможности работы с **GPS4** на БК-0010 без турбирования не совсем верно, ибо у автора этих строк ни разу не возникало проблем при работе с указанной программой на «нетурбированном» БК-0010.01.

Резидентные графические драйверы

Немного следует поговорить и о резидентных программах, позволяющих распечатывать копии экрана в процессе работы с той или иной программой. В качестве примеров (отсутствующих в статье В.П. Юрова) можно привести следующие:

— **ПС6312** (Ю.А. Зальцман) — позволяет печатать копии экрана на MC6312 из Бейсика и Фокала, а также обеспечивает «сопровождение» на принтере вывода текстов на экран в Бейсике, Фокале и режиме МСТД (коды символов одновременно передаются на экран и на принтер);

— **PRINSCR2** (Altec) — позволяет печатать копии экрана на принтерах MC6312, MC6313, Robotron, Epson LX/FX800 из Фокала и Бейсика, в том числе с предварительным редактированием изображения на экране;

— **GRAFIC.BIN** — небольшая программа, загружаемая вручную в Бейсике с помощью команды BLOAD и вызываемая для печати копии экрана на MC6312 с помощью USR.

Следует заметить, что все вышеперечисленные драйверы, собственно говоря, не являются резидентными в полном смысле этого слова. Они лишь просто заменяют нужные векторы прерываний на свои «в надежде», что другая программа не изменит эти установки и вообще не затрет «резидентный» драйвер в памяти.

Однако сегодня уже разработана настоящая резидентная программа — **SCREW ANIRAM-MIRIADA**, универсальный драйвер для ОС ANDOS с расширенным ОЗУ 16К. После инсталляции этот драйвер (имеется два варианта: для ANDOS v2.30 с SHELL8 и для ANDOS v2.50 с МАСТЕРом) все время присутствует в памяти, «заставляя» ANDOS постоянно обновлять свои

векторы прерываний и взамен дополняя его следующими полезными функциями:

— АР2+ВВОД или клавиша «PrintScreen» (прерывание по вектору 100) — запись копии экрана в файл <имя>.xxx, где xxx — номер от 000 до 999, а <имя> задается пользователем при инсталляции;

— АР2+РУС, АР2+ЛАТ — включение/выключение режима сопровождения диалога на принтере, включенном по стандарту Бейсика;

— добавляется устройство P: — «принтер». Из любой программы, не имеющей собственного вывода на принтер, но располагающей функцией записи на магнитофон/диск, можно распечатать нужный текст, выведя его на устройство P:.

Отметим также, что драйвер ANIRAM оснащен перехватом EMT14, обеспечивающим сохранение значений вектора EMT-прерывания, и сохраняет «живучесть» до тех пор, пока не будет деинсталлирован пользователем (либо пока не будет поврежден ANDOS или затерта область памяти с адресами 155000-157777). Даже если другая программа с помощью оператора MOV изменяет векторы @#30 и @#100, после рестарта ОС ANDOS драйвер вновь «готов к бою».

Файловая совместимость с другими компьютерами

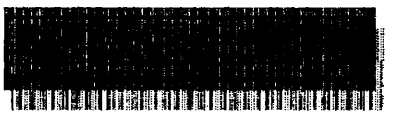
В статье В.П. Юрова достаточно подробно рассказано о программах для переноса графики на БК с IBM- и Синклеро-подобных компьютеров. Однако существует необходимость и обратного переноса — с БК на IBM. Наиболее часто это приходится делать для печати копии экрана, если нет своего принтера, подключенного к БК. Другой вариант — перенос копий экрана с БК для помещения их в публикации (при верстке с помощью Ventura или PageMaker) в качестве иллюстраций к описаниям программ. Для нужд редакции журнала «Информатика и образование» разработана программа **«УГМ SCREW»** («Универсальный графический монитор»), реализованная на IBM на Borland C++. Эта программа позволяет вывести рисунок, записанный на диск в виде копии экранного ОЗУ БК и перенесенный на IBM, например, через ОС ANDOS, на экран IBM для просмотра, «захватить» его с помощью резидентной программы перехвата прерывания по клавише «PrintScreen» (программы типа PZP) и сохранить рисунок на диске в одном из общепринятых для IBM графических форматов (PCX, TIFF и т.п.) либо распечатать на принтере (PZP позволяет в широких пределах варьировать качество печати, размер и положение твердой копии на листе, обеспечивает возможность подбора оттенков серого для различных цветов на экране и печать выбранной его части).

От аналогичных разработок «УГМ SCREW» отличается следующими преимуществами:

— максимально возможный размер изображения с БК на экране IBM (до 2/3 площади) с сохранением его пропорций;

— возможность изменения непосредственно в момент вывода графической картинки с БК режимов «цветной-монохромный» и «прямой-инверсный». В цветном режиме изображение выглядит как на подключенном к БК цветном мониторе, в монохромном — как на черно-белом;

— возможность предварительной закраски остающихся справа и снизу (из-за различной разрешающей способности экранов БК и IBM по вертикали и горизонтали) полей в один из пяти стандартных цветов (черный, белый, красный, зеленый и синий), таким образом частично «маскируя» наличие этих полей.



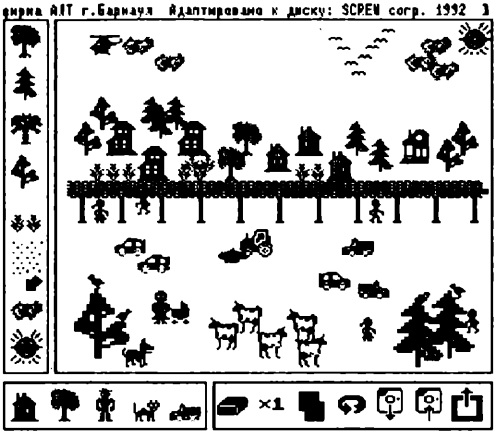
```
This programm shows a graphic  pictures from BK-0010 to IBM
PC/AT/XT (VGA videomode). Use PZP programm (with key /W) for
transfering picture in graphic formats: .PCX, .TIF, etc. and
for printing this picture at your printer.

File name (<.> - exit): ekran.bkg


 Komnand keys:
<C>olor, <M>onochrome, <I>nvert - output graphic methods;
<B>lack, <W>hite, <R>ed, <G>reen, b<L>ue - background colors;
<Q>uit - close graphics output.
```

Графические «конструкторы»

Имеется достаточно обособленный класс графических программ, предназначенных для «игрового обучения» младших школьников. Одной из наиболее известных является программа «Компьютерный мир» на Фокале, входящая в стандартный комплект, прилагаемый ранее к БК при продаже. Эта программа позволяла рисовать на экране несложные пейзажи, составляя их из стандартных «кирпи-



чиков» — рисунков (дом, дерево, облако и т.п.), размещая их в произвольных местах экрана. Однако программа «Компьютерный мир», при всех ее положительных чертах, сегодня не может считаться лучшей. Ей на смену пришла аналогичная разработка фирмы ALT из Барнаула, реализованная в кодах.

Программа «графический конструктор» оснащена удобными меню пиктограмм-примитивов, подобными используемым в БК-PAINT. Меню в левом нижнем углу экрана содержит пиктограммы «классов» объектов «компьютерного мира» — «здания», «растения», «люди», «животные», «транспорт». После выбора конкретного «класса» слева появляется набор примитивов, которые можно «извлекать» из меню и «впечатывать» в рисунок. Меню операций внизу справа позволяет стереть ненужное («ластик»), задать масштаб при выводе примитива в рабочую область тем же, что и в меню слева, или вдвое большим, установить режим наложения примитивов («прозрачный/непрозрачный»), получить зеркально отраженные примитивы, а также обеспечить чтение/запись созданных рисунков на диск и выход из программы. Указанная программа очень проста в эксплуатации и удобна для самых маленьких пользователей БК. Однако эта барнаульская разработка, как и большинство других программ фирмы ALT, рассчитана на работу с КУВТ86: операции чтения/записи рисунка на диск производятся путем обращения по последовательному каналу к РМП (рабочему месту преподавателя). Кроме того, при попытке выхода из программы посредством соответствующей пиктограммы меню операций на «индивидуальных» БК происходит зависание.

Силами редакции произведена доработка барнаульского «графического конструктора» и его адаптация к диску. Новая версия данной программы (**GRFCON.SCW**) позволяет осуществлять запись созданных рисунков на магнитофон или на диск (если дисковая система оснащена перехватом EMT36) с возможностью последующего чтения с магнитофона (диска). Рисунок записывается в виде копии экрана вместе с левым меню примитивов (при необходимости его нетрудно убрать в другом графическом редакторе). Зависание при выходе из программы также устранено.

Другой интересной программой подобного рода является **FOTOROBOT**, также разработанная фирмой ALT с помощью ею же созданной системы AVTOR для написания обучающих, демонстрационных, игровых и других программ, не прибегая к языкам программирования.

Программа FOTOROBOT позволяет «собрать» на экране «фоторобот» из предлагаемых в «раскрывающихся» меню деталей — глаз, носов, ртов и причесок. И хотя ее возможности весьма ограничены, можно думать, что эта программа доставит немало веселых минут пользователям БК, и не только самым маленьким.

По вопросам приобретения описанных программ обращайтесь в редакцию журнала по телефону: 151-19-40. E-Mail: mail@infoobr.msk.su

Д. Ю. Усенков,
г. Москва

ГЕНЕРАТОР УЗОРОВ ДЛЯ ВЯЗАНИЯ

Предлагаемая программа, видимо, будет интересна многим женщинам (да и мужчинам), занимающимся вязанием. Фактически эта программа представляет собой машинный калейдоскоп с поистине неисчерпаемым запасом узоров.

При запуске программы на экран выводится запрос количества цветов (2 или 3) узора. Трехцветные узоры имеет смысл «заказывать» только при использовании цветного монитора (на черно-белом экране они смотрятся плохо) и для машинной вязки, однако трехцветные узоры более разнообразны. После выбора количества цветов на экране заготавливаются три панели для вывода узоров, состоящих из повторяющихся элементов 4x4. Затем БК ждет ввода символа <Д> или <Н>. <Д> вызывает вывод на экран новой «порции» узоров, а <Н> — выход из программы. В данном варианте программы не предусмотрен вывод узоров на принтер, но можно распечатать содержимое экрана с помощью программ типа PRINTSCREEN.


```
10  '  ========================
20  '  !   ГЕНЕРАТОР УЗОРОВ
30  '  ! Усенков Д. МОСКВА  1988
40  '  ========================
60  DIM A(4,4)'УЗОРЫ ДЛЯ ВЯЗАНИЯ
70  CLS
80  ? AT(3,8);CHR$(156);"УЗОРЫ
    ДЛЯ ВЯЗАНИЯ.";CHR$(156)
90  ? AT(5,10);"Сколько цветов?"
100   LOCATE 5,12
110   INPUT " <2 или 3>";C
120   CLS
130   I0=10
140   FOR I=15 TO 90 STEP 5
150   J0=10
160   FOR J=15 TO 50 STEP 5
170   LINE (I0,J0)-(I,J),3,B
180   J0=J
190   NEXT
200   J0=60
210   FOR J=65 TO 100 STEP 5
220   LINE (I0,J0)-(I,J),3,B
230   J0=J
240   NEXT
250   J0=110
260   FOR J=115 TO 150 STEP 5
270   LINE (I0,J0)-(I,J),3,B
280   J0=J
290   NEXT
300   I0=I
310   NEXT
320   ? AT(14,2);"1";AT(14,7);
      "2";AT(14,12);"3"
330   FOR K=1 TO 3
340   FOR I=1 TO 4
350   FOR J=1 TO 4
360   A(I,J)=INT(RND(1)*C)
370   NEXT J,I
380   J0=12+50*(K-1)
390   J1=J0
400   I0=12
410   L=1
420   M=1
430   FOR I=I0 TO I0+15 STEP 5
440   FOR J=J0 TO J0+15 STEP 5
450   PAINT (I,J),A(M,L),3
460   L=L+1
470   IF L>4 THEN L=1
480   NEXT
490   M=M+1
500   IF M>4 THEN M=1
510   NEXT
520   I0=I0+20
530   IF I0>80 THEN 540 ELSE 410
540   J0=J0+20
550   IF J0>J1+25 THEN 560
      ELSE 400
560   NEXT
570   ? AT(3,22);"ЛИСТАТЬ ДАЛЬШЕ?
      <Д или Н>."
580   CH$=INKEY$
590   IF CH$<>"Д" AND CH$<>"Н" AND
      CH$<>"D" AND CH$<>"N" GOTO 580
600   IF CH$="Н" OR CH$="N" GOTO 630
610   ? AT(2,22);" ОДНУ МИНУТУ,
      ПОЖАЛУЙСТА !  "
620   GOTO 330
630   CLS
640   ? AT(7,10);"ДО СВИДАНИЯ !"
650   ? AT(7,12);"ЖЕЛАЮ УСПЕХА!"
660   END
```

Д. Ю. Усенков,
г. Москва

БИБЛИОТЕКА ГРАФИЧЕСКИХ ФУНКЦИЙ ДЛЯ АССЕМБЛЕРА БК-0010(.01)

Как известно, написать действительно хорошую игровую программу для БК-0010 можно только на ассемблере (это, конечно, относится и к другим программам — «системным», расчетным и т.п., но игры наиболее яркий пример). Высокая скорость работы получаемой после трансляции программы в машинных кодах позволяет реализовать достаточно динамичные игры при хорошем оформлении. Но в ассемблере, в отличие от языков высокого уровня (Фокал, Бейсик), пользователю предоставляется лишь ограниченный набор простейших команд. Что же касается более сложных операций (плавающая арифметика, работа со строковыми переменными, файловый обмен и др.), то для них каждому программисту приходится писать свои подпрограммы реализации. Вот, например, графика, без которой трудно сделать красивую игру. В Бейсике для этого есть почти все: PSET, lINE, POINT, CIRCLE, PAINT... А в ассемблере? Только EMT30 — поставить точку, да EMT32 — рисовать линию. (В БК-0011 набор «ассемблерной» графики, конечно, богаче, но речь сейчас не о ней.)

Однако из создавшегося трудного (особенно для начинающих программистов на ассемблере) положения есть выход. Большинство существующих сегодня на БК трансляторов с ассемблера имеют весьма полезную функцию — подгрузку к создаваемой кодовой программе внешних объектных модулей. (К сожалению, специфика работы с магнитофоном делает использование этой функции не слишком удобным, поэтому многие вообще избегают пользоваться ею. Но если к БК подключен дисковод, указанная функция раскрывает все свои возможности.)

Что же дает нам ее использование? Мы можем создать достаточно полный набор подпрограмм на ассемблере, реализующих операции некоторого класса, оттранслировать полученный текст, записать на диск (или на магнитофон) созданный объектный файл и затем подгружать его к любой ассемблерной программе, в которой требуется использовать нужные операции. Фактически мы имеем дело с идеей создания подгружаемых библиотек стандартных функций, широко применяемых на более мощных компьютерах (например, в языках Си или Паскаль на IBM-совместимых ПЭВМ).

Основным преимуществом такого подхода является простота использования библиотеки. Пользователю достаточно в своей ассемблерной программе написать команду JSR PC, <имя>, где <имя> — взятая из таблицы-подсказки метка соответствующей подпрограммы, и записать с помощью MOV нужные значения в регистры процессора. После компиляции достаточно вызвать команду подгрузки нужного объектного модуля, и все требуемые подпрограммы будут автоматически «привязаны» к их вызовам в основной программе. И второе преимущество: чтобы написать подпрограммы библиотеки, реализующие нужные функции, требуется значительный опыт и время. После этого готовый файл-библиотеку может использовать даже начинающий пользователь, практически не затрачивая времени на повторное «изобретение велосипеда». Заманчив и такой вариант: написанные различными авторами ассемблерные листинги могут быть опубликованы в журнале, составив в конечном счете полную библиотеку подгружаемых функций для ассемблера. А читатели, набрав листинг и откомпилировав его в объектный модуль в имеющемся у них трансляторе, получат возможность писать действительно высококачественные программы,

причем основной «мыслительный потенциал» программиста в этом случае будет направлен именно на творчество, а не на разработку того, что уже и так существует.

Недостатком применения библиотек на БК-0010 является прежде всего то, что при их подгрузке к основной программе подшиваются все содержащиеся в объектном модуле машинные коды подпрограмм, тогда как реально может требоваться всего одна-две из них. Для БК-0010 с ее «куцей» памятью это довольно расточительно, но что делать — за удобства приходится платить, в данном случае длиной получаемых кодовых файлов. Впрочем, этот недостаток не столь уж и страшен — при необходимости можно «урезать» ассемблерный листинг библиотеки, оставив в нем только необходимые подпрограммы, а затем вновь перекомпилировать объектный файл под другим именем.

Итак, вашему вниманию предлагаются два ассемблерных листинга из набора разработанных автором подгружаемых подпрограмм: библиотека графических функций и подпрограмма закрашивания фона экрана.

