КАК подписаться на «ПК БК»?

Почтовые отделения
уже прекратили прием подписки
на первое полугодие 1996 г.

НО

вы можете подписаться на журнал
«Персональный компьютер БК-0010 —БК-0011М»
непосредственно в редакции

Стоимость подписки на первое полугодие 1996 г. (без учета доставки) за три выпуска — 21 000 рублей (наличный расчет) или 36 000 рублей (безналичный расчет). Оформить подписку можно по каталожной цене с условием получения журналов в редакции (о выходе в свет очередного номера редакция сообщает подписчикам по телефону). Иногородним подписчикам журнал высылается по мере его выхода из печати. В последнем случае к указанной выше цене по каталогу добавляется стоимость пересылки из Москвы в пункт назначения трех бандеролей с журналом (эту сумму нужно узнать на почте) и 6000 рублей (орграсходы).

Оплату подписки можно произвести в редакции лично или почтовым переводом. Перевод на общую сумму (журналы, пересылка и орграсходы) нужно отправить по адресу: 125315, Москва, а/я 17, на имя редактора издательства Д. Ю. Усенкова. Одновременно по тому же адресу следует отправить письмо, где необходимо сообщить сумму перевода, количество экземпляров, а также разборчиво написать полный почтовый адрес. Оплата по безналичному расчету производится аналогично, в письмо нужно вложить копию платежного поручения.

Расчетный счет издательства: **0014-048122-002 в АКБ "Столичный" RUR корр. сч. 161706 в РКЦ ГУ ЦБ РФ г.Москвы МФО 201791.**

Адрес для переписки: **125315, Москва, а/я 17, Усенкову Дм. Юр.**

МАИ Международная Академия Информатизации

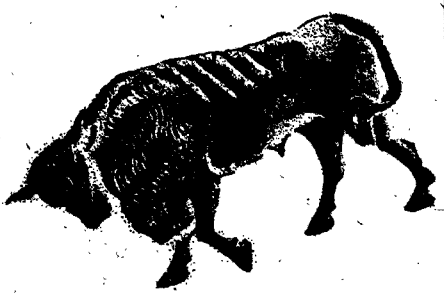
ПЕРСОНАЛЬНЫЙ КОМПЬЮТЕР БК-0010 БК-0011м

Компьютика

6'95

Издается с 1993 г.

В НОМЕРЕ



Издательство Компьютика
Москва 1995

Авторы выпуска

Вальяно Д.
Глускер А.
Кондратович М. И.
Котов Ю. В.
Кузнецов А. И.
Неробеев С. М.
Прудковский А. Г.
Рассказов Е. М.
Рожков С. П.
Сергеев Б. Н.
Серегин М.
Сорокин А. В.
Усенков Д. Ю.

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

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

ПЕРЕПЕЧАТКА МАТЕРИАЛОВ ТОЛЬКО С РАЗРЕШЕНИЯ
РЕДАКЦИИ ЖУРНАЛА

Почтовый адрес: 125315, Москва, а/я 17
E-Mail: mail@infoobr.msk.su

© Издательство «Компьютика», 1995 г.

Завершаем публикацию руководства по программированию на ассемблере БК-0011(М). Начало см. в №№4, 5 за 1995 г.

Руководство системного программиста БК-0011(М)
Базовая операционная система

Запросы драйвера экрана

Инициализация драйвера экрана .BTINI .
Макрокоманда: .BTINI
Расширение: JSR PC,@140130 (EMT 50)
Производится инициализация драйвера экрана, описанная для макрокоманды .BINIT

Установка режима драйвера экрана .BTSET .
Макрокоманда: .BTSET MODE
Расширение: MOV MODE,R0
JSR PC,@140132 (EMT 51)
Устанавливается режим работы экрана, соответствующий слову MODE. Формат слова приведен в приложении 3.

Чтение режима драйвера экрана .BTGET .
Макрокоманда: .BTGET
Расширение: JSR PC,@140134 (EMT 52)
В R0 помещается слово управления режимом драйвера экрана.

Установка цветов .BSCOL .
Макрокоманда: .BSCOL FORE,BACK
Расширение: MOV <BACK*400>+FORE,R0
JSR PC,@140136 (EMT 53)
Задаются цвета для символьного вывода на экран. Аргумент FORE задает код цвета символов и помещается в младший байт R0, BACK — код цвета фона, старший байт R0. Код, больший 200_8 означает сохранение текущего цвета. В зависимости от режима рассматриваются младшие один или два бита кода.

Чтение цветов .BGCOL .
Макрокоманда: .BGCOL
Расширение: JSR PC,@140140 (EMT 54)
В R0 помещается код текущих цветов, как описано для макрокоманды .BSCOL .

Установка/чтение палитры .BPAL .
Макрокоманда: .BPAL PALET
Расширение: MOVB PALET,R0
JSR PC,@140142 (EMT 55)

Устанавливается палитра, соответствующая аргументу PALET (0..15). Если PALET>200_8, то палитра не изменяется, а в R0 помещается код текущей палитры.

Переключение буферов экрана .BBUF .

```
Макрокоманда:   .BBUF WRITE,VIDEO
Расширение:     MOV #<WRITE*400>+VIDEO,R0
                JSR PC,@#140144 (EMT 56)
```

Устанавливаются буферы экрана для отображения (VIDEO) и записи (WRITE). Номер буфера отображения (0 или 1) помещается в младший байт R0, записи — в старший. Если бит 15 R0 установлен в «1», в R0 возвращается текущий код буферов.

Установка координат алфавитно-цифрового курсора .BSPOS .

```
Макрокоманда:   .BSPOS LINE,POS
Расширение:     MOV #<LINE*400>+POS,R0
                JSR PC,@#140146 (EMT 57)
```

Алфавитно-цифровой курсор устанавливается в указанной позиции заданной строки. Если номер строки или позиции в строке превышает допустимый, то соответствующая координата не изменяется.

Чтение координат алфавитно-цифрового курсора .BGPOS .

```
Макрокоманда:   .BGPOS
Расширение:     JSR PC,@#140150 (EMT 60)
```

В R0 помещается текущее положение курсора, аналогичное описанному для макрокоманды .BSPOS (младший байт — номер позиции, старший — номер строки).

Сдвиг экрана .BSCRL .

```
Макрокоманда:   .BSCRL NUM
Расширение:     MOV NUM,R0
                JSR PC,@#140152 (EMT 61)
```

Производится сдвиг изображения на указанное количество телевизионных строк. Положительное число означает сдвиг вверх, отрицательное — вниз.

Очистка экрана .BCLS .

```
Макрокоманда:   .BCLS
Расширение:     JSR PC,@#140154 (EMT 62)
```

Производится очистка экрана (заполнение цветом фона). Курсор устанавливается в левый верхний угол.

Вывод символа .BTOUT .

```
Макрокоманда:   .BTOUT CHAR
Расширение:     MOVB CHAR,R0
                JSR PC,@#140156 (EMT 63)
```

Производится вывод символа на экран (если символ печатаемый). Управляющие символы, интерпретируемые драйвером, описаны в приложении 2. Управляющие символы, отсутствующие в таблице, игнорируются. В режиме «БЛОК РЕД» все управляющие символы не обрабатываются, а выводятся в виде инверсных изображений латинских букв.

Вывод строки символов (формат RT-11) .BPRIN .

```
Макрокоманда:   .BPRIN ADDR
Расширение:     MOV ADDR,R0
                JSR PC,@#140160 (EMT 64)
```

Производится вывод на экран строки символов. Строка должна начинаться с адреса ADDR и заканчиваться нулевым байтом (курсор переводится в начало следующей строки) или 200 (курсор располагается за последним выведенным символом). После возврата в R0 находится адрес байта, следующего за конечным байтом выведенной строки.

Вывод строки символов .BSTR .

```
Макрокоманда:   .BSTR ADDR
Расширение:     MOV ADDR,R0
                JSR PC,@#140162 (EMT 65)
```

Производится вывод строки, заканчивающейся нулевым байтом. Курсор на новую строку не переводится.

Установка текущих графических координат .BSGRF .
Макрокоманда: .BSGRF X,Y
Расширение: MOV X,R0
MOV Y,R1
JSR PC,@140164 (EMT 66)
Устанавливаются текущие графические координаты, которые используются в других графических вызовах. Аргумент X — номер точки в строке, Y — задает номер строки растра. Верхняя левая точка имеет координаты (0,0), правая нижняя — (255,255) или (511,255) в зависимости от режима.

