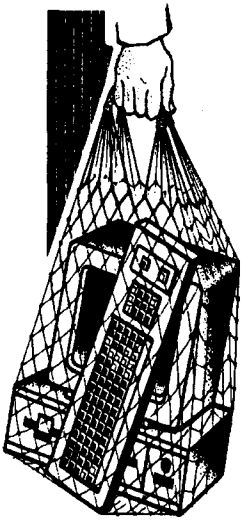
УВАЖАЕМЫЕ ЧИТАТЕЛИ!

Приглашаем вас в Центр БК, открытый совместными усилиями редакции журнала и фирмы «АльтПро». Вашему вниманию предлагается новейшее аппаратное и программное обеспечение для БК-0010(.01) и БК-0011(М):



— цветные и черно-белые мониторы;

— [redacted]роллеры для них;

— [redacted]ски с контроллерами;

— [redacted]я БК, мониторов и дисководов;

— кабели, разъемы и прочие комплектующие

а также

— новые версии операци[redacted] ANDOS, NORD и MK-DO

— база данных BASIS[redacted]

— графический реда[redacted]T;

— текстовый проце[redacted]

— музыкальный редак[redacted]

— разнообразные драйве[redacted]алиты, системные и прикладные программы;

— богатый выбор игр и мно[redacted]другое.

Специалисты Центра[redacted] предоставят вам консультации по подключени[redacted]ериферийных устройств и использованию программ[redacted]ого обеспечения, окажут помощь в ремонте БК и периферии.

Центр БК расположен в помещении редакции журнала по адресу: Москва, ул. Часовая, д. 21б, комн. 36. Для заказа каталогов имеющегося программного и аппаратного обеспечения по почте достаточно прислать в редакцию письмо с заявкой и вложенным пустым конвертом с надписанным обратным адресом и наклеенными почтовыми марками

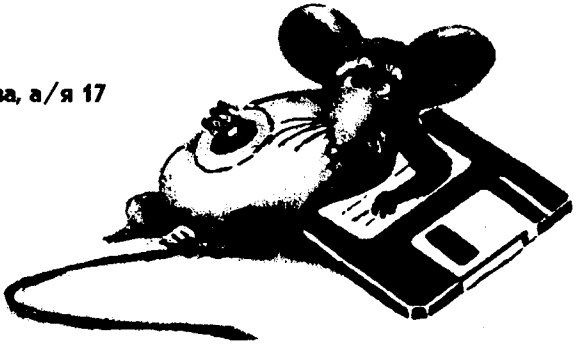
Здесь же вы можете приобрести отдельные выпуски журналов «Персональный компьютер БК-0010 — БК-0011М» и «Персональный компьютер УКНЦ», а также оформить полугодовую подписку на эти издания с получением экземпляров в редакции или по почте.

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

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

Телефон: (095) 151-19-40

E-Mail: mail@infoobr.msk.su



ПЕРСОНАЛЬНЫЙ КОМПЬЮТЕР БК-0010, БК-0011м

Приложение
к журналу
«ИНФОРМАТИКА
И ОБРАЗОВАНИЕ»

3'95 (9)

Издается с 1993 г.

В НОМЕРЕ

Москва «Информатика и образование» 1995

Авторы выпуска

Бриндеев А. В.
Глускер А. И.
Зальцман Ю. А.
Капустин Е. И.
Милюков А. В.
Неробеев С. М.
Романов Д.
Румянцев С. К.
Сорокин А. В.
Усенков Д. Ю.

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

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

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

Завершаем публикацию неформального учебника по программированию на ассемблере для начинающих пользователей БК-0010(.01). Начало см. в №1—4 за 1994 г. и №1—2 за 1995 г.

Ю. А. Зальцман,
г. Алма-Ата

МикроЭВМ БК-0010. Архитектура и программирование на языке ассемблера

Схемотехника БК-0010

У определенной части пользователей персональных компьютеров существует устойчивый интерес к схемотехнике и электронике. Оправдан ли этот интерес применительно к БК? Стоит ли изучать схемотехнику этого компьютера, и если да, то для чего?

Во-первых, любая техника рано или поздно выходит из строя. Хотя сервисная сеть по ремонту БК-0010 и существует, воспользоваться ее услугами иной раз довольно сложно (скажем, для пользователей, живущих на периферии). Зная же основы схемотехники БК, можно попытаться сделать это самостоятельно*. Однако сразу следует оговориться, что начинающим пользователям не стоит даже пытаться это сделать. Такая возможность открыта лишь для опытных радиолюбителей и для профессионалов в области микросхемотехники. Помимо теоретических знаний, отсутствующих у подавляющего большинства пользователей БК (наивно думать, что эта статья позволит приобрести их в нужном объеме), имеется еще масса «тонкостей» чисто практического свойства, известных лишь тем, кто работал с микропроцессорной техникой. Существует также множество приемов проверки и отладки микроэлектронных схем, научить которым «заочно» вообще невозможно, их каждый приобретает самостоятельно лишь после многих лет практической работы: опытный ремонтник «чувствует» неисправность, даже порой не отдавая себе отчета, каким образом это у него получается. Итак, в смысле возможностей ремонта БК эта часть книги полезна только тем, кто МОЖЕТ ЕЮ ВОСПОЛЬЗОВАТЬСЯ. Всегда помните древнейшую поговорку: «То, что позволено Юпитеру, не позволено быку» — и правильно оценивайте свои силы. То же самое замечание в полной мере относится и к случаю подключения к БК-0010 дополнительных устройств (дисковода, принтера, дополнительной памяти и т. п.)**.

Во-вторых, этот материал может оказаться полезным при изучении других ЭВМ с аналогичной организацией («Электроника-60» или ДВК), станков с числовым программным управлением и т. п.

И наконец, кто знает, не послужит ли для кого-либо этот раздел стимулом, который приведет в дальнейшем к «большой электронике»? Словом, данный материал будет полезен «для общего развития».

Было бы безнадежным пытаться изложить схемотехнику БК0010 «корректно», т. е. с введением и объяснением терминологии, описанием работы отдельных схем, узлов и т. п. — это потребовало бы нескольких книг. Поэтому всех интересующихся подробностями приходится отослать к обширной литературе по радиоэлектронике (список литературы см. в №1 за 1994 г.). После всех этих оговорок приступим к описанию схемы БК-0010. Оно будет основано на принципиальной схеме компьютера, опубликованной в журнале «Информатика и образование», №4 за 1990 г., с. 74—77, а также в журнале «Персональный компьютер БК-0010 — БК-0011М», №2 за 1994 г. Несмотря на наличие в ней значительного количества неточностей и ошибок (впрочем, многие из них

* Ремонтная документация и техническое описание БК-0010, предоставленные редакции Павловопосадским заводом, опубликованы в №2, 3 и 4 за 1994 г. — *Прим. ред.*

** По вопросам ремонта БК и периферийных устройств для нее, а также подключения периферии к компьютеру можно обращаться в редакцию по телефону: (095) 151-19-40. — *Прим. ред.*

исправлены в схеме, опубликованной в №2 за 1994 г.), это единственная доступная пользователю схема БК-0010. Об отличиях схемы БК-0010.01 мы будем упоминать по мере возможностей, они в основном незначительны и непринципиальны.

БК-0010 выполнена на основе микропроцессорного набора серии К1801. Перечень микросхем (МС) этого типа, выполненных по так называемой n-МОП технологии, включает помимо нескольких видов ЦП набор специализированных микросхем-контроллеров. Все они изготовлены на базе одной микросхемы, групповое название которой — К1801ВП1, и отличаются связями логических элементов — своего рода «микропрограммой», зашитой в них в процессе изготовления. Каждому контроллеру присвоен свой порядковый номер (например: К1801ВП1-014, К1801ВП1-037, К1801ВП1-128 и т. д.). Такая технология помимо узкой специализации и, следовательно, высоких функциональных качеств каждого контроллера (во всяком случае таковы они по мнению разработчиков, пользователи же с этим мнением не всегда согласны) позволяет легко и без больших затрат расширять их набор почти неограниченно.

Все функциональные узлы (МС, блоки и т. п.) связаны в микроЭВМ линиями связи, называемыми в совокупности СИСТЕМНОЙ МАГИСТРАЛЬЮ (СиМ). Системная магистраль имеет 16 линий «адрес—данные» (шина АД) и еще несколько «служебных линий» (СЛ) для передачи специальных сигналов. Обмен по СиМ производится АСИНХРОННО. При таком виде обмена каждое из входящих в систему устройств может быть включено в СиМ в произвольном месте и работает самостоятельно, в своем временном режиме («ритме»). Все устройства, подключенные к СиМ, делятся на АКТИВНЫЕ (способные занимать и освобождать магистраль «по своей воле») и ПАССИВНЫЕ (вынужденные ждать, пока магистраль им будет предоставлена активным устройством). Другими словами, активным считается устройство, способное самостоятельно инициировать и осуществлять обмен по СиМ, а пассивным — устройство, работающее под «надзором» активного. Активным устройством (АУс) в нашем компьютере является только ЦП, все остальные устройства в его составе — пассивные. Каждое из пассивных устройств (ПУс) имеет в общем адресном пространстве ЭВМ свой АДРЕС (или группу адресов), по которому к нему может обратиться АУс.

Как же происходит обмен информацией по СиМ? В начале цикла обмена магистраль «занята» АУс, в нашем случае ЦП. Он выставляет на шину АД СиМ адрес того ПУс, к которому хочет обратиться. Если это системный регистр, то этот адрес будет для него единственным, который вызывает реакцию, а если это, например, ОЗУ или ПЗУ, то при обращении ко всей области адресов, обслуживаемой данным ПУс, его реакция будет одинаковой. Выставив адрес, ЦП выдает по одной из СЛ специальный сигнал, называемый «синхроимпульсом активного устройства» (СИА) или сигналом SYNC (синхронизация обмена). Пассивное устройство, получив сигнал SYNC (он подается на все устройства ЭВМ одновременно), фиксирует (обычно в специальном регистре) полученный по шине АД адрес. Дальнейший порядок обмена зависит от того, что нужно сделать: передать данные в ПУс (цикл ВЫВОД, или ЗАПИСЬ) или принять их от него (цикл ВВОД, или ЧТЕНИЕ). В зависимости от этого ЦП почти одновременно с сигналом SYNC выставляет на других СЛ еще один из двух сигналов — DIN (ввод) или DOUT (вывод). Рассмотрим сначала цикл ввода.

Если принятый по сигналу SYNC адрес соответствует адресу данного ПУс, то оно по отдельной СЛ выдает сигнал ответа, называемый синхроимпульсом пассивного устройства (СИП) или сигналом RPLY. Получив его, ЦП уже знает, что вызов получен адресатом. А что случится, если вызываемое ПУс отсутствует в составе ЭВМ или попросту неисправно? Неужели ЦП будет ждать до бесконечности и работа ЭВМ на этом бесславно завершится? Конечно нет! ЦП будет ждать ровно столько, сколько положено (а именно 64 такта специальной, подаваемой на него опорной, или тактовой, частоты), а затем выполнит переход по вектору 4, тем самым констатируя, что вызываемое ПУс на запрос не ответило (произойдет прерывание по зависанию канала связи).

Итак, ответ получен. Что дальше? По-прежнему выбор порядка действий за ЦП, ему «водить» в этой игре. Вернемся чуть назад. Выдав сигнал DIN, ЦП освобождает шину АД СиМ (снимает сигналы адреса) и, как уже говорилось, ждет ответа RPLY. А ПУс одновременно с RPLY (или даже чуть раньше) выставляет на шину АД данные (которые, собственно, и требовал от него ЦП). Получив ответ ПУс, ЦП принимает данные, которые ПУс установило на шине АД (записывает их в один из своих буферных регистров) и снимает сигнал DIN, показывая тем самым ПУс, что данные приняты. ПУс в ответ на это освобождает СиМ, снимая сигнал данных и сигнал ответа RPLY. Теперь ЦП наконец-то снимает сигнал SYNC, тем самым завершая цикл обмена. Пока сигнал SYNC был на СЛ СиМ, он свидетельствовал, что идет обмен, это было как бы предупреждение всем остальным АУс (если они есть в системе): «Не мешать!» При чтении информации ЦП всегда обращается только по четному адресу (бит 00 слова адреса всегда равен нулю) и записывает в свой бу-

фер только целое слово. Если же нужен байт, то он выделяется из буфера уже по окончании обмена.

А что происходит, если нужно записать что-либо по адресу ПУс? Тогда ЦП вместо сигнала DIN выставляет по другой СЛ сигнал DOUT (вывод), причем до подачи этого сигнала он успевает освободить шину АД от адреса и выставить на ней данные, которые нужно передать ПУс. Получив сигнал DOUT, ПУс не занимает магистраль, а принимает выставленные на ней данные и только затем выдает ответ RPLY, который в данном случае означает «данные приняты», после чего ЦП снимает сигнал SYNC (и данные), завершая обмен. Для того чтобы можно было по адресу ПУс записать не только слово, но и байт, в цикле «вывод» работает еще одна СЛ, по которой в подобном случае вместе с сигналами SYNC и DOUT передается специальный сигнал WTBT, свидетельствующий о передаче байта. Естественно, что при этом передаваемый по сигналу SYNC адрес может быть нечетным (в отличие от цикла ВВОД).

Все служебные сигналы передаются по СиМ БК-0010 НИЗКИМ УРОВНЕМ, т. е. о наличии на соответствующих СЛ СиМ сигналов DIN, DOUT, RPLY, WTBT свидетельствует уровень «логического нуля», а уровень «логической единицы» означает отсутствие сигнала (как говорят, «линия не активна»).

Существуют и другие виды обмена по СиМ (например, «ввод—модификация—вывод»), но мы их рассматривать не будем, как и обмен при наличии других АУс (например, второго процессора или устройства ПДП — прямого доступа к памяти). Не станем мы разбирать и порядок предоставления СиМ процессором по требованию прерывания от ПУс — все эти виды обмена особого интереса для нас не представляют, важно лишь усвоить общие принципы асинхронного обмена. Кроме того, нужно признать, что циклы ВВОД и ВЫВОД мы рассмотрели несколько упрощенно, в действительности обмен происходит немного сложнее, но и изложенного будет для неискушенного читателя вполне достаточно.

Теперь ясно, что такая ЭВМ, как БК-0010, может практически неограниченно расширяться путем подключения новых АУс и ПУс. Важно лишь, чтобы все АУс соблюдали «правила игры» и не мешали друг другу во время обмена, а все ПУс имели различные адреса или, если их адреса одинаковы, каким-либо способом включались по очереди. Все необходимые линии СиМ выведены в БК-0010 на специальный разъем МПИ, к которому и могут при необходимости подключаться дополнительные ПУс или АУс. (Это уже используется на практике: к БК-0010.01 подключается дополнительный блок МСТД, причем отключаются установленные на основной плате БК микросхемы ПЗУ интерпретатора языка БЕЙСИК, но зато подключаются установленные в блоке МСТД ПЗУ тестов и интерпретатора языка ФОКАЛ.)

* * *

Теперь приступим непосредственно к рассмотрению схемотехники нашего компьютера. Как уже говорилось, оно будет очень кратким и поверхностным.

Конструктивно БК-0010 относится к семейству ОДНОПЛАТНЫХ ЭВМ. Это означает, что все ее основные функциональные узлы размещены на одной печатной плате*. Тем не менее схема БК может быть условно разделена на два функциональных блока — процессорный и модуль памяти. Начнем с процессорного модуля.

Все узлы нашей микроЭВМ имеют общий ТАКТОВЫЙ ГЕНЕРАТОР. Он выполнен на микросхеме D5 типа К555ЛН2. Частота генератора (12 МГц) стабилизирована кварцевым резонатором. Тактовая частота после деления на 2 (до 6 МГц) триггером D8.1 подается на контроллер ОЗУ и дисплея D19 (о котором речь пойдет дальше), а кроме того, используется для тактирования телевизионного сигнала. Следующий триггер D8.2 делит тактовую частоту еще на 2, и полученная частота 3 МГц используется для обеспечения работы ЦП типа К1801ВМ1 (D14).

Сигналы прерываний от внешних устройств на процессор поступают через промежуточный регистр D11 типа К155ИР1. Запись в этот регистр также тактируется частотой 3 МГц. Порт ввода-вывода выполнен на четырех 8-разрядных регистрах D15—D18 типа К589ИР12. Обращение к порту (выбор регистров) ЦП осуществляет специальным сигналом SEL2 через логический элемент D1.6. Такой порядок обращения к внешним устройствам по сравнению с обычным асинхронным обменом значительно упрощает схему (но надо отметить, что внутри ЦП обмен осуществляется как обычно, просто ЦП сам отвечает на свой сигнал, выступая как АУс и ПУс одновременно).

Регистры системных внешних устройств (ввода и вывода) организованы на МС D12, D13 типов К531АП2П и К155ИР1. Обращение к порту системных внешних устройств также производится специальным сигналом ЦП SEL1.

Клавиатура ЭВМ подключена к общей магистрали через специализированную МС типа К1801ВП1-014 — контроллер клавиату-

* Вторая установленная в корпусе вычислителя плата служит для размещения клавишных переключателей и установки пьезодинамика (на БК с новой «пленочной» клавиатурой пьезодинамик распаян на основной плате). — *Прим. ред.*

ры. Эта МС имеет входы «X» и «Y» для подключения клавиш, образующие двумерную матрицу. При замыкании контактом клавиши одной из шин «X» на одну из шин «Y» от контроллера к ЦП поступает запрос на прерывание от клавиатуры, а принятый код клавиши заносится в регистр 177662. Переключение режимов «РУС»/«ЛАТ» осуществляется по кодам соответствующих клавиш, включенных в общее поле клавиатуры (и непосредственно передаваемых в программу), а переключение «СТР»/«ЗАГЛ» частично организовано аппаратно на триггере D3.1, который выдает специальный сигнал на вход EC1 контроллера. Клавиши «АР2» («НР»), «СУ» и «ПР» подключены к отдельным входам контроллера, причем «ПР» — через логический элемент D2.1. Клавиша «ПРОБЕЛ» имеет отдельную аппаратную «привязку» к контроллеру на МС D2.2, при ее нажатии включается клавиша «0» по регистру «ПР» (выдающая код пробела). Такое включение позволило выделить для символа «ПРОБЕЛ» отдельную клавишу, на что контроллер клавиатуры изначально не был рассчитан, и выдавать код пробела независимо от текущего режима клавиатуры. Контроллер имеет специальные антидребезговые цепи на резисторах R3, R4 и конденсаторах C3, C4. К сожалению, работа этих цепей оставляет желать лучшего: на такую ужасную клавиатуру, какая применялась раньше на БК-0010.01, они не рассчитаны*.

Для выдачи сигнала на магнитофон с выхода МС D13 (на БК-0010.01 схема этого узла несколько изменена) служит делитель на резисторах R29—R31, а для приема сигнала с магнитофона — компаратор на транзисторах VT1, VT2. Дистанционное управление магнитофоном осуществляет реле K1 (**неправильно обозначенное на схеме как K1.2**) типа РЭС-15. Его нормально разомкнутые контакты K1.1 подключены непосредственно к разъему МГ и ни с какими цепями внутри БК не соединяются. Таким образом, на контакты ДУ разъема МГ можно подавать любое напряжение (по техническим условиям для данного реле оно не должно превышать 115 В), но, учитывая, что цепи, соединяющие реле с разъемом МГ, проходят по печатной плате компьютера, не следует использовать реле K1 для коммутации цепей с напряжением свыше 12—24 В и током более нескольких десятков миллиампер.

Все цепи внутренней магистрали ЭВМ (и некоторые другие) нагружены на буферные резисторы, объединенные в резистивные матрицы (E1—E8).

Цепь запуска микроЭВМ выполнена на МС D6 типа К561ЛЕ5 и обеспечивает определенный порядок подачи сигналов на входы DCLO и ACLO ЦП после включения питания или срабатывания переключателя ручного перезапуска системы («ПУСК-СТОП», неправильно обозначенный на схеме как SA2, на БК-0010.01 этот переключатель отсутствует**).

Центральным элементом модуля памяти (показанного на второй части схемы) является контроллер ОЗУ и дисплея на МС К1801ВП1-037 (D19). Эта микросхема многофункциональная. Получая по магистрали микроЭВМ сигналы чтения, записи и адреса, она преобразует их в обращение к конкретному адресу матрицы ОЗУ, выполненной на МС ОЗУ динамического типа (К565РУ6) DS1—DS16 общей емкостью 32 Кб. Для промежуточного хранения данных используется регистр на МС D22, D23 типа К589ИР12, а для преобразования содержимого экранного ОЗУ в последовательный растровый код — регистр сдвига на МС D24, D25 типа К155ИР13. Контроллер формирует также сигналы регенерации динамического ОЗУ и полный телевизионный (ТВ) сигнал. Дальнейшая обработка ТВ-сигнала осуществляется МС D9.2, D10, D20, D21. Выходной ТВ-сигнал черно-белого изображения снимается с выхода оконечного видеоусилителя на транзисторах VT8, VT9, а цветного («R», «G», «B» и «СИНХРО») — с выхода повторителей на транзисторах VT4—VT7 (последние четыре транзистора, как и RGB-выход вообще, на старых БК-0010 отсутствуют). С выхода E контроллера снимается сигнал, блокирующий работу ПЗУ в области адресов системных регистров (177600... 177777). Микросхемы ПЗУ DS17—DS19 являются отдельными элементами схемы, связанными с ЦП по системной магистрали асинхронно. На их подключении следует остановиться чуть подробнее.

Прежде всего, на указанной схеме приведены лишь МС типа К1801РЕ2 с номерами 017, 018 и 019, что соответствует схеме старой БК-0010 (монитор, ФОКАЛ и тест-ПЗУ). В БК-0010.01 вместо двух последних МС установлены три ПЗУ БЕЙСИКа (номера 106,

* В настоящее время на БК устанавливается так называемая «пленочная» клавиатура, отличительным признаком которой являются «ступеньки» на клавишах управления курсором «ВЛЕВО» и «ВПРАВО». Новая клавиатура отличается от прежней более мягким ходом клавиш и отсутствием дребезга. Желающие заменить старую «кнопочную» клавиатуру БК-0010(.01) на «пленочную» могут обращаться в редакцию. — *Прим. ред.*

** На схеме, опубликованной в №2 за 1994 г., данный переключатель имеет правильное обозначение SA1. Кроме того, хотя он отсутствует в БК-0010.01, соответствующие ему контактные площадки на плате имеются и могут быть использованы для подключения кнопки перезапуска «RESET». О том, как это сделать, рассказано в №1 за 1994 г., с. 71. — *Прим. ред.*

107, 108). Микросхема типа К1801РЕ2 имеет вход выборки CE. Схема БК-0010.01 устроена так, что при подключении блока МСТД микросхемы БЕЙСИКа, стоящие на плате, отключаются по этим входам и уже не отвечают на запросы ЦП, а установленные в блоке МСТД микросхемы К1801РЕ2-018 и -019 включаются в работу. Микросхема, адресное пространство которой приходится на область системных регистров (019 — тест-ПЗУ или 108 — БЕЙСИК), дополнительно блокируется сигналом запрета с выхода E контроллера D17, если ЦП обратился по адресу какого-либо системного регистра. Чтобы отключить любую из МС ПЗУ, достаточно на ее вход CE (вывод 23) подать «логическую 1», при этом МС перестает отвечать на запросы ЦП и занимать адресное пространство, она как бы не существует. Таким способом можно почти неограниченно расширять память БК-0010, «размещая» новые страницы памяти (ОЗУ или ПЗУ) в адресном пространстве ПЗУ при условии отключения последних. Конечно, ни при каких обычных условиях недопустимо отключение ПЗУ монитора (DS17), иначе ЭВМ просто не будет работать (хотя ЦП будет продолжать функционировать), ибо это ПЗУ выполняет функции виртуального процессора.

Возможные доработки БК-0010

Среди пользователей БК-0010 в последнее время появилась мода на доработки схемы БК. Считается чуть ли не неприличным, если эта схема осталась неизменной и работает так, как это предусмотрено. Нельзя не признать, что многие из доработок весьма рациональны, но есть среди них и сомнительные. Рассмотрим кратко, какие доработки БК-0010 сегодня известны, в чем их преимущества и недостатки, а также насколько велик риск выхода компьютера из строя при их реализации.

Самой распространенной доработкой является исключение блока МСТД с установкой платы блока (или непосредственно микросхем ПЗУ ФОКАЛа и БЕЙСИКа) внутрь корпуса компьютера. Один из вариантов этой доработки описан в журнале «Информатика и образование» (там же, где приведена схема БК). Доработку надо признать весьма рациональной и при аккуратном исполнении безопасной.

Одновременно можно установить переключатель перезапуска компьютера, если он отсутствует. Его удобно вынести на левую стенку корпуса, как это показано в журнале «Персональный компьютер БК-0010 — БК-0011М», №1 за 1994 г., с. 71. Если использовать кнопку типа П2К, то ее выводы точно устанавливаются в отверстия печатной платы, предназначенные для переключателя, остается только их распаять и прорезать в левой стенке корпуса окно для кнопки.

Следующим шагом может быть установка отдельных выключателей для КАЖДОЙ из микросхем ПЗУ. Для включения соответствующей МС достаточно соединить ее вывод 23 с общим проводом компьютера («логический 0»), а для отключения подать на вывод 23 «1» (соединить вывод с шиной «+ 5 В» через резистор 1...3 кОм), что и должны делать переключатели. Такая доработка может быть очень полезной. Например, если при исполнении какой-либо программы компьютер «завис» и не реагирует на нажатие клавиш, то после отключения всех МС ПЗУ (кроме ПЗУ монитора) и перезапуска содержимое ОЗУ сохранится (если, конечно, его не уничтожит программа, вызвавшая зависание) и имеющуюся в ОЗУ информацию можно использовать. Зависание системы перестает быть «фатальным»!* И совершенно необходимо выборочное отключение ПЗУ при подключении к ЭВМ дополнительной памяти, дисковода и иных внешних устройств, работающих по адресам 120000...177577. Но эта доработка уже сложнее, так как требует вмешательства в схему компьютера (в частности, необходимо освободить от печатных проводников выводы 23 всех ПЗУ, сохранив при этом другие цепи нетронутыми).

Интересной и несложной доработкой является введение переключателя «ПДП» — прямого доступа к памяти. Конечно, настоящий режим ПДП мы получить не сможем: для этого необходимо как минимум специальное устройство — контроллер ПДП, но переход в режим ПДП позволяет обратимо (на время) прервать исполнение ЛЮБОЙ программы пользователя. Для перехода в этот режим необходимо подать «0» на выводы 2, 3 и 5 ЦП, т. е. с помощью специально установленного переключателя соединить эту точку схемы (указанные выводы МС D14 в стандартной схеме БК соединены вместе и через резистор R27 подключены к шине питания «+5 В») с общим проводом. Отключать от этой точки что-либо, как это рекомендовалось в журнале «Информатика и образование», № 2 за 1990 г. (с. 46), не нужно — это только лишний риск вывести компьютер из строя, а останов ЦП будет достигнут и без этого. (В упомянутой статье помимо переключателя ПДП описана схема перевода БК-0010 в так называемый «турбо-режим» — удвоение тактовой частоты ЦП. Эта доработка призвана удвоить скорость исполнения программ на БК. При всей своей привлекательности эта доработка относится к числу НЕ РЕКОМЕНДУЕМЫХ. Во-первых,

* То же самое можно сделать и без установки «выключателя ПЗУ». Если сразу же после перезапуска нажать на клавишу «СТОП» (возможно, несколько раз), происходит выход в монитор. — *Прим. ред.*

поскольку обмен в БК асинхронный, *удвоения* скорости не произойдет, она увеличится в лучшем случае в 1,7 — 1,9 раза (а при работе с ОЗУ — всего в 1,2 — 1,6 раза). Кроме того, вообще не каждый БК-0010 будет работать при удвоенной тактовой частоте, это зависит от качества примененной в вашей ЭВМ микросхемы ЦП. И наконец, тактовая частота 6 МГц является ПРЕДЕЛЬНОЙ для ЦП данного типа и значительно увеличивает вероятность сбоев в работе ЭВМ, а также сокращает срок службы ЦП из-за его сильного перегрева*.)

Не составляет труда ввести в компьютер и дополнительное внеприоритетное прерывание ACLO (по вектору 24). Для вызова этого прерывания достаточно подать короткий положительный импульс на вывод 13 МС D6.4. Импульс может быть сформирован, если зарядить конденсатор емкостью 0,033 мкФ до напряжения + 5 В, а затем разрядить его на эту точку схемы (с помощью специального переключателя или кнопки). В МСД при этом будет выдано сообщение «СБОЙ ПИТАНИЯ». Перезадав вектор 24 и написав свою программу его обработки, можно решать самые различные задачи, например делать графическую копию экрана в любом режиме ЭВМ (надо только учитывать, что вектор 24 весьма «уязвим» — его переписывают команда EMT 14 и многие программы пользователя).

Следующая очень распространенная доработка — расширение памяти компьютера. Предложено множество схем такого расширения, все они сводятся в основном к отключению ПЗУ в области адресов 120000...177577 и подключению вместо них нового внешнего устройства — контроллера ОЗУ и матрицы-накопителя (попросту говоря, микросхем памяти). Контроллер необходим потому, что не существует микросхем ОЗУ, рассчитанных на непосредственное включение в системную магистраль БК-0010 (кроме сверхдефицитных К1809РУ1, имеющих к тому же емкость всего 2 Кб), — все они имеют организацию INTEL, а у нас в компьютере используется организация DEC. Есть, правда, МС ППЗУ типа К573РФ3 (с ультрафиолетовым стиранием) и КМ1801РР1 (с электрическим стиранием), рассчитанные на архитектуру DEC; это перепрограммируемые ПЗУ, по подключению идентичные МС К1801РЕ2. Но эти МС крайне дефицитны, для записи в них информации необходимо специальное оборудование (программатор), а срок хранения информации в них ограничен (в лучшем случае несколько тысяч часов), поэтому всерьез говорить о их широком применении в БК не приходится.

Все схемотехнические решения дополнительного ОЗУ распадаются на две ветви — статическое и динамическое ОЗУ. Динамические МС ОЗУ (преимущественно серии К565) относительно дешевы и обладают значительной емкостью (от 16 до 64 и больше килобит на корпус), но требуют периодической РЕГЕНЕРАЦИИ — восстановления информации, осуществляемого специальным устройством примерно 1000 раз в секунду. Это усложняет схемотехнику такого типа ОЗУ или требует применения специальных контроллеров (например К1801ВП1-037), а они дефицитны. Кроме того, информация в динамическом ОЗУ хранится, пока включено питание компьютера. Любой сбой питания или выключение ЭВМ приводит к необходимости загружать информацию в ОЗУ заново, а при его значительной емкости (порядка 128 Кб) такая загрузка с магнитофона будет длиться десятки минут. Ясно, что применение ОЗУ столь большой емкости оправдано в основном для промежуточного хранения и обработки информации (графической или текстовой). Для этого, естественно, требуются специализированные программы, на БК-0010 пока что отсутствующие (их разработка ведется, но ассортимент вряд ли будет широк — маловато самих дополнительных ОЗУ, не всем они доступны). Еще один «минус» динамических ОЗУ — значительное энергопотребление. Ток потребления, например, ОЗУ емкостью 128 Кб может превышать 0,5 А. На такую дополнительную нагрузку блок питания БК не рассчитан, и требуется его переделка или изготовление дополнительного блока питания. Конечно, можно довериться и «штатному» блоку питания БК (по принципу «авось пронесет») — некоторые резервы имеются, но полагаться на «авось» как-то не очень хочется.

Вторая ветвь — статические ОЗУ. Они конструируются обычно на базе МС со сверхнизким энергопотреблением, чаще всего серии К537. Эти МС, не обладающие столь большими информационными емкостями и более дорогие, тем не менее имеют ряд важных преимуществ. Прежде всего, они не требуют регенерации, поэтому схема ОЗУ получается проще. Но самое главное — потребляемый ток этих МС так мал, что они не только не создают

* По имеющейся в редакции (к сожалению, пока не подтвержденной) информации, БКманами Брянска ведется разработка одного весьма интересного устройства на основе ПДП — «телесканера» (точнее было бы назвать его «VideoBlaster», по аналогии с одноименной системой для IBM-совместимых). Основная идея состоит в считывании в ОЗУ экрана БК по нажатию специальной внешней кнопки видеосигнала с соответствующих цепей телевизора (естественно, с оцифровкой сигнала). Поскольку при этом скорость загрузки в ОЗУ является определяющим параметром, режим ПДП здесь обязателен. Редакция приглашает авторов данной разработки (и других аналогичных устройств) поделиться своим опытом с читателями. — *Прим. ред.*

перегрузок блока питания БК, но даже позволяют снабдить такое ОЗУ «буферным питанием», т. е. блоком питания от батарей или аккумуляторов, позволяющим хранить информацию при отключении ЭВМ несколько месяцев или даже лет. Благодаря такому решению уже не приходится каждый раз загружать информацию в ОЗУ заново, однажды считанные туда программы (например отладчик, ассемблер, редактор текста) могут храниться неограниченно долго, при включении ЭВМ они сразу готовы к работе, а при необходимости их очень легко заменить на другие программы. Понятно, что применение статических ОЗУ с малым энергопотреблением и буферным питанием (их еще называют энергонезависимыми ОЗУ — ЭОЗУ) гораздо предпочтительнее динамических*.

Основным преимуществом в случае расположения рабочих адресов дополнительного ОЗУ в зоне адресов ПЗУ является то, что загруженные в такое ОЗУ программы находятся в адресном пространстве БК и (если они перемещаемые) могут работать непосредственно в дополнительном ОЗУ. Если же требуется переслать такую программу в ОЗУ пользователя, то это тоже очень легко сделать, воспользовавшись, например, директивами МСД. Но существует и другой путь. Можно подключить дополнительное ОЗУ к порту БК-0010 и организовать его побайтный обмен информацией с ОЗУ пользователя, получив таким образом внешний накопитель информации — квазидиск. Преимуществом такого решения является относительная простота ОЗУ, но много и недостатков. Это и необходимость использования специальных программ обмена (типа операционных систем), и невозможность свободного подключения к порту других внешних устройств (принтера, манипулятора «мышь»), и обязательность пересылки программ для работы в ОЗУ пользователя. Разумное сочетание обоих решений (например, программы обмена и некоторые инструментальные программы — в ЭОЗУ, в общем адресном пространстве, а хранение промежуточной информации — на квазидиске) может дать хорошие результаты. Однако сегодня все упирается в отсутствие специальных программ, рассчитанных на работу с дополнительными ОЗУ, которые позволили бы в полной мере реализовать преимущества этой доработки БК-0010.