Синтаксис и доступ к функциям

Предлагаемые листинги написаны на ассемблере в стандарте транслятора M18. Допускаются метки длиной до шести символов, причем идентификаторы, начинающиеся с нецифрового знака, являются «глобальными» (т.е. метка в любом месте любого модуля доступна пользователю безо всяких ограничений, соответственно во избежание ошибок в работе получаемой кодовой программы разные метки не должны иметь одинаковые имена), а начинающиеся с цифры — «локальными» (т.е. в «блоках» листинга, разделенных хотя бы одной «глобальной» меткой, цифровые могут повторяться, соответственно цифровая метка из одного такого «блока» не доступна из другого). По этой причине желательно каким-либо образом выделить метки, используемые в библиотеках в качестве имен подпрограмм и ячеек для хранения данных, чтобы библиотеку можно было подгружать к любой ассемблерной программе, не утруждая себя заботами о возможном «перекрытии» имен. В предлагаемом листинге графических подпрограмм признаком «библиотечной» метки является первый символ «%». Так что, чтобы быть уверенным в отсутствии возможных «коллизий», не используйте в своей ассемблерной программе имена, начинающиеся с «%», кроме как для обращения к библиотеке.

Доступ к подпрограммам и ячейкам библиотеки осуществляется в полном соответствии со стандартами ассемблера. Подпрограммы вызываются командой JSR R7,<метка> (хотя в некоторых случаях в качестве регистра связи вместо R7 может быть использован и другой). Данные для подпрограмм могут быть переданы либо через регистры, либо через специально зарезервированные в библиотечном модуле ячейки. Занести же в эти ячейки нужные данные можно, просто обратившись по указанным для них меткам.

Отдельно следует поговорить о «табличных» данных. Если в библиотечном модуле имеется массив значений (в предлагаемой графической библиотеке примером является палитра цветов), можно обеспечить доступ к отдельной его ячейке, получив с помощью соответствующей подпрограммы адрес начала таблицы и вычислив адрес элемента массива с нужным индексом по несложной формуле: <адрес элемента>=<адрес начала>+<длина каждого элемента>*<индекс>. Можно использовать и более сложный способ, например MOV @#<метка>+2,R0 (доступ ко второй ячейке от начала массива), если это позволяется в имеющемся у вас трансляторе. Если же стандарт транслятора допускает две и более разных меток для одной и той же ячейки (как в M18), возможен и третий вариант: пометив, помимо метки начала массива, каждую его ячейку собственным именем, можно обеспечить возможность «двойственного» доступа к ним — как по их «собственным именам», так и по имени всего массива и индексу элемента.

Что же касается возврата данных из вызванной подпрограммы, то здесь возможны четыре варианта:

— данные возвращаются в регистрах (регистре), причем в случае ошибки в тех же регистрах возвращается специально оговоренное число — признак (значения для признаков ошибок выбираются таким образом, чтобы их легко было отличать от правильных значений. Например, если возвращается значение цвета точки, то в качестве признака ошибки можно выбрать отрицательное число);
— в регистре возвращается указатель
— значение адреса ячейки или начала массива (как, например, в подпрограмме %GETPL графической библиотеки);
— подпрограмма не возвращает значений, но в одном из регистров ей записывается «код завершения»: ноль, если подпрограмма выполнена без ошибок, либо ненулевое число — код ошибки (возможен и вариант, когда один из регистров выполняет роль «индикатора ошибок», а в других возвращаются данные);
— подпрограмма вообще ничего не возвращает (например, когда ошибка невозможна), в этом случае значения регистров, как правило, сохраняются.

Состав графической библиотеки

Функции, имеющиеся в предлагаемой графической библиотеке, позволяют реализовать операции, аналогичные графическим операторам Бейсика, в режиме 32 символа в строке (цветной). При этом пользователю доступна палитра из 16 цветов, из которых четыре (красный, зеленый, синий и черный) являются стандартными, а один («пользовательский») может быть изменен в процессе работы программы. (Впрочем, при желании можно задать и свою палитру, а затем при необходимости вернуться к стандартной.)

В библиотеку входят следующие функции:

— рисование и стирание точки;
— рисование линии по координатам концов;
— рисование прямоугольника по диагональным вершинам;
— определение цвета точки с заданными координатами;
— закраска замкнутой области;
— получение адреса палитры;
— получение адреса цвета в палитре;
— установка палитры;
— установка нового цвета пользователя.

Подпрограммы, реализующие эти функции, написаны «под влиянием» и «по мотивам» соответствующих подпрограмм, прошитых в ПЗУ Бейсика. (Отказ от прямого обращения к ним позволяет обеспечить независимость полученной программы от конфигурации БК.)

Возможно, кому-то покажется нерациональным то, что функции рисования и стирания точки и вычерчивания линии частично дублируют возможности EMT30 и EMT32. Однако эти функции занимают в объектном модуле совсем немного места, а использование их вместо EMT дает некоторые преимущества.

Следует отметить, что в данном варианте библиотеки не реализована операция рисования окружности. Причина в том, что это требует сложных для ассемблера БК расчетов, а значит, объем библиотеки рисковал вырасти почти вдвое. (Возможно, кто-либо из читателей предложит редакции свой, более простой листинг подпрограммы для построения окружности. — *Прим. ред.)*

Описание подпрограмм

1) Рисование точки:

Имя: %PSET.

Входные данные: R1, R2 — координаты точки (X,Y); R3 — двухбайтное слово, коди-

рующее цвет (например, 125252 — зеленый).

Выходные данные: нет.

2) Стирание точки:

Имя: %PREST.

Входные данные: R1, R2 — координаты точки.

Выходные данные: нет.

Примечание: точка с заданными координатами закрашивается в цвет фона, т.е. эта функция эквивалентна %PSET с R3=<цвет фона>.

3) Рисование линии:

Имя: %LINE.

Входные данные: (R1,R2), (R3,R4) — координаты концов отрезка; R5 — двухбайтное слово, кодирующее цвет.

Выходные данные: R0=0 — линия начерчена без ошибок; R0=−1 — ошибка: не включен цветной режим (32 символа в строке).

Примечание: данная функция позволяет сразу указать координаты обоих концов отрезка и цвет рисования, тогда как при использовании EMT32 для этого (в общем случае) дополнительно требуются EMT16 для задания цвета и EMT30 для задания координат начала отрезка.

4) Рисование прямоугольника:

Имя: %RECT.

Входные данные: (R1,R2), (R3,R4) — координаты концов диагонали прямоугольника; R5 — цвет рисования (два байта).

Выходные данные: R0=0 — без ошибок; R0=−1 — нецветной режим.

Примечание: данная функция аналогична оператору LINE (R1,R2)-(R3,R4),R5,B в вильнюсском Бейсике.

5) Определение цвета точки:

Имя: %POINT.

Входные данные: R1, R2 — координаты точки.

Выходные данные: R0 — номер цвета; R0=−1 — указаны координаты точки за пределами экрана.

Примечание: функция возвращает номер цвета указанной точки — число от 1 до 4 (1 — красный, ... 4 — черный). Нестандартные цвета с номерами в палитре от 5 до 16 с помощью этой функции не определяются.

6) Закрашивание замкнутой области:

Имя: %PAINT.

Входные данные: R1, R2 — координаты начальной точки; R3 — номер цвета границы (1—4); R4 — номер цвета закрашивания из палитры (0—16).

Выходные данные: R0=0 — без ошибки; R0=−1 — ошибочный номер цвета границы; R0=−2 — ошибочный номер цвета закрашивания; R0=−4 — нецветной режим.

Примечание: алгоритм реализации закрашивания представляет собой модернизацию подпрограммы PAINT из ПЗУ Бейсика и аналогичен подпрограмме 16PAINT (см. ИНФО №4, 1993).

7) Возврат адреса палитры:

Имя: %GETPL.

Входные данные: нет.

Выходные данные: R0 — адрес первого байта палитры (метки %PALTR).

Примечание: данная функция позволяет получить абсолютное значение адреса метки начала палитры, который заранее не определен, так как неизвестен начальный адрес объектного модуля графической библиотеки при его прилинковке к программе пользователя.

8) Возврат адреса цвета:

Имя: %GETPC.

Входные данные: R1 — номер цвета в палитре 0—16 (0 — последний использованный).

Выходные данные: R0 — адрес первого из четырех байтов, кодирующих цвет с заданным номером; R0=0 — задан ошибочный номер цвета.

Примечание: данная функция возвращает адрес первого байта из четырех, кодирующих указанный цвет. Если в R1 задан

ноль, в качестве номера цвета используется значение, использованное в какой-либо предыдущей операции.

9) Установка палитры пользователя:

Имя: %SETPL.

Входные данные: R0 — адрес начала новой палитры (ноль — устанавливается стандартная палитра).

Выходные данные: R0 — адрес установленной в данный момент палитры.

Примечание: с помощью этой функции можно изменить всю палитру цветов. Использованные ранее цвета из состава прежней палитры на экране не изменяются. В качестве входных данных в R0 должен быть задан адрес массива из 30 машинных слов, пары которых кодируют цвета (примером является стандартная палитра, входящая в состав библиотеки). Данная функция не производит переписывания стандартной палитры новыми цветами, она лишь указывает, что при всех последующих операциях с палитрой следует использовать массив значений кодов цветов с заданным начальным адресом. При необходимости возврата к стандартной палитре нужно задать R0=0 (в R0 в данном случае возвращается абсолютное значение адреса стандартной палитры аналогично функции %GETPL).

ВНИМАНИЕ! Данная операция должна быть выполнена прежде, чем будет вызвана какая-либо другая использующая палитру графическая функция (даже для стандартной палитры требуется вызов с R0=0).

10) Установка цвета пользователя (USER):

Имя: %SETUC.

Входные данные: R0 — указатель на 4 байта цвета пользователя.

Выходные данные: нет.

Примечание: эта функция переписывает в стандартной палитре значения двух машинных слов, кодирующих цвет пользователя (номер 16). Адрес первого из них должен быть задан в R0. ВНИМАНИЕ! Эта функция изменяет содержимое стандартной палитры!

11) Дополнительные метки:

%PALTR — метка начала массива кодировок цветов (стандартная палитра);

%A50 — адрес текущей палитры;

%A60 — подпрограмма вычисления абсолютного адреса метки, адрес возвращается в R0. Вызов:

JSR R7,%A60

.@<метка>

Подпрограмма закрашивания фона экрана

Эта подпрограмма не является частью графической библиотеки. Ее назначение — закрашивание всего экрана выбранным цветом из предложенного списка.

Эта операция может быть произведена вызовом функции %PAINT графической библиотеки после очистки экрана, однако данная подпрограмма реализует закрашивание быстрее и имеет небольшую длину, что делает ее использование более выгодным, если в программе требуется только подготовить цветной фон.

Вызов подпрограммы производится командой JSR R7,FONCOL , причем в R0 предварительно заносится номер цвета (список цветов приведен в комментариях к ассемблерному листингу), а в R1 — число-признак (0 — закрашивается весь экран, 1 — служебная строка не закрашивается).

Цвета кодируются в массиве с меткой %TABL2. Хотя в данной подпрограмме не предусмотрено средств для изменения таблицы цветов в процессе работы оттранслированной программы, это можно сделать путем перезаписи значений таблицы. Например,

MOV #123456,@#TABL2+4

MOV #123456,@#TABL2+6

позволяет изменить красный цвет на «полоски».

В подпрограмме не предусмотрено отслеживание рулонного смещения, поэтому перед ее вызовом экран должен быть «нормализован» двукратным нажатием АР2+СБР или двукратной подачей кода 140 (214 восьм.).

Подпрограмма %A13, как и %A60 графической библиотеки, возвращает абсолютное значение адреса метки, но не в R0, а в R5.