Чтение текущих графических координат .BGGRF .
Макрокоманда: .BGGRF
Расширение: JSR PC,@140166 (EMT 67)
В R0 и R1 помещаются текущие графические координаты в том же формате, что и для .BSGRF.

Установка графического цвета .BSCGR .
Макрокоманда: .BSCGR COLOR
Расширение: MOVB COLOR,R0
JSR PC,@140170 (EMT 70)
Устанавливается цвет для графического вывода. Параметр COLOR может принимать значения: 0 — черный, 1 — синий, 2 — зеленый, 3 — красный.

Чтение графического цвета .BGCGR .
Макрокоманда: .BGCGR
Расширение: JSR PC,@140172 (EMT 71)
В R0 помещается код текущего графического цвета.

Установка/чтение режима рисования .BSMOD .
Макрокоманда: .BSMOD MODE
Расширение: MOVB MODE,R0
JSR PC,@140174 (EMT 72)
Устанавливается режим рисования для всех графических вызовов. При MODE=0 устанавливается режим замещения текущим цветом, MODE=1 — режим инверсии содержимого буфера. При MODE=200_8 режим не меняется, а в R0 помещается код текущего режима рисования.

Вывод графической точки .BSPNT .
Макрокоманда: .BSPNT X,Y
Расширение: MOV X,R0
MOV Y,R1
JSR PC,@140176 (EMT 73).
На экран по заданным координатам выводится графическая точка текущего цвета. Указанные координаты (X,Y) становятся текущими.

Чтение цвета графической точки .BGPNT .
Макрокоманда: .BGPNT X,Y
Расширение: MOV X,R0
MOV Y,R1
JSR PC,@140200 (EMT 74)
В точке с заданными координатами определяется цвет и помещается в R0. Текущие координаты не изменяются.

Вывод графического вектора .BVECT .
Макрокоманда: .BVECT X,Y
Расширение: MOV X,R0
MOV Y,R1
JSR PC,@140202 (EMT 75)
Выводится графический вектор текущего цвета и текущего вида, начало которого находится в точке с текущими координатами, а конец — в заданной точке. Указанные координаты (X,Y) становятся текущими. Часть отрезка, выходящая за пределы экрана (или окна), не отображается.

Вывод закрашенного прямоугольника .BRECT .

Макрокоманда: .BRECK X,Y
Расширение: MOV X,R0
MOV Y,R1
JSR PC,@140204 (EMT 76)

Выводится прямоугольник, закрашенный текущим цветом, со сторонами, параллельными границам экрана. Один угол прямоугольника располагается в текущей точке, противоположный ему — в заданной. Указанные координаты (X,Y) становятся текущими.

Вывод дуги или окружности .BCIRC .

Макрокоманда: .BCIRC ADDR
Расширение: MOV ADDR,R0
JSR PC,@140206 (EMT 77)

Выводится окружность, дуга или сектор текущего цвета. Аргумент ADDR задает адрес блока параметров вызова. Слово 1 — координата X центра дуги (окружности), слово 2 — координата Y центра дуги (окружности), слово 3 — радиус дуги (окружности), слово 4 — начальный угол дуги, слово 5 — конечный угол, слово 6 — коэффициент сжатия по оси X, слово 7 — коэффициент сжатия по оси Y. Начальный и конечный углы задаются в соответствии с таблицей слева.

Аргумент	Значение угла (в градусах)
000000	0
.....	.....
077777	360

Если задано отрицательное значение, используется его абсолютная величина, а соответствующая граничная точка дуги соединяется с центром дуги отрезком прямой. Коэффициенты сжатия кодируются следующим образом:
0: нет сжатия;
1—177777: реальная координата = (вычисленная * коэффициент)/32767.

Закраска гранично-определенной области .BFILL .

Макрокоманда: .BFILL COLOR
Расширение: MOVB COLOR,R0
JSR PC,@140210 (EMT 100)

Производится закраска гранично-определенной 4-связной области текущим цветом начиная с текущей точки. Область должна быть ограничена одним заданным цветом (COLOR).

Выдача звука .BSOUN .

Макрокоманда: .BSOUND LENGHT,TONE
Расширение: MOV LENGHT,R0
MOV TONE,R1
JSR PC,@140212 (EMT 101)

Выдается звук с периодом TONE и длиной LENGHT периодов.

Установка графического окна .BSWIN .

Макрокоманда: .BSWIN AREA
Расширение: MOV AREA,R0
JSR PC,@140214 (EMT 102)

Производится установка границ окна для графического вывода. Аргумент AREA задает адрес области параметров.

Формат области:

```
.WORD MINX ; левая граница по X
.WORD MAXX ; правая граница по X
.WORD MINY ; верхняя граница по Y
.WORD MAXY ; нижняя граница по Y
```

Если какая-либо из границ задана вне экрана, вместо нее устанавливается граница экрана.

Чтение графического окна .BGWIN .

Макрокоманда: .BGWIN AREA
Расширение: MOV AREA,R0
JSR PC,@140216 (EMT 103)

По адресу, заданному параметром AREA, помещается область параметров, описанная в предыдущем пункте.

Установка типа вектора .BSTYP .

Макрокоманда: .BSTYP ARG
Расширение: MOV ARG,R0
JSR PC,@140222 (EMT 105)

Производится установка типа (шаблона) для рисования графических векторов. Используется для рисования прерывистых линий. Каждый бит аргумента задает признак рисования (или пропуска) двух точек, лежащих на векторе. После инициализации драйвера экрана устанавливается тип 177777_8, задающий рисование сплошных линий. Следует учитывать, что значение типа циклически сдвигается, поэтому рисование следующего вектора начинается не с начала шаблона, а с текущего положения.

Чтение типа вектора .BGTYP .
Макрокоманда: .BGTYP
Расширение: JSR PC,@140224 (EMT 106)
В R0 помещается текущий шаблон рисования векторов. Значение может быть циклически сдвинуто относительно заданного (см. предыдущий пункт).

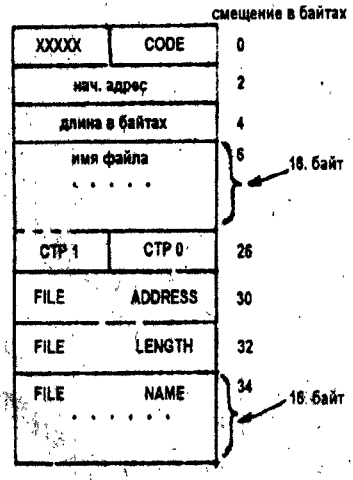
Установка и чтение ориентации графических символов .BGOSET .
Макрокоманда: .BGOSET ARG
Расширение: MOVB ARG,R0
JSR PC,@140226 (EMT 107)
Производится установка ориентации для графического вывода символов. ARG=0 означает нормальный вывод, 1 — повернутый на 180^o, 2 — на 90^o влево, 3 — на 90^o вправо. При ARG=200_8 производится чтение ориентации графических символов (код ориентации помещается в R0).

Графический вывод символа .BGOUT
Макрокоманда: .BGOUT CHAR
Расширение: MOV CHAR,R0
JSR PC,@140230 (EMT 110)
Производится графический вывод символа начиная с текущих графических координат в текущих режиме и ориентации текущим графическим цветом. Текущие координаты смещаются в соответствии с режимом и ориентацией для рисования следующего символа. Обрабатывается двойная ширина символов. В отличие от алфавитно-цифрового вывода графические символы могут быть нарисованы в любой точке экрана, однако рисование производится медленнее.

Запросы драйвера магнитофона

Управление двигателем .BMOT .
Макрокоманда: .BMOT ARG
Расширение: MOV ARG,R0
JSR PC,@140240 (EMT 114)
При ARG=1 осуществляется включение двигателя магнитофона, при ARG=0 — выключение.

Вызов формата БК-0010 .BMB10 .
Макрокоманда: .BMB10 AREA
Расширение: MOV AREA,R0
JSR PC,@140242 (EMT 115)



Выполняется операция с магнитофоном при использовании формата записи на магнитной ленте, совместимого с БК-0010. Параметр AREA — адрес области ОЗУ, через которую передаются параметры вызова.

Формат таблицы параметров вызова:

Поле CODE (байт) заполняется пользователем и содержит код операции: 0 — запись файла, 1 — чтение файла, 2 — чтение метки файла, 3 — фиктивное чтение файла.