Появление в последнее время в свободной продаже нескольких типов принтеров привело к тому, что принтер все чаще становится неотъемлемой частью комплекса на основе БК-0010, чему немало способствует наличие на БК неплохих текстовых и графических редакторов. Про программное обеспечение печати уже было рассказано в разделе «Концепция системных драйверов». Остается только указать, какой принтер наиболее предпочтителен. Сегодня это, несомненно, МС6313 с EPSON-совместимой системой команд. Ничего плохого нельзя сказать и про импортные модели принтеров (например, EPSON LX/FX-800), но где их взять рядовому пользователю БК? Да и цены кусаются. А распространившиеся было термоструйные принтеры МС6312 мало кому пришлись по нраву, так как печатающие головки-чернильницы на них одноразовые, довольно дорогие и дефицитные. Одна головка дает возможность отпечатать не более 800—1000 страниц, после чего требует замены (придуманные «умельцами» способы перезаправки головок стабильных результатов не дают). Таким образом, даже более низкая стоимость МС6312 по сравнению с другими моделями не окупает хлопот и расходов при его эксплуатации. Дополнительными недостатками этого принтера являются невозможность печати нескольких экземпляров текста одновременно и очень мелкий шрифт**.

Проблемой номер один на БК считается подключение дисковода, хотя реализована эта доработка пока очень немногими. Обычно для этого используется схема контроллера НГМД для компьютера БК-0011 (основой которого является микросхема К1801ВП1-128). Эта схема неплохо себя зарекомендовала, не требует для подключения почти никаких доработок компьютера и имеет поддержку в виде целого ряда программных средств (операционных систем). Конечно, имеющиеся программные средства поддержки дисковода пока далеки от совершенства. Сильно ограничивает возможности работы с дисководом тот факт, что программа обмена или ОС должна при работе с ним располагаться в ОЗУ пользователя, что делает невозможной работу со

* Очевидно, что основное («штатное») ОЗУ БК сконструировано по «динамическому» принципу, а расширенное ОЗУ, устанавливаемое в КНГМД, как правило, статическое. — *Прим. ред.*

** Но зато МС6312 печатает абсолютно бесшумно, «расплывание» микрокапель чернил на бумаге повышает качество печати графики (отсутствует присущая матричным принтерам «полосатость» рисунка), а мелкий шрифт в новых моделях заменен на более крупный. Что же касается МС6313, возможны некоторые проблемы с его подключением к БК (см. ряд публикаций в журналах «Информатика и образование» и «Вычислительная техника и ее применение»). Так что сказанное выше нельзя считать «не подлежащим обжалованию приговором» струйным принтерам. Кстати, «струйники» в последнее время все увереннее вытесняют привычные матричные принтеры на IBM-совместимых ПЭВМ, а мнению специалистов этой области нельзя отказать в авторитетности. — *Прим. ред.*

многими программами, требующими всего объема памяти. Обойти эту трудность можно, разместив ОС в дополнительном ОЗУ любого типа, лучше всего, конечно, в ЭОЗУ. Однако следует отметить, что необходимость подключения к БК дисковода часто переоценивается. Разумеется, существует ряд задач, для решения которых дисковод просто необходим. Но для большинства задач вполне достаточно магнитофона, особенно с последними разработками, например разработанным автором МКМ-пакетом, который позволяет вести обмен с магнитофоном на скорости 4000 бод при высокой надежности и поддерживает большинство программ пользователя и языков высокого уровня. При размещении такого драйвера в дополнительном ОЗУ удобство работы с магнитофоном возрастает столь заметно, что необходимость подключения дисковода становится неочевидиой... на некоторое время.

«Сколько на свете людей, столько и мнений»: старая поговорка права всегда, и, в полном соответствии с ней, мнение редакции журнала в данном случае не совпадает с мнением автора. Конечно, дисковод ПОКА ЕЩЕ является скорее удобством, чем необходимостью. Но для любого компьютера удобство работы — один из определяющих факторов, к тому же уже сегодня существует немало многофайловых программ, при работе подгружающих в ОЗУ тот или иной программный модуль, и количество таких программ (в связи с растущей популярностью дисководов и удобством самого модульного, или, как еще говорят, оверлейного, принципа программирования на ЭВМ с ограниченным объемом памяти) будет увеличиваться с каждым годом. Далее, в настоящее время использование КНГМД от БК-0011, упомянутых автором статьи, почти полностью прекращено, так как прошивка микросхемы ПЗУ в этой модели КНГМД обладает рядом существенных недостатков. (В частности, перед загрузкой операционной системы с диска на БК-0010(.01) требуется вручную переписать содержимое некоторых ячеек ОЗУ.) Сейчас к БК-0010(.01) подключают в основном КНГМД от новой модели БК-0011М («триста двадцать шестая» прошивка ПЗУ), соответственным образом доработанный:

- установка перемычек, блокирующих ПЗУ БЕЙСИКа при подключении блока КНГМД к разъему системной магистрали БК-0010.01 (или ПЗУ ФОКАЛа и тестов в БК-0010);
- установка дополнительного энергонезависимого ОЗУ емкостью от 8 до 32 Кб (стандарт — 16 Кб);
- прочие доработки, повышающие удобство общения с БК и расширяющие спектр его возможностей: кнопка «RESET», микросхема для временного подключения ПЗУ БЕЙСИКа взамен ДОЗУ («дисковый БЕЙСИК») и т. д.

Кроме контроллеров от БК-0011М популярны и «самостоятельные» разработки, предназначенные специально для БК-0010(.01). Из них наиболее предпочтительными, пожалуй, можно считать контроллеры фирмы «АльтПро», описание которых дано в предыдущем выпуске журнала.

Вопросам подключения дисковода к БК-0010(.01) посвящен №1 за 1993 г., имеются публикации и в других выпусках (в частности, в №5 за 1994 г. дан ассемблерный листинг прошивки ПЗУ КНГМД и объяснены основные принципы его работы, а в №2 за 1995 г. рассказано о доработке КНГМД для работы с диском в вильнюсском БЕЙСИКе БК-0010.01).

Приобрести контроллеры дисковода для БК-0010(.01) (фирмы «АльтПро») и для БК-0011(М), а также дисководы, блоки питания для них, операционные системы и даже полный комплект «винчестера» для БК-0010(.01) и БК-0011(М) (сам жесткий диск, совмещенный контроллер для него и дисководов, блок питания и программная поддержка) можно в редакции, здесь же вы можете получить справки и консультации. — *Прим. ред.*

Для БК-0010 в настоящее время разработан и ряд других дополнительных устройств, например серийно выпускаемый манипулятор «мышь» (УВК-01 «Марсианка») и несколько видов джойстиков. Если «мышь» имеет стандартную цоколевку, которую нужно учитывать только авторам использующих ее программ, то с джойстиками дело обстоит сложнее. Единого стандарта их подключения к БК не существует, так что риск несовпадения распайки у авторов программ и у пользователей довольно велик. Но проблема, как это часто бывает, преувеличена. Подавляющее большинство программ, использующих джойстик (как правило, это игры), дает возможность задания управления перед началом работы, поэтому способ подключения джойстика особого значения не имеет. Надо только найти общий вывод для всех его кнопок и подключить его к общему проводу разъема порта УП БК (например, к контакту А11), остальные же выводы подключаются к произвольным контактам порта ввода (например, к В24, А24, В23, В17, В20, А20, В22 и т. д.)*.

Прочие дополнительные устройства применяются на БК относительно редко и являются более «экзотическими». Например, в Чехословакии (по старой географической

* Варианты распайки джойстиков приведены во многих журналах, где публикуются (или публиковались ранее) материалы о БК. Примерами являются таблицы распайки в №4 за 1994 г., с. 32 нашего издания, а также в журнале «Вычислительная техника и ее применение», №8 за 1991 г., с. 38 (вариант, рекомендованный магазином «Электроника» на Ленинском проспекте в Москве). — *Прим. ред.*

терминологии) специально для БК выпускался плоттер (двухкоординатный графопостроитель) типа XY4140 (XY4150, XY4160) с соответствующим программным обеспечением, входящим в комплект поставки. Есть и программы, разработанные для него дополнительно. Например, автором этой статьи создана программа «СКР.4140», позволяющая выводить на плоттер любые изображения, подготовленные в графическом редакторе Р1.0М (доработанный БК-PAINT), в том числе и размечать с его помощью печатные платы. Автором также разработана система ввода в БК изображений с телекамеры — телевизионный сканер «ТВИНТ», имеющий очень несложную аппаратную поддержку (не считая, конечно, самой телекамеры типа «Электроника-841»!), так как большинство проблем решается программным путем. Разными авторами в разное время разрабатывались на базе БК-0010 многоголосые музыкальные синтезаторы, системы автоматизации, приставки к телефонам и т. п., но все эти устройства были достаточно сложны и не получили заметного распространения *.

«Бичом» старых БК-0010 была клавиатура. Став более удобной на БК-0010.01, она проиграла в надежности. Разработано несколько способов подключения к БК внешней клавиатуры (в том числе и от IBM) — на герконах, на магнитодиодных микросхемах и т. п., но все они достаточно сложны и дороги в реализации. Существует и целый ряд приемов борьбы с дребезгом «штатной» клавиатуры, более или менее эффективных. Но с выпуском новой клавиатуры для БК (с многослойной пленочной контактурой и характерными «ступеньками» на клавишах управления курсором) эта проблема утрачивает остроту.

Подводя итоги всему сказанному, можно заметить, что БК-0010 благодаря своей относительно низкой стоимости, широкой распространенности, возможности расширения, удобной системе команд и развитому программному обеспечению еще достаточно долго «не сойдет со сцены» и будет продолжать радовать своих владельцев все новыми возможностями, а перед программистами и схемотехниками ставить все новые задачи.

ПРИЛОЖЕНИЯ

Перечень общепринятых в литературе о БК сокращений

DIN — сигнал цикла ВВОД (чтение) активного устройства.

DOUT — сигнал цикла ВЫВОД (запись) активного устройства.

PC — программный счетчик, один из регистров общего назначения.

PS — регистр состояния процессора (в котором хранится слово состояния процессора).

PSW (ССП) — слово состояния процессора.

RPLY (СИП) — синхроимпульс (ответ) пассивного устройства.

SYNC (СИА) — синхроимпульс (синхронизация обмена) активного устройства.

SP — регистр хранения адреса вершины стека, один из регистров общего назначения.

WTBT — сигнал байтового обмена цикла ВЫВОД (запись) активного устройства.

АД — адрес—данные (шина, группа линий связи системной магистрали).

«АР2» («НР») — регистровая клавиша для ввода дополнительных кодов клавиатуры на БК — псевдографики и некоторых команд управления.

АУс — активное устройство (инициирующее обмен по системной магистрали).

АЦП — аналого-цифровой преобразователь (устройство, преобразующее сигналы тока или напряжения в цифровой код).

БК — бытовой компьютер (здесь имеется в виду компьютер БК-0010(.01) или БК-0011(М))**.

ГТ — графическая точка.

ДАМП — распечатка содержимого памяти (обычно в коде ASCII) а также специальная программа для такой распечатки.

* О том, как сделать самодельный графопостроитель, рассказано в №4 за 1994 г., а на с. 50 этого выпуска пользователи БК познакомятся и с разработанной автором данной статьи схемой сканирующей приставки к графопостроителю (или, при соответствующих изменениях в конструкции, — к принтеру). Что же касается прочих периферийных устройств, их перечень можно продолжить. Это световое перо, дигитайзер, чувствительный к нажатию планшет, модем... Последний дает возможность не только обмениваться по телефону программами с другими БКманами, но и работать в профессиональных электронных сетях типа RELCOM или FIDONET, до недавнего времени считавшихся исключительной прерогативой IBM, «Макинтошей» и других «профессионалок». — *Прим. ред.*

** Не так давно появился другой домашний компьютер: «Апогей БК-01». Несмотря на похожее имя, он никак не совместим с «настоящими» БК. Редакции же остается только предостеречь читателей от ошибки и посетовать на невнимание изготовителей «Апогея», приведшее к путанице. — *Прим. ред.*

ДВК — диалоговый вычислительный комплекс, групповое название распространенных компьютеров конфигурации DEC.

ДУ — дистанционное управление (магнитофоном или иными устройствами).

«ЗАГЛ» — регистровая клавиша, устанавливающая режим ввода кодов заглавных букв.

КС — контрольная сумма (файла, блока информации, массива) — специальным образом вычисляемое значение, позволяющее идентифицировать блок информации (как правило, используется для проверки правильности ввода с клавиатуры или чтения с внешнего носителя).

КУВТ — комплекс учебной вычислительной техники*.

«ЛАТ» — регистровая клавиша, устанавливающая режим ввода кодов латинских букв.

МДС — монитор-драйверная система, часть системного программного обеспечения БК-0010.

МЛ — магнитная лента.

МОП — одна из технологий производства микросхем (металл-окисел-полупроводник).

МПИ — разъем системной магистрали БК.

МС — микросхема.

МСД, МСТД — мониторная система тестов и диагностики, часть системного программного обеспечения БК-0010, а также блок (МСТД), подключаемый к разъему системной магистрали.

НГМД — накопитель на гибких магнитных дисках.

НМД — накопитель на жестком магнитном диске («винчестер»).

НМЛ — накопитель на магнитной ленте.

ОЗУ — оперативное запоминающее устройство.

ОС — операционная система.

ПДП — устройство (режим) прямого доступа к памяти.

ПЗУ — постоянное запоминающее устройство.

ПМ — пусковой монитор (часть системного программного обеспечения, а также один из диалоговых режимов БК-0010).

ПНЧ — преобразователь напряжения в частоту.

ПО — программное обеспечение.

ППЗУ — перепрограммируемое постоянное запоминающее устройство.

«ПР» — регистровая клавиша, устанавливающая режим ввода дополнительных кодов клавиатуры.

ПУс — пассивное устройство (устройство, подключенное к системной магистрали и работающее под управлением активного устройства).

РОН — регистр общего назначения центрального процессора.

РП — режим расширения памяти БК-0010.

«РУС» — регистровая клавиша, устанавливающая режим ввода кодов русских букв.

СИА (SYNC) — синхроимпульс (синхронизация обмена) активного устройства.

СиМ — системная магистраль.

СИП (RPLY) — синхроимпульс (ответ) пассивного устройства.

СЛ — служебная линия (системной магистрали).

СР — системный регистр.

ССД — слово состояния дисплея.

ССП (PSW) — слово состояния процессора.

«СТР» — регистровая клавиша, устанавливающая режим ввода кодов строчных букв.

«СУ» — регистровая клавиша, устанавливающая режим ввода кодов символов управления.

СУБД — система управления базами данных.

ТЛГ — телеграфный канал связи.

УсП — установочная (настроечная) последовательность.

ЦП — центральный процессор.

ЭВМ — электронно-вычислительная машина (компьютер).

Цоколевка разъемов БК-0010(.01)

Вывод разъема	Назначение	
	Разъем порта УП	Разъем магистрали МПИ
(1)	(2)	(3)
A1	СБРОС (INIT)	ОСТ (рестарт)
A2	—	Общий
A3	—	Общий
A4	—	—
A5	—	ПРТ (IRQ2)
A6	—	—
A7	Вывод 07	—
A8	+5 В	—

(1)	(2)	(3)
A9	+5 В	—
A10	—	—
A11	Общий	—
A12	—	+5 В
A13	Вывод 01	—
A14	—	БЛОК (откл. ПЗУ) [1]
A15	—	—
A16	Вывод 00	—
A17	—	—
A18	Общий	—
A19	Общий	—

* Как правило, имеется в виду состоящий из БК и ДВК (в качестве учительской машины) класс КУВТ-86 или построенный на базе БК-0011(М) комплекс УКНЦ-01 («спасибо» изготовителям за еще одну путаницу в названиях — к «одноименному» компьютеру УКНЦ данный класс не имеет никакого отношения!). — *Прим. ред.*

(1)	(2)	(3)
A20	Ввод 05	—
A21	—	—
A22	—	—
A23	Ввод 07	ВВОД (DIN)
A24	Ввод 01	ППР2 (предост. прер.)
A25	Вывод 14	АД 13
A26	Вывод 12	АД 11
A27	Вывод 10	АД 09
A28	Вывод 08	АД 05
A29	Ввод 13	ЕН (блок. СР)
A30	Ввод 15	—
A31	Ввод 09	АД 00
A32	Ввод 11	—
B1	Вход ПРТ (IRQ2)	—
B2	Прием ТЛГ[2)]	Общий
B3	Передача ТЛГ[2)]	Общий
B4	—	ППР1 (предост. прер.)
B5	Вывод 04	ТПР (требован. прер.)
B6	Вывод 06	—
B7	Вывод 05	АД 15
B8	+5 В	—

(1)	(2)	(3)
B9	+5 В	—
B10	Вывод 03	—
B11	Общий	БАЙТ (WTBT)
B12	Вывод 02	+5 В
B13	—	—
B14	—	—
B15	—	—
B16	—	—
B17	Ввод 03	—
B18	Общий	—
B19	Общий	СБРОС (INIT)
B20	Ввод 04	СИП (RPLY)
B21	—	ВЫВОД (DOUT)
B22	Ввод 06	СИА (SYNC)
B23	Ввод 02	АД 14
B24	Ввод 00	АД 12
B25	Вывод 15	АД 10
B26	Вывод 13	АД 08
B27	Вывод 11	АД 06
B28	Вывод 09	АД 04
B29	Ввод 14	АД 02
B30	Ввод 12	АД 03
B31	Ввод 08	АД 01
B32	Ввод 10	АД 07

Примечания:

1) — данный контакт используется только на БК-0010.01 и обеспечивает блокировку ПЗУ БЕЙСИКа при подключении блока МСТД.

2) — на некоторых экземплярах БК для обмена информацией по линии ТЛГ необходимо установить перемычки на печатной плате.

Некоторые полезные сведения о подключении к БК-0010(.01) контроллера дисковода

Для нормальной работы с БК-0010.01 КНГМД необходимо доработать: контакты A29 и A14 разъема МПИ нужно соединить с шиной «+5 В» через резисторы 150—200 Ом. Этим достигается отключение встроенного ПЗУ БЕЙСИКа при подключении КНГМД. Если в БК-0010.01 установлен переключатель для выборочного подключения ПЗУ БЕЙСИКа, ФОКАЛа и тестов, то такая доработка не требуется, достаточно перед подсоединением КНГМД просто отключить все встроенные ПЗУ. Для БК-0010 отключение ПЗУ по двадцать третьему выводу микросхем является единственным.

В КНГМД имеется дополнительный вывод шины МПИ (контакты под разъем на боковой стороне платы). К этим контактам можно подключать дополнительные устройства (например, блок энергонезависимого ОЗУ). Но при этом следует помнить, что линия системной магистрали WTBT на дополнительные контакты не выведена (если сигнал WTBT необходим в подключаемом устройстве, его следует снимать непосредственно с вывода B11 разъема МПИ КНГМД).

Автор выражает благодарность всем программистам и пользователям БК, с которыми он вел переписку и лично встречался за последние 7 лет, и чьи ценные предложения, советы, вопросы и пожелания помогли ему в написании данного материала:

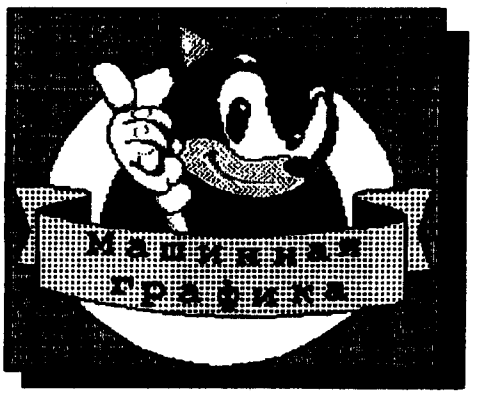
Б. Ф. Фролкину , С. А. Кумандину и Д. Г. Гаеву (Москва),

А. В. Цаплеву и А. Г. Макееву (С.-Петербург),

В. А. Михайлову (г. Алма-Ата),

С. И. Никанорову (г. Керчь)

и другим.



Дисплей, подключенный к ЭВМ, представляется мне окном в Алисину страну чудес, где программист может изображать либо объекты, описываемые хорошо известными законами природы, либо чисто воображаемые объекты, подчиняющиеся законам, записанным в программе.

Айвен Сазерленд

С. К. Румянцев,
пос. Локня Псковской обл.

Рисование окружностей и эллипсов средствами ассемблера

Во многих приложениях машинной графики возникает необходимость получения растровых изображений окружностей и эллипсов. Для решения этой задачи можно воспользоваться, например, подпрограммой реализации оператора функцией CIRCLE, прошитой в ПЗУ БЕЙСИКа БК-0010.01. Однако при программировании на ассемблере такое решение нельзя считать оптимальным. Вызов подпрограммы БЕЙСИКа из программы на ассемблере не только связан с определенными трудностями, но и вообще нежелателен, так как ассемблерная программа не должна зависеть от содержимого постоянной памяти. Ведь на месте БЕЙСИКа в ПЗУ может быть Фокал или тестовый монитор. Рациональное решение этой проблемы предложено в статье Д. Ю. Усенкова «Библиотека графических функций для ассемблера БК-0010(.01)» (№1 за 1994 г., с. 37), где для реализации базовых графических функций из программ на ассемблере предлагается использовать подгружаемую библиотеку стандартных функций.

Чтобы удовлетворить все запросы пользователей, такая библиотека кроме прочих операций должна содержать подпрограммы рисования окружностей и эллипсов. К сожалению, среди программистов бытует мнение, что реализация этих функций на ассемблере связана с чрезвычайно сложными расчетами, требующими большого объема машинного кода. Но, как говорится, «не так страшен черт, как его малюют». Задача формирования окружностей, как и любая другая, допускает несколько различных по сложности решений.

Математически окружность можно описать различными способами. Это может быть тригонометрическое выражение:

$$\begin{cases} X = R \cdot \sin(T); \\ Y = R \cdot \cos(T). \end{cases}$$

или алгебраическое: $X^2 + Y^2 = R^2$.

В обоих случаях возникает необходимость вычисления вещественных функций: синуса и косинуса или квадратного корня. Подобные расчеты не представляют трудностей на языке высокого уровня, но реализовать их на ассемблере довольно сложно. Кроме того, вещественные вычисления требуют представления чисел с «плавающей» точкой и выполняются медленнее, чем операции с целыми числами. В конечном же счете изображение на экране будет складываться из отдельных пикселов, имеющих целочисленные координаты. Так что значения, полученные при вещественных вычислениях, имеют избыточную точность, их все равно придется округлять до целых. Поэтому предпочтительнее сразу работать только с целыми числами, это упростит расчеты и поможет избежать избыточных вычислений.

Метод построения окружностей, работающий с целыми числами, известен программистам как алгоритм Брезенхема. В нем используются только операции сложения, вычитания и умножения на два, что как нельзя лучше подходит для реализации на ассемблере. Вывести этот алгоритм можно различными способами.

Рассмотрим здесь один из самых простых. Будем строить окружность с целым радиусом R, центр которой находится в начале координат. Построение будем проводить в четвертом квадранте от точки (0,−R) до (R,0) (изображение в остальных квадрантах легко получить путем отражения точек от координатных осей). Выбор именно четвертого квадранта позволит придать алгоритму особенно краткий вид.

При построении окружности координаты текущей точки в этом квадранте X и Y в этом квадранте будут изменяться только в сторону увеличения. На каждом шаге алгоритма необходимо делать выбор из трех возможностей: увеличение только координаты X, увеличение одновременно X и Y, увеличение только Y. Принцип Брезенхема предполагает выбор того направления, которое наиболее точно приближает текущую точку к идеальной окружности. Чтобы производить такой выбор, введем дополнительную величину, характеризующую степень приближения точки к идеалу. Допустим, что точка смещается в диагональном направлении. Тогда разность между квадратом ее расстояния от центра и квадратом радиуса R идеальной окружности будет равна $D = (X+1)^2+(Y+1)^2-R^2$. Эта величина является базовой. В начале процесса она равна $D = (0+1)^2+(-R+1)^2)-R^2 = 2\cdot(1-R)$. Аналогичные величины для горизонтального и вертикального направлений легко находятся по базовой:

$$\begin{cases} Dr=(X+1)^2+Y^2-R^2=D-(2\cdot Y+1); \\ Db=X^2+(Y+1)^2-R^2=D-(2\cdot X+1). \end{cases}$$

Теперь остается только выбрать, какая из этих величин ближе к нулю (т. е. наименьшую по абсолютной величине). Так как в четвертом квадранте X>0 и Y<0, очевидно, что при D≤Y ближе к нулю будет Dr и следует предпочесть горизонтальное направление. При D>X ближайшей к нулю будет Db, соответствующая вертикальному направлению. В остальных случаях предпочтение следует отдать диагональному направлению. На рис. 1 показана схема выбора направления смещения текущей точки (указанного штриховкой) в зависимости от величины D. Отобразив текущую точку на экране (а фактически четыре точки, полученные зеркальным отражением), следует изменить ее координаты X и Y в соответствии с выбранным направлением. Затем нужно увеличить базовую величину D: в горизонтальном случае на 2·X+1, в вертикальном — на 2·Y+1, в случае диагонального направления — на обе эти суммы. Такие итерации следует выполнять до тех пор, пока координата Y не достигнет нуля, после чего окружность построена и алгоритм закончен. (Блок-схема описанного алгоритма показана на рис. 2.)

На этом можно было бы остановиться, но попробуем решить более сложную общую задачу — построение эллипсов (как известно, окружность является частным случаем эллипса). Поэтому алгоритм построения эллипса удобно разрабатывать на основе уже рассмотренного для окружности. Воспользуемся тем фактом, что эллипс получается путем «растяжения» или «сжатия» окружности. Пусть заданы радиусы эллипса, параллельные координатным осям Rx и Ry. Выберем из них

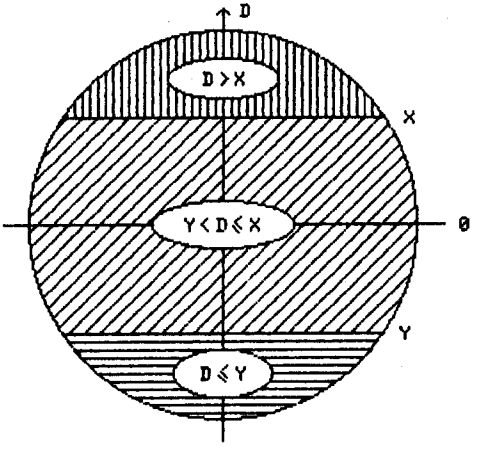


Рис. 1. Выбор направления смещения точек в алгоритме построения окружности

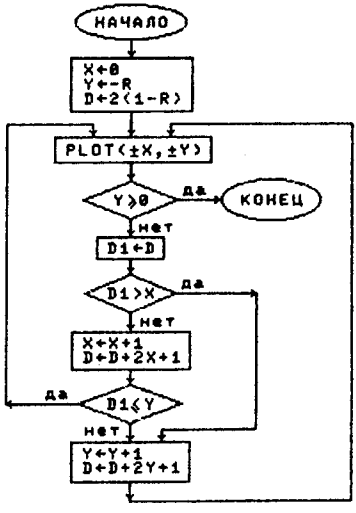


Рис. 2. Блок-схема алгоритма построения окружности

наибольший, обозначив его как H («high»), меньший же обозначим как L («low»). Сохраним также в специальной переменной K информацию, какая из величин (Rx или Ry), оказалась больше, — это пригодится в дальнейшем для ориентации эллипса. На основе наибольшего радиуса H будем строить окружность с помощью рассмотренного выше алгоритма. При этом переменная X будет использоваться в построении окружности, как обычно, изменяясь от 0 до H. Для получения же эллипса введем вспомогательную координату X1, которая будет изменяться пропорционально значению X от 0 до L.

Для соблюдения коэффициента пропорциональности воспользуемся методом, применяемым для построения отрезков прямых: введем специальную переменную U. Всякий раз, когда в соответствии с алгоритмом построения окружности X увеличивается, будем увеличивать U на величину L. Если при этом U останется меньше H, то X1 не изменяется, в противном случае X1 увеличивается на 1, а U уменьшается на H. (Смысл переменной U заключается в регулировании величины ошибки соответствия X1 идеальной пропорциональности X.)

Этот прием позволяет избавиться от операций умножения и деления (X1=X·L/H) и соответствует, в сущности, простейшей реализации алгоритма построения отрезка прямой с проекциями на оси координат, равными L и H. Для обеспечения симметрии начальное значение переменной U необходимо установить равным H/2+1.

Таким образом, при использовании координаты X в алгоритме реально на экране отображаются точки с координатой X1, соответствующие пропорционально сжатой окружности (т. е. эллипсу). При отображении на экране точки необходимо также правильно сориентировать, ведь построение велось для эллипса, большая полуось которого расположена вертикально. Если же большая полуось должна быть горизонтальна (на что указывает величина K, добавлена нами в начале алгоритма), то координаты X1 и Y нужно поменять местами. Для получения полного изображения эллипса попарно меняются знаки координат и все четыре полученные точки отображаются на экране. (Блок-схема алгоритма построения эллипса приведена на рис. 3.)

Рассмотренный алгоритм нетрудно реализовать в виде программы на ассемблере. Приведенная ниже подпрограмма предназначена для использования в составе библиотеки графических функций. В оттранслированном виде она занимает всего &O324 байта. Ее входными параметрами являются координаты центра эллипса, горизонтальный и вертикальный радиусы и цвет. Программа годится для построения как эллипсов, так и окружностей (при задании равных радиусов). Дополнительно предлагается также ее вариант в машинных кодах, предназначенный для подгрузки к всем хорошо известному графическому редактору БК-PAINT (автор А. В. Бакерин). Эти коды можно набрать вручную с адреса &O6130 взамен предыдущих кодов БК-PAINT. (По этому адресу в нем ранее находилась подпрограмма рисования окружностей, теперь же появится возможность изображать эллипсы.) Интересно, что новая подпрограмма формирования эллипсов занимает даже меньше места, чем старая для окружностей. (При подгрузке желательно пользоваться версией БК-PAINT v1.0-89, так как в других версиях подпрограмма окружностей может располагаться по другому адресу.) В качестве иллюстрации возможностей расширенной версии графического редактора приведен рис. 4.

Как видим, при построении алгоритмов рисования окружности и эллипса удалось

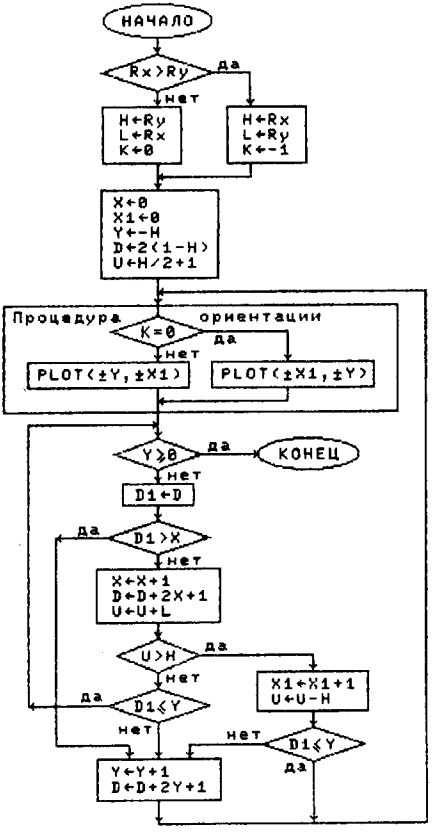


Рис. 3. Блок-схема алгоритма построения эллипса

обойтись без плавающей арифметики, хотя, казалось бы, математическое описание этих линий требует ее применения. Так, может быть, плавающая арифметика вообще предрассудок и без нее всегда можно обойтись? Конечно, не предрассудок, но замыкаться на ней тоже не стоит. Важно понимать, когда без нее можно обойтись, а когда она необходима. В разобранном алгоритме построение ведется последовательно от точки к точке, находящихся в непосредственной близости друг от друга. Математические соотношения, связывающие координаты таких контактирующих точек, оказываются очень простыми, так как большая часть необходимой информации (о положении или о величине ошибки) передается от точки к точке «по наследству». Таким образом мы избавляемся от повторных, избыточных вычислений. Построение произвольных точек окружности или эллипса, не связанных друг с другом, потребовало бы гораздо более сложных расчетов с использованием плавающей арифметики.

Принцип последовательных итераций зачастую помогает упростить вычисления, когда мы имеем дело со взаимосвязанными данными. Такой подход оказывается особенно полезен для синтеза и преобразования изображений, где всегда необходима высокая скорость обработки информации. Автор надеется, что эта статья поможет читателям открыть для себя (или реабилитировать) возможности ассемблера в области графики и избавиться от излишнего страха перед «сложной математикой».

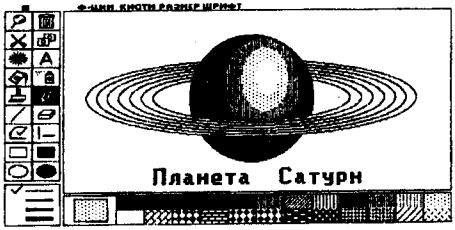


Рис. 4. Возможности программы БК-PAINT после доработки

Литература

1. *Роджерс Д.* Алгоритмические основы машинной графики: Пер. с англ. М.: Мир, 1989.

2. *Фоли Дж., Вэн Дэм А.* Основы интерактивной машинной графики: В 2 кн.: Пер. с англ. М.: Мир, 1985.