Листинг 1:

```
;     _GRAFIKA.LIB
; Библиотека графических функций.          SCREW(c)
;-------------------------------------------------------------------------------
;     Режим 32 символа в строке (цветной) обязателен !
; ------------------------------------------------------------------------------
;
;   Линия (R1,R2)-(R3,R4) цветом
;   R5 (2 байта закраски)
;     Ответ: R0=0 - O'K
;            R0= -1 - нецветной режим
;
%LINE:    TSTB @#40
          BNE 1
          MOV #-1,R0
          RTS R7
1:        MOV @#214,-(R6)
          MOV R5,@#214
          MOV #1,R0
          MOV R1,-(R6)
          MOV R2,-(R6)
          EMT 30
          MOV R3,R1
          MOV R4,R2
          EMT 32
          MOV (R6)+,R2
          MOV (R6)+,R1
          CLR R0
          MOV (R6)+,@#214
          RTS R7
;
;     Прямоугольник с диагональю
;   (R1,R2)-(R3,R4) цветом R5
;   (2 байта закраски)
;   Ответ: R0=0 - O'K
;          R0=  -1 - нецветной режим
;
%RECT:    TSTB @#40
          BNE 1
          MOV #-1,R0
          RTS R7
1:        MOV @#214,-(R6)
          MOV R5,@#214
          MOV R1,-(R6)
          MOV R2,-(R6)
          MOV R3,-(R6)
          MOV R4,-(R6)
          MOV #1,R0
          MOV @R6,R2
          MOV 2(R6),R1
          EMT 30
          MOV 6(R6),R1
          EMT 32
          MOV 4(R6),R2
          EMT 32
          MOV 2(R6),R1
          EMT 32
          MOV @R6,R2
          EMT 32
          MOV (R6)+,R4
          MOV (R6)+,R3
          MOV (R6)+,R2
          MOV (R6)+,R1
          MOV (R6)+,@#214
          CLR R0
          RTS R7
;
;     Точка (R1,R2) цвета R3 (2 байта закраски)
;   Пригодна и для 64 с/стр
;
%PSET:    MOV @#214,-(R6)
          MOV R0,-(R6)
          MOV R3,@#214
          MOV #1,R0
          EMT 30
          MOV (R6)+,R0
          MOV (R6)+,@#214
          RTS R7
;
;     Стирание точки (аналог %PSET,
;   для R3=цвету фона
;
%PREST:   MOV R0,-(R6)
          CLR R0
          EMT 30
          MOV (R6)+,R0
```

```
            RTS R7
;
;     Тест цвета точки (R1,R2)
;   Ответ: R0=цвет;
;   R0=-1 - точка за преде-
;   лами экрана
;
%POINT:   MOV #-1,R0
          CMP R1,#377
          BHI 1
          CMP R2,#357
          BHI 1
          JSR R7,%A55
          SUB #220,R0
1:        RTS R7
;
;     Закраска зоны начиная с точки
;   (R1,R2) до границы цвета R3
;   (R3 - номер цвета 1-4) цветом
;   R4 (R4- номер в палитре 0-16)
;
;     Ответ: R0=0 – O'K
;            R0= - 1 – ошибочен цвет
;                      границы
;            R0= - 2 – ошибочен цвет
;                      закраски
;            R0= - 4 – нецветной
;                      режим
;
%PAINT:   CMP R3,#1
          BLO 1
          CMP R3,#4
          BLOS 2
1:        MOV #-1,R0
          RTS R7
2:        CMP R4,#16
          BLOS 3
          MOV #-2,R0
          RTS R7
3:        TSTB @#40
          BNE 4
          MOV #-4,R0
          RTS R7
4:        MOV R3,-(R6)
          ADD #220,R3
          MOV R5,-(R6)
          MOV %A50,R5
          MOV R4,-(R6)
          ASL R4
          ASL R4
          ADD R5,R4
          MOV R4,%A51
          MOV (R6)+,R4
          MOV (R6)+,R5
          MOV R1,-(R6)
          MOV R2,-(R6)
          MOV @#214,-(R6)
          MOV R3,-(R6)
          MOV @#214,-(R6)
          MOV R4,-(R6)
          MOV R5,-(R6)
          MOV 14(R6),R1
          MOV 12(R6),R2
          MOV 10(R6),@#214
          MOV R0,R4
          MOV 6(R6),R5
          JSR R7,%A52
          BCS 26244
          MOV R1,R3
25712:    INC R1
          JSR R7,%A52
          BCC 25712
          MOV R1,R0
          MOV R3,R1
25726:    DEC R1
          JSR R7,%A52
          BCC 25726
          CLR R3
          MOV R3,-(R6)
          DEC R3
          MOV R0,-(R6)
          MOV R1,-(R6)
          MOV R2,-(R6)
          MOV R3,-(R6)
          NEG R3
          SR R7,%A53
25762:    ADD R3,R2
          INC R1
          JSR R7,%A52
          BCS 26062
          MOV R1,-(R6)
          MOV R1,-(R6)
26000:    DEC R1
          JSR R7,%A52
          BCC 26000
          SUB R1,@R6
          CMP #2,(R6)+
          BGE 26050
          SUB #6,R6
          MOV 6(R6),-(R6)
          ADD #10,R6
          DEC @R6
```

```
        MOV R1,-(R6)
        MOV R2,-(R6)
        MOV R3,-(R6)
        NEG @R6
        TST -(R6)
26050:  MOV @R6,-(R6)
        MOV R1,2(R6)
        MOV (R6)+,R1
        BR 26102
26062:  INC R1
        CMP R1,R0
        BGE 26230
        JSR R7,%A52
        BCS 26062
        MOV R1,-(R6)
        DEC @R6
26102:  INC R1
        JSR R7,%A52
        BCC 26102
        INC R0
        CMP R1,R0
        BLE 26152
        SUB #6,R6
        MOV 6(R6),-(R6)
        ADD #12,R6
        DEC R0
        MOV R1,-(R6)
        MOV R0,-(R6)
        MOV R2,-(R6)
        MOV R3,-(R6)
        NEG @R6
        BR 26214
26152:  DEC R0
        DEC R0
        CMP R0,R1
        BLE 26216
        SUB #6,R6
        MOV 6(R6),-(R6)
        ADD #12,R6
        INC R0
        MOV R0,-(R6)
        MOV R1,-(R6)
        MOV R2,-(R6)
        MOV R3,-(R6)
        SUB @R6,2(R6)
26214:  TST -(R6)
26216:  MOV R1,R0
        MOV (R6)+,R1
        JSR R7,%A53
        BR 25762
26230:  MOV (R6)+,R3
        BEQ 26244
        MOV (R6)+,R2
        MOV (R6)+,R1
        MOV (R6)+,R0
        BR 25762
26244:  MOV (R6)+,R5
        MOV (R6)+,R4
        MOV (R6)+,R3
        CMP (R6)+,(R6)+
        MOV (R6)+,R2
        MOV (R6)+,R1
        MOV #126262,R0
        JSR R7,%A54
        MOV R3,@#214
        MOV (R6)+,R3
        CLR R0
        RTS R7

%A52:   BIT R1,#177400
        BNE 26376
        BIT R2,#177400
        BNE 26376
        CMP R2,#357
        BGT 26376
        MOV R0,-(R6)
        JSR R7,%A55
        CMP R0,R5
        BEQ 26374
        CMP R0,R4
        BEQ 26374
        MOV (R6)+,R0
        CLC
        RTS R7
26374:  MOV (R6)+,R0
26376:  SEC
        RTS R7

%A55:   MOV R1,-(R6)
        MOV R2,-(R6)
        ASL R2
        ASL R2
        ASL R2
        ASL R2
        ASL R2
        ASL R2
        MOV R1,R0
        ASR R0
        ASR R0
        ADD R2,R0
        ADD @#204,R0
        BIC #140000,R0
```

```
            ADD @#202,R0
            MOVB (R0),R2
            BIC #177774,R1
            BEQ 26464
26456:      ASR R2
            ASR R2
            SOB R1,26456
26464:      BIC #177774,R2
            MOV #224,R0
            SUB R2,R0
            MOV (R6)+,R2
            MOV (R6)+,R1
            RTS R7
;
%A53:       MOV R0,-(R6)
            MOV R1,-(R6)
            INC R1
            CLR R0
            INC R0
            JSR R7,%A54
            MOV 2(R6),R1
            DEC R1
            JSR R7,%A57
            MOV (R6)+,R1
            MOV (R6)+,R0
            RTS R7
;
%A54:       MOV #104030,%A58
            JSR R7,%A59
            RTS R7
;
%A57:       MOV #104032,%A58
            SR R7,%A59
            RTS R7
;
%A59:       MOV @#214,-(R6)
            MOV R2,-(R6)
            BIC #177774,R2
            ADD %A51,R2
            MOVB @R2,@#214
            MOV (R6)+,R2
%A58:       EMT 30
            MOV (R6)+,@#214
            RTS R7
;
; Возвращает в R0 адрес палитры
;
%GETPL:     MOV %A50,R0
            RTS R7
;
;   Возвращает адрес в палитре
;   цвета 0-N (0-последний исполь-
;   зованный). Номер цвета задается
;   в R1. Каждый цвет в палитре за-
;   нимает 4 байта и возвращается
;   адрес первого байта из них
;   Ответ: R0 - адрес или
;   R0=0 при ошибке номера
;   цвета
;
%GETPC:     CMP R1,#16
            BLOS 1
            CLR R0
            RTS R7
1:          TST R1
            BNE 2
            MOV %A51,R0
            RTS R7
2:          MOV %A50,R0
            MOV R1,-(R6)
            ASL R1
            ASL R1
            ADD R1,R0
            MOV (R6)+,R1
            RTS R7
;
;   Установка палитры. В R0 зада-
;   ется новый адрес палитры поль-
;   зователя. Если R0=0, устанавли-
;   вается стандартная палитра, а
;   в R0 возвращается ее адрес
;
%SETPL:     TST R0
            BEQ 1
2:          MOV R0,%A50
            RTS R7
1:          JSR R7,%A60
            .@%PALTR
            BR 2
;
;С Т А Н Д А Р Т Н А Я  П А Л И Т Р А
%PALTR:     .#0
            .#0
            ; 0 ;                "базовый"
            .#177777
            .#177777
            ; красный          ; 1
            .#125252
            .#125252
            ; зеленый          ; 2
            .#52525
            .#52525
```

```
; синий          ; 3
.#0
.#0
; черный         ; 4
.#135756
.#135756
; желтый         ; 5
.#73735
.#73735
; фиолетовый     ; 6
.#114546
.#114546
; голубой        ; 7
.#31714
.#31714
; темно-красный  ; 10
.#21210
.#21210
;темно-зеленый   ; 11
.#10504
.#10504
;темно-синий     ; 12
.#60006
.#60006
; голубой-супер  ; 13
.#160016
.#160016
; желтый-супер   ; 14
.#150015
.#150015
; фиолет-супер   ; 15
.#165555
.#72753
```

```
                ; USER (польз)      ; 16
;
;   Установка в стандартную палит-
;   ру нового цвета пользователя
;   (USER). R0 - указатель на 4
;   байта цвета пользователя
;
;    ВНИМАНИЕ !!! Необратимо
;      изменяет стандартную палитру!
;
%SETUC:     MOV R1,-(R6)
            MOV #16,R1
            MOV R0,-(R6)
            JSR R7,%A60
            .@%PALTR
            ASL R1
            ASL R1
            ADD R0,R1
            MOV (R6)+,R0
            MOV (R0)+,(R1)+
            MOV (R0)+,(R1)+
            MOV (R6)+,R1
            RTS R7
;
%A60:       MOV @R6,R0
            ADD @R0,R0
            ADD #2,@R6
            RTS R7
%A50:       .#0
%A51:       .#0
;
```

Листинг 2:

```
;    FONCOL.OBJ
;    Входной параметр:          R0 - номер цвета
;  0 - черный;          4 - желтый;          10 - темно-зеленый;
;  1 - красный;         5 - фиолетовый;      11 - темно-синий;
;  2 - зеленый;         6 - голубой;         12 - голубой супер;
;  3 - синий;           7 - темно-красный;   13 - желтый супер;
;                    14 - фиолетовый супер;  15 - черный
;                    R1 - 0/1: закраска служ. строки - Д/Н
;  Подпрограмма закрашивания фона экрана      SCREW(c)
; ---------------------------------------------------------------------------
; ---------------------------------------------------------------------------
; Выходные параметры:   нет.
; ---------------------------------------------------------------------------
; Закрашивает весь экран выбранным цветом.
; ---------------------------------------------------------------------------
;
```

```
FONCOL: MOV R2,-(R6)
        MOV R3,-(R6)
        MOV R4,-(R6)
        MOV R5,-(R6)
        TST R1
        BEQ 1
        MOV #42000,R1
        MOV #170,R2
        BR 2
1:      MOV #40000,R1
        MOV #200,R2
2:      CMP R0,#14
        BLOS 3
        CLR R0
3:      ASL R0
        ASL R0
        JSR R7,%A13
        .@%TABL2
        ; адрес таблицы -> R5
        ADD R5,R0
        MOV (R0)+,R4
        MOV (R0)+,R0
10:     MOV #40,R3
11:     MOV R4,(R1)+
        SOB R3,11
        MOV #40,R3
12:     MOV R0,(R1)+
        SOB R3,12
        SOB R2,10
        MOV (R6)+,R5
        MOV (R6)+,R4
        MOV (R6)+,R3
        MOV (R6)+,R2
        RTS R7
%A13:   MOV (R6),R5
        ADD (R5),R5
        ADD #2,(R6)
        RTS R7
;
;       Таблица цветов:
%TABL2: .#0         ; черный
        .#0         ;
        .#177777    ; красный
        .#177777    ;
        .#125252    ; зеленый
        .#125252    ;
        .#52525     ; синий
        .#52525     ;
        .#167356    ; желтый
        .#135673    ;
        .#156735    ; фиолетовый
        .#73567     ;
        .#63146     ; голубой
        .#114631    ;
        .#146314    ; темно-красный
        .#31463     ;
        .#104210    ; темно-зеленый
        .#21042     ;
        .#42104     ; темно-синий
        .#10421     ;
        .#60140     ; голубой-супер
        .#3006      ;
        .#160340    ; желтый-супер
        .#7016      ;
        .#150320    ; фиолетовый-супер
        .#6415      ;
        .#0         ; черный
        .#0         ;
```

НЕСТАНДАРТНЫЕ ШРИФТЫ

А. В. Милюков,
г. Москва

О НЕТРАДИЦИОННОМ ИСПОЛЬЗОВАНИИ ЗНАКОГЕНЕРАТОРА БК

Предлагаемый материал рассчитан на пользователей, обладающих навыками работы на ассемблере БК и желающих более полно использовать возможности встроенного ПЗУ. Использование содержимого ПЗУ БЕЙСИКА и ФОКАЛА нельзя считать оправданным, так как в этом случае существенно ограничивается класс машин, на которых программа работоспособна. По этой причине автор интересовался содержимым именно системного ПЗУ.

Известно, что немалую часть ПЗУ монитора занимают драйвер магнитофона и знакогенератор. О работе первого написано много, и, наверное, он уже не является белым пятном для большинства пользователей и разработчиков. Работа же знакогенератора ПЗУ, по-видимому, гораздо меньше интересует программистов, поскольку крайне редко можно обнаружить его использование в программах (не считая копирования алгоритма).

По мнению автора, существуют две характерные тенденции:

— разработка собственного знакогенератора;

— нестандартные процедуры вывода на экран.

Первый способ предполагает большие затраты времени и сил на разработку знакогенератора с заданными параметрами, второй обычно требует некоторых ухищрений при стыковке со стандартным вводом-выводом. Компромиссным подходом является сочетание этих двух методов.

Чтобы оценить выигрыш в экономии памяти, достигаемый при отказе от собственного массива кодирования символов знакогенератора, умножим объем алфавита (минимум 64 символа) на количество байт, требуемых для одного символа (минимум 6). Даже для такого, весьма трудно читаемого на экране набора символов требуется порядка 400 байт ОЗУ. А если надо выводить еще и строчные буквы, и цифры, и хорошего качества? К тому же нужно учесть, что если для игровой программы лишние 2—3 кб терпимы, то системные разработки обычно ценятся обратно пропорционально их длине.

Вероятно, многие пользователи еще помнят копировщик COPIR, где для экономии места на экране, требуемого для хранения копируемого файла, были использованы символы меньшего, чем обычно, размера. Именно этот копировщик стал затем основой семейства UNIC, которые, по мнению автора, являются на сегодня одними из лучших в данном классе программ. Вместе с тем в них сохранился и главный недостаток прототипа — громадный, с точки зрения системщика, массив знакогенератора. Автор предпринял попытку доработки версии UNIC3 Баронова и Семенова с целью добавления директивы просмотра текстов в файлах. Идея состояла в том, что для изображения букв А, М, С, Е, Т, О можно использовать одинаковый шрифт в русском и латинском регистрах и сэкономить таким образом несколько десятков байт.

Ввиду варварского способа внесения корректив (в отладчике Прохорова) автор больших успехов не достиг, в чем могли убедиться пользователи версии 3+.

Следующий этап состоял в полном отказе от знакогенератора и использовании средств ПЗУ. Если посмотреть процедуру системного вывода в мониторе, то обнаружится, что для получения образа символа достаточно вывести в ОЗУ экрана «стопку» из десяти байт. В итоге получим символ, вид которого соответствует режиму 64 символа в строке. Однако первый и последний байты, т.е. на экране верхний и нижний, обычно равны нулю — это межстрочный промежуток и резерв для режима подчеркивания. Понятно, что при выводе на экран их можно пропустить. Среди восьми оставшихся байт тоже можно найти лишние без ущерба для качества изображения. Например, из букв О, Т, Л, Д, Г и других можно выбросить средний по высоте (третий по порядку) байт. Таким способом легко построить малоразмерные шрифты с полным набором знаков.

Для улучшения читаемости можно попытаться имитировать режим 32 символа в строке в смысле толщины линий символов. Эта проблема особенно актуальна для владельцев цветных видеомониторов. Автор решил эту проблему достаточно просто: символ рисуется дважды в пределах одного знакоместа со смещением двух изображений друг относительно друга на одну точку экрана (или на один бит). При этом одновременно достигается возможность выбора цвета символов путем сдвига второго изображения влево или вправо относительно первого. Чтобы избежать стирания первого рисунка вторым, при выводе используется команда ассемблера BIS. Вышесказанное иллюстрирует процедура OUT (листинг 1), за счет использования которой в версии UNIC4 автору удалось почти вдвое расширить набор директив копировщика при одновременном увеличении буфера на &O1200 и соответствующем сокращении длины собственно программы.

Другая подпрограмма (листинг 2) обеспечивает вывод пары увеличенных вчетверо символов с использованием стандартного знакогенератора, что может быть полезно, например, при выводе номера лабиринта в игре, числа пройденных уровней и т.п. Программа ориентирована на использование в режиме 32 символа в строке.

Идея программы состоит в распаковке каждого байта изображения в слово, в котором паре соседних точек (битов) соответствует один бит в исходном байте. Это слово затем копируется в двух соседних строках телевизионного растра, что обеспечивает двукратное увеличение по вертикали. Левый нижний угол изображения — общий для исходных и результирующих символов. В качестве примера в приведенном листинге показан вывод на экран числа 35, но оно может быть произвольным. Благодаря использованию ОЗУ экрана в качестве буфера преобразования выполняются достаточно быстро.

Листинг 1:

```
; ПОДПРОГРАММА ВЫВОДА НА ЭКРАН СИМВОЛА МЕНЬШЕЙ ВЫСОТЫ
; С УТОЛЩЕННЫМИ ЛИНИЯМИ
;   АВТОР: МИЛЮКОВ А.В.
OUT:    JSR R4,@#110346     ; сохранить все регистры
        BIC #177400,R0      ; выделить младший байт
        CMP R0,#200         ; вычислить новый код для символа
        BLOS 1
        SUB #40,R0
1:      SUB #20,R0
        BGT 2
        MOV #20,R0          ; непечатаемый код заменить
                            ; пробелом
2:      ASL R0              ; умножить новый код на десять
```

```
        MOV R0,R2
        ASL R0
        ASL R0
        ADD R0,R2
        ADD #112036,R2      ; прибавить адрес начала
                            ;знакогенератора в ПЗУ
        MOV #6,R0           ; счетчик числа выводимых байт
        MOV @#160,R1        ; адрес левого верхнего угла
                            ;символа на экране
        JSR PC,PRI          ; вывод верхней части символа
        INC R2              ; пропуск одного байта
3:      JSR PC,PRI          ; вывод оставшейся части символа
        SOB R0,3
        INC @#160           ; сдвиг курсора вправо
        JSR R4,@#110362     ; восстановить регистры из стека
        RTS PC
;
PRI:    INC R2              ; сдвиг указателя на один байт
                            ;в массиве кодирования символа
        MOVB @R2,@R1        ; вывод байта на экран
        ASRB @R1            ;и его сдвиг вправо
        BISB @R2,@R1        ; вывод байта-дубля
        ADD #100,R1         ; переход к следующей TV-строке
        RTS PC
```

Листинг 2:

```
; ПОДПРОГРАММА ВЫВОДА НА ЭКРАН ПАРЫ СИМВОЛОВ
;       4-КРАТНОЙ ВЕЛИЧИНЫ

BIG:    MOV #NUMBER,R1      ; адрес выводимого текста
        MOV #2,R2           ; длина текста (два символа)
        EMT 20
        MOV @#160,R2        ; адрес изображения первого символа
        SUB #4,R2
        MOV R2,R1
        SUB #1200,R1        ; адрес результирующего изображения
        MOV #12,R0          ; размер символа по вертикали
3:      MOV #4,R5           ; размер по горизонтали
4:      MOV #10,R4          ; число битов в байте
        MOV #3,R3           ; аналог маски цвета
        CLR @R1             ; очистить байт экрана
6:      RORB @R2            ; распаковать байт в слово
        BCC 5
        BIS R3,@R1
5:      ASL R3
        ASL R3
        SOB R4,6
        MOV @R1,100(R1)     ; продублировать байт
        INC R2              ; сдвиг указателей
```

```
        TST (R1)+
        SOB R5,4            ; повтор для соседних 4 байтов
        ADD #170,R1         ; сдвиг указателей
        ADD #74,R2
        SOB R0,3            ; повтор 9 раз
        RTS PC
NUMBER: .A:35               ; текст для вывода
```

Примечание редактора

Предложенные А.В. Милюковым программы являются неплохим примером использования прошитого в ПЗУ БК знакогенератора. Хотя получаемый по описываемой методике шрифт уменьшенной высоты хуже читается на экране, нежели стандартный, его использование может потребоваться при необходимости увеличения числа строк, отображаемых на экране. Подобное «усечение» символов по высоте использовано, например, в программе — сквизере дисков В.Кобякова. Полезным было бы оно также при выводе на экран Нортон-подобными оболочками списка файлов, хранящихся на дискетах (что позволило бы ускорить поиск нужного файла), а также в текстовых редакторах (обеспечение возможности вывода на экран целой страницы перед печатью либо вывод большего количества строк при работе в режиме расширенной памяти).