Аргумент <начальный адрес> задает начальный адрес области ОЗУ, из которой выводится информация в файл, или начальный адрес области ОЗУ, куда производится ввод информации. Если в команде чтения задан адрес 0, информация записывается начиная с адреса, указанного в метке файла.

Аргумент <длина> используется только в операции записи файла и задает длину в байтах записываемой области ОЗУ.

Аргумент <имя файла> содержит 16-байтное имя файла, с которым необходимо произвести операцию. Если в запросе на чтение файла имя будет начинаться с нулевого байта, то будет загружен первый встреченный файл.

Аргументы <СТР 0> и <СТР 1> задают номера страниц ОЗУ (ПЗУ), подключаемых к окнам 40000—100000 и 100000—140000 при передаче данных (страницы памяти буфера ввода-вывода). Отрицательное содержимое соответствующих байтов означает текущее распределение памяти.

Следующая 20-байтная область служит для возврата информации о текущем файле при операциях чтения и фиктивного чтения. Она содержит начальный адрес, длину и имя, прочитанные из метки текущего файла ленты.

При ошибочном завершении операции возвращается установленный в «1» бит С и код ошибки в байте 52:

- 1 — найден файл с именем, не совпадающим с заданным (Filfnd);
- 2 — ошибка контрольной суммы (Chkerr);
- 3 — недопустимый диапазон адресов (Memerr);
- 4 — прерывание по вектору 4 или клавише «СТОП» (Stperr).

При ошибке типа 1 магнитофон останавливается после прочитанной метки файла.

Запросы драйвера принтера

Инициализация драйвера принтера .BPINI .
Макрокоманда: .BPINI
Расширение: JSR PC,@140260 (EMT 124)
Осуществляется инициализация драйвера принтера, описанная для макрокоманды .BINIT.

Установка режима печати .BPSET .
Макрокоманда: .BPSET MODE
Расширение: MOV MODE,R0
JSR PC,@140262 (EMT 125)
Устанавливается режим печати в соответствии со словом MODE. Назначение разрядов:

- бит 8: входная информация: «0» — КОИ-8, «1» — КОИ-7;
- бит 9: выходная информация: «0» — КОИ-8, «1» — КОИ-7;
- бит 10: «1» — КОИ-7Н2 (при КОИ-7 на входе).

Младший байт MODE содержит число квантов ожидания готовности принтера. Один квант соответствует одному полному циклу (65536 раз) команды SOB. При равенстве нулю время ожидания бесконечно.

Чтение режима печати .BPGET .
Макрокоманда: .BPGET
Расширение: JSR PC,@140264 (EMT 126)
В R0 помещается слово режима печати (см. предыдущий пункт).

Вывод символа на печать .BPOUT .
Макрокоманда: .BPOUT CHAR
Расширение: MOVB CHAR,R0
JSR PC,@140266 (EMT 127)
Осуществляется вывод символа на печать. Если печать не выдает готовности за установленное время тайм-аута, возвращается управление с установленным битом С.

Определение готовности принтера .BPRDY .
Макрокоманда: .BPRDY
Расширение: JSR PC,@140270 (EMT 130)
Если принтер находится в состоянии готовности, в R0 помещается 1, а бит С сбрасывается в 0, иначе R0=0, C=1.

Чтение текущей позиции печати .BPPOS .
Макрокоманда: .BPPOS
Расширение: JSR PC,@140272 (EMT 131)
В R0 помещается текущая позиция печатающей головки. Пересчет при выводе символов табуляции не производится.

Запросы драйвера НГМД

Драйвер НГМД* размещен в ПЗУ, находящемся в контроллере НГМД, и доступен при подключении контроллера к микроЭВМ. Драйвер имеет несколько точек входа:

* Описание прошивки ПЗУ драйвера КНГМД с его ассемблерным дампом опубликовано в №5 за 1994 г. — *Прим. ред.*

Адрес точки входа	Функция
160000	Автоматическая загрузка
160002	Загрузка с выбранного привода
160004	Чтение-запись по номеру блока
160006	Чтение-запись по номеру сектора
160010	Инициализация рабочей области драйвера
160012	Форматирование дорожки

Примечание: здесь и далее числа без последующей точки — восьмеричные, с точкой — десятичные.

Для работы драйвера необходима область ОЗУ размером 66 байт для размещения параметров и рабочих ячеек.

Формат рабочей области:

Смещение от начала области	Длина (байт)	Символическое имя	Начальное значение	Назначение
0	2	CSRW	0	Копия по записи регистра состояния КНГМД
2	2	CURTRK	—	Адрес текущей дорожки
4	4	TRKTAB	177777 ,177777	Таблица текущих дорожек
10	2	TDOWN	20000 (для 6022)	Время опускания головки (в циклах команды SOB)
12	2	TSTEP	-"-	Время перехода с дорожки на дорожку
14	1	TRKCOR	36 (для 6022)	Номер дорожки, с которой включается предкомпенсация при записи
15	1	BRETRY	30 (рекоменд.)	Число повторных операций при ошибках
16	1	FLAGS	0	Рабочая ячейка драйвера
17	1	FILLB	Вх. параметр	Код заполнения секторов при форматировании дисков
20	2	FLGPTR	—	Указатель на байт признаков
22	4	FLGTAB	0	Таблица признаков (см. ниже)
26	2	ADDR	Вх. параметр	Адрес буфера ОЗУ
30	2	WCNT	-"-	Число пересылаемых слов
32	1	SIDE	-"-	Номер стороны диска
33	1	TRK	-"-	Номер дорожки
34	1	UNIT	-"-	Номер привода
35	1	SECTOR	-"-	Номер сектора
36	22	—	-"-	Рабочие ячейки драйвера
60	2	MAXSEC	Вх. параметр	Число секторов на дорожке
62	4	—	-"-	Рабочие ячейки

Таблица FLGTAB содержит по одному байту на привод и используется для обработки нестандартных дисков. В каждом байте задействовано два бита для указания параметров операции:

- бит 0 — при установке в «1» драйвер производит пропуск дорожек при поиске, т. е. фактически обрабатываются только четные дорожки. Эта возможность используется для обработки на 80-дорожечном дисководе 40-дорожечных дискет;
- бит 1 — при установке в «1» драйвер производит обращение только к нижней стороне дискеты. Эта возможность используется при подключении односторонних дисководов или при обработке односторонних дискет на двустороннем дисководе;
- бит 2 — при равенстве «0» драйвер обрабатывает сектора длиной 512 байт, иначе — длиной 1024 байт. (Внимание! Работа с секторами 1024 байт в драйвере «326-й прошивки» реализована неверно. — *Прим. ред.*)

Автоматическая загрузка.

Вызов: JSR PC,@#160000

Можно также произвести загрузку непосредственно из монитора командой «В». Входные параметры отсутствуют.

Производится попытка загрузить операционную систему с привода 0. Если попытка загрузки была неудачной, то драйвер последовательно пытается загрузить систему с привода 1, 2 и 3 (для БК-0011М — только с привода 1). Если все попытки оказались неудачными, драйвер возвращает управление с установленным в «1» битом С.

Алгоритм загрузки:

1) инициализируется рабочая область драйвера в ОЗУ (базовый адрес 2000) с параметрами для дисковода 6022;

2) считывается в ОЗУ начиная с адреса 1000 первый сектор нулевой дорожки выбранного дисковода;

3) в случаях, когда в приводе отсутствует дискета, не закрыта крышка привода, привод неисправен, дискета не размечена, при считывании указанного сектора возникла ошибка или на диске нет загрузчика операционной системы, попытка загрузки считается неудачной. Иначе драйвер передает управление по адресу 1000, при этом в R0 помещается номер привода, откуда произведена загрузка.

Загрузка с выбранного привода.

Вызов: MOV #N,R0
JMP @#160002

Из монитора загрузка производится командой <N>B, где <N> — номер выбранного привода. Действие аналогично предыдущему пункту, однако чтение начинается сразу с выбранного привода и при неудаче попыток загрузиться с других приводов не производится.

Чтение-запись логического блока.

Вызов: JSR PC,@#160004

Входные параметры:

- R0 — номер блока на диске,
- R1 — длина пересылаемого массива данных,
- R2 — начальный адрес массива данных,
- R3 — базовый адрес рабочей области драйвера,
- поле UNIT в рабочей области — номер привода.