3. *Усенков Д. Ю.* Библиотека графических функций для ассемблера БК-0010(.01). // Персональный компьютер БК-0010 — БК-0011М. 1994. №1.

```
;  ************************************
;  *  Подпрограмма рисования эллипса  *
;  *  Автор:  Румянцев С.К., п.Люкня  *
;  ************************************
;
;   Входные  параметры:
;   R1=X0 , R2=Y0 - координаты центра
;   R3 - радиус по оси X
;   R4 - радиус по оси Y
;   R5 - цвет (2 байта закраски)
;
%ELLIPSE:  MOV  R0, -(SP)    ; Сохранить регистры
           MOV  R1, -(SP)
           MOV  R2, -(SP)
           MOV  R3, -(SP)    ; Если X-радиус
           BPL  1            ; отрицательный,
           NEG  R3           ; то сменить знак
1:         MOV  R4, -(SP)    ; Если Y-радиус
           BPL  2            ; отрицательный,
           NEG  R4           ; то сменить знак
2:         MOV  R5, -(SP)
           CLR  -(SP)        ; Флаг нулевой, если
           CMP  R4, R3       ; эллипс вертикальный
           BHI  3
           COM  (SP)         ; Иначе флаг не нулевой
           MOV  R4, -(SP)    ; Меньший радиус в стек,
           MOV  R3, -(SP)    ; затем больший
           MOV  R3, R4
           BR   4
3:         MOV  R3, -(SP)    ; Меньший радиус в стек,
```

```
                MOV   R4, -(SP)    ; затем больший
4:              CLR   R1           ; Вычислительный X
                CLR   R3           ; Отображаемый X
                MOV   R4,R2        ; Начальный Y
                NEG   R2
                MOV   R2,R5        ; Начальная ошибка
                INC   R5           ; окружности
                ASL   R5
                ASR   R4           ; Начальная ошибка сжатия
                INC   R4
9:              JSR   PC, %PS4     ; Отобразить 4 точки
8:              TST   R2           ; Если Y не меньше нуля,
                BGE   5            ; то на выход
                MOV   R5, R0
                CMP   R0, R1       ; Только вертикальная составляющая?
                BGT   6            ; Да - на вертикальную
                INC   R1           ; Отработка горизонтальной
                ADD   R1, R5       ; составляющей смещения
                ADD   R1, R5
                INC   R5
                ADD   2(SP), R4    ; Отработка сжатия
                CMP   R4, (SP)     ; Отображаемый X менять?
                BHI   7            ; Да - увеличить
                CMP   R0, R2       ; Только горизонтальная составляющая?
                BLE   8            ; Да - точку не показывать
                BR    6            ; Нет - на вертикальную составляющую
7:              INC   R3           ; Увеличить отображаемый X
                SUB   (SP), R4     ; Скорректировать ошибку сжатия
                CMP   R0, R2       ; Только горизонтальная составляющая?
                BLE   9            ; Да - показать точку
6:              INC   R2           ; Отработка вертикальной
                ADD   R2, R5       ; составляющей смещения
                ADD   R2, R5
                INC   R5
                BR    9
5:              ADD   #6, SP  .    ; Восстановить регистры и выход
                MOV   (SP)+, R5
                MOV   (SP)+, R4
                MOV   (SP)+, R3
                MOV   (SP)+, R2
                MOV   (SP)+, R1
                MOV   (SP)+, R0
                RTS   PC
%PS4:           MOV   R1, -(SP)    ; Подпрограмма "4 точки"
                MOV   R2, -(SP)
                MOV   R3, -(SP)
                TST   14(SP)       ; Флаг ориентации нулевой?
                BNE   10           ; Нет - эллипс горизонтальный
                MOV   R3, R1       ; Да - вертикальный
                BR    11
10:             MOV   R2, R1       ; X и Y в соответствии
                MOV   R3, R2       ; с ориентацией
11:             MOV   16(SP), R3   ; Цвет
                JSR   PC, %PS1     ; Ставить точки,
                NEG   R2           ; исключая
                BEQ   12           ; дублирование
                JSR   PC, %PS1
12:             NEG   R1
                BEQ   13
                JSR   PC, %PS1
                NEG   R2
```

```
            BEQ   13
            JSR   PC, %PS1
13:         MOV   (SP)+, R3
            MOV   (SP)+, R2
            MOV   (SP)+, R1
            RTS   PC
%PS1:       MOV   R1, -(SP)     ; Подпрограмма "1 точка"
            MOV   R2, -(SP)
            ADD   34(SP), R1    ; Получить реальное положение,
            ADD   32(SP), R2    ; прибавив координаты центра
            JSR   PC, %PSET     ; Поставить реальную точку
            MOV   (SP)+, R2
            MOV   (SP)+, R1
            RTS   PC
```

Программа рисования эллипса, подгружаемая к графическому редактору БК-PAINT v1.0-89

```
                                Адрес=006130        Длина=000340
010046 005046 016746 017652 166716 017652 100001 005416
016746 017640 166716 017640 100001 005416 026616 000002
101405 012600 012601 005116 010046 010146 011604 005001
005003 010402 005402 010205 005205 006305 006204 005204
004767 000070 005702 002403 062706 000010 000207 010500
020001 003017 005201 060105 060105 005205 066604 000002
020416 101003 020002 003756 000404 005203 161604 020002
003747 005202 060205 060205 005205 000742 010146 010246
010346 010546 005766 000016 001002 010301 000403 010201
005401 010302 005766 000020 001410 004767 000054 005402
001402 004767 000044 005401 001406 004767 000034 005402
001402 004767 000024 012605 012603 012602 012601 004767
013766 000207 000000 000000 000000 010146 010246 066702
017346 016600 000026 001014 010103 005401 066701 017326
                                Контрольная сумма=035620
```

С. К. Румянцев,
пос. Локня Псковской обл.

Наклонные эллипсы в БЕЙСИКе БК-0010.01

В отличие от окружности, эллипс имеет разную протяженность по ширине и высоте, так что его можно ориентировать под различными углами к осям координат. Это наиболее общий случай расположения эллипса на плоскости, и необходимость в его реализации часто возникает в машинной графике. К сожалению, даже оператор CIRCLE, предоставляемый БЕЙСИКом, не может изобразить наклонный эллипс. Для решения этой задачи необходимо написать специальную программу, манипулирующую более простым графическим средством БЕЙСИКа — оператором LINE.

Допустим, что эллипс имеет полуоси A и B, его центр находится в начале координат — точке O, а большая полуось составляет с осью координат OX угол I. Если этот угол равен нулю, то мы имеем дело с «прямым» (не наклонным) эллипсом. В этом случае его построение удобно вести, прослеживая все координаты Y вдоль малой полуоси, от B до −B. Для каждого Y можно найти текущий радиус R,

определяющий «толщину» эллипса, по формуле: $R=(A/B)\cdot SQR(B^2-Y^2)$. Величину R надо откладывать вправо и влево от оси OY (т. е. со знаком «+» и «−»), так как эллипс симметричен.

Для того чтобы перейти к эллипсу, находящемуся под ненулевым углом, как оказалось, достаточно вести построение +R и −R относительно другой, наклонной оси. Необходимо лишь найти коэффициент «перекоса» такой наклонной оси, а также коэффициент, дающий поправку к величине R с учетом угла наклона.

На рис. 1 изображен наклонный эллипс. Его наивысшая точка M намечает положение наклонной оси OM, относительно которой будет проводиться построение. Расстояние H от точки M до оси OX определяет верхний предел изменения координаты Y. Для вычислений нам понадобится квадрат этой величины (переменная H2), который может быть найден по формуле $H2=(A\cdot SIN(I))^2+(B\cdot COS(I))^2$.

Тогда коэффициент сдвига наклонной оси равен $K1=SIN(I)\cdot COS(I)\cdot(A^2-B^2)/H2$.

Коэффициент же «толщины» эллипса получим по формуле $K2=A\cdot B/H2$.

Тогда текущий радиус наклонного эллипса будет равен $R=K2\cdot SQR(H2-Y^2)$. Эту величину радиуса мы будем откладывать симметрично относительно наклонной оси вправо и влево. Горизонтальные координаты полученных точек эллипса будут равны

$$\begin{cases} X1=K1\cdot Y+R; \\ X2=K1\cdot Y-R. \end{cases}$$

Таким образом, прослеживая все возможные координаты Y от H до −H, мы найдем все точки горизонтальных сечений, а соединив их попарно отрезками — получим полное изображение наклонного эллипса. (Эта технология была разработана автором в 1987 г.) Приведенная ниже демонстрационная программа на БЕЙСИКе реализует рассмотренный алгоритм.

Программа изображает серию эллипсов под различными углами (рис. 2), где за каждый эллипс отвечает подпрограмма со строки 1000. Ее можно использовать и в других программах, где возникает такая необходимость (например, для получения более сложного изображения, показанного на рис. 3). Для нормальной работы подпрограмме 1000 нужно передать восемь переменных, определяющих параметры эллипса:

- X0 и Y0 — координаты центра эллипса;
- A — большая полуось;
- B — малая полуось;
- I — угол наклона (вместо угла можно передать его синус S и косинус C, после чего вызывать подпрограмму сразу со строки 1030);
- C1 — цвет контура эллипса;
- C2 — цвет закраски внутренней части эллипса (при C2<0 внутренняя часть не закрашивается);
- G — коэффициент сжатия экрана. Он нужен для достижения правильных пропорций на любых типах мониторов и численно равен отношению сторон X/Y прямоугольника, который на экране вашего монитора воспринимается как точный квадрат.

Такое количество параметров необходимо для универсальности этой подпрограммы. Кроме перечисленных для внутренних вычислений используются также переменные H2, K1, K2, Y, R, X1, X2, X3, X4 (эти имена в основной программе (вызывающей данную программу) использовать нельзя. Назначение этих переменных ясно из описания алгорит-

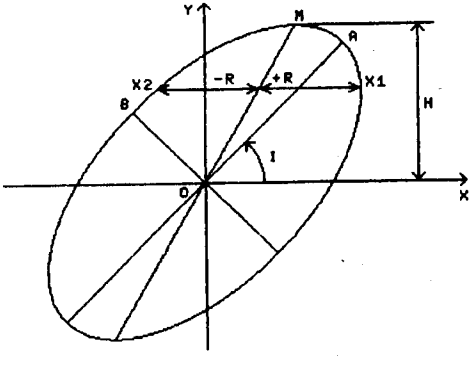


Рис. 1. Построение наклонного эллипса

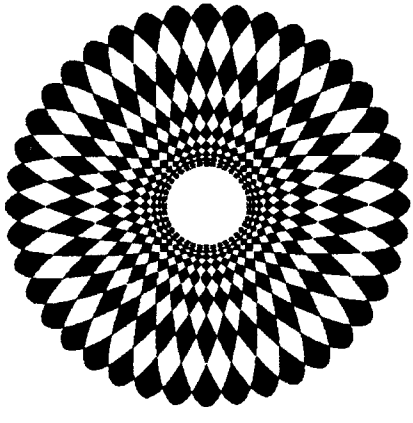


Рис. 2. Изображение, получаемое с помощью демонстрационной БЕЙСИК-программы

ма. Добавлю только, что X3 и X4 являются координатами правой и левой точек сечения на предыдущей строке и служат для проведения отрезков к аналогичным точкам на текущей строке. Для сокращения вычислений программа ведет построение в обе стороны от оси 0X (вверх и вниз), пользуясь симметрией эллипса. В конце процесса программа округляет верхушки эллипса до ближайшего целого Y. Если один из цветов — C1 или C2 — был равен нулю, то при остановке программы по клавише «СТОП» на экране могут перестать появляться буквы. Не пугайтесь, это не зависание. Просто им надо вернуть видимый цвет, нажав «АР2»+«1», а затем «1» и «ВВОД».

Рассмотренный алгоритм построения наклонного эллипса в принципе можно реализовать и на ассемблере. Конечно, он требует большого количества вычислений с «плавающей» точкой. Но если вызывать подпрограмму в кодах из программы на БЕЙСИКе, то большинство подготовительных расчетов можно выполнить в нем, а подпрограмме передавать только три параметра, округленных до целых чисел:

- высоту наклонного эллипса H=INT(SQR(H2));
- наибольший сдвиг наклонной оси INT(K1*H);
- наибольший радиус горизонтального сечения INT(K2*H).

Этих трех параметров кодовой подпрограмме построения наклонного эллипса будет достаточно. Подпрограмма в кодах, несомненно, справится с задачей рисования гораздо быстрее, чем БЕЙСИК, зато последнему будут более «по зубам» сложные подготовительные расчеты. Так можно достичь оптимального разделения труда между языками высокого и низкого уровней.

```
10 REM * Демонстрационная программа *
20 REM * рисования наклонных эллипсов *
30 REM * Автор: Румянцев С.К. *
40 CLS
50 X0=128        ' Координаты
60 Y0=120        ' центра эллипса
70 A=115         ' Большая полуось
80 B=25          ' Малая полуось
90 G=0.7         ' Коэф. сжатия экрана
100 C1=1%        ' Цвет контура
110 C2=0%        ' Цвет закраски
120 FOR I=0 TO PI STEP PI/20
                 ' Угол наклона
130 GOSUB 1000
140 NEXT I
150 STOP
```

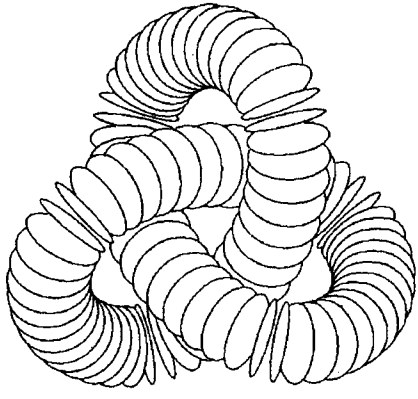
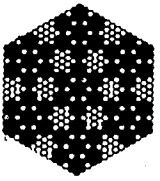


Рис. 3. Более сложная картинка из наклонных эллипсов

```
1000 REM * П/пр "Наклонный эллипс" *
1010 S=SIN(I)
1020 C=COS(I)
1030 H2=(A*S)^2%+(B*C)^2%
1040 K1=G*S*C*(A^2%-B^2%)/H2
1050 K2=G*A*B/H2
1060 Y=0
1070 IF Y^2%>H2 THEN 1230
1080 R=K2*SQR(H2-Y^2%)
1090 X1=K1*Y+R
1100 X2=K1*Y-R
1110 IF C2<0 THEN 1140
1120 LINE (X0+X1,Y0+Y)-(X0+X2,Y0+Y),C2
1130 LINE (X0-X1,Y0-Y)-(X0-X2,Y0-Y)
1140 IF Y=0 THEN 1190
1150 LINE (X0+X1,Y0+Y)-(X0+X3,
     Y0+Y-1),C1
1160 LINE (X0+X2,Y0+Y)-(X0+X4,Y0+Y-1)
1170 LINE (X0-X1,Y0-Y)-(X0-X3,Y0-Y+1)
1180 LINE (X0-X2,Y0-Y)-(X0-X4,Y0-Y+1)
1190 X3=X1
1200 X4=X2
1210 Y=Y+1
1220 GOTO 1070
1230 H2=INT(1.99*(Y-SQR(H2)))
1240 LINE (X0+X3,Y0+Y-1)-(X0+X3-R,
     Y0+Y-H2),C1
1250 LINE -(X0+X4,Y0+Y-1)
1260 LINE (X0-X3,Y0-Y+1)-(X0-X3+R,
     Y0-Y+H2)
1270 LINE -(X0-X4,Y0-Y+1)
1280 RETURN
```



Значительным удобством при работе с БК является наличие в ее мониторе стандартных графических функций для вывода точек и отрезков (EMT 30 и EMT 32), а в более поздней модели БК-0011(М) — и окружностей. Данная статья ознакомит БКманов с алгоритмом вывода точки (EMT 30) и позволит расширить его графические возможности.

Д. Ю. Усенков,
Москва

«Негативная» графика на БК

О преимуществах и недостатках реализации графики на компьютерах семейства БК сказано и написано уже немало, но еще много «подводных камней» ожидает пользователя. Многие сталкиваются с трудностями при реализации графических программ, в которых ранее нарисованные фигуры нужно удалять с экрана, не «портя» при этом остального изображения. Примером является создание «графического курсора» в графических редакторах или «выстрелов лазерных лучей» в играх. Конкретная сложность состоит в том, что при стирании, например, прямой (или при ее закраске цветом фона, что то же самое) в цвет фона закрашиваются все точки, через которые ранее проходила эта прямая. Чтобы сохранить изображение на экране под нарисованным, а потом удаленным элементом, используются разные способы, но все они обладают существенными недостатками.

Перерисовка всего изображения на экране заново: медленно и возможно только, когда сведения о всех элементах рисунка хранятся в памяти. Например, это возможно в графическом редакторе GRAF, хранящем в памяти порядок рисования картинки, но нельзя реализовать в ВК-PAINT, где изменения изображения на экране не записываются в память.

Инверсия байтов ОЗУ экрана: достаточно быстро, но имеет дискретность по горизонтали в четыре (или восемь для режима 64 символа в строке) точки. Таким способом можно организовать «инверсный курсор» в меню операций (как в PGM) или выделить на экране прямоугольную область для записи в качестве спрайта, но сложно реализовать вертикальные и наклонные линии толщиной в одну точку.

Поточечное построение и стирание с проверкой предыдущего цвета точки: на БЕЙСИКе очень медленно, в кодах сложно реализуется из-за отсутствия (по крайней мере на БК-0010(.01)) EMT-функции для определения цвета точки с заданными координатами. К тому же такой алгоритм требует перебора в операторах IF всех возможных в данном режиме номеров цветов точек для замены на «противоположные» цвета, что сильно загромождает программу.

Причина трудностей заключается в принципе реализации графики на БК. При построении точки с помощью EMT 30 соответствующему биту (или паре битов в цветном режиме) безусловно присваивается значение, кодирующее цвет (при стирании точки — цвет фона). Будем называть такой способ рисования «прямой» графикой. Этот способ является стандартным и реализован практически на всех компьютерах и во всех языках программирования, обеспеченных графическими возможностями, например Borland C++ на IBM. Но в том же C++ имеется реализация и другого способа рисования, который мы назовем «негативной» графикой. Он заключается в том, что цвет рисуемых точек задается «обратным» по отношению к прежнему цвету тех же точек. А когда эти точки рисуются повторно, их цвета приводятся к прежним, так как соответствие цветов является строго однозначным. (Отсюда и смысл названия: для двух цветов, например черного фона и белого переднего плана, рисуемая таким способом на белом фоне картинка будет «негативом» той, что рисуется на черном.)

Но ведь нам и нужно, чтобы при повторном рисовании той же самой точки на том же месте эта точка бесследно исчезала с ранее нарисованной на экране картинки. Остается реализовать подобный способ рисования на БК.

Предлагаемая несложная подпрограмма на ассемблере является несколько модернизированной версией реализации EMT 30, зашитой в ПЗУ монитора БК. В ней сохранен алгоритм нахождения битов экранного ОЗУ, соответствующих заданным координатам точки и изменен только механизм задания значения этих битов (цвета точки). Подпрограмма перемещаема и может быть использована в любой из ассемблерных программ вместо EMT 30. Желающие могут доработать ассемблерный листинг, чтобы получить USR-подпрограмму для БЕЙСИКа.

Как и в EMT 30, перед вызовом подпрограммы нужно задать значения координат точки в R1 и R2. (Значение R0, вообще говоря,

излишне, и его можно не задавать вовсе. Этот регистр и строка MOVB R0,@#153 в ассемблерном листинге зарезервированы для сохранения универсальности — см. ниже.) Подпрограмма проверяет значения координат, и, если заданная точка находится за пределами экрана, производится выход. Иначе выполняется перекраска точки в цвет, «обратный» предыдущему. При этом подпрограмма может работать как в «цветном» (32 символа в строке), так и в «черно-белом» (64) режиме. Повторный вызов подпрограммы при задании тех же самых координат стирает точку (цвет, «обратный» к «обратному» цвету, есть исходный).

Следует заметить, что при использовании приведенного здесь стандартного алгоритма можно реализовать и другие операции с графическими точками. Например, заменив блок листинга между метками 3 и 100, можно реализовать ассемблерный аналог оператора БЕЙСИКа POINT, учитывающий рулонное смещение (кстати, оно учтено и в подпрограмме «негативной» графики) и работоспособный в любом режиме «цветности». (Через R0 и ячейку @#153 можно передавать дополнительный параметр.) Текст ассемблерной вставки может быть таким

```
3:    MOVB @R1,R0        ; Текущий байт видеоОЗУ -> R0
      CLC                ; Очистить бит C
5:    ROR R2             ; Смещать биты, соответствующие
      BCS 6              ; данной точке, вправо, пока
      ROR R0             ; они не окажутся в самых
      BR 5               ; младших позициях
6:    MOVB @#155,R2      ; Маска точки #1 или #3
      COM R2             ; Выделить из байта видеоОЗУ
      BIC R2,R0          ; биты, соответствующие маске точки
      TSTB @#40          ; Режим 64 символа в строке?
      BEQ 100            ; Да - на выход
      CMP R0,#1          ; Если 32 символа
      BEQ 7              ; в строке, то
      CMP R0,#3          ; заменить
      BNE 100            ; код цвета #1
      MOV #1,R0          ; на номер 3
      BR 100             ; и наоборот
7:    MOV #3,R0          ;
      BR 100             ;
      MOV #177777,R0     ; Сюда передается управление
                         ; вместо метки 100, если точка
                         ; находится за пределами экрана
100:  . . . . .          ; Окончание программы
```

В этом случае после выхода из подпрограммы в R0 будет записан цвет точки:

— в режиме 64 символа в строке: 1 — белый, 0 — черный;

— в режиме 32 символа в строке: 1 — красный, 2 — зеленый, 3 — синий, 0 — черный.

Если заданные координаты точки выходят за пределы экрана, R0 содержит число 177777 (−1) независимо от режима.

Другим направлением творческих изысканий может послужить реализация по данному принципу аналога EMT 32 для рисования отрезков, однако стандартный алгоритм этой EMT-функции более сложен. При использовании же подпрограммы «негативной» графики в качестве USR-подпрограммы на БЕЙСИКе можно применить менее быстрый, но зато простой способ рисования прямых (и эллипсов) по точкам в цикле. Например, прямая может быть построена согласно формуле:

$$Y=Y1+\frac{X-X1}{X2-X1}\cdot(Y2-Y1)$$

```
FOR X=X1 TO X2
Y=Y1+(X-X1)/(X2-X1)*(Y2-Y1)
.............
```

где (X1,Y1) и (X2,Y2) — координаты начальной и конечной точек, а (X,Y) — вычисляемые координаты текущей точки. Для эллипса используется другая формула:

$$\begin{cases} X=X0+A\cos\dfrac{\pi i}{N}; \\ Y=Y0+B\sin\dfrac{\pi i}{N}. \end{cases}$$

```
FOR I=1 TO 2*N
X=X0+A*COS(PI*I/N)
Y=Y0+B*SIN(PI*I/N)
..............
```

где (X0,Y0) — координаты центра, i — номер текущей точки (i=1...2N), A и B — длины полуосей, а (X,Y) — вычисляемые координаты текущей (i-той) точки.

```
;  Подпрограмма рисования "негативной" точки        SCREW(c)
;  ---------------------------------------------------------
; Входные данные:      R1 - координата X;
;                      R2 - координата Y;
;                      R0 - фиктивный параметр.
; Выходные данные:     нет.
; Значения регистров: сохраняются в стеке.
;  ---------------------------------------------------------
;
NEGSET:  MOV R1,@#176          ; Сохранение в буферных переменных
         MOV R2,@#200          ;  координат X и Y, а также
         MOVB R0,@#153         ;  фиктивного параметра
         JSR R4,@#110346       ; Регистры в стек
;
         TSTB @#40             ; Проверка, не выходят ли значения
         BEQ 1                 ;  координат за пределы экрана
         ASL R1
         BCS 100
1:       CMP R1,#1000
         BCC 100
         MOV #6,R0
2:       ASL R2
         BCS 100
         SOB R0,2
         CMP R2,@#210
         BCC 100
;
         MOV R1,R3             ; Вычисление байта в ОЗУ экрана
         ASR R1                ;  и формирование маски точки:
         ASR R1                ;       (R1 - адрес,
         ASR R1                ;        R2 - маска)
         ADD R2,R1
         ADD @#204,R1
         BIC #140000,R1
         ADD @#202,R1
         MOVB @#155,R2
         BIC #177770,R3
         BEQ 3
4:       ASL R2
         SOB R3,4
;
3:       MOVB @R1,R3           ; Формирование точки "инверсного"
         COMB R2               ;  цвета
         COMB R3
         BICB R2,R3
         COMB R2
         BICB R2,@R1
         BISB R3,@R1
;
100:     JSR R4,@#110362       ; Регистры из стека
         RTS PC                ;  и выход
; ----------------------------------------------------------
;
```

Кодовый листинг

```
010137  000176  010237  000200  110037  000153  004437  110346
105737  000040  001402  006301  103444  020127  001000  103041
012700  000006  006302  103435  077003  020237  000210  103031
010103  006201  006201  006201  060201  063701  000204  042701
140000  063701  000202  113702  000155  042703  177770  001402
000302  077302  111103  105102  105103  140203  105102  140211
150311  004437  110362  000207  000000  000000  000000  000000

ДЛИНА: 000160
КОНТРОЛЬНАЯ СУММА: 156526
```

В заключение приведем сведения о системных ячейках БК, использованных в данной подпрограмме.

Восьмеричный адрес	Назначение
40 (байт)	Флаг режима: 32/64 символа в строке (377_8 или 0 соответственно)
153 (байт)	Флаг закраски точки: 1 — в цвет «переднего плана», 0 — в цвет фона
155 (байт)	Битовая маска точки: #1 — в режиме 64 символа в строке, #3 (т. е. два бита) — в режиме 32 символа в строке
176 200	Координаты текущей обрабатываемой точки (X и Y соответственно)
202	«Базовый» адрес начала экранного ОЗУ (обычно 40000_8, в режиме РП 70000_8)
204	Добавка к адресу начала экранного ОЗУ с учетом рулонного смещения
210	Объем экранного ОЗУ без учета служебной строки (обычно 36000_8, в режиме РП 5000_8)
212 214	Маски цветов фона и «переднего плана»: #0 — черный, #125252 — зеленый, #52525 — синий, #177777 — красный (или «белый» в режиме 64 символа в строке)

Примечание. Ячейки 153, 176 и 200 содержат числа, занесенные перед вызовом EMT 30 в регистры R0, R1 и R2 соответственно. После отработки EMT 30 (или EMT 32) ячейки 176 и 200 содержат координаты X и Y последней обработанной точки.

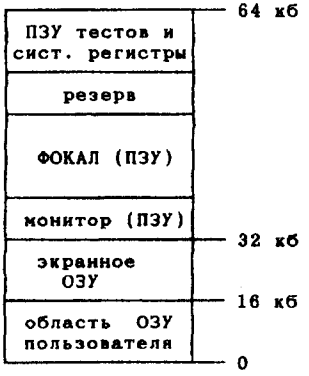
Хотя после появления БК-0010.01 с зашитым в ПЗУ вильнюсским БЕЙСИКом язык ФОКАЛ несколько утратил популярность, многие пользователи БК все же отдают ему предпочтение. В этой статье рассказывается о некоторых полезных алгоритмах, позволяющих существенно расширить графические и звуковые возможности написанных на ФОКАЛе программ.

А. В. Милюков,
Москва

ФОКАЛ: графические возможности, и не только

Работа с экранным ОЗУ БК-0010

Адресное пространство микроЭВМ БК-0010* делится на несколько больших областей.



Половина всего адресного пространства занята ПЗУ, в котором содержатся монитор, язык ФОКАЛ и встроенные тестовые программы. Адресное пространство в диапазоне 140000_8—157777_8 не используется и соответствует ПЗУ пользователя, вставляемому в резервное гнездо отсека «СТОП—ПУСК» слева от клавиатуры БК-0010 (на БК-0010.01 резервное место имеется внутри блока МСТД и не снабжено разъемом для установки микросхем). Кроме того, следует не забывать, что с адреса 177600_8 в БК-0010(.01) располагается зона адресов системных регистров, так что часть ПЗУ тестов (или БЕЙСИКа на БК-0010.01) из использования выключается (ячейки микросхемы, соответствующие адресам 177600_8—177777_8, даже не прошиваются информацией). Благодаря наличию монитора и ФОКАЛа в ПЗУ компьютер способен начать работу сразу после включения, без загрузки каких-либо программ с магнитофона.

Другая область памяти целиком занята ОЗУ, которое, в свою очередь, делится на ОЗУ пользователя и ОЗУ экрана. В ОЗУ пользователя хранятся программы и данные, которые всякий раз снова загружаются в память для каждой решаемой задачи. Экранное же ОЗУ служит только для хранения копии изображения. Когда электронный луч в кинескопе пробегает по поверхности экрана, в такт с его движением специальная микросхема на плате ЭВМ считывает одну за другой содержимое битов экранной области ОЗУ. Если данный бит содержит ноль, то соответствующая ему точка экрана остается темной, иначе она высвечивается. Пробегая по экрану с высокой скоростью, так что мелькание незаметно для глаза, луч создает изображение, повторяющее содержимое памяти машины. Чтобы напечатать что-либо на экране, программа помещает требуемое изображение в ОЗУ экрана.

Следующая несложная программа на ФОКАЛе позволит наблюдать «невооруженным глазом» процесс работы с экранной памятью.

```
1.10 S A=16384              ; C   Адрес начала ОЗУ экрана
1.20 F I=0,2,160000 ; D 5   ; C   Объем ОЗУ = 16 кб
1.30 Q                      ; C   Выход в режим диалога
5.10 S B=A+I                ; C   Задание адреса ячейки
5.20 X FX(-1,B,I)           ; C   Запись в нее значения I
5.30 R                      ; C   Возврат в цикл
```

* То же верно для БК-0010.01 с подключенным блоком МСТД.

Программа работает следующим образом. Задается адрес первой из ячеек экранного ОЗУ, равный 40000_8. Затем организуется цикл со счетчиком I, который выполнится 8000 раз (шаг задан равным 2). Этот цикл содержит вызов подпрограммы с номером 5. По окончании цикла выполнение прекращается и БК переходит в диалоговый режим (приглашение «*»). Подпрограмма 5 суммирует счетчик цикла и адрес начала памяти экрана. В результате переменная В всегда содержит адрес одной из ячеек ОЗУ экрана, и в эту ячейку записывается значение I (можно использовать вместо I любое число или переменную). Очевидно, что, хотя значение А не меняется в цикле, адрес ячейки ОЗУ будет увеличиваться на 2 при каждом вызове подпрограммы за счет увеличения счетчика цикла I. В результате запись будет происходить во все новые и новые ячейки. Результат работы программы — заполнение экрана точечным орнаментом, что наглядно демонстрирует однозначное соответствие содержимого памяти и изображения на экране. (Данная программа аналогична ассемблерной реализации теста ОЗУ, прошитой в ПЗУ диагностики.)

Интересный результат можно получить, меняя параметры цикла и значение, которым заполняется ОЗУ экрана. Так, использование в строке 5.20 вместо I констант, равных 2^n, где n — целое число, или сумме таких констант, позволит расчертить экран вертикальными полосами. Если подготовить несколько констант, например в массиве, и записывать их в экранную память с шагом цикла, равным 64, то можно создавать спрайты. (Одна строка экрана соответствует 64 (100_8) байтам памяти. Поэтому записанные с таким шагом числа будут находиться на экране одно под другим в соседних строках, образуя прямоугольник.)

Звуковые эффекты

В самом конце адресного пространства БК располагается область системных регистров. Для программиста или пользователя регистр выглядит как ячейка памяти, имеющая свой адрес. Но, в отличие от обычного ОЗУ, которым можно распоряжаться по собственному желанию, регистры чаще всего используются только монитором и ФОКАЛом «незаметно» для пользователя. Но иногда прямое обращение к регистрам оказывается необходимым, например когда нужно в ФОКАЛ-программе обеспечить требуемые для игры звуковые эффекты.

Для ЭВМ пьезодинамик (излучатель звука, размещенный под верхней крышкой корпуса) является внешним устройством. Для управления им (получения звука) особых команд ни в ассемблере, ни в ФОКАЛе не предусмотрено. Любой звуковой эффект на БК-0010.01 — это результат выполнения целой подпрограммы.

Как же получить на БК звук? Динамик подключен к специальной микросхеме, которая представлена регистром с адресом 177716_8 (-50_{10}). При записи в этот регистр процессор передает данные в указанную микросхему и уже с ее выхода снимается сигнал для динамика. Если записать в специально предназначенный для этой цели бит 6 регистра «лог. 1», то мембрана динамика отклонится в одну сторону, а если «лог. 0», то в другую. Посылая поочередно нули и единицы в течение некоторого времени, мы будем слышать звук.

Для примера рассмотрим несложную ФОКАЛ-программу, в которой циклически вызывается подпрограмма, записывающая в регистр сначала нуль, а затем ненулевое число, обеспечивая тем самым щелчок динамика.

```
1.20 F I=0,2,160000 ; D 5   ; C Цикл вызова подпрограммы
1.30 Q                      ; C Конец программы
5.10 S B=-50                ; C Адрес регистра динамика
5.20 X FX(-1,B,0)           ; C Очистка регистра
5.30 X FX(-1,B,64)          ; C Запись константы
5.40 R                      ; C Возврат из подпрограммы
```

Следует отметить, что рассматриваемый регистр управляет не только динамиком, но и чтением-записью на магнитофон и рядом других операций. Подробно об этом можно прочитать в статье Ю. А. Зальцмана (№2 за 1994 г.). — *Прим. ред.*

Рисование окружности

В отличие от БЕЙСИКа, где имеется готовый оператор CIRCLE, в ФОКАЛе средства для рисования окружности отсутствуют и для этого приходится использовать самостоятельно разработанные процедуры.

Наиболее простой и наглядный алгоритм состоит в расчете по формулам Y=R*SIN(A) и X=R*COS(A) координат каждой точки, соответствующей значениям угла А от 0 до 360°. Недостаток такого метода в том, что при малом количестве точек окружность получается не

сплошной, а для большего их количества резко увеличивается время на вычисления и построение.

Другой алгоритм предусматривает работу в полярной системе координат: $R=R_0 \cdot SIN(A)$ или $R=R_0 \cdot COS(A)$. Здесь угол меняется от 0 до 180°, а положение точки совпадает с вершиной вектора R, идущего из начала координат (0,0) под углом A к оси OX. Преимущество этого метода в том, что можно вычерчивать окружность отрезками (хордами) с помощью функции FV. При количестве хорд 12...20 ломаная достаточно близка к окружности, а время построения невелико. Недостаток же этого метода — необходимость обратного пересчета полярных координат в декартовы.

Существует и алгоритм, основанный на целочисленности экранных координат. Смысл его в том, что нельзя задать две различные точки A(12,15.7) и B(12.4,15) на экране в силу его ограниченной разрешающей способности: обе эти точки сольются в одну с координатами (12,15). Таким образом, для получения непрерывного изображения достаточно рассчитать только целые координаты точек окружности.

Для построения четверти окружности, лежащей в первом квадранте, начиная с левой точки с координатами (0,R) и до правого края дуги (R,0) выполняются следующие действия:

- нарисовать точку (X,Y);
- найти сумму квадратов координат текущей точки P;
- если эта сумма меньше или равна квадрату радиуса окружности, то увеличить координату X;
- если сумма больше, то уменьшить координату Y;
- проверить, не достигнута ли правая точка дуги;
- если нет, повторить все действия с начала.

Поскольку окружность является центрально-симметричной фигурой, можно за один цикл расчета формировать не одну, а четыре точки с противоположными относительно начала координат положениями. Для рисования окружности в произвольном месте экрана достаточно в момент вывода точки прибавлять константы к ее координатам.

```
1.10 S R=30; S Y=R           ; C Исходные значения
1.20 S X=0                   ; C
1.30 D 5                     ; C Вывод четырех симметричных точек
1.40 S P=X*X + Y*Y           ; C Квадрат радиуса
1.50 IF (P-R*R) 1.6,1.6,2.1 ; C Точка вне окружности?
1.60 S X=X+1; G 2.2          ; C Нет: шаг вправо
2.10 S Y=Y-1                 ; C Да: шаг внутрь
2.20 IF (X-R) 1.3,1.3        ; C Если не правый край дуги, продолжать
2.30 Q                       ; C Конец программы
5.10 D 7                     ; C Вывод первой точки
5.20 S X=-X; D 7             ; C Вторая - симметрично по вертикали
5.30 S Y=-Y; D 7           ; C Третья - симметрично относительно центра
5.40 S X=-X; D 7; S Y=-Y     ; C Четвертая
5.50 R                       ; C Возврат
7.10 X FT(1,X+200,Y+100); R ; C Вывод точки с координатами (X,Y)
```

При совершенствовании программы (прежде всего для повышения быстродействия) следует обратить внимание на порядок вычислений. В программе многократно используется умножение, но от этой операции можно отказаться. Из листинга видим, что абсцисса только увеличивается, а ордината только уменьшается, кроме того, вычисляются их квадраты. Таким образом, задача сводится к нахождению рекуррентного соотношения, позволяющего определять для некоторого натурального числа квадраты его предыдущего и последующего значений. Для нахождения этого соотношения достаточно школьного курса математики, поэтому оставим данную задачу читателям для самостоятельного решения.