Что же касается программы вывода символов «учетверенной» величины, ее возможности сильно ограничены из-за того, что она позволяет выводить на экран лишь два символа. Адаптация же ее к выводу текста произвольной длины сильно усложнена по той причине, что в данной программе имеется несколько констант, значения которых зависят от длины выводимого текста. Более рациональным было бы использование одной константы — собственно длины текста, с расчетом других величин программным способом. Для интересующихся приведем формулы расчета констант при изменении длины текста (в ассемблерный листинг нужно подставлять восьмеричные значения):

Номер строки листинга 2, начиная с метки BIG	Формула для расчета	Прежнее значение константы
2	dl	#2
5	2*dl	#4
9	2*dl	#4
23	&O200-dl*4	#170
24	&O100-dl*2	#74
27	текст длины dl	2 символа

Здесь dl — длина выводимого теста (не более 18 символов).

М. В. Лядвинский,
г. Александров

НОВЫЙ НАБОР СИМВОЛОВ В ПРОГРАММАХ НА БЕЙСИКЕ БК-0010.01

Если вы хотите создать собственный набор символов, то в этом вам поможет «SHRIFT EDITOR» (листинг 1). С помощью этой программы вы сможете получить до 64 символов, которые затем будут представлены в виде блока с адреса 15102 и длиной 1280 байтов.

Создание шрифта

Вначале вы должны освободить ОЗУ с адреса 15102 командой CLEAR 200,15100, а затем запустить программу 1. Компьютер запросит литеру, вместо которой знакогенератор на Бейсике будет печатать символ, определенный пользователем. После этого в окне 8x10 точек, перемещая курсор клавишами «вверх», «вниз», «вправо» и «влево», вы создаете символ, зажигая («1», «2», «3») или гася («4») точки. После завершения работы вы нажимаете «ВК», и компьютер запрашивает следующую литеру. Если вы определили все символы нового шрифта, то нужно нажать «СТОП» и записать созданный набор символов: BSAVE "<имя>",15102,16382.

Использование шрифта

Подпрограмма, генерирующая символы, находится в строках с 1000 по 1070 (программа 1). Удалив «SHRIFT EDITOR» командой DEL -350, вы можете набирать свою программу. Входные данные (код символа, который должен быть напечатан) содержит переменная SM%. Для примера использования нового шрифта можно запустить программу 2 или 3.

```
10 ? CHR$(148%);CHR$(158%);CHR$(140%);CHR$(140%);CHR$(145%); "SHRIFT
EDITOR     (C) LIMAX 1992";AT(17%,4%)"NEW";AT(17%,5%)"SYMBOL:";
20 GOSUB 180
30 ? AT(17%,2%)"SYMBOL:";CHR$(154%);
40 LINE (15%,19%)-(80%,120%),,B
50 LINE (14%,18%)-(81%,121%),1,B
60 C$=INKEY$
70 IF C$="" TH 60 ELSE IF ASC(C$)<33% OR ASC(C$)>96% TH 60
80 ? C$;
90 X%=2%
100 Y%=2%
110 ? AT(X%,Y%);CHR$(145%);
120 J$=INKEY$
130 IF J$="" TH 120 EL IF J$=CHR$(8%) AND X%>2% TH X%=X%-1% EL IF
J$=CHR$(25%) AND X%<9% TH X%=X%+1% EL IF J$=CHR$(26%) AND Y%>2% TH
Y%=Y%-1% EL IF J$=CHR$(27) AND Y%<11% TH Y%=Y%+1% EL IF ASC(J$)>48%
AND ASC(J$)<53% TH 150EL IF J$=CHR$(10%) TH 280
140 GOTO 110
150 ? CHR$(154%);CHR$(156%);CHR$(96%+ASC(J$));CHR$(127%);
CHR$(156%);CHR$(154%);CHR$(8%);CHR$(148%);
160 PSET (190%+X%,48%+Y%),ASC(J$)-48%
170 GOTO 120
180 FOR J%=33% TO 64%
190 ? AT(J%-33%,16%)CHR$(J%);AT(J%-33%,19%)CHR$(J%+32%)
200 ? AT(J%-33%,17%);
210 SM%=J%
220 GOSUB 1000
230 ? AT(J%-33%,20%);
```

```
240 SM%=J%+32%
250 GOSUB 1000
260 NEXT J%
270 RETURN
280 P%=(ASC(C$)-33%)*20%+15102%
290 P1%=20656%
300 FOR P2%=P% TO P%+18% ST 2%
310 POKE P2%,PEEK(P1%)
320 P1%=P1%+64%
330 NEXT P2%
340 ? CHR$(154%)
350 GOTO 10
1000 XY1%=17408%+640%*CSRLIN+2%*POS
1010 SM%=15102%+(SM%-33%)*20%
1020 ? CHR$(25%);
1030 FOR XY%=XY1% TO XY1%+576% ST 64%
1040 POKE XY%,PEEK(SM%)
1050 SM%=SM%+2%
1060 NEXT XY%
1070 RETURN

2010 J$="<любой текст>"
2020 FOR L%=1% TO LEN(J$)
2030 SM%=ASC(MID$(J$,L%,1%))
2040 GOSUB 1000
2050 NEXT L%
2060 END

3010 J$=INKEY$
3020 IF J$="" GOTO 3010
3030 SM%=ASC(J$)
3040 GOSUB 1000
3050 GOTO 3010
```

Примечание редактора

Предложенная идея реализации новых шрифтов пользователя интересна, а программа SHRIFT EDITOR при не слишком большом объеме достаточно удобна в работе. Однако есть у нее и крупные недостатки. Во-первых, отсутствует возможность задания в новом шрифте пробела. А так как буфер кодирования символов предварительно не очищается, при работе программ 2 и 3 вместо пробела выводится нечто невразумительное. И во-вторых, при ошибочном нажатии клавиши и входе на редактирование соответствующего ей символа нет возможности «аварийного выхода» из редактирования с сохранением содержимого буфера и приходится рисовать символ для данной клавиши заново.

Отдельно следует заметить, что подпрограмма вывода символов (строки 1000—1070) «не следит» за тем, чтобы переданный ей в SM% код не выходил за пределы допустимого интервала. Корректнее было бы производить такую проверку и в случае, когда SM%<33 или SM%>140, обеспечивать вывод кода стандартным способом: CHR$(SM%).

Р. Аскеров,
г. Брянск

О НЕКОТОРЫХ СПОСОБАХ ГЕНЕРАЦИИ ШРИФТОВ ДЛЯ БК

Далеко не всем пользователям БК нравится стандартный шрифт, зашитый в ПЗУ монитора. Хотя большинство с этим смирилось, все же создаются отдельные программные средства и даже интегрированные системы, предоставляющие возможность работы с собственными шрифтами. Примером такой системы может служить широкоизвестный Т-язык. Собственные шрифты имеют некоторые копировщики (например, серии UNIC), где это необходимо для увеличения максимальной длины копируемого файла, и некоторые текстовые редакторы типа EDIT2 — для удобства работы с текстами. Редактор же шрифтов PetSoft Font Master позволяет генерировать оригинальные шрифты и включать их в программы в кодах.

Что же нужно для написания нового шрифта? Прежде всего свободная область памяти и хороший отладчик или ассемблер — кому как удобнее. Написать драйвер на языке высокого уровня (имеется в виду стандартный Фокал и вильнюсский Бейсик. — *Прим. ред.*) невозможно — для его работы необходим перехват прерывания по вектору 30 (EMT).

Хранить изображения символов можно поточечно и векторно. В первом случае в память записывается непосредственно образ символа, а во втором — способ его построения. При использовании первого метода символ можно вывести на экран почти мгновенно, но возникают трудности с масштабированием. Векторный шрифт, напротив, не накладывает никаких ограничений на масштаб, но в остальном уступает точечному. Поскольку векторные шрифты на БК применялись только в драйвере чешского плоттера, а точечные — практически повсюду, то рассмотрим последние подробнее.

Существует несколько вариантов построения таблиц, в которых хранятся образы символов. Так, адрес начала битового образа может вычисляться по коду символа в процессе вывода. Это удобно, если замене подлежит вся кодовая таблица, а не отдельные символы или группы символов.

```
                ; Вход: в R0 код символа

                CMP R0,#100          ; Начало латинских символов
                BMI END              ; Меньше - выход
                CMP R0,#200          ; Конец латинских символов
                BPL NEXT             ; Переход, если больше
                SUB #100,R0          ; Находим смещение
GOOD:           ASL R0               ; относительно начала
                ASL R0               ; таблицы образов
                ASL R0               ; (из расчета 20 байт
                ASL R0               ; на символ)
                ADD #BEGTAB,R0       ; Находим абсолютный адрес
                BR  OUT              ; Адрес найден: он в R0
NEXT:           CMP R0,#300          ; Начало русских символов
                BMI END              ; Управляющие коды пропустим
                SUB #200,R0          ; Перерабатываем код ASCII
                                     ; символа в его табличный код
                BR  GOOD             ; Теперь находим смещение...
                . . .
```

Хотя процедура вычисления адреса и оказывается довольно громоздкой, зато не приходится тратить память под какие-либо другие таблицы и время для поиска в них значений, а символы можно подготовить в любом графическом редакторе, например ГРАФРЕДе. Этот способ активно применяется RDA corp.

Можно и не вычислять адреса — для этого достаточно хранить код символа в таблице непосредственно перед его битовым образом. Просматривая отведенные для кода байты, можно найти и сам образ. К достоинствам этого способа следует отнести легкость дополнения уже существующей таблицы новыми символами, а к недостаткам — затраты времени на поиск кодов, памяти — на их хранение, а также некоторые сложности при написании драйвера.

```
                ; Вход: в R0 код символа

                MOV  #BEGTAB,R1      ; Начало таблицы - в R1
NEXT:           CMPB (R1)+,R0        ; Сравнить код с эталоном
```

```
        BEQ  OUT            ; Нашли - в R1 адрес образа
        ADD  #20,R1         ; Нет - следующий код (из
                            ; расчета 20 байт на символ)
        CMP  R1,#ENDTAB     ; Таблица кончилась?
        BMI  NEXT           ; Нет - следующая проверка
END:    NOP                 ; Такого символа нет в таблице
        . . .
```

Так организованы шрифты в драйвере PetSoft, а также в копировщиках серии UNIC.

Третий способ является промежуточным между первыми двумя. Вместо одной таблицы здесь применяются две отдельные: для самих образов и их адресов. Это ускоряет как поиск, так и вывод символов, сами символы (в отличие от адресов) могут располагаться вразброс. Однако при этом способе возникают заметные трудности при добавлении новых символов. Кроме того, из-за наличия двух таблиц память расходуется менее экономно. Но программа вывода при этом отличается лаконичностью:

```
; Вход: R0 - код символа

ASL R0                  ; Находим смещение относительно
                        ; начала таблицы адресов
MOV TABADR(R0),R1       ; Пересылаем адрес в R1
BR  OUT                 ; Выводим символ...
. . .
```

Во всех примерах предполагается, что адреса начала таблиц жестко заданы. Впрочем, не составит никакого труда, добавив пару строк, сделать эти подпрограммы перемещаемыми.

Теперь поговорим непосредственно о выводе символа на экран. Видеопамять БК отображается на экране в виде 256 строк по 32 слова (64 байта) каждая. Процессор 1801БМ1 пересылает информацию только побайтно или пословно, поэтому получить 32 или 64 символа в строке достаточно просто, тогда как, например, реализация режима 80 символов в строке требует достаточно сложных преобразований.

Если предполагается использовать только режим «32 символа в строке», то желательно хранить символы в таблице в виде двухбайтных слов (при этом все точки символа должны быть красными), а пересылать их на экран побайтно — это позволит размещать символы на границе двух «широких» знакомест:

```
; Вход: в R1 - адрес образа
;       в R2 - адрес на экране
;       в R3 - маска цвета

        MOV  #10,R4         ; 10 строк растра на символ
MET:    MOVB (R1)+,R2       ; Вывести очередную строчку
        BICB R3,(R2)+       ; и сменить цвет. R3:
        BICB R3,(R2)+       ; 52525 - зеленый,
                            ; 125252 - синий,
                            ; прочие - полосатый
        ADD  #76,R2         ; Перейти к следующему байту
        SOB  R4,MET         ; Вывести остальные байты
        . . .
```

Здесь в R3 хранится маска, по которой сбрасываются отдельные биты, тем самым изменяется цвет символа (поэтому вначале он должен быть красным). Именно так выводятся символы в программах RDA corp.

При использовании только режима «64 символа в строке» выгодно хранить символы побайтно и побайтно же выводить их на экран. Естественно, смена цветов здесь не производится.

Если же по каким-то причинам необходимы оба режима, то возможны два варианта: «сжатие» хранящихся в памяти «широких» символов и «распаковка» «узких» с одновременной сменой цвета. Второй способ предпочтительнее из-за значительной экономии памяти, он же применяется в системном мониторе БК. Поскольку алгоритм вывода «распакованного» символа полностью аналогичен только что рассмотренному, приведу только процедуру «распаковки»:

```
                ; Вход: в R0 - байт "узкого" символа

                MOV  #10,R2         ; В байте 10 бит
MET:            ROR  R0             ; Очередной бит -
                MFPS R3             ; запомнить его в R3
                ROR  R1             ; и дописать к R1
                MTPS R3             ; "Вспомнить" бит
                ROR  R1             ; и дописать к R1
                SOB  R2,MET         ; Повторить для байта
                . . .
```

Как видно, затраты времени на «распаковку» достаточно велики — только для одной строчки растра нужно произвести 10 циклов.

Высота символов ничем не ограничивается, но лучше всего перед созданием шрифта провести несложные расчеты. Полный экран содержит 256 строк растра. Пусть наш символ имеет в высоту 7 таких строк. Добавив еще по 2 строки на промежутки между строками символов, получаем 9 строк на символ. Тогда на экране поместятся 256/9=28 строк символов и останутся свободными еще 4 телевизионные строки (например, для табулятора).

Поскольку преобразования символов в режимах ПОДЧ, ИНВ, БЛР различны для каждой программы, здесь не приводятся их описания, но путем небольшой модификации подпрограмм вывода можно получить любой необходимый эффект.

Если вас устраивают стандартные размеры шрифта, то можно, создав таблицу символов, по структуре аналогичную прошитой в ПЗУ монитора, использовать для вывода стандартные подпрограммы. Такой драйвер написан В. Коренковым:

```
                ; Вход: в R0 - код символа

                JSR  R4,@#110346    ; Сохранить регистры
                BIC  #177400,R0     ; Выделить код символа
                CMPB R0,#40         ; Служебный символ?
                BCS  2$             ; Да - стандартная обработка
                CMPB R0,#200        ; Латинский?
                BCS  1$             ; Да - вывести
                CMPB R0,#240        ; Управляющий код?
                BCS  2$             ; Да - стандартная обработка
                SUB  #40,R0         ; Преобразовать код
1$:             SUB  #40,R0         ;
                ASL  R0             ; Найти
```

```
        MOV  R0,R1          ; смещение
        ASL  R0             ; относительно
        ASL  R0             ; начала
        ADD  R0,R1          ; таблицы
        ADD  #TABADR,R1     ; Найти абсолютный адрес
        CALL @#102764       ; Вывести символ
        CALL @#102374       ; стандартными средствами
        JSR  R4,@#110362    ; Восстановить регистры
        RTI                 ; Возврат
2$:     JSR  R4,@#110362    ; Восстановить регистры
        MOV  R5,-(SP)       ; Сохранить R5
        MOV  #16,R5         ; Выполнить стандартное
        JMP  @#100126       ; EMT 16
```

И наконец, поговорим о перехвате командного прерывания EMT. В вектор 30 заносится адрес обработчика прерывания, функции которого состоят в сохранении значений регистров, сравнении номера вызова EMT (в данном случае с 16 — вывод символа на экран) и, при необходимости, вызова своей подпрограммы вывода символа. Далее обработчик восстанавливает регистры и возвращает управление вызывающей программе. Обработчики отличаются обычно только способом вызова подпрограмм, поэтому приведу, на мой взгляд, наиболее удачный вариант, написанный тем же В. Коренковым (начало которого является практически копией стандартного обработчика EMT, прошитого в ПЗУ0. — *Прим. ред.*):

```
EMT:    MOV R5,-(SP)        ; Сохранить R5
        MOV 2(SP),R5        ; Адрес EMT+2 - в R5
        MOV -(R5),R5        ; Код EMT - в R5
        BIC #177400,R5      ; Номер EMT - в R5
        CMP R5,#16          ; Это EMT 16?
        BNE OUT$            ; Нет - стандартно
        MOV (SP)+,R5        ; Восстановить R5
        BR  EMT16           ; Выполнить свою п/п
OUT$:   JMP @#100126        ; Стандартная обработка EMT
```

Впрочем, если вы создаете собственную программу, то вполне можно обойтись без обработчика — достаточно вызывать подпрограмму вывода явно, по ее адресу. Ничто не мешает также сделать эту подпрограмму перемещаемой.