Блок является логической единицей информации на внешних носителях и равен 512 байтам. В основном используемом формате записи на диск размер блока совпадает с размером сектора. Первый сектор нижней стороны нулевой дорожки соответствует блоку 0. Вычисление номера сектора и дорожки по номеру блока производится с учетом установленных признаков в таблице FLGTAB. Алгоритм этого вычисления совпадает с алгоритмом драйвера MY для микроЭВМ серии ДВК.

Длина массива данных задает количество двухбайтных слов, подлежащих пересылке, а также тип операции: положительное число — чтение, отрицательное — запись. При записи пересылаемое количество слов равно абсолютному значению длины. Если заданное количество превышает размер сектора, осуществляется автоматический переход к следующему сектору, а если необходимо, то на другую дорожку и сторону. Если при записи количество слов не кратно размеру сектора, то остаток последнего сектора заполняется нулями.

Перед самым первым вызовом драйвера поля CSRW, TRKTAB и FLAGS в рабочей области должны быть заполнены значениями, указанными в таблице, и не должны изменяться программой пользователя. Поля TDOWN, TSTEP, TRKCOR, BRETRY и FLGTAB рекомендуется заполнять значениями, указанными в таблице, однако они могут быть изменены в зависимости от конкретных условий*. Возврат из драйвера осуществляется командой RTS PC. Содержимое регистров процессора R0, R1, R2, R4, R5 **не сохраняется**.

При возникновении ошибки бит С в слове состояния процессора устанавливается в 1, а в байт с абсолютным адресом 52 помещается код ошибки.

Код	Причина ошибки
1	при чтении — ошибка CRC в зоне данных, при записи — установлена защита записи
2	ошибка CRC в зоне заголовка сектора
3	нет позиционирования на 0 дорожку

* Большинство дисковых операционных систем для БК (например, ANDOS или MKDOS) содержат в себе таблицы оптимальных значений констант для различных видов дисководов и позволяют менять эти константы или выбирать их, указывая название дисковода (например, как в программе Disk Doctor). — *Прим. ред.*

Код	Причина ошибки
4	ошибка позиционирования
5	не найден сектор
6	нет диска или диск не вращается
7	прерывание по вектору 4 при выполнении операции
10	не найден адресный маркер
11	не найден маркер данных
12	недопустимые параметры команды или нестандартный формат диска

После завершения операции драйвер оставляет двигатель дисковода включенным. Для остановки двигателя программа пользователя должна записать 0 по адресу 177130, а также в ячейку CSRW рабочей области.

Чтение-запись по номеру сектора и дорожки.
Вызов: JSR PC, @#160006
Входные параметры:
- R3 — адрес рабочей области,
- поля рабочей области:
  ADDR — начальный адрес массива данных,
  WCNT — длина массива данных,
  SIDE — номер стороны (0 — нижняя, 1 — верхняя),
  TRK — номер дорожки,
  UNIT — номер привода (0..3),
  SECTOR — номер сектора ($1..12_8$).

Действия аналогичны предыдущему пункту.

Инициализация рабочей области. Эта точка входа является вспомогательной и служит для облегчения написания программ, использующих драйвер НГМД.
Вызов: JSR PC,@#160010
Входной параметр:
- R3 — базовый адрес рабочей области.

Подпрограмма заполняет следующие поля рабочей области: CSRW = 0; TRKTAB = 177777, 177777; TDOWN = 10000.; TSTEP = 10000.; TRKCOR = 30.; BRETRY = 30.; FLAGS = 0; FLGTAB = 0, 0.

Форматирование дорожки.
Вызов: JSR PC,@#160012
Входные параметры:
- R3 — базовый адрес рабочей области,
- поле FILLB — код, которым будет заполнена зона данных сектора при форматировании,
- поле SIDE — номер стороны,
- поле TRK — номер дорожки,
- поле UNIT — номер привода.

Форматируется указанная дорожка с указанной стороны, зона данных всех секторов заполняется кодом FILLB.

Приложение 1. *Кодировка клавиш БК-0011(М): управляющие клавиши*

Клавиша	Прямой режим		Перекодировка	
		«АР2»		«АР2»
«ТАБ»	11	11	11	11
«КТ»	3	203	3	203
«↔»	213	213	33, 144	33, 144
«⇤»	26	226	33, 145	226
«↦»	27	227	27	227
«ИНД СУ»	202	202	33, 25	33, 25
«БЛОК РЕД»	Код в программу не передается			
«ШАГ»	220	220	33, 10	33, 10

Клавиша	Прямой режим		Перекодировка	
		«АР2»		«АР2»
«СБР»	14	214	14	214
«↤» («ЗАБОЙ»)	30	230	177	30
«ВС»	23	223	23	33
«↵»	12	212	15	15*
«↑»	32	232	33, 101	33, 121
«↓»	33	233	33, 102	12
«←»	10	210	33, 103	33, 120
«→»	31	231	33, 104	33, 122

Примечание. * — в режиме дополнительной клавиатуры выдается код 33, 77, 115.

Коды функциональных клавиш в режиме перекодировки и дополнительной клавиатуры

Клавиша+«АР2»	«0»	«1»	«2»	«3»	«4»
Код	33, 77, 160	33, 77, 161	33, 77, 162	33, 77, 163	33, 77, 164
«5»	«6»	«7»	«8»	«9»	«.»
33, 77, 165	33, 77, 166	33, 77, 167	33, 77, 170	33, 77, 171	33, 77, 156

Приложение 2. *Управление экраном (управляющие символы драйвера экрана)*

Код	Назначение
7	Звонок
10	Возврат на шаг (курсор влево)
11	Горизонтальная табуляция
12	Перевод строки (курсор вниз, сдвиг экрана вверх)
14	Перевод формата (пропуск 10 строк)
15	Возврат каретки (курсор в начало строки)
16	Переход на русский регистр (в КОИ-7)
17	Переход на латинский регистр (в КОИ-7)
33	ESCAPE (начало командной последовательности)

Командные последовательности

Команда	Назначение
ESC 0 (033, 060)	Устанавливает режим 32 символа в строке
ESC 1 (033, 061)	Устанавливает режим 64 символа в строке
ESC 2 (033, 062)	Устанавливает режим 80 символов в строке
ESC 3 (033, 063)	Двойная ширина символов
ESC 4 (033, 064)	Одинарная ширина символов
ESC 5 (033, 065)	Двойная высота символов (верхняя половина)
ESC 6 (033, 066)	Двойная высота символов (нижняя половина)
ESC 7 (033, 067)	Отмена двойной высоты
ESC 8 (033, 070)	Отображение курсора
ESC 9 (033, 071)	Запрет курсора
ESC : (033, 072)	Установка палитры*
ESC ; (033, 073)	Установка цветов**
ESC = (033, 075)	Режим дополнительной клавиатуры
ESC > (033, 076)	Сброс режима дополнительной клавиатуры
ESC @ (033, 100)	Раздвижка символов от курсора вправо
ESC A (033, 101)	Курсор вверх (в первой строке игнорируется)

Команда	Назначение
ESC B (033, 102)	Курсор вниз (в последней строке игнорируется)
ESC C (033, 103)	Курсор вправо (зависит от режима автопереноса)
ESC D (033, 104)	Курсор влево (зависит от режима автопереноса)
ESC E.(033, 105)	Очистка экрана
ESC H (033, 110)	Курсор в начало экрана
ESC I (033, 111)	«Обратный ПС» (сдвиг экрана вниз)
ESC J (033, 112)	Стирание от курсора до конца экрана
ESC K (033, 113)	Стирание от курсора до конца строки
ESC L (033, 114)	Раздвижка строк от курсора вниз
ESC M (033, 115)	Сдвижка строк к курсору вверх (строка, в которой находится курсор, исчезает)
ESC N (033, 116)	Подчеркивание символов
ESC O (033, 117)	Отмена подчеркивания символов
ESC P (033, 120)	Сдвижка символов к курсору влево
ESC Q (033, 121)	Курсор в начало следующей строки
ESC R (033, 122)	Курсор в начало текущей строки
ESC T (033, 124)	Инверсия символов
ESC U (033, 125)	Отмена инверсии
ESC Y (033, 131)	Прямая адресация курсора***
ESC Z (033, 132)	Идентификация****

Примечания.