Моделирование процессов средствами машинной графики

Две приведенные ниже программы реализуют простейшие возможности моделирования процессов на микроЭВМ БК-0010(.01). Первая иллюстрирует получение псевдослучайных узоров при помощи детерминированных математических зависимостей. При достаточно долгой работе программы можно наблюдать образование изображений, схожих с объектами живой природы.

Вторая программа удобна как иллюстративный материал при изучении физики: на экране создается изображение фигур Лиссажу (можно дополнить листинг фрагментом, выполняющим удобный ввод новых значений частот.

Построение псевдослучайных узоров Б. Мартина

```
1.05 S X=128; S Y=120; S A=100; S B=1.9; S C=-40
                                        ; C Исходные значения
1.10 F I=1,4000; D 2                    ; C В цикле выполнять фрагмент 2
1.20 Q                                  ; C Конец программы
2.10 X FT(1,X,Y)                        ; C Вывод точки
2.20 S U=Y-FSGN(X-128)*FSQT(ABS(B*(X-128)-C))
                                        ; C Новые данные
2.30 S W=A-X+128                        ; C
2.40 S X=U                              ; C
2.50 S Y=W                              ; C
2.60 R                                  ; C Возврат в цикл
```

Можно выбирать B из списка {0.1, 0.2, 1.9} C={−40, −20, −10}

Фигуры Лиссажу

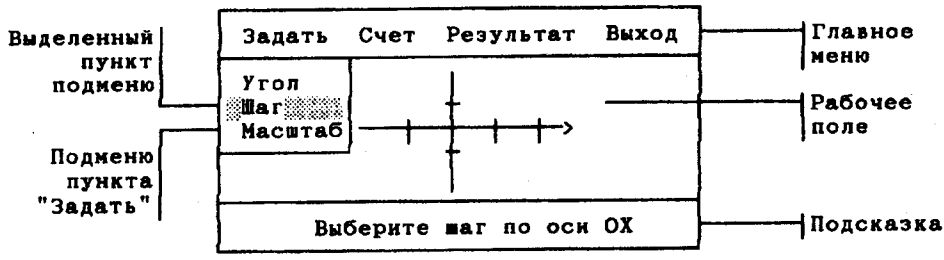
```
1.10 S X=FSIN(A/5)*30 ; C  или      X=FSIN(A/7)*100
1.20 S Y=FCOS(A/7)*50 ; C           Y=FCOS(A/3)*50
1.30 D 7              ; C  Вызов процедуры 7 (вывод точки)
1.40 S A=A+0.1        ; C  Увеличение угла
1.50 G 1.1            ; C  Продолжение расчета координат
1.60 Q                ; C  Конец программы
7.10 X FT(1,X+200,Y+100); R ; C Вывод точки на экран
```

Создание простейшего меню

В настоящее время в связи с развитием материальной базы кабинетов ИВТ, появлением цветных или графических мониторов гораздо больше внимания следует уделять оформлению программ. Основная часть УЧЕБНОЙ ПРОГРАММЫ (проще говоря, калькулятор) обычно не содержит сложных формул и расчетов, так что соответствующую часть листинга разработать легко. Вопросы же удобного ввода и вывода результатов счета, а также управления программой нередко остаются в тени. Это нельзя считать верным, так как среди двух программ аналогичного назначения пользователь, как правило, выбирает лучше оформленную и более удобную в работе.

Современные представления об организации управления программой предполагают отказ от цифрового или буквенного ввода команд, т. е. надписи типа «Для вывода таблицы нажмите 3, для расчета — 5» нежелательны. Предпочтительный вид экрана показан ниже (один из возможных вариантов).

На рисунке изображен вид некоей программы для построения семейства графиков. Имеется главное меню, содержащее названия групп операций, которое занимает верхнюю строку экрана. По строке клавишами управления курсором перемещается указатель в виде символа звездочки или инверсного изображения, и после нажатия клавиши «ВВОД»



раскрывается подменю или сразу выполняется выбранное действие. В подменю при необходимости можно уточнить, что именно нужно сделать. Для удобства пользователя в нижней строке выводится подсказка, описывающая выбранную команду или выполняемое в данный момент действие. Очевидно, что в данном случае пользователь ВСЕ нужные действия совершает, используя только пять клавиш. Нет необходимости и запоминать инструкцию, она (в виде подсказок и пунктов меню) всегда на экране.

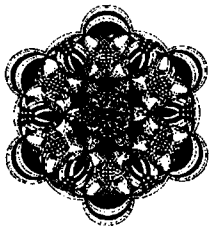
Главная проблема при создании подобного интерфейса — разработка подпрограмм меню. Вот простейшая программа выбора из трех вариантов*.

Идея состоит в том, что один из пунктов меню, выбранный в данный момент, должен на экране отличаться от остальных. Для выделения можно использовать цвет, инверсию или какой-либо символ, помещаемый рядом с этим пунктом. Достаточно удобно применять инверсию, так как включение и выключение этого режима производится одним и тем же способом — с помощью кода 156. Сначала номер выделенного пункта N будет равен 0. Распечатывая в цикле (строка 1.10) все три пункта меню, будем проверять, не совпадает ли текущий номер пункта (равный счетчику цикла I) с номером выделенного пункта (строка 5.10). Если нет, то печатаем на экране его текст, как обычно. Иначе ДО печати переходим в режим инверсии, а ПОСЛЕ снимаем этот режим. Таким образом, среди пунктов меню один будет выведен инверсно. Далее ждем реакцию пользователя (строка 1.20) и по коду нажатой им клавиши определяем, что нужно сделать. После нажатия клавиши «ВВОД» заканчиваем программу, при этом N хранит номер выбранного пункта меню. Для «стрелок» уменьшаем или увеличиваем N, обращая внимание на выход значения за пределы 0...2 (так как имеется всего три строки меню).

```
1.10 F I=0,2; D 5           ; C Вывод меню из трех пунктов
1.20 IF (FCHR(-1)-26) 1.30,1.40,1.50
                            ; C Если нажата клавиша "ВВОД", то
1.30 T N; Q                 ; C конец программы
1.40 S N=N-1                ; C Если "ВВЕРХ",  то уменьшаем N
1.41 IF (N) 1.42,1.1,1.1    ; C Если N отрицательно,  то перейти
1.42 S N=2; G 1.1           ; C на нижний пункт меню
1.50 S N=N+1                ; C Если "ВНИЗ", то увеличиваем N
1.51 IF (N-2) 1.1,1.1       ; C Если была нижняя строка, то
1.52 S N=0; G 1.1           ; C перейти на верхнюю
5.10 IF (I-N) 5.3,5.2,5.3   ; C Если строка выделена,
5.20 X FCHR(156)            ; C то дать код инверсии символов
5.30 X FK(11,I)             ; C Установить курсор
5.40 S V=I+10               ; C Вычислить номер строки с пунктом меню
5.50 D V                    ; C Напечатать текст пункта
5.60 IF (I-N) 5.8,5.7,5.8   ; C Если нужно, снять инверсию
5.70 X FCHR(156)            ; C
5.80 R                      ; C Возврат в цикл
10.10 T "ЯБЛОКО"            ; C Первая строка меню
10.20 R                     ; C
11.10 T "ГРУША "            ; C Вторая строка
11.20 R                     ; C
12.10 T "СЛИВА "            ; C Третья строка
12.20 R                     ; C
```

Данная программа содержит две весьма полезные строки — 5.40 и 5.50. Известно, что в ФОКАЛе крайне неудобно работать с символьными данными. Понятно также, что для каждого меню не надо писать собственную процедуру, так как приведенная выше может работать с любым меню (достаточно вместо константы 10 в строке 5.40 поместить переменную, указывающую номер строки с первым пунктом нужного меню). Упомянутые строки программы иллюстрируют метод работы с символьными константами и показывают способ использования одной процедуры для вывода нескольких меню. Язык ФОКАЛ дает такую возможность благодаря допустимости переменного номера строки в операторе D (DO).

* Листинги подпрограмм для реализации меню и окон на ФОКАЛе, БЕЙСИКе и ассемблере даны в №3 за 1994 г. — *Прим. ред.*



Пользователи, работающие с графическими изображениями, часто сталкиваются с необходимостью изменить размеры растровой картинки. Например, чтобы получить распечатку экрана в прямоугольнике с заданными длинами сторон или придать правильные пропорции изображению на данном мониторе, требуется растянуть или сжать картинку по вертикали или по горизонтали. К сожалению, эта задача пока не под силу имеющимся сегодня растровым графическим редакторам для БК0010, а между тем она вполне разрешима средствами ассемблера.

С. К. Румянцев,
пос. Локня Псковской обл.

Масштабирование растровых картинок

Допустим, мы работаем с черно-белым изображением, расположенным в стандартной видеопамяти с восьмеричного адреса 42000 и имеющим длину 36000 байт. Возможны два различных варианта масштабирования: увеличение и уменьшение. В первом случае некоторая часть исходного изображения, скажем, расположенная в его левом верхнем углу, будет растягиваться до размеров всего экрана. При этом необходимо дублировать некоторые столбцы и строки пикселов. Во втором случае полное изображение на экране будет сжиматься до размеров заданного прямоугольника в левом верхнем углу экрана. Понятно, что некоторые строки и столбцы при этом придется пропускать, а еще лучше — логически складывать с соседними. Таким образом, необходимы две различные программы масштабирования: для увеличения и для уменьшения.

Вся хитрость заключается в том, чтобы определить, какие именно строки и столбцы нужно дублировать при увеличении или пропускать при уменьшении. Ясно, что они должны быть равномерно распределены по всему изображению, чтобы не нарушались его пропорции. Если прослеживать координаты точек экрана вдоль одной из координатных осей от 0 до максимальной величины HIGH с шагом 1, то соответствующая пропорциональная координата масштабируемого изображения должна изменяться от 0 до заданной величины LOW, увеличиваясь с шагом LOW/HIGH. Суммировать эти изменения пропорциональной координаты мы будем в счетчике SUM. В начале процесса он содержит 0, а когда по мере накопления изменений его величина превосходит 1, это означает, что целая часть пропорциональной координаты увеличивается на 1, после чего SUM необходимо уменьшить на 1, чтобы продолжать накопление изменений относительно уже достигнутой целой величины.

Так как смещение LOW/HIGH — дробная величина, удобно накапливать в счетчике SUM не саму ее, а в HIGH раз большую целую величину LOW, а сравнение счетчика производить не с 1, а с HIGH. Суть алгоритма от этого не изменится, зато не нужно будет работать с дробными числами, что гораздо удобнее. Итак, счетчик SUM будет служить индикатором идеальной пропорциональности масштабируемой координаты. Накапливая в нем величину LOW и сравнивая его значение с HIGH, можно сделать вывод о необходимости увеличения пропорциональной координаты на 1 в случае превышения содержимого SUM величины HIGH (после чего из SUM необходимо вычесть HIGH), в противном случае пропорциональная координата не изменяется.

Этими двумя координатами — прямой, которая всегда изменяется на 1, и пропорциональной ей, изменяемой на 1 или оставляемой без изменений, — мы и воспользуемся для построения масштабированного изображения. Чтобы получить увеличенную картинку, будем проверять отдельные пикселы вдоль пропорциональной координаты и в случае, когда пиксел установлен, отображать соответствующую точку вдоль прямой координаты. Для получения уменьшенной картинки, напротив, будем проверять пикселы вдоль прямой координаты, а отображать вдоль пропорциональной. Прямая и пропорциональная координаты являются выражением разных масштабов вдоль одной из координатных осей. Так как изображение имеет два измерения, то рассмотренное построение необходимо выполнять дважды, для каждой из координатных осей.

Эту задачу решают две приведенные ниже подпрограммы на ассемблере: SCALE+ (для увеличения картинки) и SCALE- (для уменьшения). Эти подпрограммы работают исключительно в экранной области памяти, не используя никаких дополнительных буферов, что особенно важно при малых ресурсах памяти БК-0010. После трансляции их можно использовать для масштабирования изображений, путем вызова командой JSR PC, <адрес>. При этом им необходимо передать сле-

дующие параметры изображения: в регистре R1 — координату X, задающую ширину картинки, в регистре R2 — координату Y, задающую ее высоту. Фактически при условии, что левый верхний угол изображения на экране совпадает с началом экранных координат, значения R1 и R2 представляют собой координаты точки, принадлежащей его правому нижнему углу.

Параметр X может принимать восьмеричные значения от 0 до 777, Y — от 0 до 357. При этом подпрограмма SCALE+ увеличит картинку (X,Y) до размеров полного экрана, а SCALE- уменьшит полный экран до размеров (X,Y). Если заданные параметры не находятся в допустимых пределах, изображение не изменяется. Варьируя X и Y, можно подобрать необходимое соотношение размеров масштабируемого изображения. Таким образом, из исходной картинки можно получить 512×360 = 184320 различных вариантов масштабированного изображения при увеличении и еще столько же при уменьшении. (Некоторые из возможных вариантов показаны на рисунке.)

Как можно судить по диапазону изменения координаты X, исходное изображение должно быть в формате 64 символа в строке, т. е. черно-белым. Однако рассмотренные подпрограммы можно успешно применять и для обработки цветных изображений (32 символа в строке), например для преобразования изображения из одного формата в другой. Для перевода цветной картинки в черно-белую необходимо передать параметры X=377 и Y=357 подпрограмме SCALE-. Те же параметры, переданные SCALE+, задают обратный перевод. Полученная таким образом цветная картинка будет иметь только два цвета — черный и красный. При других значениях параметров масштабирования на цветном изображении могут появиться искажения цветов, связанные с тем, что цвета отдельных пикселов в режиме 32 символа в строке кодируются парами битов, в то время как программы масштабирования работают с пикселами, представленными одним битом. Таким образом, применение рассмотренных программ для цветных изображений связано с определенными ограничениями, которые нужно учитывать.

```
;     Подпрограмма SCALE+ (увеличение изображений)
;     Автор  Румянцев С. К.     п. Локня        1994
;
; Входные параметры:    R1=X,   R2=Y - координаты,
; задающие  границу  увеличиваемого  изображения
;
SCALE+:   CMP  #1000, R1        ; Проверка допустимости
          BLOS 12               ; входных  координат
          CMP  #360, R2
          BLOS 12
          MOV  R2, -(SP)        ; Высота - в стек
          INC  (SP)
          MOV  R1, -(SP)        ; Ширина - в стек
          INC  (SP)
          SWAB R2               ; Формирование
          ROR  R2               ; начального адреса
          ASR  R2               ; исходной картинки
          ADD  #42076, R2
          MOV  R1, -(SP)        ; Формирование
          ASR  (SP)             ; смещения  адреса
          ASR  (SP)             ; к  новой  строке
          ASR  (SP)             ; исходного изображения
          BIC  #177701, (SP)
          SUB  #76, (SP)
          MOV  #1, -(SP)        ; Формирование
          BIC  #177760, R1      ; начальной маски
          BEQ  2                ; для исходного изображения
1:        ASL  (SP)
          SOB  R1, 1
2:        CLR  -(SP)            ; Счетчик ошибки по Y
          CLR  -(SP)            ; Счетчик ошибки по X
          MOV  #100000, R4      ; Нач. адр.  увелич. изображения
3:        MOV  #40, R1          ; Слов в строке
          ADD  6(SP), R2        ; К новой строке исходного изображения
          MOV  4(SP), R5        ; Маска-тест исходного изображения
4:        CLR  R3               ; Накопитель битов
          MOV  #20, R0          ; Количество битов в слове
```

```
5:      ASL   R3              ; Цикл накопления битов
        BIT   R5, (R2)        ; Есть бит?
        BEQ   6               ; Нет
        INC   R3              ; Да - в накопитель
6:      ADD   10(SP), (SP)    ; Увеличить ошибку по X
        CMP   #1000, (SP)     ; Ошибка в норме?
        BHI   7               ; Да
        SUB   #1000, (SP)     ; Нет - коррекция ошибки
        ROR   R5              ; Модификация маски
        BCC   7               ; Если слово исходного изображения
        ROR   R5              ; кончилось, то
        TST   -(R2)           ; на следующее слово
7:      SOB   R0, 5           ; Счет битов  увеличенного изображения
        MOV   R3, -(R4)       ; Слово для увеличенного изображения
        SOB   R1, 4           ; И так для всей строки
8:      CMP   #42000, R4      ; Есть еще строки увеличенного изображения?
        BHIS  11              ; Нет - на выход
        ADD   12(SP), 2(SP)   ; Увеличить ошибку по Y
        CMP   #360, 2(SP)     ; Ошибка в норме?
        BHI   9               ; Да
        SUB   #360, 2(SP)     ; Нет - коррекция ошибки
        BR    3               ; и на продолжение процесса
9:      MOV   #40, R1         ; Продублировать
10:     MOV   76(R4), -(R4)   ; строку увеличенного
        SOB   R1, 10          ; изображения
        BR    8               ; и на продолжение
11:     ADD   #14, SP         ; Коррекция стека
12:     RTS   PC              ; Выход из подпрограммы

;    Подпрограмма  SCALE-  (уменьшение изображений)
;    Автор  Румянцев С. К.      п. Люкня        1994
;
; Входные параметры:   R1=X,  R2=Y - координаты,
; задающие  границу  уменьшенного  изображения
;
SCALE-: CMP   #1000, R1       ; Проверка допустимости
        BLOS  8               ; входных  координат
        CMP   #360, R2
        BLOS  8
        MOV   R2, -(SP)       ; Высота - в стек
        INC   (SP)
        MOV   R1, -(SP)       ; Ширина - в стек
        INC   (SP)
        MOV   #42000, R2      ; Начальный адрес исходного изображения
        MOV   R2, R4          ; Начальный адрес уменьшенного изображения
        MOV   R4, -(SP)       ; Адрес текущей строки
        CLR   -(SP)           ; Счетчик ошибки по Y
        CLR   -(SP)           ; Счетчик ошибки по X
1:      MOV   #40, R1         ; Слов в строке
        MOV   #1, R5          ; Маска-формирователь
        CLR   R3              ; Накопитель  битов
2:      MOV   #20, R0         ; Количество битов в слове
3:      ROR   (R2)            ; Есть бит?
        BCC   4               ; Нет
        BIS   R5, R3          ; Да - в накопитель
4:      ADD   6(SP), (SP)     ; Увеличить ошибку по X
        CMP   #1000, (SP)     ; Ошибка в норме?
        BHI   5               ; Да
        SUB   #1000, (SP)     ; Нет - коррекция ошибки
        ASL   R5              ; Модификация маски
        BCC   5               ; Если накопитель весь,
        ROL   R5              ; то  установить  его
        BIS   R3, (R4)+       ; в уменьшенное изображ.
```

```
        CLR   R3              ; Сбросить  накопление битов
5:      SOB   R0, 3           ; и на продолжение
        TST   (R2)+           ; Следующее слово исходного изображения
        SOB   R1, 2           ; И так все слова в строке
        TST   R3              ; Накопитель  битов  пуст?
        BEQ   6               ; Да
        BIS   R3, (R4)        ; Остаток - в последнее слово
6:      ADD   10(SP), 2(SP)   ; Увеличить ошибку по Y
        CMP   #360, 2(SP)     ; Ошибка в норме?
        BHI   7               ; Да
        SUB   #360, 2(SP)     ; Нет - коррекция ошибки и переход
        ADD   #100, 4(SP)     ; на следующую строку уменьшенного изображения
7:      MOV   4(SP), R4       ; Новая строка  уменьшенного изображения
        TST   R2              ; Еще есть строки исходного изображения?
        BPL   1               ; Да - на продолжение
        ADD   #12, SP         ; Нет - коррекция стека
8:      RTS   PC              ; Выход из подпрограммы
```

При написании игровых программ одной из наиболее часто выполняемых операций является вывод спрайтов — фаз движения персонажей игры, деталей «окружающей обстановки» и т. п. В статье рассказано, как наиболее простыми средствами обеспечить наложение спрайта «поверх» сложного, содержащего большое количество многоцветных деталей, фона без его «смешения» с изображением на спрайте.

Д. Ю. Усенков,
Москва

Вывод спрайтов на рисованный фон

Необходимость работы со спрайтами на БК возникает достаточно часто, особенно при создании игровых программ. Но при этом нужно учитывать, что никакого стандартного механизма работы со спрайтами (как, например, в MSX-БЕЙСИКе) в БК не имеется и соответствующие подпрограммы нужно разрабатывать самостоятельно.

Как правило, спрайты используются в игровых программах для создания эффекта мультипликации (последовательный вывод различных фаз движения игрового персонажа с одновременным смещением спрайта, если это нужно, по координатам X и Y). И чаще всего фоном является не пустой экран, а какое-либо изображение «места действия».

Как правило, для вывода спрайтов используются два варианта операций: прямая запись в видеоОЗУ (MOV) и комбинирование спрайта с участком фона по логике «исключающее ИЛИ» (XOR). Оба они имеют недостатки.

При выводе по методу прямой записи вместе с самим спрайтом на экран переносятся и участки его фона. На пустом (например, черном) экране это незаметно, а вот на рисованном фоне при движении спрайта по экрану отчетливо видно, что его окружает зона прямоугольной формы, затирающая рисунок фона.

При выводе же по команде «исключающее ИЛИ» фон не нарушается, но цвета самого спрайта искажаются в зависимости от цветов участков фона.

По этой причине приходится использовать достаточно сложные алгоритмы наложения спрайта на копию участка фона (причем обработка ведется не на экране, а в отведенных в качестве буфера областях ОЗУ) и последующего вывода спрайта уже вместе с фоном. Один из примеров опубликован в журнале «Вычислительная техника и ее применение», №1 за 1991 г., с. 33.

То же самое можно сделать и проще — непосредственно на экране. Для этого следует использовать операции BIC и BIS. (И разумеется, нужно не забывать сохранить в отдельный буфер «старый» участок фона, чтобы, когда потребуется убрать спрайт, вывести его на прежнее место.)

При использовании команды BIS единичные биты в байте спрайта, используемом в качестве маски, устанавливают в единицу соответствующие биты в байте экрана. В результате в черно-белом режиме белые участки спрайта будут наложены поверх фона, а черные (в том числе и фон спрайта) окажутся «прозрачными» и сквозь них будет виден фон (старое изображение на экране). При использовании команды BIC участки экрана, соответствующие белым фрагментам спрайта, гасятся (закрашиваются в черный цвет), а соответствующие черным — не изменяются. Это можно использовать, например, для создания «тени» или для рисования на самом спрайте «непрозрачных» черных участков. Для этого нужно заранее заготовить дополнительную «спрайт-маску», где места, которые должны быть черными на спрайте, будут белыми на черном фоне. Таким образом, комбинируя операции BIC и BIS, можно прямо на экране достаточно быстро и незаметно для пользователя вначале записать «поверх» фона сам спрайт (BIS), а затем «зачернить» нужные его участки (BIC).

В цветном режиме дело обстоит сложнее. Ведь здесь цветные точки кодируются парами битов. Поэтому, если мы, например, выводим зеленый спрайт поверх синего фона, он станет красным, а попытка применить в этом же случае операцию BIC и вовсе ничего не даст — ведь соответствующие биты в экранном ОЗУ уже погашены.

Однако заметим, что, если мы выводим спрайт, полностью окрашенный в красный цвет, все будет нормально. Поэтому при работе в цветном режиме мы можем использовать тот же самый метод с использованием BIC и BIS, если добавить еще одну «спрайт-маску».

Итак, для вывода в цветном режиме многоцветных спрайтов на рисованный фон (также состоящий из разноцветных деталей) нужно заготовить для каждого спрайта вспомогательную «спрайт-маску», изображающую силуэт спрайта: участки, соответствующие самому

спрайту, должны быть в ней закрашены красным цветом, а участки его фона (которые должны быть «прозрачными» при выводе на экран) — черным. Для этого с помощью любого редактора спрайтов, оснащенного режимом линзы, нужно перекрасить все точки собственно спрайта в красный цвет. После этого элементарный цикл спрайт-мультипликации будет состоять из четырех этапов:

- сохранить в буфере участок экранного ОЗУ, куда будет выведен спрайт;
- вывести в нужное место «спрайт-маску» силуэта спрайта (BIC);
- вывести поверх силуэта сам спрайт (BIS);
- для стирания спрайта с экрана вывести сохраненный в буфере участок фона.

Если немного дорисовать «спрайт-маску», закрасив красным, кроме силуэта спрайта, еще и часть его фона, то при описанном способе вывода спрайт будет «отбрасывать тень».

Конечно, предлагаемый способ не слишком оптимален — в ОЗУ приходится хранить, по сути, две копии каждого спрайта. Однако если спрайтов немного, то дополнительный расход памяти окажется невелик, тем более что другие способы (например, приведенный в «Вычислительной технике», №1 за 1991 г.) тоже требуют немало памяти под буфер. Но предлагаемый метод намного проще, и, скорее всего, придется по вкусу начинающим пользователям БК.

Таблица преобразование цветов при выводе спрайтов

Операция	Цвет спрайта	Цвет на экране	Результат
MOV	красный	любой	красный
	зеленый	любой	зеленый
	синий	любой	синий
	черный	любой	черный
XOR	красный	красный	черный
	красный	зеленый	синий
	красный	синий	зеленый
	красный	черный	красный
	зеленый	красный	синий
	зеленый	зеленый	черный
	зеленый	синий	красный
	зеленый	черный	зеленый
	синий	красный	зеленый
	синий	зеленый	красный
	синий	синий	черный
	синий	черный	синий
	черный	любой	не меняется
BIS	красный	любой	красный
	черный	любой	не меняется
	зеленый	красный	красный
	зеленый	зеленый	зеленый
	зеленый	синий	красный
	зеленый	черный	зеленый
	синий	красный	красный
	синий	зеленый	красный
	синий	синий	синий
	синий	черный	синий
BIC	красный	любой	черный
	черный	любой	не меняется
	зеленый	красный	синий
	зеленый	зеленый	черный
	зеленый	синий	синий
	зеленый	черный	черный
	синий	красный	зеленый
	синий	зеленый	зеленый
	синий	синий	черный
	синий	черный	черный

Те, кто пока еще не знаком с ассемблером, могут поэкспериментировать на вильнюсском БЕЙСИКе. В нем для сохранения спрайта в массиве используется оператор PEEK (в цикле по X и Y), а для вывода из массива на экран по методу «прямой записи» — POKE (также в цикле). А программная строка POKE <A>,(PEEK(<A>) XOR <B>), где <A> — текущий адрес ячейки видеоОЗУ, а <B> — очередные два байта спрайта (константа или содержимое элемента массива) позволяет реализовать вывод по методу «исключающее ИЛИ». К сожалению, аналоги ассемблерных команд BIC и BIS в БЕЙСИКе отсутствуют, но можно поэкспериментировать с близкими к ним функциями AND и OR, подставляемыми в показанную выше БЕЙСИК-строку вместо XOR. В остальных же случаях (в том числе для вывода спрайтов по одному байту) БЕЙСИКу приходится прибегать к использованию USR-подпрограмм.

В статье описывается реализация операций сохранения и вывода спрайтов на экран. В отличие от ранее предлагавшихся вариантов для хранения спрайта здесь используется массив, целиком передаваемый в кодовые подпрограммы как аргумент оператора USR.

Спрайты в вильнюсском БЕЙСИКе БК-0010.01

Среди операторов, имеющихся в мощных версиях БЕЙСИКа для IBM-совместимых компьютеров, но отсутствующих в вильнюсском БЕЙСИКе БК-0010.01, можно назвать и функции для работы со спрайтами GET и PUT. Первая служит для занесения в массив с указанным именем части экранного изображения (спрайта заданных размеров), а вторая — для вывода спрайта из массива на экран. (Подробнее об этих функциях можно прочитать, например, в книге: *Грайс Д.* Графические средства персонального компьютера. М.: Мир, 1989. С. 46.) Занесенную в массив копию можно затем переписывать (в том числе отдельными фрагментами) в другой массив, изменять (скажем, накладывать один спрайт на другой) и т. д.

Для БЕЙСИКа БК-0010.01 уже неоднократно предлагались различные реализации простейших спрайтовых операций, как правило выполненные по одному из двух способов. Первый предполагает использование для сохранения/восстановления спрайтов подпрограмм в машинных кодах, вызываемых с помощью DEF USR. Второй — непосредственное считывание содержимого экранного ОЗУ в цикле с помощью операторов БЕЙСИКа PEEK и POKE. Однако оба способа имеют свои недостатки. Так, практически все USR-подпрограммы производят сохранение спрайта в некоторую буферную область ОЗУ, а не в массив, что затрудняет доступ к сохраненному спрайту (для каких-либо его преобразований, как правило, требуется также подпрограмма в машинных кодах). Обращение же с помощью PEEK и POKE, напротив, позволяет обеспечить легкость доступа к сохраненному спрайту (его можно хранить и в массиве), но такая конструкция громоздка и замедляет работу программы, что особенно нежелательно для игр.

Можно ли объединить достоинства этих методов, одновременно освободившись от их недостатков? Предлагаемые здесь реализации операторов PUT и GET, выполненные в виде кодовых подпрограмм и вызываемые с помощью USR в одной программной строке, позволяют сохранять спрайт в массиве, имя которого указано пользователем. Приведем ассемблерный листинг для подпрограммы GET.

```
                        ; восьм. код
GET:    MOV R4,R0       ; 010400 ; поиск в ОЗУ дескриптора
        SUB #14,R0      ; 162700 ;  и содержимого массива,
                        ; 000014 ;
        MOV @R0,R0      ; 011000 ;  элемент которого задан
        MOV @R0,R0      ; 011000 ;  в операторе USR
        MOV (R0)+,R1    ; 012001 ; -  R1 - число столбцов (X)
        MOV (R0)+,R2    ; 012002 ; -  R2 - число строк (Y)
        MOV 2(R5),R3    ; 016503 ; адрес в экранном ОЗУ
                        ; 000002 ;
 ; стандартная подпрограмма сохранения спрайта
2:      MOV R1,R4       ; 010104 ;
1:      MOV (R3)+,(R0)+ ; 012320 ;
        ; для PUT - MOV (R0)+,(R3)+ ; 012023 ;
        SOB R4,1        ; 077402 ;
        ADD #100,R3     ; 062703 ;
                        ; 000100 ;
        SUB R1,R3       ; 160103 ;
        SUB R1,R3       ; 160103 ;
        SOB R2,2        ; 077210 ;
        RTS R7          ; 000207 ;
```

Непосредственный доступ из кодовой подпрограммы к массиву (начало листинга) обеспечивается по методу, подробно описанному в №2 за 1995 г., с. 21. Вторая часть отличается от стандартной подпрограммы для работы со спрайтами (см. Вычислительная техника и ее применение. 1991. №5. С. 33) только тем, что ширина спрайта X здесь задается не в байтах, а в двухбайтных машинных словах (так как каждый элемент целочисленного массива занимает два байта).

Подпрограмма для реализации оператора PUT выглядит аналогично, и ее листинг отличается от приведенного выше только одной строкой с меткой 1 (как это указано в комментарии к листингу).

Работа с подпрограммами PUT и GET в БЕЙСИКе очень проста. Оттранслированные с ассемблера в коды и записанные на магнитофон (возможно, единым блоком; имена файлов должны состоять из шести символов и иметь расширение .BIN), они перемещаемы и загружаются в БЕЙСИК командой BLOAD с любого адреса (следует предварительно освободить часть ОЗУ с помощью CLEAR), после чего нужно определить с помощью оператора DEF USR две USR-подпрограммы, адреса запуска которых соответствуют началу подпрограмм (метки PUT и GET).

Вызов осуществляется строкой вида:

A=<адр>=USR<n>(<элемент массива>) ,

где A — некая буферная переменная (содержимое которой не требуется в дальнейших вычислениях), **<адр>** — адрес верхнего левого угла спрайта в ОЗУ экрана, а номер USR-функции **<n>** определяет тип спрайтовой операции: сохранение или вывод (например, USR1 — для вызова подпрограммы GET, а USR2 — для PUT). В скобках указывается любой элемент массива (обязательно целого типа!), предназначенного под спрайт, причем важно только его имя, а значения индексов могут быть любыми.

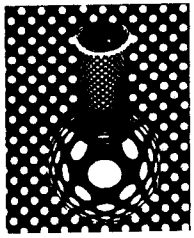
Величина спрайта полностью зависит от размеров отведенного для него массива: ширина спрайта X (в двухбайтных машинных словах) равна количеству столбцов, а высота Y (в графических строках) — количеству строк массива. (Не забывайте, что количества столбцов и строк массива на 1 больше чисел, указанных в операторе DIM, так как отсчет индексов ведется с нуля.) Спрайт считывается в массив построчно, т. е. верхний левый байт (первый) соответствует элементу массива с индексами (0,0), следующий за ним (второй) — (0,1), затем (0,2) и т. д.

В качестве иллюстрации приведем элементарный пример. БЕЙСИК-программа (ее листинг приведен ниже) считывает с экрана в качестве спрайта фрагмент, адрес верхнего левого угла которого равен &O42000, и выводит его с адреса &O50000, а также — частично и в инвертированном виде — с адреса &O50040 (перед запуском нужно нормализовать рулонное смещение экрана двухкратным нажатием «АР2»+«СБР»). Предполагается, что функция USR1 «отвечает» за сохранение спрайта, а USR2 — за его восстановление.

```
10 DIM A%(19,9), B%(9,4)
20 GOSUB... ' нарисовать картинку в левом верхнем углу экрана
30 A=&O42000=USR1(A%(0,0)) ' индексы для A%() могут быть любыми
40 FOR I=0 TO 9
50 FOR J=0 TO 4
60 B%(I,J)= NOT A%(I,J) ' скопировали часть спрайта в массив
70 NEXT J,I              ' B% с одновременной инверсией
80 A=&O50000=USR2(A%(0,0)) ' вывод копии спрайта
90 A=&O50040=USR2(B%(0,0)) ' вывод его инвертированного фрагмента
100 END
```

Размер спрайта в данном случае равен 20 байт (10 двухбайтных слов) · 20 строк, а его инвертированного фрагмента — 10 байт · 10 строк.

Из недостатков предложенной здесь реализации спрайтовых функций можно назвать следующие. Во-первых, размер спрайта жестко связан с размерами массива (а значит, нельзя использовать один и тот же «буферный» массив для сохранения спрайтов различных размеров). Во-вторых, в целях упрощения подпрограмм в них не предусмотрен контроль выхода за границы экрана (т. е. слишком широкие и начинающиеся чересчур близко к правой границе экрана спрайты могут частично «вылезать» слева, а если спрайт слишком высок и выводится в самом низу экрана, возможен аварийный останов программы из-за попытки записи информации в ПЗУ). Желающие могут самостоятельно устранить эти недостатки, разработав свои, более сложные подпрограммы.