Теперь остается только объединить нужные подпрограммы с обработчиком и таблицей шрифта — и БК порадует вас своим новым почерком!

От редакции

ВЕКТОРНЫЙ ШРИФТ ДЛЯ БЕЙСИКА БК-0010.01

На многих компьютерах помимо обычного растрового шрифта (знаки которого представляются в виде матрицы точек, часть которых высвечивается, формируя начертание символа) используется векторный, где символы вычерчиваются отрезками прямых. Как правило, подобный тип шрифта применяется в графических редакторах, входящих в комплект систем САПР

(например, AutoCAD или PCAD), а также в программах, предназначенных для работы с графопостроителями, однако его реализации имеются и в графических библиотеках многих языков высокого уровня, например Borland C++. В чем же секрет подобной популярности? Векторный шрифт обладает по сравнению с растровым рядом существенных преимуществ:

— произвольное местоположение текста на экране (растровые символы жестко «привязаны» к знакоместам экрана и их положение можно менять только дискретно);

— произвольный размер символов по горизонтали и вертикали;

— произвольная толщина линий вычерчивания;

— возможность задания произвольного угла поворота при выводе текста;

— возможность изменения масштаба всего рисунка, содержащего надписи векторным шрифтом (при уменьшении растровой картинки нередко случается, что текст надписей на ней становится нечитаемым, так как оно чаще всего производится путем удаления части точек изображения).

Из недостатков векторного шрифта, пожалуй, можно назвать только один — «банк закодированных изображений символов» занимает в памяти немало места. На компьютерах типа IBM, о которых, собственно, и говорилось до сих пор, с этим еще можно как-то мириться, а как быть на БК, у которой ресурсы оперативной памяти ограничены?

Были испробованы различные способы кодирования векторного шрифта на БК, от использования стандартного режима «ГРАФ, ЗАП, СТИР» до такого экзотического, как вызов из кодовой USR-подпрограммы зашитой в ПЗУ Бейсика программы реализации оператора DRAW с передачей ей текстовой строки, кодирующей символ. Во всех случаях размер «банка символов» в несколько раз превышал объем собственно программы вывода векторного шрифта.

Решение задачи оказалось довольно простым: можно использовать стандартный, прошитый в ПЗУ монитора БК растровый шрифт, преобразуя его в векторное начертание. Простейший алгоритм такого преобразования может быть таким:

— перебор (например, построчно) точек, составляющих матрицу символа, с проверкой их цвета;

— если цвет соответствует цвету фона, то переход к следующей точке;

— если точка «высвечена», то проверка цвета соседних с ней точек по горизонтали, вертикали и диагонали;

— вычерчивание отрезков, соответствующих каждой паре соседних точек исходного растрового символа.

Описанный алгоритм проще всего реализуется средствами Бейсика. Кроме того, легкость освоения этого языка и понятность листинга позволят разобраться в работе программы даже начинающим пользователям. Те же, кто неплохо знает и программирование на ассемблере, могут попытаться написать по этому принципу кодовую USR-подпрограмму, сократив, таким образом, занимаемый объем памяти и повысив универсальность применения.

В данной программе исходный растровый символ выводится в позицию (0,0) экрана, а затем просматривается в цикле FOR оператором POINT. Более интересным и удобным (но и более сложным) может быть другое решение — непосредственный анализ содержимого «банка изображений символов» в ПЗУ. На Бейсике такой способ организовать трудно, так как оператор INP не обеспечивает доступа к байтам с нечетными адресами. На ассемблере же это реализуется достаточно легко с помощью оператора BITB #<маска>,@#<адрес>. Для справки приведем алгоритм вычисления соответствующего коду символа адреса первого байта его изображения в «банке» (в виде фрагмента Бейсик-программы):

```
' KS% - код символа
IF KS%>159% THEN KS%=KS%-64%
    ELSE IF (KS%>31%)AND(KS%<128%) THEN KS%=KS%-32%
        ELSE KS%=0%
ADR%=&O112276+KS%*10% ' ADR% - адрес первого байта
```

В данном случае непечатаемые символы автоматически заменяются пробелами. Другой вариант проще для реализации алгоритма на ассемблере, но для защиты от ошибок необходим предварительный контроль кода символа:

```
' KS% - код символа (KS%>159% или 31%<KS%<128%)
IF KS%>159% THEN KS%=KS%-32%
KS%=KS%-32%
ADR%=&O112276+KS%*10% ' ADR% - адрес первого байта
```

Приведенный далее листинг показывает часть возможностей, доступных пользователю при использовании векторного шрифта, а также способы «управления» этими возможностями. Блок, реализующий собственно преобразование растровых символов в векторные и их вывод на экран, оформлен в виде подпрограммы (строки 1000 — 1540, из них 1500-1540 — вычерчивание линий заданной толщины). Необходимые значения передаются в подпрограмму с помощью переменных:

S$ — текстовая константа (строка) для вывода в векторном виде;

KS$ — код очередного символа, выделяемый из строки S$ (внутренняя переменная подпрограммы);

X0%,Y0% — координаты «точки привязки» (верхнего левого угла прямоугольной зоны на экране, в которую будет вписан первый символ строки);

MX%,MY% — масштаб увеличения по горизонтали и по вертикали, значения «1» соответствуют размерам стандартного растрового шрифта в режиме 32 символа в строке (величины MX% и MY% могут быть заданы независимо);

T% — толщина линий вычерчивания символов;

JS% — направление вывода текста: 0 — слева направо, 1 — снизу вверх, 2 — справа налево, 3 — сверху вниз (сами символы не поворачиваются);

C% — цвет вывода (1 — красный, 2 — зеленый, 3 — синий);

MS% — междусимвольный интервал;

X%,Y%,XK%,YK%,K%,J%,I%,L% — рабочие переменные.

Примечания:

1. Этот вариант программы рассчитан на использование режима 32 символа в строке и черного фона. Нежелательно также использование режимов подчеркивания и инверсии символов и экрана.

2. В режиме БлокРед можно получить векторное начертание символов, соответствующих кодам редактирования.

```
5 ' Демонстрация работы алгоритма преобразования растрового шрифта в векторный
10 CLS
20 FOR II=1 TO 7
```

```
30 READ X0%,Y0%,MX%,MY%,T%,JS%,C%,MS%,S$
40 GOSUB 1000
50 NEXT
60 END
70 DATA 100,10,1,2,1,0,1,0,"ПРИМЕР"
80 DATA 10,0,4,3,3,3,2,-7,"ВЕКТОРНОГО"
90 DATA 200,0,4,3,3,3,2,-7,"ВЕКТОРНОГО"
100 DATA -5,200,8,5,5,0,3,-23,"ШРИФТА"
110 DATA 50,50,2,6,2,0,3,1,"***"
120 DATA 100,50,6,2,2,0,3,-15,"***"
130 DATA 50,100,3,3,1,0,1,1,"$&#▒@"
990 ' Подпрограмма построения векторного шрифта
995 ' ----------------------------------------
1000 FOR K%=1% TO LEN(S$)
1010 KS$=""+MID$(S$,K%,1%) ' выделить очередной символ
1020 ? AT(0,0);KS$; ' "буферная" позиция экрана
1030 Y%=Y0%
1040 FOR J%=0% TO 8%
1050 X%=X0%
1060 FOR I%=0% TO 7%
1065 IF POINT(I%,J%)=4 THEN 1190
1070 IF I%=0% THEN 1100
1075 IF POINT(I%-1%,J%+1%)<>4 THEN XK%=X%-MX% ELSE 1100
1080 YK%=Y%+MY%
1090 GOSUB 1500
1100 IF POINT(I%,J%+1%)<>4 THEN XK%=X% ELSE 1130
1110 YK%=Y%+MY%
1120 GOSUB 1500
1130 IF I%=7% THEN 1160
1135 IF POINT(I%+1%,J%+1%)<>4 THEN XK%=X%+MX% ELSE 1160
1140 YK%=Y%+MY%
1150 GOSUB 1500
1160 IF I%=7% THEN 1190
1165 IF POINT(I%+1%,J%)<>4 THEN XK%=X%+MX% ELSE 1190
1170 YK%=Y%
1180 GOSUB 1500
1190 X%=X%+MX%
1200 NEXT I%
1210 Y%=Y%+MY%
1220 NEXT J%
1230 ON JS% GOTO 1260,1280,1300
1240 X0%=X0%+8%*MX%+MS%
1250 GOTO 1310
1260 Y0%=Y0%-10%*MY%-MS%
1270 GOTO 1310
1280 X0%=X0%-8%*MX%-MS%
1290 GOTO 1310
1300 Y0%=Y0%+10%*MY%+MS%
1310 NEXT K%
1320 RETURN
```

```
1490 ' Подпрограмма рисования линии (X%,Y%)-(XK%,YK%) цветом C% и толщиной T%
1500 IF T% MOD 2% =0% THEN T%=T%+1% ' T% обязательно должно быть нечетным!
1510 FOR L%=-(T%-1%)/2% TO (T%-1%)/2%
1520 LINE (X%+L%,Y%+L%)-(XK%+L%,YK%+L%),C%
1530 NEXT L%
1540 RETURN
```

Р. А. Рахманкулов,
г. Ташкент

БОЛЬШИЕ СИМВОЛЫ С ТЕНЯМИ НА БК

Автор предлагает читателям журнала программу рисования больших букв на Бейсике на БК-0010.01.

Строка A$, передаваемая подпрограмме (строки 90-620), имеет формат:

<I0><J0><M><N><CO(1)><CO(2)><DX><DY><строка>

Здесь:

— <I0><J0> — координаты верхней левой точки текста на экране (по три знака в символьном представлении);

— <M><N> — коэффициенты увеличения размеров букв по X и Y (по два знака);

— <CO(1)><CO(2)> — цвета букв и тени (по одному знаку);

— <DX><DY> — сдвиг тени по X и Y (по два знака);

— <строка> — выводимый текст (количество символов не должно превышать «информационной емкости» экрана по ширине при заданном масштабе увеличения).

```
10 CLS
20 A$="010010050513-1-4Привет"
30 GOSUB 90
40 A$="0100500610310301всем"
50 GOSUB 90
60 A$="0051200315120301БК-шникам!"
70 GOSUB 90
80 END
90 I0%=VAL(MID$(A$,1,3))
100 J0%=VAL(MID$(A$,4,3))
110 M%=VAL(MID$(A$,7,2))
120 N%=VAL(MID$(A$,9,2))
130 CO%(1)=VAL(MID$(A$,11,1))
140 CO%(2)=VAL(MID$(A$,12,1))
150 DX%=VAL(MID$(A$,13,2))
160 DY%=VAL(MID$(A$,15,2))
170 GOSUB 470
180 GOSUB 550
190 L%=LEN(A$)-16
200 FOR O%=1% TO L%
210 ? AT(0,0);MID$(A$,16+O%,1)
220 FOR I%=T1% TO T2% ST T3%
230 FOR J%=V1% TO V2% ST V3%
240 IF POINT(I%,J%)=1% TH GOS 380
250 NEXT J%
260 NEXT I%
270 NEXT O%
280 ? AT(0,0);" ";
290 RETURN
300 X%=I0%+I%*M%+(O%-1)*8*M%+D1%
310 Y%=J0%+J%*N%+D2%
320 FOR I1%=1 TO M%
330 FOR I2%=1 TO N%
340 PSET (X%+I1%,Y%+I2%),C%
350 NEXT I2%
360 NEXT I1%
```

```
370 RETURN
380 D1%=DX%
390 D2%=DY%
400 C%=CO%(2)
410 GOSUB 300
420 D1%=0
430 D2%=0
440 C%=CO%(1)
450 GOSUB 300
460 RETURN
470 T1%=0
480 T2%=9
490 T3%=1
500 IF DX%>=0 TH RET
510 T1%=9
520 T2%=0
530 T3%=-1
540 RETURN
550 V1%=0
550 V1%=0
560 V2%=8
570 V3%=1
580 IF DY%>=0 TH RET
590 V1%=8
600 V2%=0
610 V3%=-1
620 RETURN
```

В. И. Рогов, А. В. Кузнецов,
г. Москва

УВЕЛИЧЕНИЕ СИМВОЛОВ НА ФОКАЛЕ

Предлагаем простую подпрограмму на FOCALe для БК-0010, которая позволяет увеличивать масштаб написания букв в 11 раз.

В первой группе строк программы символы, которые необходимо увеличить, выводятся в первую строку экрана. Во второй группе строк создается цикл проверки экранного ОЗУ (адрес первой ячейки соответствует 17472). В третьей группе проверяется каждый бит экранной ячейки. Если бит равен 1, то он заменяется любым алфавитно-цифровым символом (в данном случае — пробелом, выводимым в режиме инверсии символов). В десятой группе реализована возможность записи в служебную строку на языке FOCAL.

```
C: ФОКАЛ-БК0010
1.10 X FCHR(140,140)
1.20 T "RK corp."
1.30 X FCHR(154)
2.10 E; S Y=5; F I=17472,2,18048; D 3
2.20 X FCHR(18); T "(C)RK corp. 1991"; G 10.2
3.10 I (J-4)3.15; S I=I+56; S Y=Y+1; S J=0
3.15 F X=0,15; I (FX(0,I,2^X))3.9,3.9; D 5
3.20 S J=J+1
3.90 R
5.10 S XX=J*16+X; X FK(XX,Y); X FCHR(156); T " "; X FCHR(156)
10.10 C ЗАПИСЬ В СЛУЖЕБНУЮ СТРОКУ
10.20 F I=17472,2,18048; S Y=I-1088; S X=FX(1,I); X      FX(-1,Y,X)
10.30 X FCHR(18,19)
```

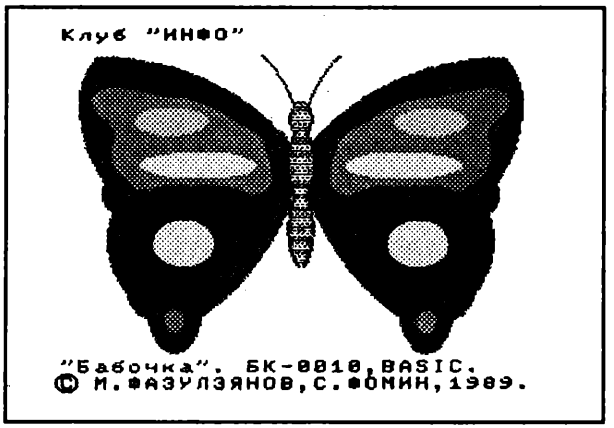
В. В Константинов,
г. Москва

ПРОГРАММА «КРУЖЕВНИЦА»

Предлагаемая вниманию читателей программа выводит на экран череду сменяющих друг друга красочных узоров, напоминающих кружева. Программа запускается с адреса 1000. При останове происходит выход в пусковой монитор (символ диалога "?") без стирания ОЗУ пользователя, т.е. можно запустить программу повторно, изменив начальные параметры с помощью режима ТС. (Перейти в него можно с помощью команд: Т, ВВОД, РУС, ТС - появится приглашение ¤). Можно варьировать стиль рисунков, изменяя значение по адресу 1004 (рекомендуется 44400 и 40400). Скорость вывода узоров задается значением по адресу 1112 (от 1 до 70). Если у вас черно-белый монитор, то отсутствие цвета компенсируется более высокой разрешающей способностью.

НАЧАЛЬНЫЙ АДРЕС			001000		ДЛИНА		000122
104014	012700	066555	012701	040000	012702	020000	010021
006100	004767	000062	005200	077206	012703	025456	012700
111111	012701	040000	012702	020000	074021	006003	006000
004767	000024	077206	005303	005200	160003	006300	077401
077401	077401	000167	177732	012705	000015	000240	077502
000207							

КОНТРОЛЬНАЯ СУММА : 1643738



В. В. Юров, В. П. Юров,
г. Москва

КАТАЛОГИЗАТОР ДИСКАТ2.МВИ

В одном из первых каталогизаторов файлов на кассете БК (варианты имени «KATL», «KATL.M» или «KATL.ED») предусмотрено только чтение имен файлов до прерывания операции клавишей СТОП с последующей записью созданного списка в виде текстового файла под именем «КАТАЛОГ». В нем в таблице в одну колонку содержатся имена файлов на кассете, их адреса загрузки и длины. Над таблицей помещена «шапка», отделенная от нее двумя горизонтальными линиями. Таблица заканчивается такими же линиями, под которыми указывается общее количество файлов. Созданный список можно загружать в текстовый редактор для последующей обработки: нумерации файлов, записи показаний счетчика магнитофона, сортировки, добавления комментариев, объединения с другими списками и т. д.

В дальнейшем появились каталогизаторы с дополнительными функциями: проверка читаемости файлов с соответствующей отметкой об этом в каталоге, определение скорости записи файлов, добавление дополнительных сведений в таблицу, одновременный вывод на принтер, были попытки обеспечить автоматическое отслеживание показаний счетчика магнитофона и времени и т.д. Но эти каталогизаторы обычно удовлетворяли только их создателей, так как в них была заложена жесткая форма представления каталога, изменить которую можно было только после загрузки полученного файла в редактор, что по затратам времени соизмеримо с ручным составлением каталога.