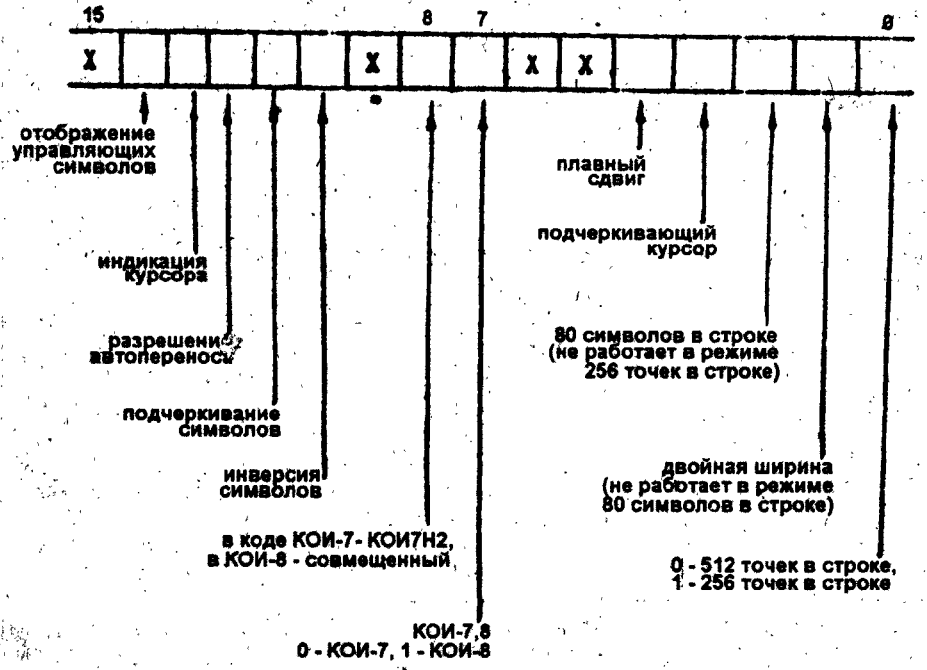
* — после команды должен следовать байт, младшие четыре бита которого указывают код палитры.

** — после команды должны следовать два байта, указывающие цвета фона и символов.

*** — после команды должны следовать коды строки и позиции в строке. (Пример: код 40 соответствует левой позиции в верхней строке.)

**** — в ответ на эту команду в драйвер клавиатуры передается последовательность ESC K.

Приложение 3. *Формат слова управления режимом экрана*



Продолжаем публикацию материалов по практическому освоению компьютера. Начало см. в №4, 5 за 1995 г.

Б. Н. Сергеев,
Москва

Практикум на ассемблере

Апостроф и команда BR

Вывод символов на экран с помощью команды EMT 16 на предыдущих занятиях мы делали так:

```
MOV #<код символа>,R0
EMT 16
```

Так как подобное действие требуется в программах достаточно часто, в ассемблер-трансляторе предусмотрена возможность упрощения записи — вместо кода пишут сам символ, вставляя перед ним знак апострофа («'»)* (клавиша 7 по нижнему регистру).

Пример

```
MOV #'Д,R0    EMT 16   ;1; вывод
MOV #'в,R0    EMT 16   ;2; слова
MOV #'а,R0    EMT 16   ;3; "Два"
.END                   ;4;
```

Последовательность исполнения этой программы по строкам: 1—2—3—4. Вставим после строки 1 команду BR N:

```
N:  MOV #'Д,R0    EMT 16   ;1;
    BR N                   ;2;
    MOV #'в,R0    EMT 16   ;3;
    MOV #'а,R0    EMT 16   ;4;
    .END                   ;5;
```

Последовательность выполнения: 1—2—1—2—1—2 и т. д. (бесконечный цикл). Изменим команду BR N на BR M:

```
N:  MOV #'Д,R0    EMT 16   ;1;
    BR M                   ;2;
    MOV #'в,R0    EMT 16   ;3;
M:  MOV #'а,R0    EMT 16   ;4;
    .END                   ;5;
```

Последовательность выполнения стала такой: 1—2—4—5. Изменим BR M на NOP («нет операции»):

```
N:  MOV #'Д,R0    EMT 16   ;1;
    NOP                    ;2;
    MOV #'в,R0    EMT 16   ;3;
M:  MOV #'а,R0    EMT 16   ;4;
    .END                   ;5;
```

Последовательность работы: 1—2—3—4—5.

Четыре приведенных выше листинга являются кроме примера использования апострофа для записи символов также демонстрацией безусловных переходов по команде BR. Первая программа выводит на экран слово «Два». Вставив команду BR N после первой строки и затем заменяя ее на другие, мы наблюдаем соответствующие изменения в поведении программы. Так, ее второй вариант непрерывно печатает букву «Д», а остальные символы не используются совсем (для останова нажмите «СТОП»). Третий листинг печатает слово «Да», а четвертый, с командой NOP, выводит на экран слово «Два», как и первый.

Команда NOP, которой мы только что воспользовались для «отмены» перехода BR, может применяться для изменения хода исполнения программы во время ее отладки. Вписав NOP вместо какой-либо команды, мы как бы удаляем ее из листинга, не затрагивая остальные строки. Теперь, запустив исправленную программу, можно понаблюдать последствия доработки и сделать выводы о назначении удаленной команды. Для восстановления прежнего текста надо сделать обратную замену, т. е. вместо NOP вписать ранее удаленную команду. А чтобы не забыть, какой она была, лучше всего переписать ее на бумагу.

---

* Напоминаем, что в данной статье в качестве ассемблер-транслятора используется Turbo. В других трансляторах символ в операторе MOV нужно ограничивать апострофами с обеих сторон, например: MOV 'a',R0. — *Прим. ред.*

В ассемблер-трансляторе Turbo то же достигается более удобным способом — команда не удаляется совсем, а временно «отключается», для чего достаточно в начале соответствующей строки вставить символ «;» («ТОЧКА С ЗАПЯТОЙ»). Тем самым строка превращается в КОММЕНТАРИЙ, который при трансляции игнорируется. Но в то же время строка остается на экране и мы ее уже не забудем.

Для вставки в Turbo дополнительной команды в середину листинга можно раздвинуть строки нажатием клавиши «ВВОД». В отладчике OTL16 такой возможности нет, а в MIRAGE есть (об этом мы поговорим позже).

Коды клавиш

Каждая клавиша БК имеет свой код, который может изменяться в зависимости от режима «РУС»/«ЛАТ», «ЗАГЛ»/«СТИР», а также если одновременно с клавишей нажать «АР2», «СУ» или «НР». Следующая программа позволяет увидеть коды нажимаемых клавиш (в различных комбинациях).

```
      MOV #14,R0     EMT 16          ; очистим экран          ;1
      MOV #T1,R0     CLR R2   EMT 20 ; вывод текста T1        ;2
      MOV #12,R0     EMT 16          ; перевод строки         ;3
N:    EMT 6          EMT 16          ; вывод нажатой клавиши  ;4
      MOV #T2,R1     CLR R2   EMT 20 ; вывод текста T2        ;5
      MOV R0,R4      CAL W8          ; вывод содержимого R0   ;6
      MOV #12,R0     EMT 16          ; перевод строки         ;7
      BR N                           ;                        ;8
      HALT
T1:   .ASCIZ/Нажимайте клавиши!/
T2:   .ASCIZ/=/
      .EVEN
; --------- подпрограмма "163220" - вывод восьмеричного числа
W8:   MOV #6,R5    CLR R0
1:    ROL R4       ROL R0    ADD #60,R0    EMT 16    DEC R5
      BEQ 2        CLR R0    ROL R4        ROL R0    ROL R4
      ROL R0       BR 1
2:    MOV #40,R0   EMT 16    RET
      .END
```

Рассмотрим листинг более подробно. Строка 4 нам уже знакома — она печатает только что введенный символ. Строка 6 пересылает код нажатой клавиши из регистра R0, в который он попадает после EMT6, в R4. А оттуда его извлекает подпрограмма W8 и выводит на экран в восьмеричном виде.

Подпрограмма W8 выводит на экран содержимое регистра R4 — пока это все, что нам надо о ней знать. Разбирать ее здесь не будем. Придумал ее тоже не автор данной статьи — она прошита в ПЗУ БК по адресам 163220—163226, и ее можно увидеть в отладчике, подав команды 163220A ЗАПЯТАЯ ... (если только у вас контроллер дисковода с дополнительным ОЗУ не на 32 кб, а на 16 или меньше).

В данной программе появилась новая для нас макрокоманда .EVEN . Обсудим ее назначение. Если посмотреть таблицу клавиш или запустить только что набранную программу, вы увидите, что код любой клавиши занимает один байт. Для записи последовательности символов в память используется цепочка байтовых ячеек. Процессор читает эти байты один за другим, тогда как обычно он работает с целыми словами.