Каждому программисту наверняка хочется, чтобы созданная им компьютерная игра отличалась эффектной графикой, стремительностью происходящих на экране событий, реалистичностью объектов игры. Разумеется, при наличии большого объема памяти для хранения промежуточных фаз движений, а также аппаратных средств, ускоряющих вывод на экран графической информации, проблем нет. Но как быть, если компьютер не располагает такими ресурсами? На помощь в этом случае могут прийти некоторые психологические особенности человеческого зрения. Если невозможно воспроизвести на экране движение, можно создать его иллюзию.

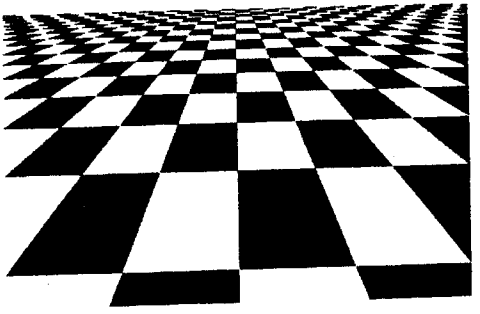
С. К. Румянцев,
пос. Локня Псковской обл.

Волшебная инверсия

Из разнообразных зрительных иллюзий нам может пригодиться так называемый муаровый эффект. Если наложить под небольшим углом две расчески одна на другую и плавно передвигать их, между зубьями будут видны своеобразные волны. Причем эти волны могут смещаться в ином направлении и с большей скоростью, чем создающие их предметы, — это и есть муаровый эффект. Порой он бывает нежелателен: когда на телевизионном экране демонстрируется множество параллельных линий (например, изображение жалюзей), появляется неприятное мерцание.

Попробуем, однако, извлечь пользу из муарового эффекта. В этом нам поможет микропроцессорная команда COM — «инверсия». Чем же она замечательна? С одной стороны, она полностью преобразует аргумент: каждый единичный бит становится нулевым, а нулевой — единичным. С другой, она не уничтожает прежнюю информацию совсем: если применить инверсию повторно, прежнее значение аргумента восстановится. Это в некотором смысле «интеллектуальная» команда, она «сама решает», что нужно делать. Применив в нужном месте инверсию, можно избежать многих вычислений.

Давайте разберем небольшую программу на ассемблере, демонстрирующую возможности инверсии в обработке графической информации. (Листинг содержит также машинные коды, которые можно набрать вручную с удобного вам адреса, так как программа перемещаема, и сразу запустить на исполнение на любом компьютере серии БК.)



Вообразите, что ваш вездеход плавно перемещается по поверхности фантастической планеты. Окружающая местность, словно бесконечная шахматная доска, испещрена черными и белыми квадратиками, которые, расступаясь, проплывают за ветровым стеклом... Чтобы изобразить эту картину на экране, посмотрим, как должны располагаться линии, образующие каркас шахматного поля.

Предположим, что эти линии представляют собой сетку взаимно перпендикулярных прямых, отстоящих на равных расстояниях друг от друга. Пусть мы «движемся» вдоль одной из таких прямых, тогда она расположится вертикально в центре экрана, а прямые, параллельные ей на плоскости, подобно вееру, будут распределены симметрично справа и слева. По закону перспективы все они пересекутся в «точке схода» в верхней части экрана, через которую проходит «линия горизонта». Если прямые расположены на равных расстояниях друг от друга, то на каждой из строк экрана они отсекают равные отрезки, длина которых пропорциональна расстоянию от линии горизонта. На этом и основан принцип построения полос, вдоль которых происходит «движение».

Вначале программа последовательно, сверху вниз, просматривает строки экрана и изображает на них чередующиеся черные и белые отрезки, по длине равные половине расстояния на экране данной строки от линии горизонта. Попутно вычисляются и помещаются в буфер обратные величины этого расстояния (эти данные пригодятся в дальнейшем). Когда текущая координата Y выходит за нижнюю границу экрана, построение заканчивается. Из черных и белых отрезков складывается изображение полос, веером сходящихся в точке схода.

Полученная картина является основой для создания клетчатого поля. Для этого надо добавить изображение горизонтальных полос (перпендикулярных построенным). Интерес-

но, что все дальнейшие преобразования на экране будут проводиться только при помощи команды инверсии. В самом деле, чтобы получить клетчатое поле, нам осталось только инвертировать некоторые строки экрана. Но как узнать, какие именно? Здесь и пригодятся данные, ранее сохраненные в буфере (они были получены делением масштабного коэффициента на экранное расстояние строки от линии горизонта). Нетрудно подсчитать, что эта величина представляет собой истинное расстояние до линии на плоскости, соответствующей данной строке на экране. Масштаб подобран таким образом, что разряд единиц в этих числах приходится на седьмой (знаковый) разряд представляющих их байтов. Остальные разряды в байте выражают дробную часть.

Если предположить, что расстояние между квадратиками клетчатого поля равно единице, то можно по знаковому разряду этих чисел судить, на какую полосу (черную или белую) приходится данная строка экрана, и соответственно окрашивать ее. Отметим, что нас интересует не полное расстояние до поля, а только его дробная часть и разряд единиц, так как именно эти величины выражают чередование цвета на плоскости. Поэтому в буфере сохранялись только байты, хотя для полных расстояний могли потребоваться двухбайтные машинные слова.

Используя таблицу байтов расстояний, можно показать на экране не только вид клетчатого поля в перспективе, но и процесс движения по нему. Прибавим к каждому байту расстояний некоторое постоянное число, оно задает фазу смещения. Тогда знаковый разряд результата будет определять цвет точки на плоскости, смещенной в направлении движения. Прибавляя же различные значения фазы, изменяемые циклически, мы можем заставить клетчатое поле двигаться с различной скоростью вперед или назад либо останавливаться. Скорость изменения фазы в программе регулируется клавишами «СТРЕЛКА ВВЕРХ» и «СТРЕЛКА ВНИЗ». Всего движение шахматного поля может состоять из 128 фаз, что придает ему более чем необходимую плавность.

Как же происходит построение отдельных кадров этого «шахматного» мультфильма? Для каждой строки экрана программа вычисляет сумму байта расстояний и фазы, после чего по знаку полученного байта определяет цвет соответствующей полосы на плоскости. Если строка на экране уже окрашена в этот цвет (после предыдущего кадра), она пропускается, если же цвета не совпадают, строка инвертируется. Это сравнение цветов осуществляется с помощью операции XOR («исключающее ИЛИ») между байтом суммы и байтом экранной строки, примыкающим к вертикальной оси экрана (это позволяет исключить влияние исходного «веерного» фона). Кстати, операция XOR представляет собой не что иное, как «побитную инверсию», она прямо на экране определяет дальнейшую стратегию: инвертировать или пропускать данную строку. Фактически инверсии подвергается лишь небольшая часть строк экрана, отсюда значительная скорость процесса.

Дойдя до нижней границы экрана, программа увеличивает фазу, а также (если нажата соответствующая клавиша) изменяет константу увеличения фазы. Далее процесс модификации строк экрана циклически повторяется.

Присмотримся внимательнее к тому, что происходит на экране. Кажется, что квадратики шахматного поля движутся не только по вертикали, но также вправо и влево от осевой линии, увеличивая свои размеры по мере приближения. Трудно поверить, что на самом деле процесс инверсии строк идет только сверху вниз. Вот где проявляются «интеллектуальные» свойства инверсии! Она «конструирует» следующий кадр из предыдущего, из него же добывая «строительный материал». Конечно, это всего лишь иллюзия, вызванная муаровым эффектом между наклонными линиями фона и горизонтальными полосами инверсии. Но цель достигнута: мы отчетливо видим картину движения по шахматной плоскости.

Тем, кого заинтересовали эти методы программирования и кто хотел бы использовать данную программу для оформления игр, будут полезны сведения о назначении некоторых ее ячеек (предполагаем, что программа загружена с адреса 1000, здесь и далее все числа восьмеричные):

- **1002** — адрес, начиная с которого производится вывод изображения. Его значение должно находиться в диапазоне 40000—77700 и быть кратным 100, соответственно меняются размеры картинки;
- **1016** — высота линии горизонта над верхней границей кадра. Ее лучше установить чуть больше нуля, например равной 20, чтобы полосы не располагались слишком часто. При значении 177400 изображение получается перевернутым, а при 177600 линия горизонта находится в центре кадра (эффект также может быть интересен);
- **1022** — адрес буфера байтов расстояний. В данном варианте программы равен 300, так что при максимальной величине буфера 400 байт порча содержимого стека нам не грозит, хотя область данных магнитофона оказывается затерта. Значение адреса можно изменить, перенеся таким образом буфер в другую зону ОЗУ;
- **1136** — маска цвета, при значении 125252 или 52525 изображение становится четырехцветным (при использовании цветного монитора);
- **1162** — масштабный коэффициент. При больших значениях горизонтальные полосы располагаются чаще, при меньших — реже;

- 1222 — кратность сжатия по вертикали, в программе равна 1. Это значение можно немного увеличить (например, до 2 или 3), при этом точка восприятия как бы понижается.

Программа не зависит от операционной системы и нормально работает на любом компьютере серии БК. Перед ее запуском желательно нормализовать экран. Для этого на БК-0010(.01) следует нажать «АР2»+«СБР» два раза, а на БК-0011М просто «СБР». Кроме того, на БК-0011М надо настроить экранную область на адрес 40000 командой 5;0С.

```
;Демонстрационная программа имитации
;движения по шахматной плоскости
;Автор: Румянцев С.К., пос.Локня, 1994

001000  012700  MOV  #40000, R0      ;Адрес начала картинки
001002  040000
001004  062700  ADD  #36, R0         ;Влево от осевой линии
001006  000036
001010  010001  MOV  R0, R1          ;Вправо от осевой линии
001012  005721  TST  (R1)+
001014  012704  MOV  #20, R4         ;Высота линии горизонта
001016  000020
001020  012746  MOV  #300, -(SP)     ;Адрес буфера сохранить
001022  000300
001024  010446  MOV  R4, -(SP)       ;Текущее расстояние
001026  100001  BPL  1032            ;от линии горизонта
001030  005404  NEG  R4              ;сделать положительным
001032  001001  BNE  1036            ;и не равным нулю
001034  005204  INC  R4
001036  010446  MOV  R4, -(SP)       ;Сохранить R4 в стеке
001040  010405  MOV  R4, R5          ;Вычисление
001042  006205  ASR  R5              ;длин отрезков
001044  001002  BNE  1052            ;на текущей строке,
001046  005205  INC  R5              ;равных половине
001050  005204  INC  R4              ;расстояния до линии
001052  160504  SUB  R5, R4          ;горизонта
001054  012702  MOV  #140000, R2     ;Маска точек для
001056  140000                       ;левой половины экрана
001060  012703  MOV  #3, R3          ;Маска точек для
001062  000003                       ;правой половины экрана
001064  010446  MOV  R4, -(SP)       ;Сохранить длины
001066  010546  MOV  R5, -(SP)       ;отрезков
001070  005304  DEC  R4              ;Первая закончилась?
001072  100005  BPL  1106            ;Нет - поставить точки
001074  005305  DEC  R5              ;Вторая закончилась?
001076  100006  BPL  1114            ;Нет - поставить точки
001100  012605  MOV  (SP)+, R5       ;Восстановить длины
001102  012604  MOV  (SP)+, R4       ;отрезков
001104  000767  BR   1064            ;и продолжать процесс
001106  050210  BIS  R2, (R0)        ;Установить левую точку
001110  040311  BIC  R3, (R1)        ;и сбросить правую
001112  000402  BR   1120
001114  040210  BIC  R2, (R0)        ;Сбросить левую точку
001116  050311  BIS  R3, (R1)        ;и установить правую
001120  006303  ASL  R3              ;Вычисление масок
001122  006002  ROR  R2              ;следующих точек
001124  006103  ROL  R3
001126  006002  ROR  R2              ;Слово закончилось?
001130  103357  BCC  1070            ;Нет - на продолжение
001132  006103  ROL  R3              ;Закончить маску
001134  042710  BIC  #0, (R0)        ;Установить цвет слева
001136  000000
001140  005740  TST  -(R0)           ;Уменьшить левый адрес
001142  046721  BIC  1136, (R1)+     ;Установить цвет справа,
001144  177770                       ;увеличив правый адрес
001146  032701  BIT  #40, R1         ;Строка закончилась?
001150  000040
001152  001346  BNE  1070            ;Нет - продолжение
001154  022626  CMP  (SP)+, (SP)+    ;Продвинуть стек
001156  012604  MOV  (SP)+, R4       ;Восстановить расстояние
```

```
001160  012703  MOV  #40000, R3       ;Масштабный коэффициент
001162  040000                        ;делить на расстояние,
001164  005005  CLR  R5                ;для этого обнулить
001166  005002  CLR  R2                ;счетчик и частное
001170  006304  ASL  R4                ;Сдвигать делитель,
001172  005205  INC  R5                ;увеличивая счетчик,
001174  103375  BCC  1170              ;пока не будет переноса
001176  006004  ROR  R4                ;Вернуть перенос
001200  005202  INC  R2                ;Увеличивать частное,
001202  160403  SUB  R4, R3            ;пока делитель
001204  103375  BCC  1200              ;умещается в делимом
001206  060403  ADD  R4, R3            ;Иначе отступить
001210  005302  DEC  R2                ;на шаг назад,
001212  006302  ASL  R2                ;сдвинуть частное
001214  077510  SOB  R5, 1176          ;и продолжать деление
001216  012604  MOV  (SP)+, R4         ;Восстановить расстояние
001220  062704  ADD  #1, R4            ;от линии горизонта
001222  000001                        ;и увеличить его
001224  012603  MOV  (SP)+, R3         ;Восстановить адрес буфера
001226  110223  MOVB R2, (R3)+         ;Байт частного - в буфер
001230  062700  ADD  #140, R0          ;Левая половина
001232  000140                        ;следующей строки
001234  062701  ADD  #40, R1           ;Правая половина
001236  000040                        ;следующей строки
001240  100402  BMI  1246              ;Экран закончился?
001242  010346  MOV  R3, -(SP)         ;Нет - сохранить буфер
001244  000667  BR   1024              ;и на продолжение
001246  005005  CLR  R5                ;Начальная фаза - ноль
001250  016700  MOV  1002, R0          ;Адрес начала картинки
001252  177526
001254  016703  MOV  1022, R3          ;Адрес буфера байтов
001256  177542                        ;истинных расстояний
001260  112302  MOVB (R3)+, R2         ;Расстояние - из буфера
001262  060502  ADD  R5, R2            ;Прибавить фазу
001264  116004  MOVB 37(R0), R4        ;Состояние текущей
001266  000037                        ;строки экрана
001270  074402  XOR  R4, R2            ;наложить с инверсией
001272  105702  TSTB R2                ;Строка требует
001274  100403  BMI  1304              ;инверсии?
001276  062700  ADD  #100, R0          ;Нет - пропустить
001300  000100                        ;строку
001302  000404  BR   1314
001304  012704  MOV  #40, R4           ;Да - инвертировать
001306  000040                        ;строку
001310  005120  COM  (R0)+
001312  077402  SOB  R4, 1310
001314  005700  TST  R0                ;Экран закончился?
001316  100360  BPL  1260              ;Нет - на продолжение
001320  062705  ADD  #10, R5           ;Увеличить фазу
001322  000010
001324  032737  BIT  #100, @#177716    ;Клавиша нажата?
001326  000100
001330  177716
001332  001346  BNE  1250              ;Нет - следующий кадр
001334  022737  CMP  #33, @#177662     ;Нажата клавиша
001336  000033                        ;"стрелка вниз"?
001340  177662
001342  001003  BNE  1352              ;Нет - проверяем дальше
001344  105267  INCB 1322              ;Да - увеличить
001346  177752                        ;изменение фазы и перейти
001350  000737  BR   1250              ;на следующий кадр
001352  022737  CMP  #32, @#177662     ;Нажата клавиша
001354  000032                        ;"стрелка вверх"?
001356  177662
001360  001333  BNE  1250              ;Нет - следующий кадр
001362  105367  DECB 1322              ;Да - уменьшить
001364  177734                        ;изменение фазы и перейти
001366  000730  BR   1250              ;на следующий кадр
```

При оформлении программ и документаций к ним значительного внешнего эффекта можно добиться, заменяя стандартный шрифт БК на более оригинальный, используя необычные приемы вывода на экран (например, «бегущей строкой») и т. д. Ниже вниманию читателей предлагается ряд интересных программ на ассемблере.

А. И. Глускер,

Москва

Нестандартные шрифты

Предлагаю несколько интересных листингов, реализующих шрифты нестандартного вида на БК-0010(.01). Листинги написаны в стандарте ассемблер-транслятора TURBO6M и работают в ОС ANDOS версии 2.50.

Необычный цвет символов

Исходный режим экрана: 32 символа в строке, красный цвет переднего плана, черный фон.

```
        MOV @#30,EMT1
; Запоминание адреса EMT
        MOV #EMT,@#30
; Установка своего адреса
        RET
; Возврат в вызвавшую программу
EMT:    MOV R5,-(SP)
; Выборка команды EMT
        MOV 2(SP),R5
        MOV -(R5),R5
        CMP R5,#104016
; EMT 16?
        BEQ EMT16
        CMP R5,#104020
; EMT 20?
        BEQ EMT20
        CMP R5,#104036
; EMT 36?
        BEQ EMT36
        CALL @-4000(R5)
; Иначе стандартная обработка
        MOV #EMT,@#30
; Восстановить EMT
        BR AA
EMT36:  MOV (PC)+,@(PC)+
; Предыдущий EMT
EMT1:   .WORD 0,30
        EMT 36
        MOV #EMT,@#30
        BR AA
EMT16:  CMPB R0,#40
; Отсортировка непечатаемых кодов
        BLO 10
        CMPB R0,#200
        BLO 11
        CMPB R0,#240
        BLO 10
11:     MOV @#160,-(SP)
        CALL @#100016
; Печать символа
        MOV (SP)+,R5
; В R5 - адрес его верхней строки
        MOV R4,-(SP)
        MOV R3,-(SP)
        MOV #12,R4

1:      MOV (R5),R3
; Копирование в соседние точки экрана
        BIS R3,-100(R5)
        CLC
        ROR R3
        ASR R3
        BIS R3,(R5)
        BIS R3,-100(R5)

        BIT #100,R5
        BNE 3

        BIC #114631,-100(R5)
; Установка нужного цвета
        BR 4
3:      BIC #63146,-100(R5)

4:      ADD #100,R5
        BPL 2
        BIT #100,R5
        BNE 6
        BIC #114631,-100(R5)
        BR 7
6:      BIC #63146,-100(R5)
7:      SUB @#206,R5
2:      SOB R4,1
        MOV (SP)+,R3
        MOV (SP)+,R4
        BR AA
10:     CALL @#100016
; Служебный код
AA:     MOV (SP)+,R5
        RTI
EMT20:  MOV R0,-(SP)
; Аналог прошивки ПЗУ для EMT20, но
; вместо CALL @#... используется EMT 16
        MOV R3,-(SP)
        MOV  R2,R3
```

```
        SWAB R3
        BIC #177400,R3
1:      MOVB (R1)+,R0
        EMT 16
        CMPB R2,R0
        BEQ 2
        SOB R3,1
2:      MOV (SP)+,R3
        MOV (SP)+,R0
        RR A4
        .END
```

Бегущая строка

В отличие от имеющихся вариантов здесь бегущая строка не «прыгает», всегда одного цвета, занимает не очень много памяти и не использует EMT16.

```
NAC:    MOV #100000,@#60
        MOV @#60,@#274
        MOV #TXT,R1
10:     MOVB (R1)+,R0
        BNE 1
        BR NAC
1:      MOV R1,-(SP)

        BIC #177400,R0
; Выделение младшего байта R0
        CMP R0,#200
        BLO 19

        SUB #40,R0
19:     SUB #40,R0
        ASL R0
        MOV R0,R2
        ASL R0
        ASL R0
        ADD R2,R0
        ADD #112276,R0
        MOV #BUF,R3
; Символ во внутренний буфер
        MOV #5,R2
200:    MOV (R0)+,(R3)
        COM (R3)+
; Инверсия
        SOB R2,200
        MOV #10,R4
4:      MOV #BUF+11,R0
        MOV #70000,R3
; Адрес вывода
        MOV #12,R5
3:      MOV #40,R2
        CLR R1
; Сдвижка строки
        RORB (R0)
        ROR R1
        ASR R1
        SWAB R1
        ASR R1
        ASR R1
        ASR R1
        ASR R1
        ASR R1
2:      ROR R1
        ROR -(R3)
        ROR R1
        ROR (R3)
        ROR R1
        SWAB R1
        ASR R1
        ASR R1
        ASR R1
        ASR R1
        ASR R1
        ASR R1
        SOB R2,2
        DEC R0
        SOB R5,3
        SOB R4,4
        MOV (SP)+,R1
        BR 10
BUF:    .BLKB 12
TXT:    .ASCIZ /Привет, друзья./
        .END
```

Наклонный векторный шрифт

```
        EMT 14
; Загрузчик нестандартного шрифта
        MOV #233,R0
        EMT 16
        DEC R0
        EMT 16
        MOV #40000,R0
;  Сброс экрана
12:     CLR (R0)+
        TST R0
        BPL 12
        MOV #EMT,@#30
; EMT-перехват
        MOV #177777,@#212
; Новые цвета
        MOV #52525,@#214
        MOV #TXT,R1
2:      MOVB (R1)+,R0
;  Вывод текста
        BEQ 1
        EMT 16
        BR 2
1:      EMT 0
        JMP @#100442
```

```
;  Теперь можно
поэкспериментировать,
;  набирая текст в мониторе
EMT:    MOV R5,-(SP)
        MOV 2(SP),R5
        MOV -(R5),R5
        CMP R5,#104016
;  EMT 16?
        BEQ EMT16
        CMP R5,#104030
;  EMT 30?
        BEQ EMT30
        CALL @-4000(R5)
;  Остальное
        MOV #EMT,@#30
        MOV (SP)+,R5
        RTI
EMT30:  CALL @#120002
;  Обращение к диску в ANDOS
        MOV (SP)+,R5
        RTI
EMT16:  JSR R4,@#110346
; Регистры в стек
        BIC #177400,R0
; Выделение младшего байта R0
        CMPB R0,#40
; Отсортировка непечатаемых кодов
        BLO 1
        CMPB R0,#200
        BLO 2
        CMPB R0,#240
        BLO 1
        SUB #40,R0
2:      SUB #40,R0
        ASL R0
        MOV R0,R1
        ASL R0
        ASL R0
        ADD R1,R0
        ADD #112270,R0
        MOV R0,R3
        EMT 26
;  Откуда начинать вывод
;  векторного шрифта
        ASL R2
        ASL R2
        MOV R2,R0
        ASL R2
        ASL R2
        ADD R0,R2
        ASL R1
        ASL R1
        ASL R1
        MOV R2,R0
        ASR R0
        ADD R0,R1
        MOV #12,R5
5:      MOVB (R3)+,R4
        MOV R5,-(SP)
        MOV #10,R3
4:      CLR R0
        RORB R4
        ADC R0
        BEQ 13
        EMT 30
;  Вывод графической точки
        INC R2
        EMT 30
        INC R1
        EMT 30
        INC R2
        EMT 30
        SUB #2,R2
        DEC R1
13:     INC R1
        SOB R3,4
        MOV (SP)+,R3
        SUB #7,R1
;  Окончить вывод и сместиться
вправо
        ADD #2,R2
        SOB R5,5
        MOV #31,R0
        EMT 16
6:      JSR R4,@#110362
        MOV (SP)+,R5
        RTI
1:      CALL @#100016
;  Служебный код
        BR 6
TXT:    .ASCII /Привет/
        .ASCII <12>/всем/
        .ASCII <12>/БКшники./
        .ASCIZ <12>/ГАМ/
        .END
```

Напоследок приведу наилучший (по краткости листинга и скорости работы) вариант программы для очистки экрана (со служебной строкой).

```
        MOV #40000, R0  ; MOV #42000,R0 - без сл. строки
1:      CLR (R0)+
        TST R0          ; Обычно используется CMP R0,#100000
        BPL 1
```

Заменив константу в первом операторе, можно использовать этот фрагмент для очистки части экрана до его нижнего края.

Возможно, решаемые приведенными в этой статье программами на БЕЙСИКе задачи кому-то покажутся слишком примитивными. Но, думается, начинающим осваивать БЕЙСИК будет небезынтересно «покопаться» в них — исследовать алгоритмы, попытаться расширить возможности программ или сделать их листинги более компактными.

Д. Романов,
г. Саратов

Узоры для БК

Предлагаемые ниже программы пригодятся для создания оригинальных заставок в играх и для разработки орнаментов (например, при моделировании рисунка обоев). Автор не ставил задачу достижения максимально возможной скорости работы программ, так что пользователям остается значительный простор для оптимизации и доработок.

Круговой узор

```
10 INPUT "H";H%
20 CLS
30 FOR I%=1% TO 100% STEP H%
40 CIRCLE (I%,I%),I%,1%
50 CIRCLE (255%-I%,I%),I%,1%
60 CIRCLE (I%,239%-I%),I%,1%
70 CIRCLE (255%-I%,239%-I%),I%,1%
80 NEXT I%
90 END
```

Цветочный орнамент

```
10 INPUT A%,B%,C%
20 CLS
30 FOR X%=0% TO A%*5%
40 FOR Y%=0% TO A%*5%
50 IF (X%\32%) MOD 2% XOR
   (Y%\32%) MOD 2% THEN PSET(X%,Y%),1-
   SGN(ABS(C%-((X%*Y%) MOD A%
   MOD B%))) ELSE 70
60 GOTO 80
70 PSET (X%,Y%),SGN(ABS(C%-((X%*Y%)
   MOD A% MOD B%)))
80 NEXT Y%
90 NEXT X%
100 GOTO 10
```

Значения параметров, задаваемых в строке 10, определяют характер орнамента. Возможные варианты:

- A%=32%, B%=3%, C%=1%;
- A%=32%, B%=6%, C%=4%;
- A%=32%, B%=7%, C%=2%.

Быстрый цветочный орнамент

```
10 DIM A%(2%,32%)
20 ? CHR¤(140)CHR¤(140)
30 GOSUB 320
40 GOSUB 200
50 GOSUB 350
60 FOR I%=16384% TO 30719% STEP 2048%
70 FOR J%=0% TO 63% STEP 4%
80 GOSUB 130
90 NEXT J%
100 NEXT I%
110 GOSUB 370
120 END
130 FOR K%=0 TO 31%
140 K1%=A%(1%,K%)
150 K2%=A%(2%,K%)
160 POKE I%+J%+64%*K%,K1%
170 POKE I%+J%+64%*K%+2%,K2%
180 NEXT K%
190 RETURN
200 FOR X%=0% TO 32%
210 FOR Y%=0% TO 32%
220 IF (X%*Y%) MOD A% MOD B% = C%
    THEN PSET (X%,Y%),1%
230 NEXT Y%
240 NEXT X%
250 FOR I%=1% TO 2%
260 FOR J%=1% TO 32%
270 A%(I%,J%)=PEEK(&O42000+I%*1%+J%*64%)
280 ' POKE &O42000+I%+J%*64%,0%
290 NEXT J%
300 NEXT I%
310 RETURN
320 LOCATE 1,21
330 INPUT "A,B,C";A%,B%,C%
340 ' A%=16% B%=3% C%=2%
341 ' A%=8% B%=3% C%=2%
342 ' A%=8% B%=3% C%=1%
343 ' A%=30% B%=7% C%=3%
344 RETURN
350 ? AT(24,21)"Поехали..."
360 RETURN
370 ? AT(24,21)"(C) Romanoff D. "
380 ? AT(0,21)"          " 20 пробелов
390 RETURN
```

Строки 340—343 содержат возможные значения параметров, вводимых с клавиатуры в строке 330.

«Звуковизор»

```
10 CLS
20 Y%=0%
30 FOR X%=0% TO 510%
40 K%=PEEK(&O177716)
50 IF K%=-32576 THEN Y%=(Y%+1%)
   MOD 200% ELSE Y%=(Y%-1%) MOD 200%
60 IF Y%<0% THEN Y%=0%
70 IF Y%>200% THEN Y%=200%
80 PSET (X%,200%-Y%),1%
90 NEXT X%
100 GOTO 10
```

После запуска программы нужно подать звуковой сигнал с подключенного к БК магнитофона (воспроизведение записи с кассеты или усиление речи с микрофона в режиме записи с нажатой кнопкой паузы).

В процессе преподавания математики часто возникает необходимость графически иллюстрировать математические понятия, формулы, решения конкретных задач. Компьютер с его богатыми графическими возможностями в этом случае позволяет сделать то, что не под силу другим техническим средствам. А чтобы облегчить учителям (да и не только им) написание программ для построения графиков функций, автор статьи предлагает простую и удобную подпрограмму вычерчивания осей координат.

Е. И. Капустин,

г. Невинномысск

Построение и масштабирование декартовых осей координат

При иллюстрировании различных математических зависимостей чаще всего приходится сталкиваться с построением в декартовой системе координат графиков функций одной переменной. Очень удобно использовать для этого программный фрагмент, позволяющий строить и масштабировать декартову систему координат в зависимости от наименьшего (A) и наибольшего (B) значений аргумента (как правило, задаваемых пользователем), а также наименьшего N и наибольшего M значений функции на отрезке AB (чаще всего вычисляемых по ходу выполнения программы). Строки 10 и 20 в данном листинге приводятся только для отладки.

Возможности программы можно расширить: например, если множество значений функции велико, то ось Y лучше масштабировать как 1:10, добавив строку:

```
645 IF M>30 OR (M-N)>30 TH TY=ZY*10 EL TY=ZY
```

и заменив в строках 660 и 690 ZY на TY.

В полученной системе координат теперь легко строится график непрерывной функции, заданной до строки 500 оператором DEF FNA. (Например так, как показано в строках 800 — 840 приведенного листинга.)

```
5 DEF FNA(X)=(X*X-1)*SIN(X/2-.5)
10 INPUT "ВВЕДИТЕ ОТРЕЗОК ИЗМЕНЕНИЯ        АРГУМЕНТА X";A,B
20 INPUT "ВВЕДИТЕ ОТРЕЗОК ИЗМЕНЕНИЯ        ФУНКЦИИ Y";N,M
30 ' ..........................................................
500 IF A>=0 AND B>0 TH ZX=250%/B EL 520
510 U%=1%
520 IF A<0 AND B>0 TH ZX=250%/(B-A) EL 540
530 U%=ABS(A)*ZX
540 IF A<0 AND B<=0 TH ZX=250%/ABS(A) EL 560
550 U%=254%
560 IF N>=0 AND M>0 TH ZY=180%/M EL 580
570 V%=187%
580 IF N<0 AND M>0 TH ZY=180%/(M-N) EL 600
590 V%=200%-ABS(N)*ZY
600 IF N<0 AND M<=0 TH ZY=180%/ABS(N) EL 620
```

```
610 V%=1%
620 ? CHR$(12%)
630 LINE (U%,0%)-(U%,201%),3%
640 LINE (0%,V%)-(255%,V%),3%
650 FOR I%=1% TO INT(ABS(N))
660 LINE (U%-1%,V%-SGN(N)*ZY*I%)-(U%+1%,V%-SGN(N)*ZY*I%),3%
670 NEXT I%
680 FOR I%=1% TO INT(ABS(M))
690 LINE (U%-1%,V%-SGN(M)*ZY*I%)-(U%+1%,V%-SGN(M)*ZY*I%),3%
700 NEXT I%
710 FOR I%=1% TO CINT(ABS(A))
720 LINE (U%+SGN(A)*ZX*I%,V%+1%)-(U%+SGN(A)*ZX*I%,V%-1%),3%
730 NEXT I%
740 FOR I%=1% TO CINT(ABS(B))
750 LINE (U%+SGN(B)*ZX*I%,V%+1%)-(U%+SGN(B)*ZX*I%,V%-1%),3%
760 NEXT I%
770 ' ........................................
800 H=(B-A)/50
810 PSET (U%+A*ZX,V%-FNA(A)*ZY),1
820 FOR I%=1 TO 50
830 LINE -(U%+(A+I%*H)*ZX,V%-FNA(A+I%*H)*ZY),1
840 NEXT I%
```

Эта заметка предназначена для начинающих программистов на БЕЙСИКе и призвана помочь им освоить кодирование произвольного цвета символов.

А. В. Бриндеев,
Москва

Выбор цвета символов средствами БЕЙСИКа

В режиме 32 символа в строке каждое знакоместо состоит из восьми вертикальных линий, цвета которых кодируются содержимым ячейки @#214. (Точнее, каждое знакоместо состоит из десяти одинаковых горизонтальных полосок из восьми точек, расположенных по вертикали одна под другой. Содержимое ячейки @#214 определяет цвет каждой из этих восьми точек, а одинаковость полосок порождает вертикальные линии. — *Прим. ред.*)

Цвет каждой линии кодируется парами битов (11 — красный, 10 — зеленый, 01 — синий, 00 — черный), причем их запись в число производится от правой линии к левой. Например, для кодировки «КЗСКЗСКЗ», где «К», «З» и «С» — красный, зеленый и синий цвета соответственно, получаем число 1011011011011011_2. Занести это значение в ячейку @#214 можно с помощью оператора POKE &O214,&B1011011011011011 или в более короткой шестнадцатеричной записи — POKE &O214,&HB6DB (таблица перекодировки в шестнадцатеричный формат приведена в №5 за 1994 г., с. 47). После этой команды курсор принимает желаемый цвет, а каждый символ будет выглядеть как прорезь соответствующей формы в непрозрачной плашке, наложенной на курсор, так что цветные вертикальные линии последнего просвечивают сквозь прорезь.

(Аналогичным способом кодируется произвольный цвет фона экрана и служебной строки с помощью ячейки @#212. Подробнее об этом можно прочитать в журнале «Вычислительная техника и ее применение», №12 за 1990 г. (с. 33, п. 4), а об использовании ячеек @#212 и @#214 для организации окон и меню в ассемблере и на БЕЙСИКе — в журнале «Персональный компьютер БК-0010 — БК-0011М», №3 за 1994 г. (с. 31 и примечание редактора на с. 36). — *Прим. ред.*)

Есть идея!

Творцам ANDOS'а

1. В программе для форматирования дискет ANFORMAT из комплекта ANDOS очень полезной была бы функция Safe Format (по аналогии с Norton Utilites на IBM), выполняющая верификацию дискеты с переформатированием отдельных (сбойных) дорожек и инициализацией диска после завершения этой операции.

2. В той же программе ANFORMAT желательно обеспечить возможность задания в отдельном пункте основного меню номеров первой и последней форматируемых дорожек (с вводом по умолчанию значений 0 и 79 соответственно).