По тому же пути вначале наметилось развитие и каталогизаторов дискет:

— утилита DISKAT1.MVI записывает в текстовый файл неполный (имя, адрес и длина файла) директорий и работает в системах MicroDOS, NORTON, NORD, а подпрограмма SAVDIR.SHL для оболочки SHELL — в системе ANDOS;

— утилита ANPRIDIR (система ANDOS) выводит неполный директорий на принтер в одну колонку с «шапкой», а утилита DIR.us VitalProgramms — полный (имя, адрес, длина файла в байтах и блоках и его местоположение) в две колонки (системы MicroDOS и NORTON). Утилита DISKAT.MVI преобразует файл резервного директория, создаваемый утилитой DIR.us, в текстовый с неполным директорием;

— утилита DirNord-5.0 записывает полный директорий дискеты системы NORD в текстовый файл, при этом каталог оформлен в виде обрамленных и расчерченных в одну колонку таблиц с «шапками» для каждого поддиректория. Под каждой таблицей подведен итог, который суммируется в конце каталога. Файлы имеют сквозную нумерацию;

— московским клубом БК для MDOS'a и NORTON'a разработана утилита KAT12_1.C, записывающая в текстовый файл директорий дискеты в трех вариантах: неполный в одну колонку и полный в одну или две колонки. Те же варианты директория могут быть выведены на принтер. В каталоге файлы пронумерованы, а каждая таблица имеет «шапку» и заголовок. При построении каталога в две

колонки каждая строка отделена горизонтальными линиями, а файлы по колонкам расположены следующим образом: первый файл — в первой колонке, второй — во второй колонке, третий — снова в первой и т.д. В конце списка подведен итог. Для изменения формы каталога требуется повторное считывание директория. Кроме того, такое разбиение файлов по колонкам скорее обусловлено удобством программирования, чем удобством использования. Такой вывод напрашивается не только из-за того, что такая форма разбиения по колонкам встречается крайне редко, но и потому, что эта утилита явно сделана на скорую руку и работает неустойчиво.

Из приведенного списка видно, что при преобразовании директория, как правило, реализуются три варианта: директорий без изменений записывается в текстовый файл для последующей его обработки в редакторе или для вывода на принтер; директорий изменяется под конкретную форму и записывается в текстовый файл и/или выводится на принтер; директорий без изменений или с изменениями под различные, но конкретные формы преобразуется и записывается в текстовый файл и/или выводится на принтер. (В третьем варианте объединены два первых)

Аналогичным образом обстоит дело и на других компьютерах. Лучшими (из известных авторам) являются утилиты Directory Master v1.1 для компьютера COMMODORE AMIGA и PISCES DIREKTORY LISTER v1.0 для компьютера ATARI ST. Эти программы могут преобразовывать директории в различные формы, кроме изготовления этикеток для дискет. Для последнего случая существуют специальные утилиты, например DISK LABELER v1.02 (ATARI ST) для изготовления этикеток на 3,5-дюймовые дискеты.

В утилите ДИСКАТ2.МВИ (программист Владислав Юров), предназначенной для серии БК, сделана попытка обеспечить максимально многофункциональные (с учетом ограниченной памяти компьютера) возможности обработки директориев дискет и кассет для различных применений при использовании «дружественного интерфейса». Утилита читает директории дискет MDOS, NORTON, NORD (без разбиения файлов по поддиректориям), ANDOS, а также магнитофонных кассет. Она может быть запущена в любой из этих систем или с магнитофона.

Основу всех форм каталога образует имя файла, которое может быть дополнено в различных сочетаниях его атрибутами: адресом загрузки, длиной и сведениями о расположении на дискете (начальный блок или кластер и их общее количество, занятое файлом). Во всех формах файлы могут быть либо размещены по порядку, в котором они находятся в директории или на кассете, либо отсортированы по алфавиту, адресу или длине. При этом имеется возможность пронумеровать файлы в том порядке, в котором они находились до сортировки, или присвоить им номера после сортировки, а также сгруппировать файлы в несколько колонок (от одной до восьми). Разбиение по колонкам последовательное: файлы по порядку располагаются в первой колонке, продолжение списка — во второй и т. д. Под таблицей может быть указано общее количество файлов и количество занятых ими блоков или кластеров. Как дань традиции для каждого варианта над таблицей может быть помещена «шапка» с оглавлениями колонок.

Полученный каталог можно записать в виде текстового файла на один из двух дисководов или на магнитофон либо сразу же вывести на принтер в любой из выбранных форм (возможно 1024 варианта без учета изменения шрифтов принтера). Для контроля предусмотрена возможность параллельного вывода каталога на экран и принтер. Повторного считывания директория в случае представления каталога в нескольких формах или для изменения выбранной формы не требуется.

При выводе каталога на принтер предусмотрен выбор шрифтов (черновой, жирный,

двойной, качественный), плотности печати (80, 96 и 132 знака в строке), отступа от левого края (от 0 до 131 символа), кода принтера (прямой или инверсный), вида его подсоединения (по стандарту БК-0010.01 или БК-0011М) и типа (МС6312, МС6313, МС6313Е, D100M, Robotron CM6329), а также предоставлена возможность ввода с клавиатуры заголовка каталога и дополнительных управляющих кодов принтера. Управляющие коды могут действовать как на весь каталог, так и только на строку заголовка. В последнем случае предусмотрен специальный «ограничивающий» код. Использование управляющих кодов принтера расширяет возможности оформления этикеток кассет и дискет.

Выбор всех режимов работы осуществляется с помощью двух меню: основного (рис. 1), которое постоянно находится на экране, и меню вывода каталога на принтер (рис. 2), выводимое на экран при выборе соответствующей команды в основном меню. Кроме того, предусмотрен вывод на экран по запросу листка подсказки (рис. 3) и перезапроса (рис. 4)

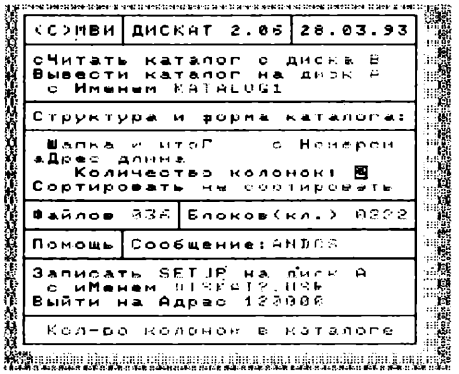


Рис. 1. Основное меню

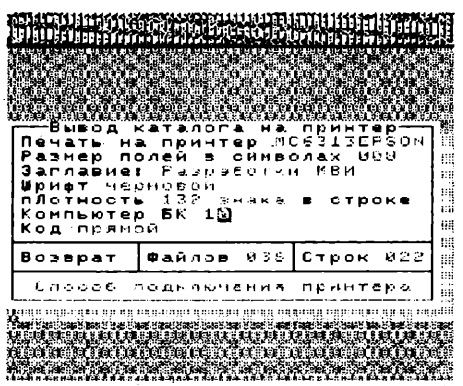


Рис. 2. Меню вывода каталога на принтер

Рис. 3. Листок подсказки

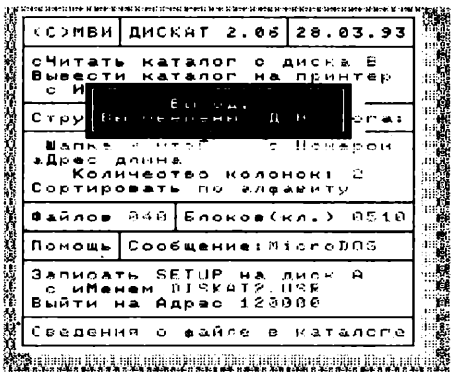
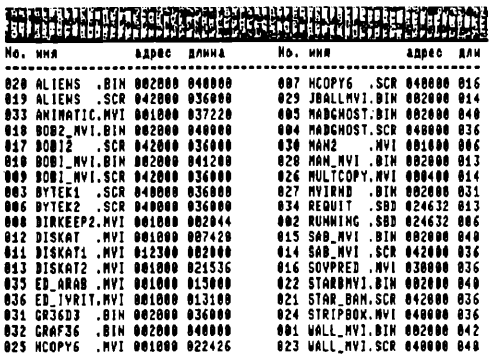


Рис. 4. Перезапрос

при выходе из программы, а также каталога для контроля (рис. 5). Меню одинаково различимы при работе и с монохромным монитором, и с цветным.

В основном меню строки с главными командами отделены от режимов установок, задающих структуру и форму каталога, и от вспомогательных функций, определяющих имя файла с установками пользователя и адрес выхода из программы. После команд, определяющих структуру и форму каталога, размещены две строки с дополнительной информацией. В первой из них приводятся ито-



No.	имя	адрес	длина	No.	имя	адрес	дли
020	ALIENS .BIN	002000	040000	007	HCOPY6 .SCR	040000	016
019	ALIENS .SCR	042000	036000	029	JBALLMVI.BIN	002000	014
033	ANIMATIC.MVI	001000	037220	005	MADGHOST.BIN	002000	040
018	BOB2_MVI.BIN	002000	040000	004	MADGHOST.SCR	040000	036
017	BOB12 .SCR	042000	036000	030	MAN2 .MVI	001000	006
010	BOB1_MVI.BIN	002000	041200	028	MAN_MVI .BIN	002000	013
009	BOB1_MVI.SCR	042000	036000	026	MULTCOPY.MVI	000400	014
003	BYTEK1 .SCR	040000	036000	027	MVIRND .BIN	002000	031
006	BYTEK2 .SCR	040000	036000	034	REQUIT .SBD	024632	013
008	BIRKEEP2.MVI	001000	002044	002	RUNNING .SBD	024632	006
012	DISKAT .MVI	001000	007420	015	SAB_MVI .BIN	002000	040
011	DISKAT1 .MVI	012300	002000	014	SAB_MVI .SCR	042000	036
013	DISKAT2 .MVI	001000	021536	016	SOVPRED .MVI	030000	036
035	ED_ARAB .MVI	001000	015000	022	STARBMVI.BIN	002000	040
036	ED_IVRIT.MVI	001000	013100	021	STAR_BAM.SCR	042000	036
031	GR36D3 .BIN	002000	036000	024	STRIPBOK.MVI	040000	036
032	GRAF36 .BIN	002000	040000	001	WALL_MVI.BIN	002000	042
025	HCOPY6 .MVI	001000	022426	023	WALL_MVI.SCR	040000	040

Рис. 5. Контроль формы каталога (на экране видно только 63 символа каждой строки полной формы каталога, организованного в две колонки)

говые сведения о количестве файлов в считанном директории и о количестве занятых блоков или кластеров. Во второй — выводится сообщение о возможности вызова подсказки по командам меню, а также сообщается о результатах выполнения команд и о системе, в которой записана считываемая дискета. В конце меню размещена третья информационная строка, в которой при перемещении курсора по командам меню выводится подсказка об их назначении.

Перемещение по меню осуществляется клавишами управления курсором. Строка, на которой находится курсор, инвертируется. Для удобства работы используется автоповтор нажатой клавиши. Выбор установок осуществляется клавишей «ВС», если курсор находится на строке с нужными установками, или «горячими клавишами» — независимо от места нахождения курсора. Исполнение команды также осуществляется двумя способами — клавишей «ВВОД» или «горячими» клавишами в сочетании с «АР2». «Горячие» клавиши в строках меню выделены цветом и обозначены заглавными буквами, входящими в наименование установок и команд. Они могут использоваться независимо от состояния регистров (РУС, ЛАТ, ЗАГЛ, СТР). В некоторых случаях для одной и той же строки установок задействовано несколько «горячих» клавиш. Это сделано для того, чтобы на экране всегда были видны обозначения «горячих» клавиш, входящих в название установок, индицируемых в данный момент в соответствующей строке.

По такому же принципу построено и меню вывода каталога на принтер, но в информационной строке вместо числа занятых блоков или кластеров сообщается о количестве строк в каталоге. Для установки величины полей, которые могут изменяться от 0 до 131 символа, дополнительно к клавише «ВС» и «горячей» клавише «О» задействованы клавиши управления перемещения курсора по горизонтали.

Все заданные установки в основном меню и в меню вывода каталога на принтер запоминаются до окончания работы с утилитой или до их очередного изменения. Кроме того, утилита может быть самопереписана с требуемыми установками, которые будут автоматически использоваться при ее повторном запуске, или записана под разными именами с различными вариантами установок. Для этого в меню предусмотрена соответствующая функция и возможность изменения имени утилиты для записи. В основном меню предусмотрено изменение имени, под которым будет записываться каталог. Для уменьшения затрат времени на очередное редактирование имени файла положение курсора при предыдущих изменениях запоминается.

Для обеспечения возврата в нужный режим после окончания работы с утилитой в ее основном меню предусмотрена возможность задания адреса выхода. При этом предлагаются стандартные ситуации выхода, типичные для различных систем. Заданный адрес запоминается вместе с другими установками при перезаписи утилиты.

После запуска утилиты на экран выводится листок-подсказка. При нажатии на любую клавишу он удаляется и курсор устанавливается на команду считывания директория. После считывания курсор перемещается на команду записи каталога или вывода его на

принтер. После записи курсор вновь возвращается на команду считывания директория. При выходе в меню вывода каталога на принтер курсор указывает на команду печати.

Некоторые строки меню содержат «адресацию» к конкретному устройству ввода-вывода, например: «считать каталог с диска В». После установки инверсного курсора на эту строку клавиша «Ввод» вызывает выполнение команды применительно к указанному устройству, а клавиша «ВС» осуществляет смену целевого устройства («магн.», «диск А» или «диск В»).

Если не устраивает структура файла, отраженная в строке «аДрес длина Блоки(кластеры)», то для изменений можно переместить в эту строку курсор и нажать клавишу «ВС» или, не перемещая курсор, нажать клавишу «Д» либо «Б». При нажатии указанных клавиш эта строка будет изменяться следующим образом: «Без дополнительных сведений», «аДрес длина», «Блоки(кластеры)», и вновь «аДрес длина Блоки(кластеры)». Аналогичным образом можно изменять и другие строки установок, что обеспечивает 1024 различные формы представления каталога. Дополнительное разнообразие (еще несколько тысяч вариантов) обеспечивается за счет выбора режимов работы принтера, предусмотренных в меню. Несмотря на такое обилие форм представления каталога, в отдельных случаях, которые для пользователя могут быть решающими, предусмотрено дальнейшее изменение формы представления за счет ввода в строке «Заглавие» дополнительных управляющих кодов принтера, с помощью которых можно задействовать другие шрифты (например, субскрипт) или изменить расстояние между строками. Эти коды можно ввести как вместе с заглавием, так и без него. Все сделанные в строке «Заглавие» изменения также запоминаются и могут быть сохранены при перезаписи утилиты.

Таким образом, не перегружая меню, удалось реализовать возможность предоставления различных режимов работы и обеспечить получение большого числа форм каталога для различных применений, в том числе для этикеток кассет и дискет, сделав при этом работу с утилитой достаточно простой.

Для приобретения утилиты ДИСКАТ2.МВИ, а также других программ обращайтесь по адресу: 127349, Москва, а/я 9, Юров Вячеслав Петрович. Условия можно узнать по телефону: (095) 908-22-12 с 10 до 21 часа ежедневно.

От редакции: утилита ДИСКАТ2.МВИ и файл документации к ней входят в комплект поставки дисковой операционной системы для БК ANDOS v2.30.

ОБМЕН ОПЫТОМ

БК-0011М — 64 ЗНАКА В СТРОКЕ

Если на БК-0011 можно было осуществлять переключение режимов 32/64 символа в строке с клавиатуры нажатием АР2+«;» (как и в БК-0010), то на БК-0011М это можно сделать только программно, с помощью функции EMT51. Перед ее вызовом необходимо занести в регистр R0 адрес начала буферного массива параметров драйвера дисплея (длина 13 байт). Первые два бита в нулевом (относительно начала массива) байте определяют количество символов в строке.

На Бейсике это можно сделать с помощью несложной USR-подпрограммы:

```
R3264:  MOV #BUF,R0
        EMT 64
        RTS R7
;                   ; дес.      ;   двоичное
BUF:    .#20002     ; 8194      ; 0 010 000 000 000 010 (*)
        .#54202     ; 22658     ; 0 101 100 010 000 010
        .#132754    ; -18964    ; 1 011 010 111 101 100
        .#54202     ; 22658     ; 0 101 100 010 000 010
```

Изменяя значение в строке, помеченной звездочкой (*), можно получить различные режимы вывода текста:

Пара битов	Десятичное число	Режим
00	8192	128 графических точек в строке
01	8193	32 символа в строке
10	8194	64 символа в строке

(Указанный факт редакцией не проверялся. — *Прим. ред.*)

По материалам «Бюллетеня БК» № 3 за 1990 г. (МНТП «Импакт», Киев).