В процессе чтения кодов команд ЦП отрабатывает такую последовательность действий: перейдя к очередному адресу, он вначале УВЕЛИЧИВАЕТ программный счетчик (регистр R7, или PC) на 2, а затем выполняет считанную команду. Поэтому адрес очередной команды в PC всегда четный. При чтении же байтов PC увеличивается на 1. Если хранящаяся в памяти строка текста состоит из ЧЕТНОГО числа символов (байтов), то после считывания последнего символа PC останется четным и после перехода в режим чтения по словам ЦП нормально продолжит свою работу. Если же количество прочитанных символов окажется НЕЧЕТНЫМ, то очередной адрес, записанный в PC, также будет нечетным. Процессор приступит к выполнению очередной словной команды с «испорченного» начала, и работа программы окажется нарушенной.

Для предотвращения такого беспорядка в конце строки с нечетным количеством символов надо дописать макрокоманду .EVEN . Она проверяет четность количества байтов в строке и добавляет к ней один или два завершающих нулевых байта, всегда приводя хранящийся в PC адрес к четному.

В качестве домашнего задания поэкспериментируйте с приведенным выше листингом:

— изменяйте текст, делая количество символов в нем четным или нечетным;

— «отключите» команду .EVEN (вставив перед ней «;») и снова попробуйте менять количество символов в тексте.

Объясните полученный результат.

Регистры общего назначения

Вспомните свою самую первую программу:

```
MOV #270,R0 EMT 16 .END
```

Заново наберите ее в Turbo (или загрузите с диска), оттранслируйте, перейдите в отладчик и просмотрите в нем (1000A ЗАПЯТАЯ и т. д.):

```
001000 MOV #270,R0    012700 300.25 ю
                      000270 270.0  ♦
001004 EMT 16         104016 16.210
001006 HALT           000000 0.0
```

Подайте команду «R» — на экране появится сводная таблица — справка по информации, записанной в регистры общего назначения (РОН).

```
R0=000000
R1=000000
R2=000000
R3=000000
R4=000000
R5=000000
R6=000740   <- Указатель стека      (SP)
R7=000000   <- Программный счетчик  (PC)
NZVC=0000   <- Флаги условий (слово состояния процессора, или ССП)
```

Эти регистры обычно можно использовать при написании программ по своему усмотрению

В регистры R0...R7 числа могут быть записаны только в восьмеричном виде (пример — команда MOV #270,R0).

Программный счетчик (PC) показывает адрес команды программы, которая будет выполнена на следующем шаге. Запись значений в него обычно производит сам центральный процессор.

Четыре флага-признака условий N, Z, V и C в ССП также выставляются самим процессором в соответствующие позиции числа, в данном случае обозначенные четырьмя нулями, и могут принимать только значения «0» или «1». Если флаг-признак равен «1», то он называется «УСТАНОВЛЕННЫМ», иначе — «СБРОШЕННЫМ». Это не просто числа, а ПРИЗНАКИ, сообщающие процессору (и программисту) о происходящих в программе событиях. В принципе, вместо «0» и «1» можно было бы написать любые символы, лишь бы это было понятно нам (хотя, как вы уже знаете, сам процессор «понимает» только битовые числа «0» и «1»). Так, в отладчике MIRAGE вместо «1» выводится буквенное обозначение того или иного флага, например, если признак N установлен, то в соответствующую позицию обозначения ССП записывается символ «N», а если сброшен, то символ «–».

Что же касается назначения битов ССП, то флаг N (от слова «negative» — «отрицательный») устанавливается в «ЕДИНИЦУ», если результат последней выполненной команды программы имеет ОТРИЦАТЕЛЬНОЕ значение. В противном случае флаг N равен нулю.

Флаг Z («zero» — «нуль») устанавливается в ЕДИНИЦУ, если результат последней выполненной команды равен НУЛЮ. В противном случае Z = 0. (Обратите внимание на кажущееся противоречие: Z=1 при нулевом результате и наоборот.)

Флаги V, C и регистр SP мы пока трогать не будем — уровень наших знаний для работы с ними еще недостаточен.

Запустим рассмотренную выше программу командой 1000Г. На экране появится символ «♦», произошли какие-то изменения и в таблице РОН. Чтобы не спеша проследить, что делается в регистрах при выполнении каждой строки программы, запустим ее в ПОШАГОВОМ режиме, когда после выполнения очередной строки ее работа приостанавливается до подачи нами разрешения на выполнение следующей.

Подавая команды пошагового исполнения по одной: 1000Ш Ш Ш ..., понаблюдайте за изменениями, которые будут происходить в поле мнемоники и в содержимом регистров.

А теперь расшифруем смысл команды пошагового исполнения. Запись 1000Ш указывает процессору номер строки, которую надо выполнить первой и на время остановиться. Если ввести «Ш» без указания адреса, то ЦП первой выполнит команду, адрес которой

указан в регистре PC (R7). Но там может находиться число, не имеющее никакого отношения к нашей программе. Поэтому мы даем команду полностью: 1000Ш. После ее выполнения содержимое PC изменилось на 1004, это говорит о том, что на следующем шаге будет выполняться команда с адресом 1004.

Посмотрим на выведенные в таблице регистров значения. В R0 появилось число 270 — результат действий команды MOV #270,R0.

Для выполнения следующего шага теперь достаточно дать команду Ш без указания адреса. Процессор «заглянет» в R7 (счетчик команд), «увидит» там адрес очередной команды программы (1004) и выполнит ее.

Снова нажмем клавишу «Ш» — R7 принял значение 1006, а на экране появился символ «бубна» (это сделала команда EMT 16).

Еще раз «Ш» — в R7 оказалось значение 1010, а в поле мнемоники выдано слово HALT. Пошаговый прогон программы закончен.

Для лучшего понимания причин изменения содержимого РОН попробуйте написать и просмотреть в пошаговом режиме такую «тренировочную» программу:

```
MOV #222, R0
MOV R0,R1
CLR R0
MOV R1,R2
CLR R1
MOV R2,R3
ADD #2,R3
MOV R3,R4
SUB R3,R4
SUB R3,R2
ADD #2,R2
SWAB R3
INC R3
DEC R2
...
```

а дальше придумайте что-нибудь свое.

Начать ввод команд можно с любого адреса, но его же нужно указать и в первой команде <адрес>Ш. При просмотре наблюдайте за изменениями в РОН и постарайтесь найти им объяснения. Особое внимание уделите флагам слова состояния процессора — это один из фундаментальных аспектов в понимании алгоритма работы программы.

* * *

Теперь рассмотрим еще один вариант программы вывода кодов клавиш. Здесь хорошо видно изменение содержимого R0. Полученный результат можно сличить с таблицей кодов БК.

```
1000 MOVB #41,R0
1004 EMT 16
1006 INCB R0
1010 BR 1004
```

Эта программа печатает символ в поле мнемоники, а код этого символа можно видеть в R0. Это, конечно, менее удобно, чем мы ранее делали в Turbo, но зато сама программа намного проще.

Модернизируем эту программу и напишем ее в Turbo:

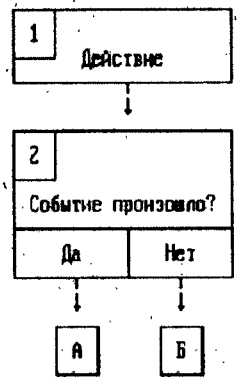
```
; Вывод печатаемых символов (с кодами 41-177, 240-377)
     MOV #41,R0      ; код первого символа              ;1
M1:  EMT 16          ; вывести на экран                 ;2
     INC R0          ; увеличить код на 1               ;3
     CMP #200,R0     ; это код 200?                     ;4
     BEQ M2          ; если ДА, то перейти к коду 240   ;5
     CMP #400,R0     ; это код 400 (последний)?         ;6
     BEQ END         ; если ДА, закончить вывод         ;7
     BR M1           ;                                  ;8
M2:  MOV #240,R0     ; код символа                      ;9
     BR M1           ;                                  ;10
END: .END            ;                                  ;11
```

Запустим ее. На экран выводятся ВСЕ печатаемые символы с кодами от 41 до 177 и от 240 до 377. Коды 200—237 не используются умышленно, так как они не имеют «печатаемого» отображения. Если же все-таки попробовать вывести их на экран, то эффект будет... Попробуйте сами! Для этого «отключите» строки 4 и 5, поставив перед ними в поле меток (по левому краю) символ «;».