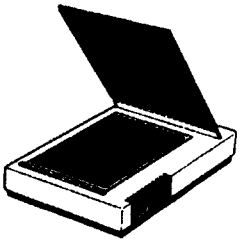
3. При работе с ANFORMAT нередки ситуации, когда даже после нескольких попыток какую-либо дорожку так и не удается успешно отформатировать. В таком случае (после трех попыток) программа прекращает работу, даже не пытаясь инициализировать диск. Лучше было бы при наличии поврежденной дорожки или ее части (когда даже с трех попыток форматирование не удается) помечать ущербные кластеры в FAT как сбойные с выполнением операции инициализирования. А после окончания операции форматирования неплохо выдавать сведения об общем объеме диска и о его «полезной» емкости за вычетом сбойных кластеров (как это сделано в утилите FORMAT на IBM).

4. Среди разнообразных возможностей, предоставляемых имеющейся на IBM программой PCTOOLS, существует операция верификации файлов. Пользователю предлагается выбрать (пометить в выведенном на экран каталоге диска) нужные файлы, а затем каждый из них читается для контроля качества записи в ОЗУ блоками по нескольку кластеров. Такая возможность проверки отдельных файлов намного удобнее, чем реализованные в различных программах для БК (в частности, в ANFORMAT) верификаторы всей дискеты целиком.

(Примечание. В операционной системе ANDOS можно проверить качество записи отдельного файла, вызвав его на просмотр по команде View («AP2»+«3») и пролистав от начала до конца. В версии ANDOS 3.1 при этом рекомендуется включить режим «текстового дампа» с помощью клавиши «ТАБ».)

5. При нажатии на клавишу «ВВОД» в DiskMaster производится загрузка файла и попытка его запуска с начала (конечно, если курсор был установлен не на имя подкаталога или «..»). При этом контроль расширения имени не производится. Очень заманчивой была бы возможность контроля расширения имени и при определенных сочетаниях символов — запуска какой-либо служебной программы. Например, при попытке «запуска» файла с расширением «.VXT» должен производиться запуск VorteX! и загрузка в него данного файла, при расширении «.COD» — запуск дискового БЕЙСИКа и отработка оператора CLOAD и т. д. Конкретные действия для каждого типа расширения было бы удобно задавать в специальном меню аналогично Extension Menu, реализованному в Norton Commander на IBM.

6. Одним из удобств при копировании файлов средствами оболочки DiskMaster является проверка качества записи и возможность помечать нечитаемые кластеры как сбойные (опция «Mark»). Было бы неплохо дополнить аналогичной возможностью стандартную функцию записи в ОС ANDOS. (Пока же при любой ошибке записи на диск выдается диагностическое сообщение и после нажатия любой клавиши делается попытка продолжить запись, а по «СТОП» происходит прерывание операции. Если же после нажатия на «СТОП» повторить запись файла, он снова попадает на то же подпорченное место дискеты. Тогда единственный способ все-же записать файл — установка в дисковод другого диска, но, согласитесь, это очень неудобный выход из положения.)



Заманчивая идея подключить к БК сканер обуревает многих БКманов уже давно. Так, еще в 1990 г. Ю. А. Зальцманом был разработан ТВ-сканер на основе видеокамеры «Электроника-841», однако из-за дороговизны и дефицитности последней ТВ-сканер так и не стал популярным.

Предлагаемая ниже разработка, напротив, проста по конструкции и не содержит каких-либо слишком дорогих и дефицитных деталей, что наверняка обеспечит ей множество поклонников. Хотя данное фотосчитывающее устройство рассчитано на работу с графопостроителем XY—4140 (или аналогичным), не слишком сложно будет доработать конструкцию для установки на принтеры разных типов (с соответствующей переделкой программы).

Ю. А. Зальцман,
г. Алма-Ата

Сканер для БК-0010

Сканером называется устройство для ввода в компьютер графической информации. Обычно это весьма сложная и дорогостоящая система, состоящая из сканирующего устройства (ручного или автоматического), специализированного программного обеспечения и одной или нескольких печатных плат, устанавливаемых в компьютер. (Сканер для IBM PC, например, стоит от 300 до 3000 долларов.)

На БК-0010 тоже хотелось бы иметь устройство для ввода графической информации — схем, чертежей, рисунков и т. д. Но сканеры от прочих компьютеров к БК не подходят (не говоря уже об их практически недоступной для БКманов цене), а разработанная автором несколько лет назад телевизионная система ввода «ТВИНТ» не обеспечивает необходимой точности полученного изображения. Но возможен компромисс — использование имеющихся на БК периферийных устройств в нестандартной роли при условии их доработки.

Предлагаемый программно-аппаратный комплекс состоит из широко распространенного и работающего с БК-0010 плоттера типа XY—4140 (XY—4150, XY—4160 или аналогичного), дополнительного устройства (устанавливаемой на плоттер самодельной фотоголовки с электронным преобразователем) и специализированной программы.

Схема преобразователя

Принципиальная схема устройства, показанная на рис. 1, состоит из входных ключей на логических микросхемах DD1—DD2, цифро-аналогового преобразователя (ЦАП) на ре-

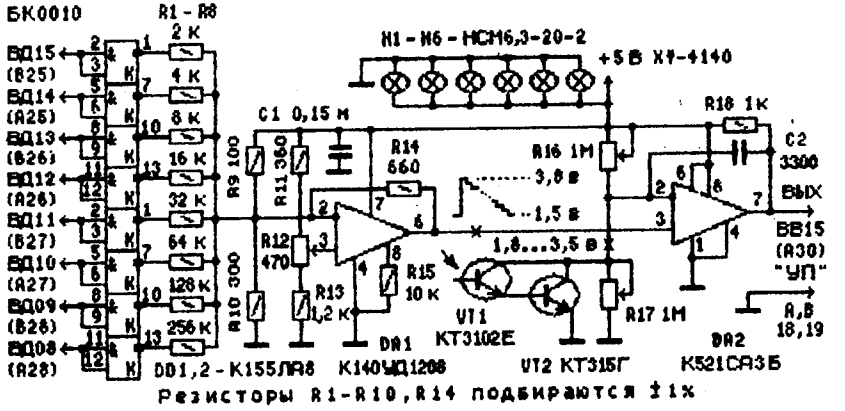


Рис. 1. Аналого-цифровой преобразователь сканера. Принципиальная схема

зисторах R1—R8 и операционном усилителе (ОУ) DA1 и фотоусилителя-компаратора DA2.

Схема работает следующим образом. На входы ключей DD1—DD2 с порта вывода БК-0010 подается 8-разрядный двоичный код. Ключ, на который подана «логическая 1», открывается, и через соответствующий резистор R1—R8 протекает ток, сила которого обратно пропорциональна сопротивлению резистора. Если открыто несколько ключей, токи через соответствующие резисторы суммируются на входе ОУ DA1, на котором выполнен преобразователь ток—напряжение.

Величины сопротивлений резисторов R1—R8 пропорциональны «весам» двоичных разрядов порта, а следовательно, напряжение на выходе ОУ пропорционально поданному на входы ключей двоичному коду. Таким образом осуществляется цифро-аналоговое преобразование «код—напряжение».

Фототранзистор VT1, являющийся чувствительным элементом сканера, при его освещении выдает сигнал, усиливаемый транзистором VT2 и поступающий на один из входов компаратора DA2. Высокий коэффициент усиления применяемой микросхемы-компаратора (около 150000) делает ненужным дальнейшее усиление сигнала фототранзистора — роль усилителя играет компаратор. На второй вход компаратора подается напряжение с выхода ЦАП, и оба сигнала сравниваются между собой. В зависимости от того, напряжение какого из сигналов больше, на выходе компаратора появляется «0» или «1», поступающая на один из разрядов порта ввода БК. Специальная программа, работающая в БК-0010, определяет характер поданного сигнала и, изменяя код на разрядах порта ввода, подбирает такое его значение, чтобы добиться равенства напряжений, поступающих с фототранзистора и ЦАП. Когда это условие выполнено, можно считать, что код, поданный на порт вывода, численно соответствует яркости точки объекта, считанной фототранзистором, — остается обработать код по определенному алгоритму и приписать полученное значение яркости определенной точке экрана.

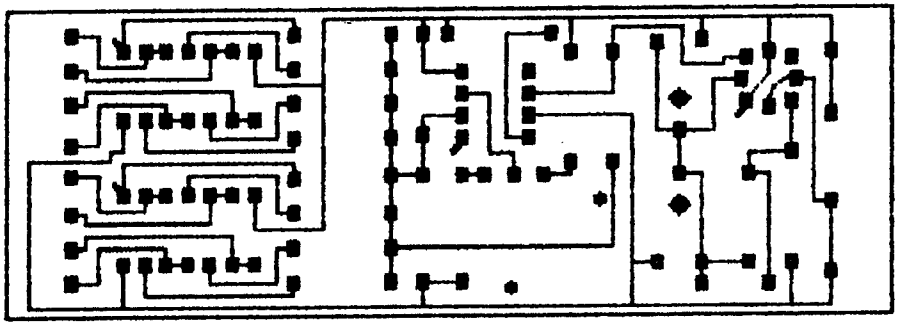


Рис. 2. Аналого-цифровой преобразователь сканера. Чертеж печатной платы

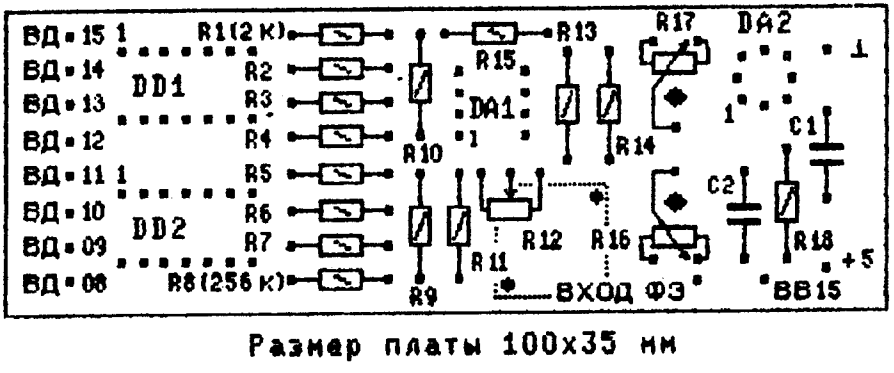


Рис. 3. Аналого-цифровой преобразователь сканера. Схема размещения элементов на печатной плате

Все это вместе (электронное устройство, программа и БК-0010) представляет из себя аналого-цифровой преобразователь (АЦП), преобразующий яркость сканируемого объекта в соответствующий ей цифровой код в диапазоне 0—255.

Остается заставить фотоголовку, установленную на плоттере, перемещаться по точкам объекта (рисунка, чертежа и т.п.) и последовательно вводить в компьютер значения их яркости. Это также выполняет программа, работающая на БК-0010 и управляющая плоттером.

Изготовление фотоголовки

Схематический чертеж фотоголовки в разрезе показан на рис. 4. Размеры на чертеже не приведены, так как они зависят от размеров и характеристик объектива, имеющегося в распоряжении пользователя.

Фотоголовка состоит из корпуса (1), выполненного из непрозрачного материала (например, эбонита). В нижней стенке корпуса (2) закреплен объектив (3). В качестве последнего может быть использован подходящий объектив или окуляр от любого оптического прибора, с диаметром входной линзы 10—15 мм и фокусным расстоянием 5—15 мм. За объективом, на оптической оси фотоголовки установлен фототранзистор (4), причем расстояние от сканируемого объекта до объектива должно составлять примерно 1/5 расстояния от объектива до фототранзистора (так как требуемое увеличение оптической системы фотоголовки — 5 раз). Другое обязательное условие — объектив должен проецировать изображение сканируемого объекта (5) на фототранзистор, находящийся в его фокальной плоскости. Как этого добиться, не зная технических характеристик объектива и не имея навыков расчета оптических систем? Как ни странно, довольно просто.

Возьмите имеющийся объектив и кусочек матового стекла (применяемого в фотографии для наводки на резкость) и закрепите их параллельно друг другу на подходящей маленькой рейке (шариковой ручке или карандаше) с помощью кусочка пластилина. На той же рейке закрепите перед объективом какой-либо контрастный объект (клочок плотной бумаги с напечатанным на нем типографским шрифтом). Хорошо осветив объект (но стараясь в то же время, чтобы прямой свет не попадал на матовое стекло) и двигая относительно друг друга объектив, объект и стекло, добейтесь получения на стекле четкого, увеличенного примерно в 5 раз изображения. Если при этом расстояние от объекта до стекла превысит 50—70 мм, то это значит, что выбранный объектив имеет слишком большое фокусное расстояние и непригоден для применения в фотоголовке — придется поискать другой. В крайнем случае вместо объектива можно использовать просто небольшую линзу с соответствующим фокусным расстоянием (но разрешение сканера при этом несколько снизится).

Получив четкое изображение объекта на матовом стекле, тщательно измерьте все расстояния в полученной оптической системе и изготовьте корпус фотоголовки. Установите в него объектив, а на нижней стенке закрепите опорную «ножку» (6), с помощью которой будет фиксироваться расстояние от объектива до бумаги на плоттере. Нижний конец ножки должен быть закруглен и тщательно отшлифован, так как он при сканировании должен легко скользить по бумаге. Сбоку на корпусе (1) закрепите вырезанный из тонкой жести кронштейн (7), за который фотоголовка будет крепиться на плоттере вместо его пишущего узла, причем, когда ножка (6) касается бумаги, фотоголовка должна распола-

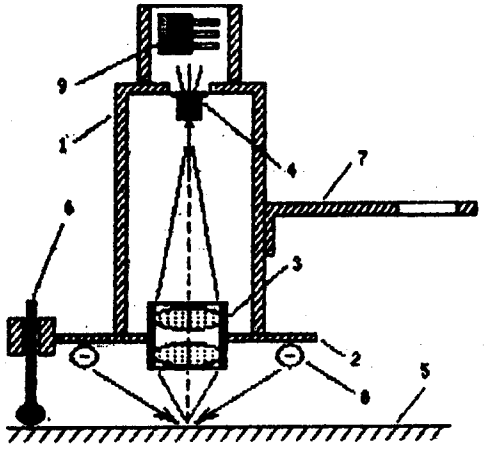


Рис. 4. Схема оптической системы сканера

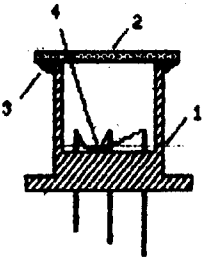


Рис. 5. Доработка транзистора КТ3102

гаться перпендикулярно к ней, что достигается подгибанием кронштейна (7).

Чтобы внешнее освещение не влияло на работу сканера, на фотоголовке размещена собственная осветительная система, состоящая из шести сверхминиатюрных лампочек типа НСМ6,3-20-2 (Н1—Н6 на рис. 1), соединенных параллельно и питаемых напряжением 5 В. Лампочки (8) размещаются вокруг объектива по окружности диаметром 20—30 мм таким образом, чтобы расположенный в центре участок объекта не затенялся краем объектива.

Установив фотоголовку на плоттер и включив подсветку, поместите матовое стеклышко в фокусе, где будет впоследствии установлен фототранзистор (4), и окончательно наведите объектив на резкость, изменяя длину ножки (6) и подгибая кронштейн (7). Удалив матовое стекло, установите фототранзистор (4) (о его изготовлении сказано ниже) и закрепите его, а также транзистор VT2 (9) любым подходящим клеем. При установке необходимо, чтобы кристалл фототранзистора располагался точно в плоскости, в которой было установлено матовое стекло. Предварительно нужно выполнить необходимые соединения транзисторов тонким многожильным монтажным проводом и сделать тем же проводом вывод из фотоголовки (длиной 30—40 см).

При изготовлении фотоголовки нужно достичь как можно меньших ее размеров и массы, так как плоттеру придется ее перемещать вместо пера.

Изготовление фототранзистора

Выпускаемые промышленностью фототранзисторы имеют большие линейные размеры фоточувствительных элементов и нас не устраивают. Придется изготовить фототранзистор самостоятельно.

За основу берется транзистор типа КТ3102 (с любой буквой, но с коэффициентом усиления по току не ниже 400). Крышка корпуса (1) транзистора (рис. 5) аккуратно удаляется (это легче всего сделать, спилив ее углы надфилем) и вместо нее подклеивается стекло (2). В качестве последнего лучше всего использовать покровное стекло для работы с микропрепаратами, толщиной 0.5—1 мм. Поставив транзистор на стекло, нанесите снаружи на стык корпуса со стеклом эпоксидный клей (3) и после его затвердевания лишнее стекло осторожно обломите (или опилите алмазным диском либо надфилем). Возможно, придется испортить несколько стекол, прежде чем получится приемлемый результат. В крайнем случае можно использовать вместо стекла прозрачную пластмассу (оргстекло, полистирол), но при этом трудно добиться герметичной склейки, а ее оптические свойства намного хуже.

После изготовления фототранзистора измерьте его параметры — коэффициент усиления и начальный ток коллектора (результаты не должны отличаться от измеренных вначале более чем на 10%), а затем обрежьте вывод базы на уровне корпуса.

Размеры кристалла (4) транзистора таковы (примерно 0.6 x 0.6 мм), что при увеличении оптической системы в 5 раз получается разрешение сканера порядка 0.1 мм, или 250 линий на дюйм.

Сборка и детали сканера

Схема АЦП собирается на печатной плате из фольгированного стеклотекстолита толщиной 1.5—2 мм согласно чертежу, приведенному на рис. 2 и 3.

Примененный в схеме ОУ типа К140УД1208 можно заменить на К140УД12. Применение других типов ОУ нецелесообразно, так как они требуют напряжение источника питания в лучшем случае порядка 10—15 В, а К140УД1208 может работать на 3—5 В. Вместо транзистора VT2 можно использовать любой кремниевый n-p-n транзистор с коэффициентом усиления по току не менее 200. Подстроечные резисторы R16, R17 — типа СП3-27б-0.125 Вт, R12 — многооборотный типа СП5-3.

Резисторы R1—R10 и R14 должны быть подобраны с точностью не хуже 1%. Это можно сделать, отбирая их с помощью цифрового омметра или моста из числа близких по номиналу обычных резисторов типа МЛТ либо подгоняя их сопротивление, сошлифовывая часть проводящего слоя ластиком. Можно, разумеется, применить и специальные прецизионные резисторы, например типа С2-29 (если их удастся достать!). Остальные резисторы — типа МЛТ-0.25-5%, конденсаторы — типа КМ.

Собранную печатную плату АЦП поместите в корпус плоттера, головку закрепите вместо пера в пишущем узле. Выполните все указанные на схеме соединения с БК-0010 и блоком питания плоттера (плоттер подключается к порту «УП» БК стандартным образом, как указано в его описании, поэтому общий провод схемы достаточно соединить с общим проводом плоттера). Блок питания плоттера, выдающий стабилизированное напряжение + 5 В (интегральный стабилизатор МА7805), как правило, выдерживает дополнительную нагрузку без проблем. Если возникают признаки перегрузки (перегрев МА7805, выпрямительных диодов или снижение напряжения более чем на 0.05 В), следует вместо имеющихся в нем диодов поставить более мощные, а микросхему МА7805 фирмы TESLA заменить на однотипную любой другой фирмы (например, на UA7805). Можно также запитать лампочки — основной потребитель тока сканера от БК-0010 или от отдельного блока питания. Для соединения схемы сканера с БК-0010 лучше всего установить на корпусе плоттера дополнительный разъем.

Налаживание схемы

Для налаживания потребуется осциллограф с возможностью измерения входных напряжений с точностью не хуже 10% и входным сопротивлением не менее 10 МОм, например С1-48 или С1-64 с выносным делителем 1:10. Подключив ЦАП к компьютеру (к старшим разрядам порта вывода, так как от младших управляется плоттер!), нужно написать простейшую программу на ассемблере, формирующую на выходе порта нарастающий код, например:

```
    CLR @#177714
0:  ADD #400,@#177714
    BR 0
```

Запустив ее и подключив осциллограф на выход ЦАП (вывод 6 DA1), с помощью подстроечного резистора R12 выставьте полученную на экране осциллографа «лесенку» из 256 ступеней точно посередине зоны усиления ОУ (не должен быть «завален» ни ее верх, ни низ). Амплитуды начала и конца «лесенки» указаны на принципиальной схеме и составляют примерно от 3.8 до 1.5 В относительно общего провода схемы соответственно. Если при любом положении резистора R12 «лесенка» ограничивается сверху или снизу (это возможно, если резисторы R1—R10 и R14 подобраны неточно), следует уменьшить сопротивление резистора R14. Затем подключите фотоголовку с осветителем и поставьте ее на белую бумагу. Подстроечным резистором R16 установите напряжение на входе 2 компаратора равным приблизительно 1.5 В. Перенесите фотоголовку на черную бумагу и с помощью резистора R17 установите напряжение около 3.5 В. Эту операцию подгонки уровня «белого» и «черного» фототока придется повторить несколько раз до получения стабильных результатов.

Подключив осциллограф к выходу (вывод 7 компаратора), наблюдайте за импульсами — их скважность должна меняться при изменении освещенности фототранзистора. На этом налаживание схемы можно считать законченным.

Программа сканирования

Программа сканирования построена следующим образом. После запуска и начальных установок начинается движение фотоголовки по сканируемому объекту. В каждой точке объекта головка считывает его яркость, а АЦП преобразует ее в 8-разрядный двоичный код. Алгоритм преобразования — двухступенчатый встречный, позволяющий сократить время преобразования в среднем в 8 раз и обеспечивающий достаточную точность. К сожалению, низкое быстродействие примененного ОУ не позволяет использовать алгоритм последовательных приближений, который сократил бы время преобразования еще в 4 раза.

Полученное цифровое значение яркости очередной точки обрабатывается по довольно сложному алгоритму, обеспечивающему получение трех градаций яркости для вывода на экран без подстройки на большинстве объектов. По этому алгоритму текущее значение яркости сравнивается с переменными MIN и MAX, и если оно больше MAX или меньше MIN, то значение MAX или MIN соответственно заменяется на текущее. Затем определяется «уровень черного» по формуле:

$$URB = \frac{2 \cdot URP + MIN + MAX}{4},$$

где URP — яркость предыдущей точки. «Уровень серого» определяется по другой формуле:

$$URG = \frac{2 \cdot URP + MIN + URB}{4}.$$

Таким образом, установленные уровни сильно зависят от текущей яркости объекта, выражаемой через URP, и данный алгоритм не только определяет среднюю яркость через MIN и MAX, но и позволяет выявлять небольшие текущие перепады яркости. Ввод в формулу URG значения MIN учитывает квадратичную зависимость фототока транзистора от яркости.

Кроме того, предусмотрена возможность ввода ручной поправки яркости в виде константы, складываемой с обоими уровнями (URB и URG), к чему приходится прибегать в редких случаях — при сканировании очень темных или очень светлых малоконтрастных объектов.

При установке режима «3 уровня яркости» переменная URB используется для определения граничной яркости четных точек строки экрана, а URG — нечетных. В режиме «2 уровня яркости» переменной URG присваивается значение URB.

Работа со сканером

Установив фотоголовку на плоттере и выполнив все подключения, запускают программу сканирования.

Программа выдает краткий перечень команд и запрос на загрузку файла в формате редактора P1.MZ (дисковая, значительно измененная автором версия редактора БК-PAINT А. В. Бакерина). Ответив «Y», можно загрузить файл (например, масштабную сетку), и сканированный рисунок будет накладываться поверх него. Любой иной ответ выводит на экран «пустой» файл P1.MZ, в котором нанесены масштабные оси с оцифровкой в сантиметрах.

Следует запрос на установку головки плоттера. Стрелками управления курсором надо поставить фотоголовку на нужную точку объекта (обычно левый верхний угол сканируемого поля) и нажать «ВВОД».

Затем следует запрос масштабных коэффициентов Мх и Му. Изменяя эти коэффициенты, можно произвольно в широких пределах менять масштаб вводимого сканером изображения. Если же предполагается сканирование в масштабе 1:1, можно просто дважды нажать «ВВОД» — коэффициенты будут заданы по умолчанию.

После этого следует запрос на установку правой границы поля экрана и курсора. Следует подвести курсор к предполагаемой правой границе поля изображения и нажать «КТ» — появится вертикальная черта (при сканировании она будет стерта). Если граница установлена ошибочно, можно сместить курсор и снова нажать «КТ» — черта будет перенесена. Ограничение ширины сканирования позволяет в большинстве случаев значительно сократить время ввода изображения. Затем следует установить курсор на то место экрана, откуда надо начать сканирование (левая граница), и нажать «ВВОД».

Начнется сканирование со скоростью около 180 точек экрана в секунду, при движении головки в обе стороны. Скорость можно менять с помощью клавиш управления курсором «ВВЕРХ» и «ВНИЗ». При снижении скорости несколько возрастает четкость ввода штриховых рисунков. Выбранная скорость поддерживается системным таймером БК-0010 (регистры @#177706—@#177712), поэтому он должен быть исправен. При максимальной скорости (константа 40, установлена изначально) сканирование всего экрана в масштабе 1:1 занимает примерно 8 минут.

Можно изменить алгоритм вывода изображения (с двумя или тремя градациями яркости), если в процессе сканирования нажать клавиши «2» или «3» соответственно. Исходно установлен режим с тремя градациями яркости. Переход на две градации целесообразен при вводе контрастных штриховых изображений (чертежей).

Инвертировать выводимое изображение можно командами «Р» или «N» — позитив или негатив. Исходно установлен режим «N», когда черные линии чертежа отображаются на экране белыми, а полученное изображение (в том числе, например, с фотографии) можно напрямую, без инверсии печатать на принтере.

Нажимая клавиши управления курсором «ВПРАВО» и «ВЛЕВО», можно вводить ручную поправку яркости изображения (исходно она равна 0). Увеличение положительных значений в режиме «НЕГАТИВ» приводит к повышению средней яркости на экране, а в режиме «ПОЗИТИВ» — наоборот. Обычно на практике вводимые поправки не превышают ±20. Изменение поправки даже на 1 в ту или иную сторону может иногда существенно изменить качество изображения (что говорит о высокой оптимальности выбранного алгоритма автоматической настройки) и обычно требует повторного сканирования с предварительно выбранной поправкой. Для пробного сканирования целесообразно задать узкий «столбик» изображения на его наиболее информативном месте, с целью меньших затрат времени. Но в девяти случаях из десяти ручная подстройка вообще не требуется.

В служебной строке индицируются установленные режимы работы программы (Mode: N;P / 2;3), а также константа скорости (Speed:) и поправка яркости (Braight:) со знаком, все цифровые данные восьмеричные.

Нажав клавишу «КТ», можно прервать сканирование в любой момент и записать полученное изображение (имя файла в формате P1.MZ запрашивается). Автоматическое окончание сканирования и запись происходит также по достижении изображением нижней границы экрана. Запрос на запись файла по окончании сканирования во всех случаях выдается только в крайнем левом положении головки, что позволяет продолжить сканирование объекта в следующий экран без ее смещения.

Если в распоряжении пользователя нет редактора P1.MZ, то полученный графический файл (его начальный адрес 42100, длина 32000) можно использовать просто как копию экрана либо переведя его в формат одного из графических редакторов БК-0010 с помощью конвертора графики.

По клавише «СТОП» происходит перезапуск программы с потерей изображения. Последнее также возможно, если при вводе коэффициентов Мх, Му или имени файла нажать какую-либо редактирующую клавишу, кроме «ЗАБОЙ» (код 30), например клавишу «СБР» (точно так же, как это происходит в мониторе БК), поэтому следует соблюдать осторожность.

Если требуется сканирование какого-либо рельефного объекта (например, печатной платы), можно прикрыть этот объект тонкой, но жесткой прозрачной пленкой для скольжения по ней фотоголовки. Но при удалении головки от объекта (толстая пленка, стекло) нарушается фокусировка объектива и четкость изображения быстро падает.

Следует учитывать, что сканирование изображения с данным типом фототранзистора и осветителя происходит преимущественно в инфракрасной зоне спектра, поэтому практически не читаются рисунки, выполненные фломастером или шариковой ручкой любого цвета, кроме черного. Зато вполне удовлетворительно читаются карандашные и типографские рисунки, а цветная типографская краска может создавать в инфракрасном свете заметный контраст там, где глаз его почти не видит.

Масштабирование изображения

Как уже говорилось, на запрос программы о масштабных коэффициентах можно просто дважды нажать клавишу «ВВОД», тогда изо

бражение будет вводиться в масштабе 1:1 — при его последующей распечатке на принтере МС6313 с помощью драйвера «ПС», в операционной системе автора MDOS.13, либо в редакторе P1.MZ будет воспроизведен оригинал, с которого производилось сканирование, с точностью до ±0.2 мм (±1 точка экрана). При выводе изображения на плоттер (с помощью программы СКР.4140) точность еще выше. Но иногда необходимо изменить масштаб вводимого изображения, например чтобы «втиснуть» лист обычного формата А4 в экран БК-0010 или, наоборот, «растянуть» изображение со спичечной этикетки или почтовой марки до размеров экрана. Для этого и служат масштабные коэффициенты.

«По умолчанию» коэффициенты равны: Mx = 54734, My = 43660 (числа восьмеричные, в десятичном виде — соответственно 23004 и 18352). Необходимый коэффициент по X или Y может быть вычислен по формуле: Mi = Mo·Ni, где Mo — масштабный коэффициент «по умолчанию» по данной оси, а Ni — изменение масштаба (число раз). Например, если нужно увеличить изображение в два раза, то Ni будет равно 2, а если, наоборот, уменьшить в два раза, то Ni будет равно 0.5. При вычислениях нужно помнить, что масштаб уменьшения ограничен размерами рабочего поля плоттера, а увеличения — тем, что вводимые восьмеричные коэффициенты не могут превышать число 177777 (65535_{10}).

Ошибки при вводе коэффициентов можно исправить, нажав любую нецифровую клавишу и повторив набор числа сначала. Незначащие нули в начале числа можно не набирать. По окончании набора следует нажать «ВВОД».

В заключение несколько слов о реальных возможностях сканера. Его разрешение таково, что после сканирования в масштабе 1:1 на экране БК можно легко читать типографский шрифт крупного формата (например, шрифт обычной пишущей машинки) и довольно уверено — шрифт принтера МС6313 обычного размера (плотность «ПИКА»). Неплохо читаются схемы и чертежи, отпечатанные на принтере в редакторе БКРАINT (за исключением обозначений, которые, как правило, слишком мелки). Отлично получаются штриховые рисунки и графики. В большинстве случаев хорошо сканируются изображения с фотографий нормальной плотности, причем при трех градациях яркости не только обеспечена «узнаваемость» лиц людей, но иногда изображение получается даже более выразительным, чем оригинал, за счет исключения мелких малозначительных деталей.

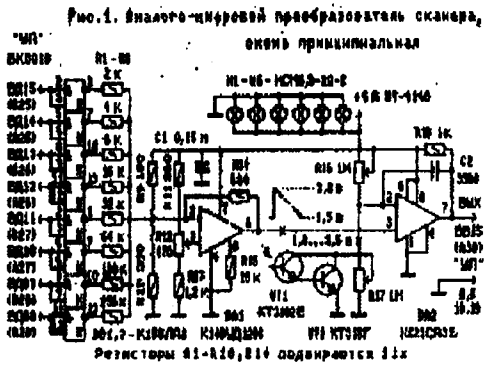


Рис. 6.

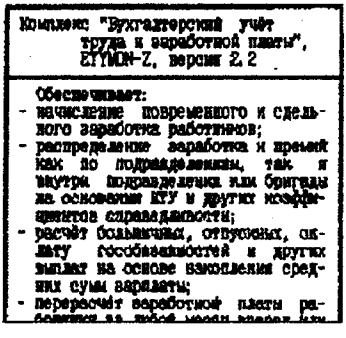


Рис. 8

Рис. 7

Рис. 9

Сказанное проиллюстрируем примерами изображений, полученных с помощью сканера для различных оригиналов. На рис. 6 показана копия принципиальной схемы блока АЦП (сравните его с рис. 1). Рис. 7 — фотография автора (три градации яркости), а рис. 8 демонстрирует копию текста, отпечатанного на лазерном принтере (как можно видеть, качество изображения вполне достаточное, чтобы попытаться реализовать даже оптический распознаватель текста!). На рис. 9 показана сканированная копия цветной картинки (заставка к статье в одном из номеров журнала «Радио»), а оригиналом для рис. 10 послужил рекламный штриховой рисунок из алма-атинской газеты «КомпИнформ».

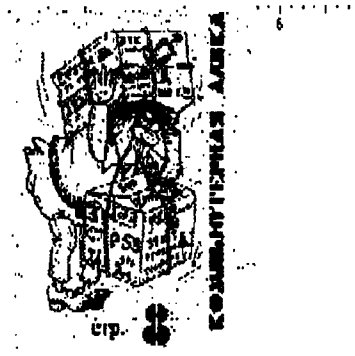


Рис. 10

Листинги программы сканирования

Ниже приводится ассемблерный листинг основной части программы сканирования, позволяющий понять алгоритм ее работы. Но после ее трансляции с ассемблера к полученному кодовому модулю нужно еще пристыковать файл исходной картинки в формате P1.MZ. Поэтому для получения работоспособной программы лучше всего ввести машинные коды, приведенные после ассемблерного листинга блоками по 2000 байт (для проверки приводятся контрольные суммы каждого блока).