ИСПОЛЬЗОВАНИЕ ГРАФИЧЕСКИХ СРЕДСТВ БЕЙСИКА БК-0010 В ПРОГРАММАХ НА АССЕМБЛЕРЕ

Рисование с помощью ассемблерной программы точек и линий проблем не представляет — разработчики драйвер-мониторной системы БК любезно предоставили пользователям соответствующие функции EMT30 и EMT32. А вот с графическими операциями «более высокого уровня» (закрашивание области экрана и рисование окружности или эллипса) дело обстоит сложнее: в БК-0010 эти функции реализованы только в БЕЙСИКе.

Но если программа должна работать в мониторе БК-0010.01 с отсоединенным блоком МСТД (т. е. при «активном» ПЗУ вильнюсского БЕЙСИКа), то можно воспользоваться подпрограммами реализации операторов БЕЙСИКа PAINT и CIRCLE, прошитыми в ПЗУ.

1. Закрашивание области экрана, ограниченной точками заданного цвета:

```
MOV #X,-(SP)
MOV #Y,-(SP)
MOV #COLOR,-(SP)
MOV #LCOLOR,-(SP)
TST -(SP)
MOV #ADDR,-(SP)
TST -(SP)
JMP @#125656
```

Здесь X, Y — координаты точки, с которой начинается закраска, COLOR — код цвета закраски (221—224), LCOLOR — код цвета границы (цвет закраски также является граничным), ADDR — адрес ячейки, в которой содержится адрес выхода из подпрограммы закрашивания.

При выполнении программы происходит обнуление ячейки по адресу 2134.

Программа активно использует стек (при закраске сложных фигур требуемый объем стека может превышать 3 Кб).

2. Черчение эллипса:

```
MOV #X,-(SP)
MOV #Y,-(SP)
MOV #RAD,-(SP)
MOV #COL,-(SP)
MOV 2(SP),-(SP)     ; Ввод координат
ADD 10(SP),(SP)     ; начала
MOV 6(SP),-(SP)     ; дуги
MOV 6(SP),-(SP)     ; Ввод координат
ADD 14(SP),(SP)     ; конца
MOV 12(SP),-(SP)    ; дуги
MOV #KY,-(SP)
MOV #KX,-(SP)
MOV @#214,-(SP)
MOV #ADDR,-(SP)
TST -(SP)
JMP @#127350
```

Здесь X, Y — координаты центра эллипса, RAD — радиус эллипса (длина большой полуоси), COL — цвет, KY, KX — коэффициенты сжатия по Y и X (обычно равны длинам полуосей, но при черчении эллипсов большого диаметра должны быть пропорционально уменьшены во избежание сбоя программы), ADDR — адрес ячейки, содержащей адрес выхода из программы.

В данном варианте программа чертит полную окружность или эллипс. Если необходимо начертить дугу, нужно изменить программу, задав координаты начала и конца дуги.

А. В. Новиков,
г. Ярославль

КНОПКА RESET ДЛЯ БК-0010(.01)

При отладке программ БК иногда «зависает», и если программа не была записана, она безвозвратно теряется. Чтобы избежать этого, надо установить переключатель, обеспечивающий перезапуск процессора. Его можно подключить, не разбирая компьютера, к контактам разъема МПИ А1 и А2.

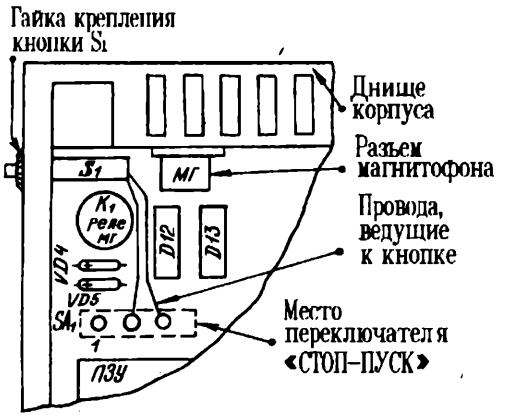
Теперь, если на короткое время замкнуть контакты и несколько раз быстро нажать клавишу СТОП, то произойдет выход в монитор без потери программы. Выключатель удобно разместить вместо вставки на задней стороне блока МСТД. Таким способом можно выходить из программ, блокирующих нажатие клавиши СТОП, а также отсоединять блок МСТД без потери программы.

От редакции

Контакты порта МПИ А1 и А2, по сути, дублируют собой контакты переключателя «СТОП-ПУСК», который устанавливался в отсеке пользователя на старой модели БК-0010. При их замыкании срабатывает прерывание IRQ1 («Останов»), а после размыкания БК перезапускается с адреса #100000, запуская затем программу по адресу #120000. Такой метод, хорошо известный под названием «кнопка RESET» многим пользователям БК, работающим с дисководом и дисковой операционной системой в расширенном ОЗУ, весьма удобен для возврата в дисковую ОС из программ, не имеющих «стандартного» выхода. Если же контроллера дисковода нет или он не подключен, то происходит перезапуск ФОКАЛа или БЕЙСИКа соответственно. Кроме того, перезапуск с помощью кнопки RESET более щадящ для элементов схемы компьютера, чем выключение и повторное включение.

Действительно, если сразу же после размыкания контактов нажать клавишу СТОП (неоднократно, так как с первого раза она, как правило, не срабатывает), то происходит выход в монитор. Но нажимать СТОП нужно быстро, иначе успеет произойти запуск по адресу #120000, и программа пользователя в ОЗУ может быть потеряна.

Следует отметить, что способ подключения кнопки RESET к контактам А1 и А2, хотя и позволяет не разбирать БК, но не очень удобен, так как в этом случае для работы с кнопкой блок МСТД (или контроллер дисковода) должен быть обязательно подключен к БК. Поэтому, если ваш компьютер уже не стоит на гарантии, лучше все же подключить кнопку перезапуска непосредственно в схему, на то место, где ранее устанавливался заводской переключатель «СТОП-ПУСК». Для этого нужно снять крышку корпуса БК вместе с платой клавиатуры (крышка крепится к днищу пятью шурупами). Контактные площадки, на которых ранее устанавливался переключатель «СТОП-ПУСК», расположены в верхнем левом углу системной платы (обозначение SA1). Выводы от кнопки подпаиваются к среднему и правому контактам (см. рис.).



Саму ее при этом удобно разместить в верхнем левом углу днища корпуса, для чего в нем сверлится отверстие. Кнопка может быть любого типа, но обязательно пружинная, не фиксирующаяся в нажатом положении. Теперь для перезапуска достаточно нажать и отпустить ее, что удобнее, чем щелкать переключателем. Кстати, и само расположение кнопки с левого «борта» БК более удобно, чем сзади на блоке МСТД или КНГМД, в чем легко можно убедиться на практике.

С. Л. Ерохин,
г.Харьков

ИСПРАВЛЕНИЕ ОШИБКИ РЕАЛИЗАЦИИ ОПЕРАТОРА LINE В БЕЙСИКЕ БК-0011

При работе с машиной БК-0011 на языке Бейсик обнаружилась одна неприятная особенность работы графики. Если при использовании операторов LINE и DRAW линия чертится справа налево, то машина проводит вертикальную линию и может зависнуть или начнутся разные «фокусы». По этой же причине машина не рисует прямоугольники. Данная программа позволяет решить эту проблему. Программа перехватывает вектор EMT на себя, и дальнейшая обработка идет через нее. При обнаружении команды EMT 32, которая отвечает за проведение линии, программа преобразует линию справа налево в аналогичную линию слева направо

```
10 REM ЕРОХИН С.Л. ХАРЬКОВ 1991
20 A1%=&O1000
30 A%=A1%
40 IF PEEK(&O30)=A% THEN 160
50 READ D%
60 POKE A%,D%
70 A%=A%+2%
80 IF D%<>&O137 THEN 50
90 POKE A%,PEEK(&O30)
100 POKE &O30,A1%
110 DATA &O10546,&O16605,&O2,&O14505,
    &O22705
120 DATA &O104032,&O1022,&O26060,
    &O2,&O6,&O103416
130 DATA &O16046,&O2,&O16046,&O4,
    &O16060,&O6,&O2
140 DATA &O16060,&O10,&O4,&O12660,
    &O10,&O12660
150 DATA &O6,&O12605,&O137
160 END
```

Программу достаточно запустить один раз (например, перед началом занятия). Можно также вставить ее в начало программы, использующей графику. После запуска программа будет работать до выключения машины.

(Проверка работоспособности предложенной программы на БК-0011 не проводилась. — *Прим. ред.*).

А. Н. Копосов,
г. Пыщуг

ПОДКЛЮЧЕНИЕ МЫШИ К БК-0011(М)

Чтобы полноценно работать в среде ОС БК-0011 с мышью, например в графическом редакторе PNT11M.SAV, пользователь БК-0011М, работавший до этого в режиме эмуляции БК-0010, вынужден перепаять строб мыши с вывода В10 УП на В25 или запаять В10 и В25 параллельно. Большинство программ, работающих с принтером, этого не замечают, однако некоторые из них при таком способе подключения вносят искажения в передаваемые по интерфейсу ИРПР данные, что приводит к печати совершенно абсурдной информации или к постоянному переводу формата.

Желающим сохранить полную работоспособность таких профессиональных программ для БК, как HCOPY6.MVI и EDALT3M при работе в средах NORD и ANDOS, и в то же время иметь возможность работы с мышью в ОС БК-0011 советую предусмотреть переключатель строба мыши. Правда, для полноценной работы с принтером как в ОС БК-0011, так и в средах эмуляторов БК-0010 пользователю придется предусмотреть также переключатели двух стандартов распайки сигналов ЗАНЯТО и СТРОБ — старого В31 и А28 и нового В29 и А25 соответственно. Как показывает практика, в корпусе разъема УП хватает места для всех необходимых переключателей и даже для микросхемы инвертирования сигналов при подключении принтера МС6313.

Примечание. Сказанное выше касается тех пользователей, которые подключили принтер, мышь и джойстик не через блок КМ, а напрямую к разъему УП.

«И вот мы в мире, исполненном умопостижимой красоты»

Н. Мальбранш

В. Е. Иванов,
г. Сергиев-Посад

ИГРА «STEP»

Перед вами окантованное кирпичами игровое поле. Когда делается очередной шаг, в том месте, где вы стояли, появляется новый кирпич (его вид можно изменить на любой другой). Таким способом нужно заполнить кирпичами все поле, не оставляя пустых мест (исключение — шестой этап). Если в течение заданного короткого промежутка времени вы не указали направление хода нажатием соответствующей клавиши, то компьютер делает ход самостоятельно.

В игре восемь этапов. У вас 10 попыток. При переходе на следующий этап добавляется одна попытка. Предусмотрена пауза, действующая, пока нажата клавиша пробела. Имеется хорошее звуковое сопровождение.

В строках 240—560 расположена подпрограмма, рисующая кирпич. Строки 1540—1890 проверяют, свободна ли клетка, в которую вы (или компьютер) собираетесь перейти. Если она занята — выбирается какая-либо из двух соседних клеток. Если и они заняты, значит, вы попали в тупик — сбрасывается одна попытка и тот же этап предлагается пройти с начала.

В каждом этапе на поле уже стоят несколько кирпичей, что увеличивает сложность игры. Их расположение программируется следующим образом. В каждом операторе DATA в строках 2020—2350 записываются по порядку следующие величины:

— число кирпичей на поле;

— координаты X и Y старта (в игре он обозначен буквой «Н» и рисуется оператором DRAW);

— направление, в котором начнется движение, если первый ход будет делать компьютер (1—влево, 2—вверх, 3—вниз, 4—вправо);

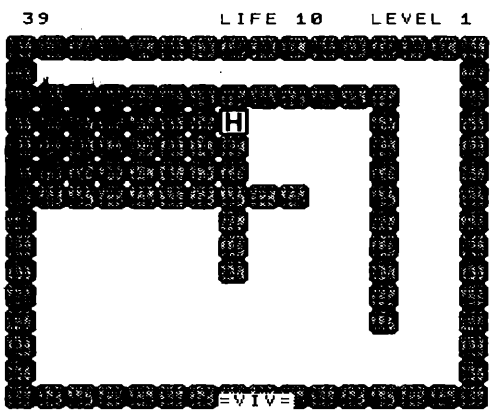
— количество пустых клеток, которые необходимо заполнить кирпичами (это число в сумме с количеством кирпичей, уже имеющихся на поле, не должно превышать 181);

— список адресов кирпичей (их должно

быть столько, сколько указано кирпичей в первом аргументе);

— адрес клетки, с которой начнется движение (должен указывать место, на котором расположен старт).