Запишем эту программу на диск (на всякий случай) командами ST и SA (т. е. исходный текст и оттранслированный в коды) и перейдем в отладчик. Запустим ее в пошаговом режиме и проследим изменения в РОН и ССП.

Когда в R0 появится код 200, команда в строке 5 обнаружит это по установленному в «1» флагу Z и изменит ход действий в нашей программе так, чтобы коды 200—237 в R0 не попадали. Аналогично задача решается и при появлении в R0 кода 400.

Ветвления



Как мы увидели выше, значение флагов ССП используется для управления работой программы в зависимости от результата предыдущего действия. Такое изменение дальнейшего хода вычислений называется «условным ветвлением». Схематично это представлено на рисунке слева. Здесь цифра «1» обозначает шаг программы, в котором ВСЕГДА совершается какое-либо действие, результат которого ВСЕГДА сопровождается выдачей конкретных значений флагов ССП. Цифра же «2» показывает шаг, в котором решается вопрос о результатах предыдущего действия, и, в зависимости от значений флагов, ход программы направляется по ОДНОМУ ИЗ ДВУХ возможных путей (А или Б).

Рассмотренная ситуация очень часто встречается и в обычной жизни. Например, вы спешите в школу, институт или на работу, подходите к автобусной остановке и смотрите: виден ли «на горизонте» автобус (произошло ли ожидаемое событие). Если ДА, то вы, расталкивая локтями конкурентов (или не расталкивая — в зависимости от воспитанности), лезете в салон (вариант А). Если НЕТ — подозреваете, что он может подойти недопустимо поздно, и быстрым шагом топаете на трамвай (вариант Б).

Или другой пример. Вы — водитель автомашины. Подъезжая к перекрестку, вы смотрите на светофор: «Зеленый горит?» Если ДА, то проезжаете, не снижая скорости, а если НЕТ — жмете на педаль тормоза и останавливаетесь в ожидании разрешения проезда*.

Разберем несложный пример (наберите приведенный ниже листинг в Turbo):

```
; Условное ветвление.
      A=1                  ; прямое присваивание        ;1
      MOV #A,R0            ; число А заносится в R0     ;2
      SUB #3,R0            ; A - 3 -> R0                ;3
      BLT M                ; если меньше 3              ;4
      BEQ R                ; если равно 3               ;5
      BGT B                ; если больше 3              ;6
M:    MOV #TM,R1   CLR R2   EMT 20   HALT                ;7
R:    MOV #TR,R1   CLR R2   EMT 20   HALT                ;8
B:    MOV #TB,R1   CLR R2   EMT 20   HALT                ;9
TM:   .ASCIZ/Заданное число меньше 3./                   ;10
TR:   .ASCIZ/Заданное число равно 3./                    ;11
TB:   .ASCIZ/Заданное число больше 3./                   ;12
      .END
```

Здесь опять встретилась новая для нас запись в строке 1 — ПРЯМОЕ ПРИСВАИВАНИЕ. Переменной А здесь присваивается значение 1, и теперь в любом месте программы вместо

* Во всех перечисленных случаях приходится иметь дело с выбором одной из двух альтернатив. Реально же возможных вариантов действий обычно бывает три и больше. Программный аналог выбора одной альтернативы из многих (оператор CASE) имеется в ряде языков программирования высокого уровня (например, в СИ или ПАСКАЛе), но в ассемблере, к сожалению, отсутствует и реализуется в виде набора команд условного перехода. — *Прим. ред.*

цифры 1 можно писать букву А. Вместо нее может быть выбран и любой другой символ в зависимости от вашего желания. Но прямое присваивание ОБЯЗАТЕЛЬНО должно быть сделано ДО начала самой программы! (Конечно, в нашем коротком примере всего этого можно было бы и не делать, но автору хотелось показать читателям и такую возможность ассемблера. Для «длинных» программ с частым использованием констант (т. е. величин всегда постоянных) в виде громоздких или труднозапоминаемых чисел это большое удобство! Скажем, прямое присваивание адреса порта ввода-вывода PR=177714 для работы, например, с принтером при многократных обращениях к порту заметно укорачивает листинг и улучшает его читаемость.)

Приведенная здесь программа сравнивает число в переменной А с содержимым R0 и сообщает нам свое заключение.

Строка 2 в данном случае совершает ДЕЙСТВИЕ, в результате которого флаги ССП принимают конкретные значения (правда, сейчас они нам не требуются и мы на них не обращаем внимания).

Строка 3 совершает другое ДЕЙСТВИЕ, в результате которого флаги ССП примут иные значения, которые процессор использует для принятия решения о дальнейшем ходе программы (строки 4, 5 или 6).

Строка 4 вступает в работу первой. Она проверяет условие «заданное число МЕНЬШЕ числа 3?». Если ДА, то выполнение программы направляется на выдачу соответствующего сообщения. Если НЕТ, управление передается на расположенную ниже строку 5.

Строка 5, если строка 4 передала ей управление, проверяет условие «заданное число РАВНО числу 3?» и аналогичным способом передает управление на выдачу соответствующего текста.

Строка 6 действует подобно строке 5 (если вообще до нее дойдет очередь) и проверяет условие «заданное число БОЛЬШЕ числа 3?».

Меняя значение А в операторе прямого присваивания (например, от 1 до 5, но можно брать и любые другие числа), запускайте программу — и убедитесь, что БК никогда не ошибается в своем суждении.

Отладчик MIRAGE

Несколько раньше для лучшего понимания причин изменения содержимого РОН мы прогоняли в OTL16 в пошаговом режиме (по-другому это действие называется пошаговой отладкой) такую «тренировочную» программу:

```
MOV #222, R0
MOV R0,R1
CLR R0
MOV R1,R2
CLR R1
MOV R2,R3
ADD #2,R3
MOV R3,R4
SUB R3,R4
SUB R3,R2
```

А теперь перейдите в отладчик. Оператор прямого присваивания исчез, но заданное в нем значение вписано в первой строке программы. Прогоните ее в пошаговом режиме, изменяя записываемое в R0 по команде MOV #1,R0 число 1 на другое. Проследите, как меняются флаги N и Z в зависимости от заданных вами чисел и как в конечном счете завершается работа программы.

* * *

Пришла пора немного отдохнуть от «высоких материй». Введите следующий листинг (его автор — некто Панфилов, других сведений о нем нет) и запустите на выполнение.

```
M3:     MOV #100,R2
            ; число можно менять
M2:     MOV #1000,R1
            ; число можно менять
M1:     BIC R0,@#177716
        ADD R2,R0
        CLR @#177716
        SOB R1,M1
        SOB R2,M2
        BR M3
        .END
;-----------------------------
```

```
ADD #2,R2
SWAB R3
INC R3
DEC R2
...
```

Во время отладки мы видели состояние РОН после каждого шага, но для лучшей оценки происходящих изменений может потребоваться сравнивать новое состояние РОН и ССП с предыдущим, т. е. помнить два, три, а то и десяток чисел. Но не у всех же память натренирована, как у Штирлица!

«Не прошедших экзамен на разведчика» выручит другой отладчик — MIRAGE, — позволяющий в пошаговом режиме видеть на

экране состояния РОН и ССП за восемь последних шагов. Это значительное преимущество перед OTL16, хотя в некоторых других случаях MIRAGE менее удобен. (Наиболее чувствительные его недостатки — неперемещаемость и отсутствие функции вывода на принтер. Нет возможности и для записи текста в виде букв вместо числовых кодов.)

Вообще же лучший выход из подобной ситуации — самому написать отладчик по собственному вкусу, чего вам и желаю. А так как пока мы этого не умеем, придется пользоваться тем, что есть, выбирая OTL16, MIRAGE или другой отладчик в зависимости от решаемой задачи.

Если в OTL16 состояние флагов ССП отображается цифрами «1» и «0», то в MIRAGE — символами «N», «Z», «V» и «C» или «—» (как уже было отмечено ранее).

Отметим и еще одно существенное преимущество MIRAGE — возможность вставлять в любое место уже написанной программы команду NOP с раздвижкой строк — адресов (по клавише «ВС») и удалять любую команду со сдвижкой строк — адресов (по клавише «СБР»). Такими возможностями обычно обладают только ассемблер-трансляторы. (Аналогично команде NOP можно вставлять в листинг любые команды по своему желанию для модернизации уже написанной программы.)