Работоспособную копию программы можно получить в редакции журнала. Справки по телефону (095) 151-19-40. — Прим. ред.

```
;  FADC.4140 - программа подготовки файлов P1.MZ
; с помощью АЦП-фотосканирования на плоттере XY-4140
;  Файл "ЭКРАН+" подгружается по метке <END:>
;  Зальцман Ю.А., тел.691797, г.Алма-Ата, 25.05.94
;  Ассемблер МИКРО.10К-РП, МИКРО.10-01К БК-0010

B1:     EMT     14              Сброс

; Начальные установки
BEG:    MOV     #100,SPD        Скорость 100
        CLRB    NEG             Негатив
        CLRB    UR              3 уровня яркости
        CLR     CKR             Коррекция = 0
        MOV     #140,URP        Средний
        MOV     #100,MIN         начальный
        MOV     #200,MAX         уровень MIN и MAX
        MOV     PC,R1           Вектор @#4 -
        ADD     (PC)+,R1         на
        .@B1+2                   перезапуск
        MOV     R1,@#4           программы
        MOV     PC,R1
        ADD     (PC)+,R1        Инструкция,
        .@TE0+2                  сброс рулона,
        CLR     R2               запрос на
        EMT     20               чтение файла
        EMT     6               Нажать клавишу
        BIC     #240,R0         Ответ "Y"/"N"
        CMPB    'Y',R0          Чтение файла?
        BNE     0               Нет
        MOV     #42100,R5       Адрес загрузки
        MOV     #32,R0          Курсор
        EMT     16               вверх
```

```
        CALL    @#100536        Имя, загрузка
        TSTB    @#321           Ошибка ?
        BEQ     2               Нет
        EMT     6               Нажать клавишу
        BR      BEG             К началу

0:      MOV     #2440,R0        Длина пересылки
        MOV     PC,R1           Адрес
        ADD     (PC)+,R1         файла
        .@END+2                  <ЭКРАН+>
        MOV     #42122,R2       Адрес пересылки
1:      MOV     (R1)+,(R2)+     Пересылка и
        SOB     R0,1             разархивация
        CALL    @#42122          экрана
2:      MOV     #1,R0           Рамка
        MOV     #103,R1          рабочего
        CLR     R2               поля
        EMT     30               P1.MZ
        MOV     #777,R1          сверху
        EMT     32

; Установка головки и курсора
UST:    MOV     PC,R1           Сообщение:
        ADD     (PC)+,R1        <установите
        .@TE1+2                 головку...>
        CLR     R2
        EMT     20
        MOV     #-1,MX          Минимальный
        MOV     #-1,MY           шаг по X, Y
0:      BIT     #100,@#177716   Клавиша нажата ?
        BNE     0               Нет
        MOV     @#177662,R0     Команда
        CMPB    #12,R0          Конец?
        BEQ     6               Да
1:      CMPB    #31,R0          Вправо?
        BNE     2               Нет
        CALL    STR             Шаг вправо
        BR      0

2:      CMPB    #32,R0          Верх?
        BNE     3               Нет
        CALL    STU             Шаг вверх
        BR      0

3:      CMPB    #10,R0          Влево?
        BNE     4               Нет
        CALL    STL             Шаг влево
        BR      0

4:      CMPB    #33,R0          Вниз?
        BNE     0               Нет
        CALL    STD             Шаг вниз
        BR      0

6:      CLR     R2              Сообщение:
        EMT     20              <Масш.коэф.,Mx=>
        CLRB    @#105           Сброс клавиши
        CALL    @#100472        Ввод Mx
        TST     R5              По умолчанию?
        BNE     61              Нет
        MOV     #54734,R5       Коэфф.по X 1:1
```

```
61:     MOV     R5,MX           Масштаб по X
        CLR     R2              Сообщение:
        EMT     20              <My=>
        CALL    @#100472        Ввод My
        TST     R5              По умолчанию?
        BNE     62              Нет
        MOV     #43660,R5       Коэфф.по Y 1:1
62:     MOV     R5,MY           Масштаб по Y
        CLR     R2              Сообщение:
        EMT     20              <Уст.пр.границу и кур.>
        MOV     #11,R1          Начальная
        CLR     R2               установка
        EMT     24               курсора
        MOV     #225,R0         Графический
        EMT     16               режим
        MOV     #33,R0          Курсор
        EMT     16               вниз на 1 точку
        CLR     PRG             Удалить правую границу
7:      BIT     #100,@#177716   Клавиша нажата ?
        BEQ     7               Да - ждать
71:     BIT     #100,@#177716   Клавиша нажата ?
        BNE     71              Нет - ждать
        MOV     @#177662,R0     Команда
        CALL    KKK             Выполнить
        CMPB    #3,R0           Правая граница ?
        BNE     72              Нет
        MOV     R1,-(SP)        Сохранить границу
        MOV     PRG,R1          Прежняя граница
        BEQ     73              Если 0 - не стирать
        CLR     R0              Стирание
        CALL    GR               границы
73:     MOV     (SP)+,R1        Правая
        MOV     R1,PRG           граница
        MOV     #1,R0           Нанесение
        CALL    GR               границы
72:     CMPB    #12,R0          Конец?
        BNE     71              Нет
        CMP     PRG,R1          Курсор левее ?
        BLOS    71              Нет
        MOV     #100000,R0      Стирание
8:      CLR     -(R0)            низа
        CMP     #74200,R0        экрана
        BNE     8

; Начальные установки ячеек для сканирования
        CLR     PKX             Начало
        CLR     PKY              отсчета
        MOV     #-1,MKX         Абс.
        MOV     #-1,MKY          коорд.
        CLRB    NSP             Сброс шагов
        MOV     #40,SPD         Скорость максим.
        JSR     R4,@#110346     Сохр.регистры
        CLR     R1              0-я позиция
        MOV     PC,R2           Адрес
        ADD     (PC)+,R2         текста
        .@TE5+2                  сл.строки
9:      MOVB    (R2)+,R0        След.байт
        BEQ     91              Если 0 - конец
        EMT     22              Вывод в сл.строку
        INC     R1              След.позиция
        BR      9
```

```
91:     JSR     R4,@#110362     Восст.регистры
        MOV     #232,R0         Убрать
        EMT     16               курсор с экрана

; Сканирование
A0:     CALL    STR             Шаг
        INC     R1               вправо
        CALL    A1              Точка
        CMP     PRG,R1          Правый край ?
        BNE     A0              Нет
        INC     R1              Поправка по X
        CALL    STD             Шаг
        INC     R2               вниз
        CALL    PAU             Пауза
0:      CALL    STL             Шаг
        DEC     R1               влево
        CALL    A1              Точка
3:      CMP     #104,R1         Левый край ?
        BNE     0               Нет
        CMPB    #3,@#177662     Конец ?
        BNE     2               Нет
4:      JMP     ZAP             Запись файла

2:      CMP     #311,R2         Низ экрана ?
        BLOS    4               Да - запись файла
        DEC     R1              Поправка по X
        CALL    STD             Шаг
        INC     R2               вниз
        CALL    PAU             Пауза
        BR      A0              Реверс вправо

; Нанесение точки, опрос клавиатуры
A1:     CALL    ADC             Преобразование
        BIT     #1,R1           Четная точка ?
        BEQ     05              Да
        MOV     URB,URK         Уровень черного
        BR      06

05:     MOV     URG,URK         Уровень серого
06:     ADD     CKR,URK         Коррекция уровня
        BPL     07              URK >= 0
        CLR     URK             Иначе 0
07:     CLR     R0              Стереть точку
        TSTB    NEG             Негатив ?
        BNE     03              Нет
        CMPB    URT,URK         Уровень черного ?
        BHI     0               Нет
        BR      04

03:     CMPB    URT,URK         Уровень черного ?
        BLO     0               Да
04:     MOV     #1,R0           Поставить
0:      EMT     30               точку
000:    BIT     #100,@#177716   Клавиша нажата ?
        BEQ     01              Да
        RET

01:     MOV     @#177662,R0     Код клавиши
        CMPB    #32,R0          Скорость больше ?
        BNE     1               Нет
```

```
        DEC     SPD             Уменьш.задержки
        CMP     SPD,#37         Предел скорости ?
        BEQ     11              Да
02:     JSR     R4,@#110346     Сохр.рег.
        MOV     #24,R1          Позиция 20
        MOV     SPD,R2          Скорость
        BR      23              Вывод в сл.строку

1:      CMPB    #33,R0          Скорость меньше ?
        BNE     2               Нет
11:     INC     SPD             Увелич.задержки
        BR      02              Вывод

2:      CMPB    #10,R0          Ярк.меньше ?
        BNE     3               Нет
        DEC     CKR             - 1

; Вывод восьмеричного числа со знаком
20:     JSR     R4,@#110346     Сохр.рег.
        MOV     CKR,R2          Число положительное?
        BMI     21              Нет
        MOV     '+',R0          Символ "+"
        BR      22              Вывод

21:     MOV     '-',R0          Символ "-"
        NEG     R2              Абс.значение CKR
22:     MOV     #46,R1          Знак
        EMT     22               в сл.строку
23:     CLR     R5              Буфер
        MOV     #6,R3           6 итераций
        CLR     R0              Буфер символа
24:     INC     R1              След.поз.
        ROL     R2              Первая
        ROL     R0               цифра
        ADD     R0,R5           Незнач. 0 ?
        BEQ     25              Да
        ADD     #60,R0          Цифра - в код ASCII
        BR      26              Вывод

25:     MOVB    #40,R0          Пробел
26:     EMT     22              В сл.строку
        DEC     R3              - итерация
        BEQ     27              Конец
        CLR     R0              Очистить буфер
        ROL     R2              След.
        ROL     R0               цифра
        ROL     R2               в
        ROL     R0               R0
        BR      24              Продолжать

27:     CALL    PAU             Пауза
        JSR     R4,@#110362     Восст.рег.
        BR      000             Чтение клав.

3:      CMPB    #31,R0          Ярк.больше ?
        BNE     50              Нет
        INC     CKR             + 1
        BR      20              Вывод значения

50:     CMPB    #63,R0          3 уровня ?
        BNE     52              Нет
        CLRB    UR              3 уров.
```

```
51:     MOV     #13,R5          12-я поз.
        BR      41              В сл.строку
        RET

52:     CMPB    #62,R0          2 уровня ?
        BNE     4               Нет
        MOVB    #1,UR           2 уров.
        BR      51              В сл.строку

4:      BIC     #240,R0         Код ЛАТ-ЗАГЛ
        CMPB    'N',R0          Негатив ?
        BNE     5               Нет
        CLRB    NEG             Негатив
40:     MOV     #7,R5           6-я позиция
41:     MOV     R1,-(SP)        Вывод
        MOV     R5,R1            в
        EMT     22               сл.
        MOV     (SP)+,R1         строку
6:      RET

5:      CMPB    'P',R0          Позитив ?
        BNE     6               Нет
        MOVB    #1,NEG          Позитив
        BR      40              В сл.строку

; Пауза
PAU:    MOV     #20000,R0       Задержка
0:      SOB     R0,0             повтора
        RET

; Подпрограмма записи экрана
ZAP:    MOV     #225,R0         Символьный
        EMT     16               режим
        CLR     @#104           Сброс символа
0:      CLR     R1              Курсор
        MOV     #27,R2           на низ
        EMT     24               экрана
        MOV     PC,R1           Стирание
        ADD     (PC)+,R1         нижнего
        .@TE4+2                  поля
        CLR     R2               экрана,
        EMT     20               <ЗАПИСЬ>
        MOV     #320,R1         Адр.блока параметров
        MOV     #2,(R1)+        Команда "запись"
        MOV     #42100,(R1)+    Адрес файла
        MOV     #32000,(R1)+    Длина файла
        CALL    @#100552        <ИМЯ?>, запись файла
        TSTB    @#321           Ошибка ?
        BNE     1               Да
        JMP     BEG             Перезапуск

1:      EMT     6               Нажать клавишу
        BR      0               Повтор записи

; П/программа контроля координат курсора в экране
KKK:    EMT     16              Перемещение
        EMT     26              Коорд.курс.
        CMP     #103,R1         Левый край?
        BNE     0               Нет
        MOV     #31,R0          Сдвиг
        BR      KKK              вправо
```

```
0:      CMP     #777,R1         Правый край?
        BNE     1               Нет
        MOV     #10,R0          Сдвиг
        BR      KKK              влево

1:      TST     R2              Верхний край?
        BNE     2               Нет
        MOV     #33,R0          Сдвиг
        BR      KKK              вниз

2:      CMP     #313,R2         Нижний край?
        BNE     3               Нет
        MOV     #32,R0          Сдвиг
        BR      KKK              вверх

3:      RET

; Шаг плоттера
STR:    MOV     #4,R4           Вправо
0:      INCB    NSP             Подсчет числа шагов в масштабе
        ADD     MX,PKX          Абс.координаты
        BCC     0               Продолжение
        BR      ST0             Конец шага

STU:    MOV     #10,R4          Вверх
0:      INCB    NSP
        SUB     MY,MKY
        BCC     0
        BR      ST0

STL:    MOV     #14,R4          Влево
0:      INCB    NSP
        SUB     MX,MKX
        BCC     0
        BR      ST0

STD:    CLR     R4              Вниз
0:      INCB    NSP
        ADD     MY,PKY
        BCC     0

ST0:    BIT     #20,@#177712    Время истекло ?
        BNE     ST0             Нет
        MOV     #177714,R3      Адрес порта
        MOV     R4,@R3          Код в порт
        BIS     #2,R4           Строб
        MOV     R4,@R3           в порт
        BIC     #2,R4           Сброс строба
        MOV     R4,@R3          В порт
        MOV     SPD,@#177706    Скорость - уст.таймера
        MOV     #30,@#177712    Пуск таймера
        DECB    NSP             Конец шага?
        BNE     ST0             Нет
        RET

; Нанесение/удаление правой границы экрана
GR:     MOV     #10,R2          Верхняя точка
        EMT     30              Поставить точку
        MOV     #303,R2         Нижняя точка
        EMT     32              Провести вектор
        RET
```

```
; Аналого-цифровое преобразование текущей точки
ADC:    MOV     R3,-(SP)
        MOV     #177400,R0      Начальный код
        MOV     #177714,R3      Адрес порта
01:     MOV     R0,@R3          Код на ЦАП
        SUB     #10000,R0       - ступень "грубо"
        TST     @R3             Сигнал АЦП есть ?
        BPL     01              Нет - продолжать
        ADD     #10000,R0       Восстановить код ЦАП

0:      MOV     R0,@R3          Код на ЦАП
        ADD     #400,R0         + ступень "точно"
        TST     @R3             Сигнал АЦП есть ?
        BMI     0               Да - конец
        SUB     #400,R0         Восстановить код ЦАП

        SWAB    R0              Перевести код в мл.байт
        CMPB    #377,R0         Сбой АЦП ?
        BNE     1               Нет
        MOV     URP,R0          Иначе взять пред.значение
1:      MOV     R0,URT          Текущее значение
        CMP     MAX,R0          URT > MAX ?
        BHIS    2               Нет
        MOV     R0,MAX          MAX = URT
2:      CMP     MIN,R0          URT < MIN ?
        BLOS    3               Нет
        MOV     R0,MIN          MIN = URT
3:      MOV     URP,R3          Предыдущее значение
        ASL     R3              * 2
        ADD     MIN,R3          + MIN
        ADD     MAX,R3          + MAX
        ASR     R3              Разделить
        ASR     R3               на 4
        MOV     R3,URB          Уровень черного
        TSTB    UR              3 уровня ?
        BNE     4               Нет
        MOV     URP,R3          Предыдущее значение
        ASL     R3              * 2
        ADD     URB,R3          + URB
        ADD     MIN,R3          + MIN
        ASR     R3              Разделить
        ASR     R3               на 4
4:      MOV     R3,URG          Уровень серого
        MOV     R0,URP          Новое значение
        MOV     (SP)+,R3
        RET

; Системные переменные
        PKX:    .E              Абсолют.
        PKY:    .E               коорд.
        MKX:    .E              X,Y
        MKY:    .E               по + и -
        MX:     .E              Масшт.
        MY:     .E               коэфф.X,Y
        NSP:    .E              Число шагов/точку
        SPD:    .E              Скорость сканирования
        PRG:    .E              Правая граница
        NEG:    .E              Негатив/позитив
        UR:     .E              2/3 уровня
        URT:    .E              Уровень текущ.точки
```

```
URP:    .E              Уровень пред.точки
MIN:    .E              Мин.уровень
MAX:    .E              Макс.уровень
URB:    .E              Уровень черного
URG:    .E              Уровень серого
URK:    .E              Уровень компарации
CKR:    .E              Константа коррекции

; Текстовые сообщения и команды дисплея
TE0:    .B:214,214,12,12,12
.A:          Программа фотосканирования изображений¬
.A:               с помощью плоттера XY-4140¬¬¬
.A:            Зальцман Ю.А., г. Алматы, V-250594¬¬
.A:     Масштабные коэффициенты "по умолчанию", для M=1:1¬
.A:                 Mx = 54734; My = 43660¬¬
.A:     При сканировании: КТ - конец, запись файла P1.MZ;¬
.A: ↑/↓ - скорость больше/меньше; →/← - яркость больше/меньше;¬
.A:   2/3 - 2 или 3 уровня яркости; N/P - негатив/позитив¬¬
.A:              Чтение файла P1.MZ (Y/N) ?¬
.A:¬¬¬¬¬\

TE1:    .B:32,23,23,23,12
        .A: Установите головку плоттера: ↑ ↓ ← → ВВОД\

TE2:    .B:32,32,12,23,23
        .A: Введите масштабные коэффициенты:¬ Mx = \
        .B:32,31,31,31,31,31,31,31,31,31,31,31,31,31,31,31
        .A: My = \

TE3:    .B:32,32,23
        .A: Установите правую границу экрана ( ↑ ↓ ← → КТ )¬
        .A: и курсор ( ↑ ↓ ← → ВВОД )\

TE4:    .B:231,32,231,32,231,232
        .A: Запись файла P1.MZ. \

TE5:    .A: Mode: N / 3 ♦ Speed:    40 ♦ Bright: +     0
        .E

END:    END
```

Листинг в машинных кодах

Начальный адрес: 001000

```
104014  012767  000100  002270  105067  002270  105067  002266
005067  002302  012767  000140  002260  012767  000100  002254
012767  000200  002250  010701  062701  177730  010137  000004
010701  062701  002246  005002  104020  104006  042700  000240
122700  000131  001014  012705  042100  012700  000032  104016
004737  100536  105737  000321  001415  104006  000722  012700
002440  010701  062701  003524  012702  042122  012122  077002
004737  042122  012700  000001  012701  000103  005002  104030
012701  000777  104032  010701  062701  003035  005002  104020
012767  177777  002044  012767  177777  002040  032737  000100
177716  001374  013700  177662  122700  000012  001430  122700
000031  001003  004767  001366  000761  122700  000032  001003
004767  001374  000753  122700  000010  001003  004767  001402
000745  122700  000033  001342  004767  001410  000737  005002
104020  105037  000105  004737  100472  005705  001002  012705
054734  010567  001704  005002  104020  004737  100472  005705
```

```
001002 012705 043660 010567 001662 005002 104020 012701
000011 005002 104024 012700 000225 104016 012700 000033
104016 005067 001634 032737 000100 177716 001774 032737
000100 177716 001374 013700 177662 004767 001074 122700
000003 001016 010146 016701 001570 001403 005000 004767
001316 012601 010167 001552 012700 000001 004767 001300
122700 000012 001344 026701 001530 101741 012700 100000
005040 022700 074200 001374 005067 001466 005067 001464
012767 177777 001460 012767 177777 001454 105067 001456
012767 000040 001452 004437 110346 005001 010702 062702
002747 112200 001403 104022 005201 000773 004437 110362
012700 000232 104016 004767 000764 005201 004767 000102
026701 001376 001370 005201 004767 001030 005202 004767
000542 004767 000774 005301 004767 000046 022701 000104
001370 122737 000003 177662 001002 000167 000516 022702
000311 101773 005301 004767 000752 005202 004767 000464
000732 004767 001050 032701 000001 001404 016767 001300
001302 000403 016767 001272 001272 066767 001270 001264
100002 005067 001256 005000 105767 001230 001005 126767
001226 001240 101007 000404 126767 001214 001226 103402
012700 000001 104030 032737 000100 177716 001401 000207
013700 177662 122700 000032 001015 005367 001142 026727
001136 000037 001412 004437 110346 012701 000024 016702
001116 000431 122700 000033 001003 005267 001102 000763
122700 000010 001054 005367 001114 004437 110346 016702
001104 100403 012700 000053 000403 012700 000055 005402
012701 000046 104022 005005 012703 000006 005000 005201
006102 006100 060005 001403 062700 000060 000402 112700
000040 104022 005303 001406 005000 006102 006100 006102
006100 000755 004767 000134 004437 110362 000664 122700
000031 001003 005267 000756 000720 122700 000063 001006
105067 000722 012705 000013 000421 000207 122700 000062
001004 112767 000001 000676 000765 042700 000240 122700
000116 001011 105067 000654 012705 000007 010146 010501
104022 012601 000207 122700 000120 001374 112767 000001
000622 000762 012700 020000 077001 000207 012700 000225
104016 005037 000104 005001 012702 000027 104024 010701
062701 002031 005002 104020 012701 000320 012721 000002
012721 042100 012721 032000 004737 100552 105737 000321
001002 000167 176214 104006 000746 104016 104026 022701
000103 001003 012700 000031 000770 022701 000777 001003
012700 000010 000762 005702 001003 012700 000033 000755
022702 000313 001003 012700 000032 000747 000207 012704
000004 105267 000410 066767 000400 000366 103372 000431
012704 000010 105267 000366 166767 000360 000352 103372
000420 012704 000014 105267 000344 166767 000334 000326
103372 000407 005004 105267 000324 066767 000316 000304
103372 032737 000020 177712 001374 012703 177714 010413
```

Контрольная сумма: 113563

Начальный адрес: 003000

```
052704 000002 010413 042704 000002 010413 016737 000260
177706 012737 000030 177712 105367 000242 001352 000207
012702 000010 104030 012702 000303 104032 000207 010346
012700 177400 012703 177714 010013 162700 010000 005713
100373 062700 010000 010013 062700 000400 005713 100773
162700 000400 000300 122700 000377 001002 016700 000152
010067 000144 026700 000146 103002 010067 000140 026700
000132 101402 010067 000124 016703 000116 006303 066703
000112 066703 000110 006203 006203 010367 000102 105767
000064 001011 016703 000062 006303 066703 000062 066703
000052 006203 006203 010367 000050 010067 000034 012603
000207 000000 000000 000000 000000 000000 000000 000000
000000 000000 000000 000000 000000 000000 000000 000000
```

```
000000 000000 000000 000000 106214 005012 020012 020040
020040 020040 020040 170040 147722 151307 146701 140715
143040 152317 151717 140713 144716 147722 140727 144716
020321 155311 141317 140722 142726 144716 005312 020040
020040 020040 020040 020040 020040 020040 020040 020323
147720 147715 154335 020300 146320 152317 142724 140722
054040 026531 030464 030064 005012 020012 020040 020040
020040 020040 020040 175040 146301 141730 140715 020316
027340 027341 020054 027307 160440 146714 152301 026331
053040 031055 030065 034465 005064 020012 020040 020040
140755 155723 140724 147302 142731 145440 156317 143306
141711 142711 152316 020331 150042 020317 146725 146317
140736 144716 021300 020054 146304 020321 036515 035061
005061 020040 020040 020040 020040 020040 020040 020040
020040 046440 020170 020075 032065 031467 035464 046440
020171 020075 031464 033066 005060 020012 020040 020040
151360 020311 145723 147301 151311 153717 147301 144711
020072 172353 026440 145440 147317 141705 020054 140732
144720 154323 143040 145301 140714 050040 027061 055115
005073 027663 020255 020055 145723 151317 151717 154324
141040 146317 155730 027705 142715 154316 142733 020073
027676 020261 020055 151321 147713 152323 020330 147702
154314 142733 146457 147305 155730 035705 020012 020040
027462 020063 020055 020062 146311 020311 020063 151325
153717 150716 150440 145722 151717 144724 020073 027516
020120 020055 142716 140707 144724 027727 147720 144732
144724 005327 020012 020040 020040 020040 020040 020040
020040 020040 152376 147305 142711 143040 145301 140714
050040 027061 055115 024040 027531 024516 037440 005012
005012 005012 015000 011423 005023 172440 152323 147301
153717 152311 020305 147707 147714 145727 020325 146320
152317 142724 140722 020072 020263 020255 020261 020276
173767 162357 015000 005032 011423 173440 142727 144704
142724 146440 151701 152333 141301 154716 020305 147713
143334 144706 144703 147305 154724 005072 046440 020170
020075 015000 014431 014431 014431 014431 014431 014431
014431 020031 074515 036440 000040 015032 020023 151765
140724 147716 144727 142724 150040 140722 152727 020300
151307 147301 141711 020325 145734 140722 140716 024040
131440 126440 130440 137040 165440 020364 005051 144440
145440 151325 147723 020322 020050 020263 020255 020261
020276 173767 162357 024440 114400 114432 114432 020232
140772 144720 154323 143040 145301 140714 050040 027061
055115 020056 020000 067515 062544 020072 020116 020057
020063 020270 070123 062545 035144 020040 020040 030064
134040 041040 064562 064147 035164 025440 020040 020040
030040 020040 020040 000000 010704 062704 000236 010400
012703 037070 005001 012702 000020 005740 005043 006110
103012 112405 162705 000157 100404 000305 152405 162705
010757 160501 010113 077216 020327 036070 001354 012702
000034 014043 077202 062700 004745 012702 000553 000113
                    Контрольная сумма: 015010

                    Начальный адрес:   005000
114044 077202 062704 032000 114001 006301 016101 036470
001410 005301 001411 102404 062701 137000 010144 000410
005300 111044 114044 000760 114002 011444 105302 001375
020400 001352 000207 000000 177000 067764 171742 137767
067054 157337 077566 067174 000000 000000 000000 077300
177256 137734 167776 037576 067156 167576 171170 020574
136575 114173 063570 145576 171170 166170 067577 067566
167575 027757 053541 065143 066156 067155 067156 067155
063553 063547 054156 067147 054147 050156 060156 034076
067147 170357 064156 173177 170357 077156 067360 167577
```

```
171177 067155 065554 173174 033550 027177 170176 077547
077746 075370 067367 167572 077150 067367 167575 170177
167164 167570 075457 077061 066760 050156 147174 167557
061533 047566 167564 167574 171160 067155 067155 067155
066552 063156 061527 067154 060156 053537 066551 037537
065153 067071 171553 031457 067154 163515 067031 142433
077556 142232 077705 073731 077236 071777 073362 063114
056130 076431 146542 016001 106715 056256 076431 146542
146764 127215 014527 061175 033715 162715 056255 076431
146542 146475 127476 014534 061175 043715 067315 056254
076431 146542 146541 146541 014534 061175 060715 060715
056315 076431 146542 076403 056143 076431 146542 146541
146541 014532 061175 060715 060715 056315 076431 146542
146541 146541 014534 061175 043715 072261 056227 076431
146542 127476 117346 014534 061175 000715 120005 163200
101034 014534 061175 000715 133012 116746 014534 061175
000715 024024 163200 045235 076431 146542 024001 100024
023553 056202 076431 146542 050001 100012 012124 056202
076431 146542 120001 100005 002337 056202 076431 146542
145173 176327 056203 076431 146542 141564 144727 014534
061175 000715 131176 144727 014534 061175 000715 146063
157600 055311 076431 146542 014601 100630 144524 014534
061175 100715 130015 065601 056303 076431 146542 160067
060600 056315 076431 146542 146541 146541 014534 061175
060715 060715 056315 076431 146542 076403 056143 076431
146542 146541 146541 014527 061175 060715 060715 056315
076431 146542 146713 146541 014534 061175 000715 144610
145713 014534 061175 000715 022220 075600 056312 076431
146542 121401 100022 145173 014534 061175 000715 105244
151201 056305 076431 146542 154301 100171 142722 014532
061175 000715 136757 137720 014534 061175 150315 103776
141342 014534 061175 160715 137577 016341 056200 076431
146542 171564 165600 056274 076431 146542 015574 067603
173271 076431 146542 050601 100045 134557 014755 061175
000715 142510 172200 100160 010370 076431 146542 022001
100011 132471 014766 061175 163315 100021 132471 014501
061175 060715 060713 164715 076431 146542 146541 146541
014534 061175 060715 060715 056315 076431 146542 076403
056143 076431 146542 146541 146541 014532 061175 060715
060715 056315 076431 146542 146713 146541 014534 061175
075715 110712 135044 014534 061175 151315 100076 135141
014534 061175 161315 100370 000441 056262 076431 146542
160353 060603 056254 076431 146542 107771 120667 014527
061175 000715 107506 124713 014534 061175 000715 060743
145607 056251 076431 146542 033561 145607 056245 076431
146542 174301 103437 124313 014534 061175 160715 007740
145607 056250 076431 146542 100601 101407 122713 014534
061175 033715 101403 124713 014532 061175 000715 114634
124713 014534 061175 071315 145651 056241 076431 146542
124722 146541 014534 061175 060715 060715 056315 076431
146542 146541 146541 014534 061175 001715 061575 014534
061175 060715 062315 045143 076431 146542 146541 061544
014534 061175 075715 062303 056147 076431 146542 144732
141316 056377 076431 146542 144732 117316 056377 076431
```

Контрольная сумма: 000435

Начальный адрес: 007000

```
146542 141573 065334 014534 061175 075715 156303 056152
076431 146542 141573 064614 014532 061175 075715 106303
056151 076431 146542 120556 064170 014534 061175 043715
074251 056150 076431 146542 124507 063163 014534 061175
043715 071651 056146 076431 146542 052401 100525 062572
014534 061175 100715 125252 033200 053545 076431 146542
052726 033200 056143 076431 146542 177741 075271 056143
```

```
076431 146542 146541 173776 056143 076431 146542 146541
061544 014534 061175 060715 062315 056143 076431 146542
076403 056143 076431 146542 146541 146541 014532 061175
060715 060715 056315 076431 146542 127541 146541 014534
061175 060715 060665 056315 076431 146542 135541 146541
014534 061175 060715 071274 175227 076431 146542 141141
115375 014766 061175 060715 035705 101200 005370 076431
146542 145141 040102 164602 076431 146542 145713 020107
175202 076431 146542 146573 017555 175201 076431 146542
146722 130115 014534 061175 161315 046715 056266 076431
146542 146753 135515 014534 061175 174715 066715 100037
014532 061175 034715 060715 056315 076431 146542 146476
146541 014534 061175 042715 060715 056315 076431 146542
146541 146541 014534 061175 060715 060715 056315 076431
146542 076403 056143 076431 146542 146541 146541 014527
061175 060715 060715 056315 076431 146542 146541 146541
014534 061175 071315 103777 146505 014534 061175 165715
042632 056315 076431 146542 123771 146505 014534 061175
034715 042657 056315 076431 146542 132476 146505 014532
061175 100715 135701 146505 014534 061175 140715 014600
042600 056315 076431 146542 136513 146505 014534 061175
045715 042705 056315 076431 146542 146513 113561 014534
061175 045715 070715 056227 076431 146542 146513 146505
014512 061175 065715 042641 056315 076431 146542 146541
146505 014534 061175 060715 060715 056315 076431 146542
146541 146541 014534 061175 060715 060715 056315 076431
146542 076403 056143 076431 146542 146541 146541 014532
061175 060715 060715 056315 076431 146542 146541 146541
014534 061175 060715 060715 056315 076431 146542 113553
113553 014534 061175 052315 153236 056230 076431 146542
117124 115611 014534 061175 052315 153236 053630 076431
146542 117124 115611 014534 061175 052315 153236 056230
076431 146542 117124 115611 014534 061175 052315 153236
056230 076431 146542 117124 115611 014534 061175 052315
153236 056230 076431 146542 117124 115611 014532 061175
065715 065627 056227 076431 146542 146541 146541 014534
061175 060715 060715 056315 076431 146542 146541 146541
014534 061175 060715 060715 173315 076431 146542 076403
040543 076431 146542 146541 146541 007370 076431 146542
146541 146541 014755 061175 060715 060715 166715 076431
146542 136562 136562 014766 061175 165315 000664 076536
056200 076431 146542 123075 125601 100752 014534 061175
045715 153232 056230 076431 146542 112117 125341 055222
076431 146542 113130 112723 014534 061175 054315 105626
056222 076431 146542 113130 112723 014534 061175 054315
105626 056222 076431 146542 112117 052723 056207 076431
146542 115113 125211 056203 076431 146542 123075 052601
100725 014527 061175 165315 076264 100172 014534 061175
071315 071275 056275 076431 146542 146541 146541 014534
061175 060715 060715 056315 076431 146542 146541 146541
014534 061175 001715 061575 014534 061175 001715 061175
014532 061175 001715 061175 014534 061175 001715 061175
014534 061175 001715 061575 014534 061175 060715 061142
056315 076431 146542 055141 146542 014534 061175 060715
000014 146542 014534 061175 060715 000006 146542 014512
061175 060715 061137 056315 076431 146542 060631 146542
014534 061175 000715 000306 061777 056254 076431 146542
```

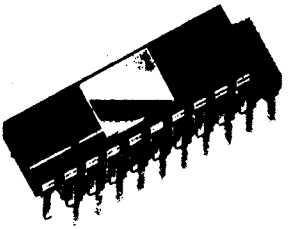
Контрольная сумма: 172414

Начальный адрес: 011000

```
061321 146542 014534 061175 160715 061142 056315 076431
146542 010001 061142 063715 076432 060543 061142 015715
061175 061141 146542 076433 060542 061142 063715 076432
060543 061142 056315 061142 061715 072543 105165 152212
```

074566	001571	104175	167210	061616	152143	167166	043616
100401	061141	146542	061134	146542	061543	073567	173567
073324	073324	031463	131463	021042	121042	042104	142104
107356	146542	146542	146542	146542	146541	146541	020010
040100	041116	002002	010362	134040	041116	002004	020362
134040	041111	171373	134331	001111	175402	010020	130731
001111	060604	060715	056315	047126	036103	164363	061315
056541	054534	044526	177103	155763	061315	000002	055535
047131	113111	146541	145173	061134	061116	061074	061350
060542	056142	053142	041542	171542	146542	061142	061135
061131	146511	136573	037401	100374	061134	061116	061074
061350	060542	056142	053142	041542	171542	146542	061142
061135	061131	146511	135742	123505	014534	061175	000715
133710	120555	014534	061175	150715	037257	056257	076402
035142	076403	060542	076402	051142	076402	166142	076403
030542	001000	061175	146765	130073	126745	001132	061175
004400	076402	146542	001141	061175	001133	061175	001344
061175	051715	076402	060142	146422	130073	127215	001134
061175	001376	061175	060733	076402	070142	001000	061175
001754	061175	000151	076402	040142	146420	156371	106600
056256	076402	035142	076402	000142	057700	076402	046142
076402	162142	076403	054542	001000	061175	146765	043775
106600	056256	076402	044542	076403	060542	076402	046142
076402	162142	076403	051542	076402	040142	146402	135742
127215	001134	061175	007400	076403	060542	076402	051142
076402	166142	076402	146542	000063	076402	160142	146436
174001	106675	000256	151705	147305	142711	152040	157317
144713	020054	150317	147722	020323	146313	153701	140711
152724	154722	040412	035061	041411	046101	004514	042101
004503	170011	142722	141317	140722	147732	140727	144716
005305	041011	052111	021411	026061	030522	004411	142776
147324	150701	152040	157317	140713	037440	004412	042502
004521	032460	004411	140744	004412	047515	004526	051125
026102	051125	004513	172411	147722	142727	154316	157040
151305	147716	147707	004412	051102	030011	005066	030012
035065	046411	053117	052411	043522	052454	045522	004411
151365	153717	147305	020330	142723	143722	005317	033060
004472	042101	004504	045503	026122	051125	004513	165411
151317	142722	141713	150711	152440	147722	147327	005321
041011	046120	030011	004467	052411	045522	037040	020075
005060	041411	051114	052411	045522	004411	147351	157301
020305	005060	033460	004472	046103	004522	030122	004411
152363	151305	152305	020330	147724	145736	005325	052011
052123	004502	042516	004507	167011	143705	152301	153711
037440	004412	047102	004505	031460	004411	142756	005324
041411	050115	004502	051125	026124	051125	004513	172411
147722	142727	154316	157040	151305	147716	147707	037440
004412	044102	004511	004460	167011	152305	004412	051102
030011	005064	030012	035063	041411	050115	004502	051125
026124	051125	004513	172411	147722	142727	154316	157040
151305	147716	147707	037440	004412	046102	004517	004460
162011	005301	032060	004472	047515	004526	030443	051054
004460	170011	151717	140724	144727	154324	030012	004472
046505	004524	030063	004411	152040	157317	152713	030012
030060	004472	044502	004524	030443	030060	040054	030443
033467	030467	004466	146353	153701	155711	020301	140716
140726	140724	037440	004412	042502	004521	030460	004411
140744	004412	042522	005124	030012	035061	046411	053117
040011	030443	033467	033066	026062	030122	165411	140714
144727	140733	004412	046503	041120	021411	031063	051054

Контрольная сумма: 006344



Пользователи БК-0011(М) довольно часто сталкиваются с невозможностью запуска программ для БК-0010(.01), рассчитанных на работу в расширенном ОЗУ. Не спасают даже мониторы эмуляции десятой модели, имеющиеся практически в любой дисковой ОС: в БК-0011(М) по адресам 140000—157777 находится только ПЗУ, а в диапазоне 120000—137777 ОЗУ занято самой операционной системой. Как быть?