В качестве адресов в операторах DATA задаются восьмеричные числа, которые в сумме с &O40000 дают адреса экранного ОЗУ, с которых будут рисоваться кирпичи или метка старта: 0 — верхняя левая позиция экрана, 36074 — нижняя правая. (Размеры «кирпича» равны 16x16 точек. — *Прим. ред.*) Следует также учесть, что окантовка из 58 кирпичей (пределы игровой площадки) рисуется самой программой, следовательно, нет смысла помещать кирпич на границе поля, а также в верхней строке экрана, служащей для отображения информации.




```
10 RESTORE 30
20 READ L%,E%,V$
30 DATA 10%,1%
40 DATA "R15D15L15U15R1D15U15R13
   D14BM-4,U12D5L6U5D11C2R1U5R4BM-3,-2
   L1U4BM+6,-1D12;"
50 ? CHR$(140)CHR$(140)
60 GOTO 570
70 COLOR ,1
80 COLOR ,4
90 E%=E%+1%
100 IF E%=9%TH120ELSE L%=L%+1%
110 GOTO 690
120 POKE 112%,&H4000
130 ? "    ПРИМИТЕ ПОЗДРАВЛЕНИЯ !  "
140 CLS
150 GOTO 820
160 B%=0%
170 B%=B%+1%
180 POKE -50%,224%
190 B1%=0%
200 B1%=B1%+1%
210 IF B1%<B2%TH200
220 POKE -50%,160%
230 IF B%<B3%TH170 EL RET
240 POKE &O40000+M,&O177760
250 POKE &O40002+M,&O7777
260 POKE &O40100+M,&O105214
270 POKE &O40102+M,&O36210
280 POKE &O40200+M,&O21043
290 POKE &O40202+M,&O171052
300 POKE &O40300+M,&O125253
310 POKE &O40302+M,&O175252
320 POKE &O40400+M,&O21043
330 POKE &O40402+M,&O171052
340 POKE &O40500+M,&O105213
350 POKE &O40502+M,&O174210
360 POKE &O40600+M,&O21043
370 POKE &O40602+M,&O171052
380 POKE &O40700+M,&O105213
390 POKE &O40702+M,&O174210
400 POKE &O41000+M,&O21043
410 POKE &O41002+M,&O171052
420 POKE &O41100+M,&O105213
430 POKE &O41102+M,&O174210
440 POKE &O41200+M,&O125253
450 POKE &O41202+M,&O171252
460 POKE &O41300+M,&O105213
470 POKE &O41302+M,&O174210
480 POKE &O41400+M,&O21043
490 POKE &O41402+M,&O171052
500 POKE &O41500+M,&O105217
510 POKE &O41502+M,&O176210
520 POKE &O41600+M,&O177774
530 POKE &O41602+M,&O37777
540 POKE &O41700+M,&O177760
550 POKE &O41702+M,&O7777
560 RETURN
570 ? CHR$(&H94)CHR$(&H9E)CHR$(&H91)
580 FOR I=1 TO36
590 READ M
600 GOSUB 240
610 NEXT I
620  DATA &O4016,&O2012,&O2006,&O4002,
     &O6002,&O10006,&O10012, &O12016
630 DATA &O14016,&O16012,&O16006,
     &O14002,&O6026,&O6032, &O6036,&O10032
640  DATA &O12032,&O14032,&O16032,&O6046,
     &O6052,&O10046, &O12046,&O12052
650 DATA &O14046,&O16046,&O16052,
     &O16062,&O14062, &O12062,&O10062
```

```
660 DATA &O6062,&O6066,&O7072,
    &O11072,&O12066
670 ? AT(5,19)"ИВАНОВ ВЛАДИМИР 2-67-03"
AT(5,20)"Г.СЕРГИЕВ ПОСАД
1991Г."
680 IF INKEY$=""TH680
690 CLS
700 POKE 112%,&H4000
710 ? "                                    ";E%
720 POKE 112%,&H4000
730 COLOR 2
740 ? "                              LEVEL"
750 COLOR 1,0
760 POKE 112%,&H4000
770 ? "                    ";L%
780 COLOR 2
790 POKE 112%,&H4000
800 ? "            LIFE"
810 COLOR 1,0
820 RESTORE 870
830 FOR I=1 TO58
840 READ M
850 GOSUB 240
860 NEXT I
870 DATA &O2000,&O2004,&O2010,&O2014,
    &O2020,&O2024, &O2030,&O2034
880 DATA &O2040,&O2044,&O2050,&O2054,
    &O2060,&O2064, &O2070,&O2074
890 DATA &O4000,&O6000,&O10000,&O12000,
    &O14000,&O16000&O20000,&O22000
900 DATA &O24000,&O26000,&O30000,&O32000,
    &O34000,&O36000,&O36004,&O36010
910 DATA &O36014,&O36020,&O36024,&O36030,
    &O36034,&O36040,&O36044,&O36050
920 DATA &O36054,&O36060,&O36064,&O36070,
    &O36074,&O34074,&O32074,&O30074
930 DATA &O26074,&O24074,&O22074,&O20074,
    &O16074,&O14074,&O12074,&O10074
940 DATA &O6074,&O4074
950 ON E% GOTO 2000,2050,2100,2140,
    2190,2230,2270,2310,2360
960 READ U%,X%,Y%,A%,Q%
970 COLOR 2
980 ? AT(14,23)"=VIV="AT(0,0)
990 COLOR 1
1000 FOR I=1 TOU%
1010 READ M
1020 GOSUB 240
1030 NEXT I
1040 LINE (X%,Y%)-(X%+15,Y%+15),3,B
1050 PAINT (X%+4,Y%+4),4,3
1060 DRAW "BM=X%;,=Y%;XV$;"
1070 READ M
1080 B2%=2%
1090 B3%=240%
1100 FOR I=1 TO3
1110 GOSUB 160
1120 FOR W=1 TO300
1130 NEXT W,I
1140 B3%=440%
1150 B2%=1%
1160 GOSUB 160
1170 FOR I=1 TO350
1180 NEXT
1190 G=0
1200 FOR B=0 TO8
1210 POKE -50%,224%
1220 POKE -50%,160%
1230 NEXT B
1240 POKE 112%,&H4000
1250 ? ;F%
1260 F%=F%+1%
1270 G=G+1
1280 IF G=Q%+1% TH70
1290 LOCATE 0,0,0
1300 C%=0%
1310 C%=C%+1%
1320 A$=INKEY$
1330 DZ=INP(&O177716,&O100)
1340 IF PEEK(&O177662)=32 AND DZ=0 TH1460
1350 IF A$<>""TH1400
1360 IF C%<100 TH1310
1370 ON A% GOTO 1720,1630,1540,1810
1380 GOSUB 240
1390 GOTO 1200
1400 IF ASC(A$)=8 THA%=1%
1410 IF ASC(A$)=26 THA%=2%
1420 IF ASC(A$)=27 THA%=3%
1430 IF ASC(A$)=25 THA%=4%
1440 IF ASC(A$)=32 TH1460
1450 GOTO 1370
1460 IF INP(&O177716,&O100)=0
     THF%=F%-1%EL1320
1470 POKE 112%,&H4000
1480 ? ;F%
1490 LOCATE 0,0,0
1500 EF=0
1510 EF=EF+1
1520 IF EF<300 TH1510
1530 IF F%<=0 TH1900EL1460
1540 IF PEEK(&O40000+M+(&O2000))<>0TH1570
1550 M=M+(&O2000)
1560 GOTO 1380
1570 IF PEEK(&O40000+M-(&O4))<>0TH
     Z=PEEK(&O40000+M+(&O4))EL1610
1580 IF Z<>0TH1900
1590 A%=4%
1600 GOTO 1370
1610 A%=1%
1620 GOTO 1370
1630 IF PEEK(&O40000+M-(&O2000))<>0TH1660
1640 M=M-(&O2000)
```

```
1650 GOTO 1380
1660 IF PEEK(&O40000+M+(&O4))<>0
     TH Z=PEEK(&O40000+M-(&O4))EL1700
1670 IF Z<>0TH1900
1680 A%=1%
1690 GOTO 1370
1700 A%=4%
1710 GOTO 1370
1720 IF PEEK(&O40000+M-(&O4))<>0TH1750
1730 M=M-(&O4)
1740 GOTO 1380
1750 IF PEEK(&O40000+M+(&O2000))<>6TH
      Z=PEEK(&O40000+M-(&O2000))EL1790
1760 IF Z<>0TH1900
1770 A%=2%
1780 GOTO 1370
1790 A%=3%
1800 GOTO 1370
1810 IF PEEK(&O40000+M+(&O4))<>0TH1840
1820 M=M+(&O4)
1830 GOTO 1380
1840 IF PEEK(&O40000+M-(&O2000))<>0TH
      Z=PEEK(&O40000+M+(&O2000))EL1880
1850 IF Z<>0TH1900
1860 A%=3%
1870 GOTO 1370
1880 A%=2%
1890 GOTO 1370
1900 B=0
1910 B=B+1
1920 POKE -50%,224%
1930 POKE -50%,160%
1940 IF B<600 TH1910
1950 IF L%-1%<0% TH1970 ELSE L%=L%-1
1960 GOTO 690
1970 POKE 112%,&H4000
1980 ? "          НЕУДАЧА          "
1990 IF INKEY$="" TH1990 EL10
2000 RESTORE 2020
2010 GOTO 960
2020 DATA 14%,112%,48%,2%,167%,&O16004,
     &O16010,&O16014,&O16020,&O16024
2030 DATA &O16030,&O16034,&O16040,&O16044,
     &O12034,&O14034,&O20034,&O22034
2040 DATA &O24034,&O10034
2050 RESTORE 2070
2060 GOTO 960
2070 DATA 15%,176%,32%,1%,166%,&O6020,
     &O10020,&O12020, &O26020,&O30020
2080 DATA &O32020,&O20030,&O20034,&O20040,
     &O20044,&O32054,&O30054,&O26054
2090 DATA &O12054,&O10054,&O6054
2100 RESTORE 2110
2110 GOTO 960
2120 DATA 7%,32%,32%,1%,174%,&O14010,
     &O22010,&O30010,&O30064,&O22064
2130 DATA &O14064,&O6064,&O6010
2140 RESTORE 2160
2150 GOTO 960
2160 DATA 14%,112%,48%,2%,167%,&O16004,
     &O16010,&O16014,&O16020,&O16024
2170 DATA &O16030,&O16034,&O16040,&O16044,
     &O12034,&O14034,&O20034,&O22034
2180 DATA &O24034,&O10034
2190 RESTORE 2210
2200 GOTO 960
2210 DATA 13%,96%,80%,2%,168%,&O14024,
     &O16024,&O20030,&O16030,&O16034
2220 DATA &O14034,&O22040,&O24040,&O24044,
     &O22044,&O20044,&O22050,&O24050,&O14030
2230 RESTORE 2250
2240 GOTO 960
2250 DATA 9%,80%,112%,4%,164%,&O16030,
     &O22030,&O14034,&O24034,&O14040
2260 DATA &O24040,&O16044,&O22044,
     &O20050,&O20024
2270 RESTORE 2290
2280 GOTO 960
2290 DATA 7%,176%,64%,3%,174%,&O26020,
     &O24024,&O22030,&O20034,&O20040
2300 DATA &O16044,&O14050,&O12054
2310 RESTORE 2330
2320 GOTO 960
2330 DATA 19%,176%,144%,4%,162%,&O12014,
     &O12020,&O12024,&O12050,&O12054
2340 DATA &O12060,&O14020,&O14054,&O20004,
     &O20070,&O24020,&O26014,&O26020
2350 DATA &O26024,&O26050,&O26054,
     &O26060,&O32010, &O32064,&O24054
2360 ? AT(5,8)"ПРОДОЛЖЕНИЕ СЛЕДУЕТ !!!"
2370 A$=INKEY$
2380 FOR N=1 TO4
2390 RESTORE 2530
2400 FOR I=1 TO4
2410 READ B2%,B3%
2420 GOSUB 160
2430 IF INKEY$<>""TH2540
2440 NEXT I,N
2450 RESTORE 2520
2460 FOR I=1 TO12
2470 READ B2%,B3%
2480 GOSUB 160
2490 IF INKEY$<>""TH2540
2500 NEXT
2510 GOTO 2380
2520 DATA 8,37,7,43,6,50,7,43,6,50,5,63,
     6,50,5,63, 4,75,5,63,4,75,3,100
2530 DATA 2,100,3,100,4,75,1,100,2,150,3,100
2540 B2%=16%
2550 B3%=400%
2560 GOSUB 200
```

Уважаемые читатели!

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

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

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

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

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

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

Из программ в первую очередь рассматриваются распечатанные на принтере. Все,

Уважаемые читатели!

Подписаться на журнал «Персональный компьютер БК-0010 — БК-0011М» на 2-е полугодие 1994 года можно в любом отделении связи. В каталоге ЦРПА «Роспечать» данные о журнале вы найдете на стр.82.

Чтобы получать журнал через редакцию:

* частным лицам необходимо перечислить за каждый выпуск 8 000 руб. для жителей России и 11 000 руб. для жителей ближнего зарубежья на расчетный счет редакции;

* предприятиям и организциям необходимо перечислить за каждый выпуск 16 000 руб. для России и 22 000 руб. для ближнего зарубежья на расчетный счет редакции.

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

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

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

101835, Москва, Садовая Сухаревская, 16.

Редакция журнала «Информатика и образование».

Стоимость услуги складывается из подписной цены журнала, стоимости пересылки и общеиздательских расходов.

что вручную переписано с экрана, как правило, содержит ошибки, а следовательно, может потребовать серьезной доработки, что, в свою очередь, еще больше увеличивает время подготовки статьи к печати.

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

Аналогичным способом могут быть подготовлены иллюстрации к статье. Формат представления — файл, содержащий копию экранного ОЗУ (начиная с адреса 40000 или 42000, т.е. с учетом служебной строки или без нее), причем рулонное смещение должно быть нормализовано (адрес 40000 должен соответствовать верхнему левому углу изображения). При выполнении рисунков в графических редакторах серии GRAF или БК-PAINT редакции могут быть переданы файлы, записанные в формате этих программ. И наконец, при подготовке иллюстраций на IBM-совместимых компьютерах нужно применять формат .PCX (черно-белый вариант). Для этой цели проще всего использовать резидентную программу типа PZP, позволяющую сохранять в файле (либо распечатывать на принтере) копию экрана IBM. (В меню PZP нужно выбрать следующие установки: режим качества «B&W Standart», без инверсии, окно выделения во весь экран, размеры для печати соответствуют листу A4, формат при записи в файл — «Ventura».)

Адрес редакции: 103051, Москва, ул.Садовая Сухаревская, д.16, комн.9.
Телефон: (095) 151-19-40.
Факс: (095) 208-67-37.
E-Mail: mail@infoobr.msk.su

ЗАЯВКА

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

_____

(адрес подписчика с почтовым индексом)

_____

(фамилия, имя, отчество полностью)

_____

(номер выпуска и год издания)

_____

(общее количество экземпляров)

Перечислено на расчетный счет _____

_____ руб.

(общее количество экземпляров x *стоимость одного экземпляра)*

Платежное поручение № _____ от _____ 199___ г.

СОДЕРЖАНИЕ

ПЕРСОНАЛЬНЫЙ КОМПЬЮТЕР БК-0010 – БК-0011М

Главный редактор
Васильев Б. М.

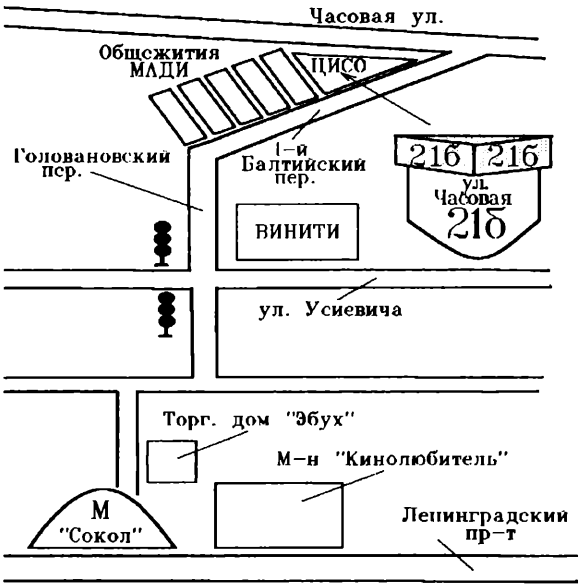
Редактор
Усенков Д. Ю.

Художник
Смирнов А. А.

Корректор
Антонова В. С.

Компьютерная верстка
Панченко О. Н.

Наш адрес: Москва, ул. Часовая, 21Б, помещение ЦИСО, комн.20
Как к нам добраться:

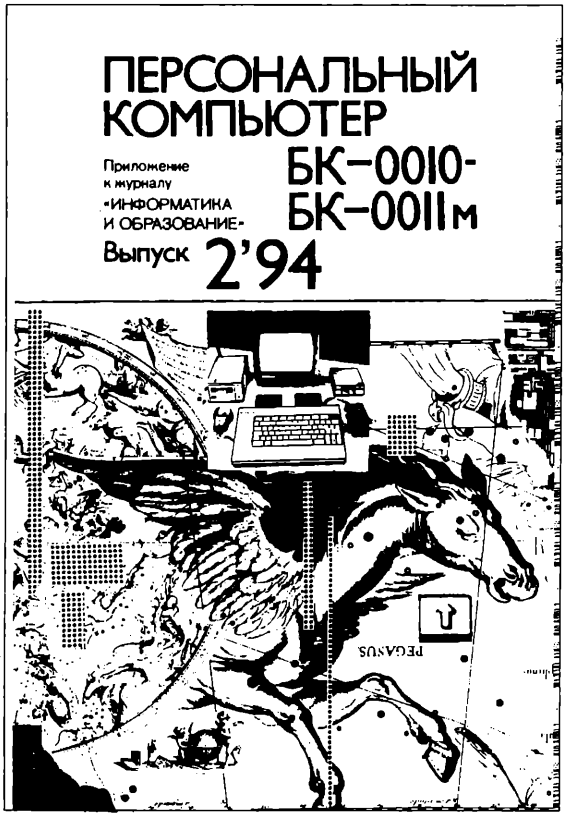


ПЕРСОНАЛЬНЫЙ КОМПЬЮТЕР БК-0010 - БК-0011М

Подписано в печать с оригинал-макета издательства «Информатика и образование» 10.05.1994 г. Формат 70×100 $^{1}/_{16}$. Бумага офсетная. Усл.печ.л. 6,5. Тираж 3000 экз. Заказ №2516 Цена 800 руб. (по подписке). В розничной продаже цена договорная.

Ордена Трудового Красного Знамени Чеховский полиграфический комбинат Комитета Российской Федерации по печати
142300, г. Чехов Московской области

СОДЕРЖАНИЕ ВЫПУСКА 2,94:



Цена 800 руб. по подписке

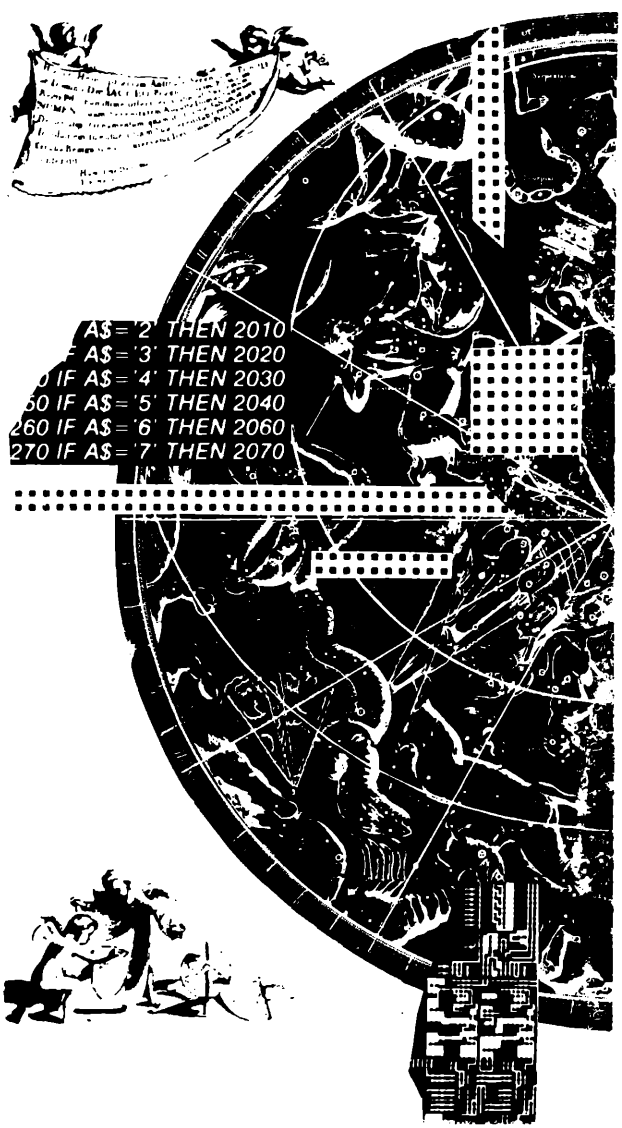
ИНФО

ПЕРСОНАЛЬНЫЙ КОМПЬЮТЕР БК-0010, БК-0011м

Приложение к журналу «ИНФОРМАТИКА И ОБРАЗОВАНИЕ»

Выпуск 1'94

Индекс 73177



"Во всех случаях, когда речь идет о создании нового, будь то храм, бактерия, сонет, фуга или текстовый процессор, главное - это архитектура, а не материал."

Алан Кэй

...А для низкой жизни были числа,
Как домашний, подъяремный скот,
Потому что все оттенки смысла
Умное число передает.

Н.Гумилев