Несложная доработка, описанная в этой статье, позволит не только обеспечить работу на БК-0011(М) практически любых программ для БК-0010(.01), но и превратить старую модель БК-0011 в новую (БК-0011М), а также существенно расширит возможности этой ПЭВМ.

С. М. Неробеев, А. В. Сорокин,
Москва

Доработка ОЗУ 16 кб для БК-0011(М)

При работе в операционных системах ANDOS, MKDOS, NORD и других, основная задача которых — эмуляция БК-0010(.01), в большинстве случаев монитор БК-0011(М), расположенный по адресам 140000—157776, уже не нужен, при этом оказываются незадействованными 8 кб адресного пространства. Так как оперативная память БК-0011(М) составляет 128 кб, то на такую «мелочь», казалось бы, можно не обращать внимания. Но в то же время, когда для работы нужна резидентная программа, рассчитанная в БК-0010(.01) на расширенную память, приходится использовать специальные средства типа PAGE MANEGER, чтобы установить ее в одну из свободных страниц ОЗУ. Можно также воспользоваться операционными системами, которые, находясь в седьмой (системной) странице, оставляют свободной память с адреса 120000 (едва ли не единственным представителем таких ОС является AO-DOS v 2.10).

Гораздо изящнее эта проблема решена на БК-0010(.01), где острый дефицит памяти вынуждает эффективнее использовать имеющиеся 64 кб адресного пространства. Вместо неиспользуемых ПЗУ БЕЙСИКа (адреса 120000—157776) подключена оперативная память. Причем по адресам 120000—137776, как и на БК-0011(М), устанавливается операционная система, а с адреса 140000 можно загрузить любую резидентную программу: отладчик, ассемблер-систему, модули издательской системы Vortex и т. п.

Но что мешает сделать то же самое и на БК-0011(М)? В этом случае можно будет использовать все программное обеспечение, написанное для работы на БК-0010(.01) в диапазоне адресов 140000—157776. Например, операционная система MKDOS при перезагрузке с включенным ОЗУ устанавливает по этим адресам свою оболочку и потом подгружает ее именно оттуда, что позволяет удалить файл «МС» с электронного диска и, соответственно, увеличить объем диска Е: до 150 блоков.

Еще более заманчиво подключить ОЗУ по адресам 160000—173776. Это даст возможность устанавливать любой драйвер дисковода — от прошивки 253 до нового драйвера фирмы Altec, позволяющего без дополнительных сложностей работать с дисками, отформатированными на IBM.

Большие перспективы открываются и для владельцев БК-0011. Если в ОЗУ, подключенное по адресам 140000—173776, загрузить монитор БК-0011М и прошивку 326 (предварительно сделав небольшую доработку схемы излучателя звука — перепаяв перемычку), то компьютер превратится в БК-0011М. Стоимость доработки при этом будет на порядок ниже затрат на переделку БК-0011 в БК-0011М обычным способом (с заменой ПЗУ монитора).

Прежде всего вам потребуется плата дополнительного ОЗУ 16К для КНГМД БК-0010(.01). Доработка заключается в изготовлении трех основных модулей, собранных на микросхемах DD1—DD3 соответственно.

Модуль на микросхеме DD1 служит для отключения микросхем ПЗУ БК-0011(М) и подачи сигнала синхронизации SYNC на плату дополнительного ОЗУ. Его схема показана на рис. 1.

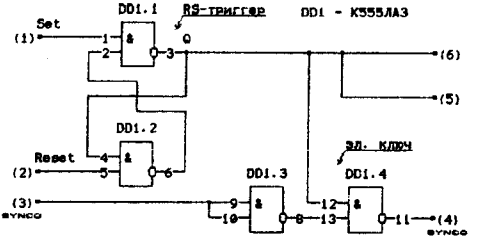


Рис. 1

Подключение модуля выполняется следующим образом:

(1) — к выводу 38 D1 (К1801ВП1-128*);
(2) — к выводу А1 или Б19 разъема МПИ;
(3) — к выводу 5 D10*;
(4) — к выводу 17 бокового разъема ОЗУ;
(5) — к контакту Б6 МПИ (вывод 23 DS17);
(6) — к выводу 23 D11*.

(Звездочкой отмечены обозначения микросхем, соответствующие принципиальной схеме КНГМД.)

Второй модуль служит для перенаправления обращений по адресам 120000—157776 на адреса 140000—174000. Модуль собран на микросхеме DD2 и изображен на рис. 2.

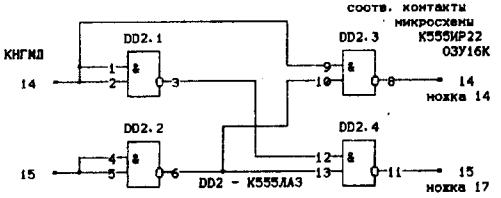


Рис. 2

В таблице показана логика отображения адресных разрядов AD13 и AD14 с шины КНГМД на защелку адреса (отсоединенные ножки микросхемы К555ИР22) ОЗУ.

Таблица отображения адресных разрядов

Шина КНГМД	120000	Модуль	Защелка	140000
AD15 AD14 (15) AD13 (14)	1 0 1	=>	AD15 AD14 (15) AD13 (14)	1 1 0
	140000			160000
	1 1 0	=>		1 1 1

Третий модуль служит для коррекции зоны отклика дополнительного ОЗУ и нужен из-за того, что в стандартном КНГМД происходит

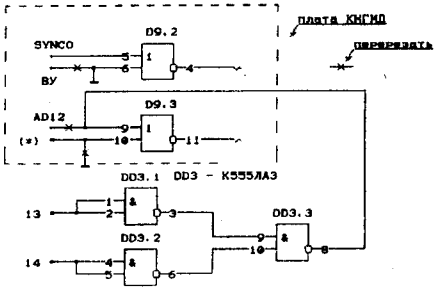


Рис. 3. Вывод (*) соединить с AD11 (контакт 12 бокового разъема КНГМД) для обеспечения отклика ОЗУ (ПЗУ) до адреса 174000

блокировка обращений ко всем адресам при установке бита 12 (AD12). Этот модуль производит изменение зоны отклика. После его установки блокируется обращение только к адресам 170000—177777 (если произвести доработку, которая описана чуть ниже, то к адресам 174000—177777). Схема третьего модуля приведена на рис. 3.

Далее необходимо доработать ОЗУ, КНГМД и БК-0011(М). Рассмотрим эти доработки на примере ОЗУ 16 кб фирмы ANCO (то же верно и для любой схемы расширенного ОЗУ на 16 кб).

- Найдите выводы микросхемы К555ИР22 (ИР23), к которым идут дорожки от контактов 14 и 15 бокового разъема ОЗУ, и отогните их, как показано на рис. 4. Отсоедините также вывод 17 бокового разъема ОЗУ от платы КНГМД.

- На плате КНГМД перережьте выводы 9, 6 и 10 микросхемы D9 (К559ИП2). Выполните соединения, как показано на рис. 3 (вверху). Разорвите цепь, соединяющую вывод 23 микросхемы ПЗУ D11 (К1801РЕ2-326 {-253}), и в разрыв подпаяйте резистор с сопротивлением 1 КОм.

- На плате БК-0011(М) перережьте цепь, соединяющую вывод 23 микросхемы ПЗУ базовой операционной системы DS17 (К1801РЕ2-324), и в разрыв подпаяйте резистор с сопротивлением 1 КОм. Соедините вывод 23 ПЗУ БОС с контактом Б6 разъема МПИ БК-0011(М).

После выполнения указанной доработки КНГМД, ОЗУ и БК-0011(М) можно собрать и подключить к ним описанные в начале статьи дополнительные модули. Устройство в наладке не нуждается и при исправных деталях и правильном монтаже начинает работать сразу после изготовления.

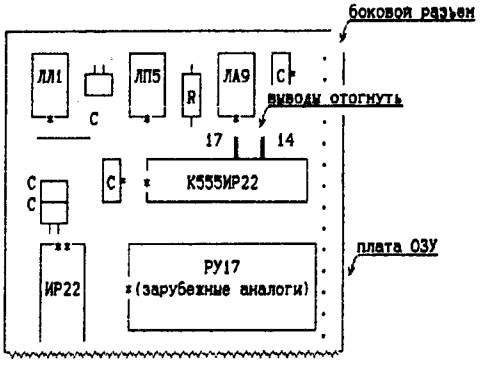


Рис. 4. Размещение элементов на плате ОЗУ

Программное обеспечение дополнительного ОЗУ 16 кб для БК-0011(М)

Ниже приводится листинг драйвера, который позволяет устанавливать и запускать программы для работы в дополнительном ОЗУ, доработанном описанным здесь способом (ассемблер стандарта TURBO7M.

Включение ОЗУ осуществляется следующей парой команд:

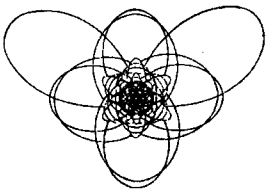
```
MOV #10,@#177130
CLR @#177130
```

Выключение ОЗУ происходит по команде RESET или при перезапуске машины (как именно — зависит от конкретного варианта подключения первого модуля).

```
; START  v 1.01
; Утилита для операционной системы
; БК-0011М  МК-DOS v 2.xx и выше
; Автор: Maker production
; Открытая версия
; Имя файла после компиляции : START или ST .

;ТАБЛИЦА ПОДПРОГРАММ И ЯЧЕЕК МК-DOS

        DIR=120014
        TXT=120050
        BEGIN=10000

START:  MOV     #330,@#177664   ; Включение режима РП
        MOV     #232,R0         ; Выключение курсора
        EMT     16              ;
        CALL    CLEAR           ; Очистка экрана
        RESET                   ; Запрещение щелчка
        MOV     #META,@#4       ; Перевод информации
        MOV     #40000,R0       ; из ПЗУ в дополнительное
        MOV     #20000,R1       ; ОЗУ 16 кб
        MOV     #140000,R2      ;
        MOV     (R2)+,(R0)+     ;
        SOB     R1,.-2          ;
META:   MOV     #ERROR,@#4      ; Если ОЗУ нет...
        MOV     #10,@#177130    ; Включение
        CLR     @#177130        ; дополнительного ОЗУ
        MOV     #40000,R0       ; Монитор 11М в
        MOV     #140000,R1      ; дополнительное ОЗУ
        MOV     #10000,R2       ;
        MOV     (R0)+,(R1)+     ;
        SOB     R2,.-2          ;
        MOV     #METB,@#4       ; Если ОЗУ кончилось...
        MOV     #60000,R0       ; ПЗУ КНГМД в
        MOV     #160000,R1      ; дополнительное ОЗУ
        MOV     #10000,R2       ;
        MOV     (R0)+,(R1)+     ;
        SOB     R2,.-2          ;
METB:   CALL    CLEAR           ; Очистка экрана
        MOV     #1330,@#177664  ; Отключение режима РП
        MOV     @#DIR,R3        ; Адрес каталога в R3
        MOV     30(R3),R2       ; Количество записей
        ADD     #450,R3         ;
        MOV     #BUF,R5         ; Адрес буфера
1:      ADD     #30,R3          ;
        CMPB    #377,@R3        ; Файл удален?
```

```
        BEQ     1               ;
        CMPB    #200,@R3        ; BAD-файл?
        BEQ     2               ;
        CMPB    #177,2(R3)      ; Подкаталог?
        BEQ     2               ;
        TST     22(R3)          ; Выделен в МС?
        BPL     2               ;
        MOV     #16,R0          ; Длина имени файла
        MOV     R3,R1           ;
        TST     (R1)+           ;
        MOVB    (R1)+,(R5)+     ; Заносим имя в буфер
        SOB     R0,.-2          ;
        MOV     24(R3),ADRES    ; Адрес загрузки
        MOV     26(R3),LONG     ; Длина файла
        BR      10              ;
2:      SOB     R2,1            ; Следующий файл в МС
10:     MOV     #BUF,R0         ; BUF -> R0
        MOV     #16,R1          ; Длина имени в буфере
7:      CMPB    #56,(R0)+       ; Это точка?
        BEQ     6               ; Да
        SOB     R1,7            ;
        BR      8               ; Нет -> на выход
6:      MOV     #3,R2           ; Длина расширения
        MOV     #BUF1,R3        ; Адрес расширения
12:     CMPB    (R0)+,(R3)+     ; Это 'DRV'?
        BNE     8               ; Нет
        SOB     R2,12           ;
        BR      LOAD            ; Загрузка программы в
                                ; память с адреса 10000
8:      JMP     TITLE           ; в МК-ДОС

LOAD:   MOV     #3,@#320        ; Создание блока параметров
        MOV     #BEGIN,R0       ;
        MOV     R0,@#322        ; Адрес загрузки
        CLR     @#324           ;
        MOV     #326,R1         ;
        MOV     #BUF,R0         ;
        MOV     #16,R2          ; Имя из буфера в блок
        MOVB    (R0)+,(R1)+     ; параметров
        SOB     R2,.-2          ;
        BR      ZAPUSK          ; На загрузку
BUF1:   .ASCII  /DRV/           ; Расширение
        .EVEN                   ;
ZAPUSK: MOV     #320,R1         ; Адрес блока параметров
        CALL    @#120002        ; Вместо EMT36 (MKDOS)
        TSTB    1(R1)           ; Ошибка?
        BNE     STOP            ; Да
        MOV     #PUP,@#4        ; Самое главное!
        MOV     #10000,R0       ;
        MOV     LONG,R2         ;
        MOV     ADRES,R1        ;
        MOVB    (R0)+,(R1)+     ;
        SOB     R2,.-2          ;
PUP:    CMP     ADRES,#160000   ;
        BLO     STA             ;
END:    EMT     4               ;
        JMP     @#100270        ;
STA:    JMP     @ADRES          ;

ADRES:  .WORD   40000           ; Адрес в МК-ДОС
LONG:   .WORD   40000           ; Длина в МК-ДОС
```

```
BUF:    .BLKB   20              ; Резерв для имени файла
STOP:   HALT                    ; При ошибке чтения...

ERROR:  CALL    CLEAR           ; Если нет ОЗУ...
        MOV     #1330,@#177664  ;

        JSR     R1,@TXT         ; Вывод сообщения

        .BYTE  12,12,12
        .ASCII /┌═══ (C) Maker ═══ START V1.01 ═══┐/<12>
        .ASCII /│                                  │/<12>
        .ASCII /│ Для использования данной програм─ │/<12>
        .ASCII /│  мы необходимо ОЗУ16К по адресам: │/<12>
        .ASCII /│         140000-173777.           │/<12>
        .ASCII /│                    1995 год ─┘   │/<12>
        .ASCII /└──────────────────────────────────┘/<12>
        .EVEN
        CALL    KEY             ; Нажатие на клавишу?
        JMP     END             ; На выход

KEY:    RESET                   ; Щелчок
        BIT     #101,@#177716   ; Корректная обработка
        BEQ     KEY             ; нажатия на клавишу
2:      BIT     #101,@#177716   ;
        BNE     2               ;
        RET                     ; Возврат из П/П

CLEAR:  MOV     #40000,R0       ; Очистка экрана
        MOV     #20000,R1       ; (очень медленная,
        CLR     (R0)+           ; но для всех
        SOB     R1,.-2          ; понятная)
        RET                     ;

TITLE:  JSR     R1,@TXT         ; Если файл не в формате
                                ; START'ера...

        .BYTE  12,12
        .ASCII /┌═══ (C) Maker ═══ START V1.01 ═══┐/<12>
        .ASCII /│                                  │/<12>
        .ASCII /│ Программа предназначена для уста─ │/<12>
        .ASCII /│ новки ... по адресам 140000-173777 │/<12>
        .ASCII /│──────────────────────────────────│/<12>
        .ASCII /│ Формат файла :     < NAME >.DRV  │/<12>
        .ASCII /│──────────────────────────────────│/<12>
        .ASCII /│ Формат вызова из МС :            │/<12>
        .ASCII /│    1) Подвести инверсную строку на │/<12>
        .ASCII /│       файл ;                     │/<12>
        .ASCII /│    2) в служебной строке набрать : │/<12>
        .ASCII /│       < DEV >:\>START ←┘ , где   │/<12>
        .ASCII /│       DEV = A,B,C,E .            │/<12>
        .ASCII /│                    1995 год ─┘   │/<12>
        .ASCII /└──────────────────────────────────┘/<12>
        .EVEN

        CALL    KEY             ; Нажатие на клавишу?
        JMP     END             ; На выход

        .END                    ; Конец программы START.
;
```



В пятом выпуске нашего журнала за 1994 г. было опубликовано описание программы NEWMON — загружаемого в ОЗУ БК-0011М расширенного аналога монитора БК-0010, обладающего дополнительными графическими возможностями для создания и печати изображений большого размера (при печати — копия формата А4 с четкостью не хуже, чем в режиме SVGA на IBM-совместимых ПЭВМ). Эта программа существенно переработана автором и получила название NEWMON95. В настоящей статье рассматриваются основные отличия от предыдущей версии.

С. К. Румянцев,
пос. Локня Псковской обл.

NEWMON95: новая версия монитора для БК-0011М

Давайте вместе поработаем с программой NEWMON95, попутно замечая ее отличия от старой версии. Загрузка NEWMON95 производится с адреса 1000, после запуска с которого программа автоматически настраивает память, пересылает свой текст в рабочий фрагмент ОЗУ (адрес 100000) и передает на него управление. Таким образом, всю работу по предварительной настройке программа берет на себя. Пользователю необходимо лишь загрузить ее в компьютер БК0011М с магнитофона или диска и запустить с начального адреса. После запуска на экране появляется символ «¤», означающий, что мы находимся в редакторе-отладчике, аналогичном прошитому в МСД БК-0010. (Директивы этого отладчика описаны в «Руководстве системного программиста БК-0010», а также в статье Ю. А. Зальцмана «Об архитектуре и программировании на ассемблере» в №1 за 1994 г.).

Кроме директив отладки с клавиатуры можно подавать на исполнение специальные коды управления графическими буферами (нажатие алфавитных клавиш одновременно с клавишей «СУ»):

Восьмеричный код	Десятичный код	Клавиши	Назначение
1	1	«СУ»+«A»	Обнуление рабочего буфера
2	2	«СУ»+«B»	Инверсия рабочего буфера
3	3	«СУ»+«C» или «КТ»	Обмен буферов
4	4	«СУ»+«D»	Очистка по маске
5	5	«СУ»+«E»	Установка по маске
6	6	«СУ»+«F»	Инверсия по маске
11	9	«СУ»+«I»	Печать смежного буфера
13	11	«СУ»+«K»	Масштабирование смежного буфера
21	17	«СУ»+«Q»	Круговая перестановка частей рабочего буфера

Схема взаимодействия графических буферов при подаче этих кодов дана на рис. 1.

Как видим, в NEWMON95 появился новый код управления 21, которого не было в предыдущей версии. Он позволяет просматривать рабочий буфер графического изображения и редактировать картинку по частям. Каждая из трех частей рабочего буфера выводится в экранную область пользователя (с адреса 42000, окно объемом 36000 байт) и не занимает область служебной строки (как в предыдущей версии). Благодаря этому появляется возможность редактировать изображение средствами режима «ГРАФ», причем реализован автоповтор при нажатии редактирующих клавиш, а также имеется возможность управления графическим курсором при помощи мыши (рис. 2). При нажатой левой кнопке мыши точки записываются (аналогично режиму «ЗАП»), при нажатой правой — стираются («СТИР»), при обеих нажатых кнопках мыши точки инвертируются. Для точного позиционирования графического курсора с клавиатуры автоповтор можно блокировать клавишей «БЛОК РЕД». При рисовании вблизи границ экрана можно не бояться испортить невидимые в данный момент части картинки, так как смещение курсора за пределы текущего рабочего окна блокировано.

Попробуем нарисовать с помощью NEWMON95 несложную картинку. Допустим, вы увлекаетесь астрологией и хотели бы составить гороскоп. Для этого можно воспользоваться простейшей заготовкой (рис. 3): здесь основная часть изображения — это круг, разделенный на двенадцать равных секторов по числу знаков зодиака. Проще всего получить такой рисунок с помощью программы на языке высокого уровня, например на ФОКАЛе. (Транслятор этого языка должен быть заранее скопирован на магнитную ленту из ПЗУ БК-0010.) ФОКАЛ нужно загрузить в память БК-0011М по адресу 120000 с помощью директивы МЧ отладчика NEWMON95 и запустить с этого же адреса.

При составлении программы на ФОКАЛе, рисующей картинку, необходимо учесть некоторые особенности новой версии NEWMON95 по сравнению со старой. Так, система координат при построении точек и отрезков теперь полностью совпадает с принятой в БК0010: координата X отсчитывается от левого верхнего угла экрана слева направо (десятичные значения от 0 до 511 в режиме 64 символа в строке и от 0 до 255 в режиме 32 символа в строке), а координата Y отсчитывается сверху вниз и может принимать значения от 0 до 719 (на экране отображаются точки с координатами от 0 до 239, остальные находятся в скрытой памяти). Такая система координат является более привычной для программистов, работающих с БК. Вот листинг программы для рисования заготовки гороскопа:

```
1.1 X FCHR(12,1,3,1)
1.2 S X=256;S Y=360;S R=240;S L=0.5236
1.3 F I=L,L,12*L;D 2
1.4 X FCHR (3,11);Q
2.1 X FT(1,X,Y);F J=0,L/5,L;D 3
3.1 X FV(1,X+R*FSIN(I+J),Y-R*FCOS(I+J))
```

В строке 1.1 производится ряд подготовительных действий: нормализация экрана (код 12) и очистка обоих графических буферов (комбинация кодов 1, 3, 1). Эта типовая строка потребуется вам и в дальнейшем при работе с ФОКАЛом в NEWMON95. В строке 1.3 осуществляется построение окружности и секторов. Завершается работа программы строкой 1.4, где полученное в рабочем буфере изображение переносится в смежный буфер (код 3) и демонстрируется в уменьшенном виде на экране с (десятичный код 11). Это уменьшенное изображение масштабировано в таких пропорциях, которые оно будет иметь после распечатки на принтере, что позволяет правильно оценивать качество полученной картинки. Если все в порядке, изображение можно сразу распечатать на принтере, набрав команду X FCHR(9) и нажав «ВВОД». Если же вы захотите отредактировать картинку (например, добавить изображения знаков зодиака, которые трудно получить программно), это можно сделать вручную в режиме «ГРАФ».

Теперь ФОКАЛ нам больше не понадобится, и мы выходим из него по команде РМ (PASS MONITOR), попадая в отладчик NEWMON95. Займемся редактированием полученной картинки. (Для этого желательно подключить к компьютеру мышь.) Набрав на клавиатуре «АР2»+«СУ»+«U», входим в режим «ГРАФ» и, нажав клавишу «КТ», пересылаем изображение из смежного буфера в рабочий. Теперь, манипулируя мышью или редактирующими клавишами, в картинку можно вносить любые изменения и дополнения. Для редактирования частей изображения, находящихся за пределами экрана, следует нажать «СУ»+«Q» (перестановка по кругу частей рабочего буфера). Эту операцию можно сравнить с поворотом барабана в револьвере, в то время как обмен буферов по клавише «КТ» аналогичен замене всего барабана целиком. Закончив редактирование, нужно вернуть части картинки в исходное положение («СУ»+«Q») и переслать изображение в смежный буфер («КТ»), после чего его можно просмотреть в уменьшенном виде («СУ»+«К») или распечатать («СУ»+«I»).

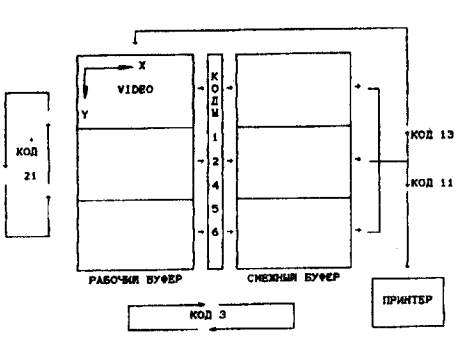


Рис. 1

Рассмотренный механизм перестановки частей рабочего буфера можно успешно применять и в других целях, не имеющих отношения к рисованию. Например, чтобы временно сохранять в памяти несколько программ длиной до 36000 байт каждая, организовав что-то наподобие электронного диска с «ручным управлением». Допустим, вы хотите сохранить игру, расположенную с адреса 1000 и имеющую длину 36000. Для этого надо войти в графический режим («АР2»+«СУ»+«U»), отключить курсор («АР2»+«:»), нормализовать экран («СБР») набрать адрес и длину: 1000А 36000Д, после чего переслать массив байтов на экран: 42000П. Теперь программу можно спрятать, «повернув барабан» («СУ»+«Q») или «заменив обойму» («КТ»). Таким способом вы можете сохранить до пяти

длинных программ, возвращая их по мере надобности на старое место. Нужно только записывать порядок заполнения буферов или различать программы по их виду в экранной памяти.

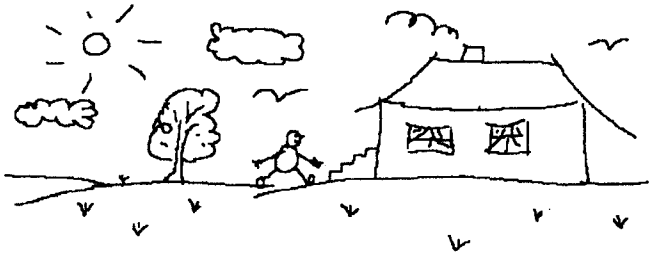


Рис. 2

Для редактирования графических изображений были бы полезны и коды, доставшимися «в наследство» от БК-0010: раздвижка и сдвижка строк, возврат строки, горизонтальная табуляция, а также буквенно-цифровые символы для ввода надписей. Но в режиме «ГРАФ» эти коды не работают, а в текстовом картинка будет засоряться служебными символами отладчика. Чтобы этого не происходило, можно воспользоваться простейшей кодовой программой эхо-печати всех вводимых с клавиатуры символов. Находясь в отладчике, наберите с любого удобного адреса три восьмеричных числа: 104006, 104016 и 775, после чего запустите эту программу. Теперь для редактирования изображений будут доступны все средства, предоставляемые базовым монитором БК. При определенном навыке можно, пользуясь вводимыми с клавиатуры редактирующими кодами, получать разнообразные изобразительные эффекты: накладывать одно изображение на другое, размножать части картинки, делать вырезку и вставку фрагментов и т. д. При этом NEWMON95 превращается в простейший графический редактор, позволяющий при сочетании программных построений с ручным творческим редактированием создавать интересные компьютерные композиции.

И наконец, несколько слов любителям модернизации. При создании NEWMON95 старая версия NEWMON была перекомпонована, так что внутренние адреса, приводившиеся в №5 за 1994 г., изменились. Резервными кодами теперь являются 0, 16 и 17. Вот их адреса:

```
114414    JMP @#114346    ; Код 0
114424    JMP @#114346    ; Код 16
114430    JMP @#114346    ; Код 17
```

Напомню, что переход JMP здесь указывает на команду возврата из подпрограммы RTS PC, находящуюся по адресу 114346. Вместо этого адреса в командах перехода можно поставить другие, указывающие точки входа в ваши подпрограммы. Адреса драйвера принтера в новой версии также изменены. Подпрограмма печати одной страницы смежного буфера находится по адресу 114564, подпрограмма передачи кода на принтер — 114746, резерв драйвера принтера (команды NOP) — 114752. Адрес же таблицы структур закрасок (117620) остался прежним. При создании NEWMON95 автор руководствовался принципом: побольше добавить, ничего не убавляя, т. е. вносить новые возможности при старом объеме памяти, за счет оптимизации других частей. В результате NEWMON95 оптимизирован до такой степени, что там просто «яблоку негде упасть». Тем не менее хотелось бы добавить кое-что еще, например операции обмена с дисководом. Пожертвовать для этой цели драйверами ТЛГ-канала (EMT 40—50)? Или лучше использовать сторонние ресурсы памяти? Автор обсудит эти вопросы со всеми, кого это заинтересует.

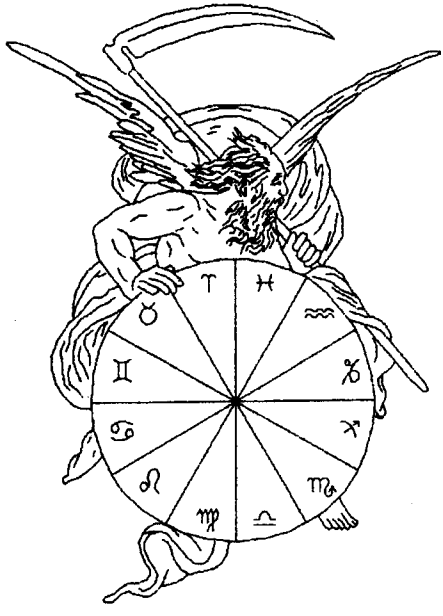
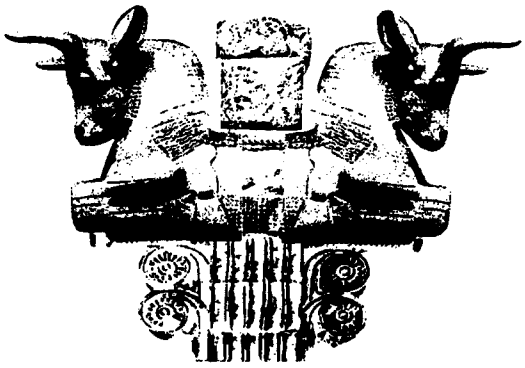


Рис. 3

По вопросам приобретения новой версии NEWMON95 можно обращаться в редакцию журнала «Персональный компьютер БК-0010 — БК-0011М». Справки по телефону: (095) 151-19-40. Письма с замечаниями и предложениями для автора NEWMON95 также следует направлять в редакцию по адресу: 125315, Москва, а/я 17.

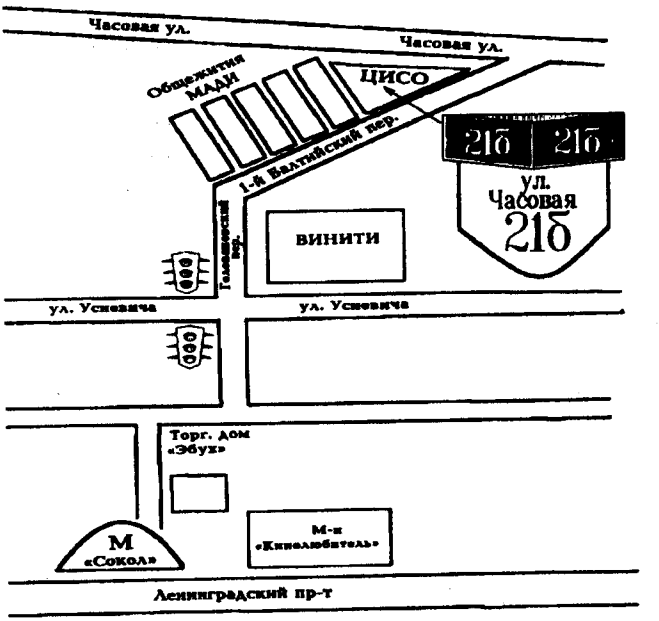


СОДЕРЖАНИЕ

ПЕРСОНАЛЬНЫЙ КОМПЬЮТЕР БК-0010 – БК-0011М

Наш адрес: Москва, ул. Часовая, 21Б, помещение Центра Интерактивных Средств Обучения (ЦИСО), комн. 36
Телефон: (095) 151-19-40

Как к нам добраться:



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

ПЕРСОНАЛЬНЫЙ КОМПЬЮТЕР
БК-0010 - БК-0011М

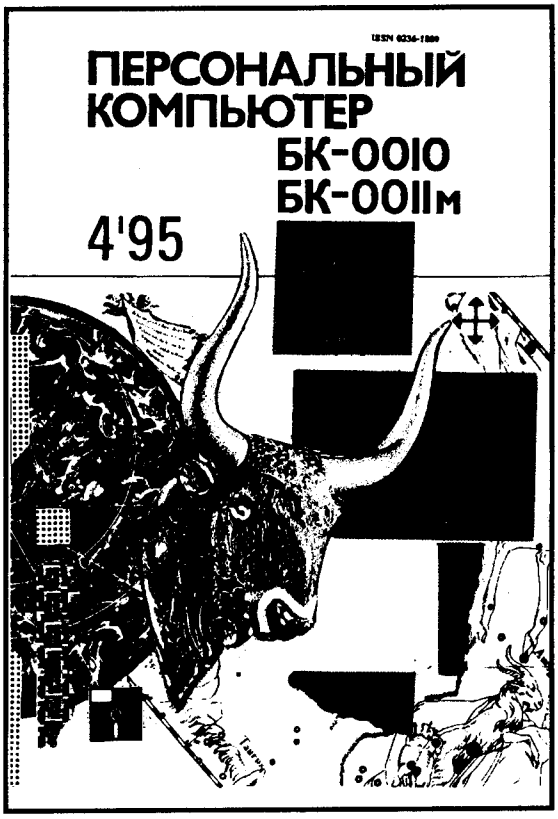
Главный редактор
Васильев Б. М.

Редактор
Усенков Д. Ю.

Корректор
Антонова В. С.

Компьютерная верстка
Лагунова Э. Е.

Подписано в печать с оригинал-макета издательства «Информатика и образование» 29.05.95. Тираж 1500 экз. Формат 70×100 1/16. Бумага офсетная. Усл.печ.л. 6,5. Заказ № 705. . Цена 4000 руб. (по подписке).
В розничной продаже цена договорная.
Ордена Трудового Красного Знамени Чеховский полиграфический комбинат Комитета Российской Федерации по печати.
142300, Чехов Московской обл.



Планируемое содержание выпуска 4'95

- Практикум программирования на ассемблере
- БК-0011М: руководство системного программиста
- Команды условного перехода в ассемблере
- Рекомендации пользователям БК-0010(.01): аппаратные доработки заводской конструкции
- Руководство по программированию музыкального сопроцессора AY-8910
- Справочный листок: микросхемы ПЗУ для БК

HARD & SOFT

- Математический сопроцессор для БК
- Дополнительный блок ОЗУ и ПЗУ для БК-0010.01
- Электронный диск для БК-0010.01
- Джойстик для БК
- Каталогизатор дискет для ANDOS

Есть идея

Начинающему пользователю

- Подключение дисковода к БК
- Что такое машинные коды
- О зависаниях
- Программные прерывания