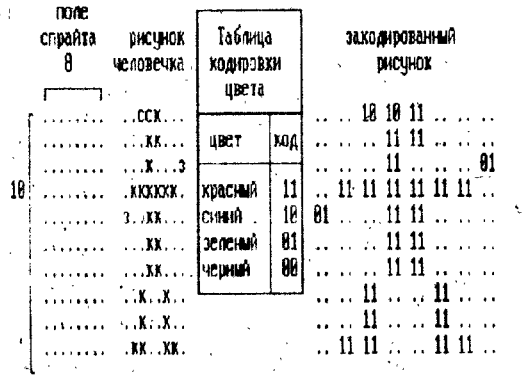
Наберите предложенную выше «тренировочную» программу в MIRAGE и прогоните ее в пошаговом режиме, наблюдая состояние регистров и флагов ССП.

Спрайты

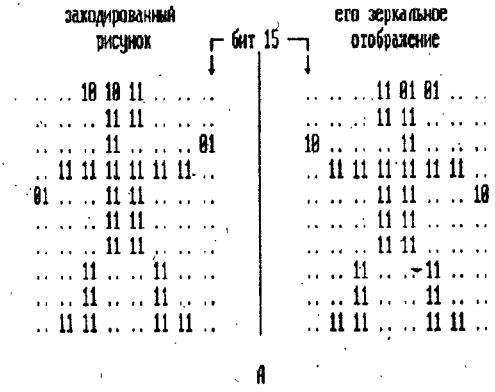
Спрайтами обычно называют маленькие рисунки, которые могут быть составными частями большой картинки. Для начала нарисуем спрайт с изображением ЧЕЛОВЕЧКА.

Пусть размеры спрайта (см. рисунок) равны величине одного цветного курсора, т. е. шириной в 16 «черно-белых» или 8 «цветных» точек (2 байта) и высотой 10 графических строк. Нарисуем на бумаге цветными точками фигурку человечка, обозначая цвета точек начальными буквами их названий: «к» — красный, «з» — зеленый, «с» — синий, а черный цвет обозначим точкой.

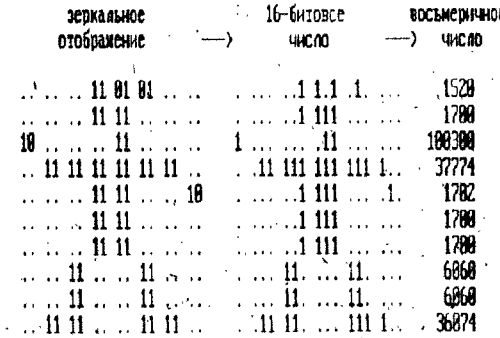
Закодируем каждую цветную точку двухбитовой комбинацией, в которой число «00» пока будем для большей наглядности рисунка изображать парой точек.



Преобразуем закодированный рисунок в его зеркальное отображение, повернув его вокруг оси А на 180°:



Полученное зеркальное отображение преобразуем в набор двоичных 16-битовых чисел, а затем в восьмеричные числа, которые можно записать в тексте программы.



Напомним, что в восьмеричных числах незначащие нули в старших разрядах для краткости не пишутся.

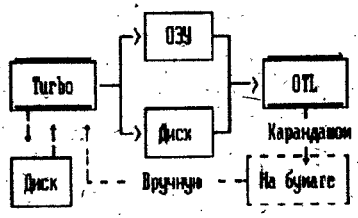
Теперь введите в MIRAGE (или в OTL16) следующую программу, запишите ее на диск по команде SAVE CHL/1000/1076 и запустите по команде G1000. Если все сделано без ошибок, то полученное изображение будет похоже на фигурку человечка, причем на цветном дисплее будут видны точки всех трех цветов.

```
1000 MOV #1520,@#56050
.... MOV #1700,@#56150
.... MOV #100300,@#56250
.... MOV #37774,@#56350
.... MOV #1702,@#56450
.... MOV #1700,@#56550
.... MOV #1700,@#56650
.... MOV #6060,@#56750
.... MOV #6060,@#57050
.... MOV #36074,@#57150
1074 HALT
```

(Программу обязательно сохраните на диске — она нам еще понадобится.)

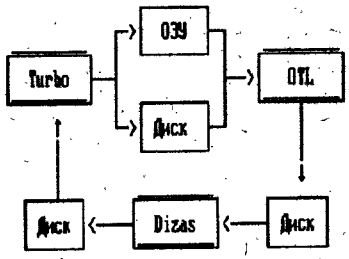
Дезассемблер

Ранее уже говорилось, что, написав программу в Turbo, можно перейти в отладчик для ее пошагового прогона. При этом, возможно, потребуется внести в программу какие-либо изменения, но, записав ее из отладчика на диск и загрузив в Turbo, мы ничего вразумительного не увидим! Поэтому в Turbo перед переходом в отладчик надо сохранять программы на диске по команде ST. А изменения, сделанные в отладчике, тогда остается только записывать вручную карандашиком на бумаге, чтобы потом, запустив Turbo, загрузить с диска сохраненную там старую (неисправленную) программу и вручную же вносить в нее все изменения «по бумажке», следя за отсутствием ошибок. (А это не всегда удается, особенно когда изменений достаточно много.) Этот процесс условно можно представить в виде схемы:



Путь от Turbo к OTL через ОЗУ возможен не во всех дисковых операционных системах, например в ANDOS это невозможно. Поэтому лучше использовать «путь через диск», записывая туда свою программу командой SA и затем вновь считывая в память.

Для облегчения внесения в ассемблерный листинг программы изменений, сделанных во время прогона в отладчике, вы можете воспользоваться системной программой Dizas.WS. Она ДЕЗАССЕМБЛИРУЕТ написанную в отладчике кодовую программу, преобразуя ее в текст, который можно загрузить в Turbo по команде LO. Этот процесс удобно представить такой схемой (сравните ее с предыдущей):



Здесь самое ценное состоит в том, что процесс в какой-то мере автоматизируется и позволяет исключить из цепочки выполняемых операций самое ненадежное звено — карандаш и бумагу в руках усталого пользователя...

Действуя по этой схеме, возьмите программу рисования человечка, которую мы ранее написали в отладчике, обработайте ее в Dizas и загрузите в Turbo. Надеюсь, вы это сможете сделать без дополнительных подсказок.

Для закрепления полученных знаний предлагаем проделать такое упражнение: напишите программу в Turbo, доработайте ее в MIRAGE и «верните» в Turbo (естественно, с помощью Dizas) для дальнейшего совершенствования.

Пример.

Пишем в Turbo:

```
MOV #'A,R0
EMT 16
.END
```

Начинаем подавать команды:

- «ВС», «ТАБ» — переходим в режим команд ассемблера;
- СО — транслируем нацисанную программу в коды;

- RU — убеждаемся в ее работоспособности;
- SA A1 «ВВОД» — записываем на диск под именем «A1»;
- MO «AP2»+«2» — выходим в ANDOS;
- MIRAGE «ВВОД» — запускаем отладчик;
- F1000/2000/0 «ВВОД» — очищаем участок памяти;
- LOAD A1 «ВВОД» — загружаем программу с диска;
- A1000 «ВВОД» — входим в редактор отладчика и видим нашу программу.

Внесем в листинг дополнения, скажем, такие:

```
1000 MOV #101,R0
1004 EMT 16
1006 MOV #20,R3
1012 MOV #7,R0
1016 EMT 16
1020 SOB R3,1012
1022 HALT
```

Снова «командуем»:

- «СТОП» — выходим в командный режим;
- G1000 — проверяем программу в работе;
- SAVE M1/1000/1024 «ВВОД» — записываем под именем «M1»;
- MON «ВВОД», «AP2»+«2» — выходим в ANDOS;
- Dizas «ВВОД» — запускаем дезассемблер;
- LOA M1 «ВВОД» — загружаем программу, полученную в отладчике;
- DFI «ВВОД», «ВВОД»... — просматриваем программу;
- «СТОП» DIZ «ВВОД» — дезассемблируем.

На экране видим строки:

```
        MOV #101,R0
        EMT 16
        MOV #20,R3
L1012:  MOV #7,R0
        EMT 16
        SOB R3,L1012
        HALT
        .END
```

Имя файла:

Здесь Dizas сгенерировал метку L1012, наличие которой и отличает текст в ассемблере от текста в отладчике. Цифровая часть метки совпадает с адресом данной строки в отладчике. Знать его иногда бывает очень полезно при рассмотрении ассемблерного текста без загрузки в отладчик. Сам Turbo не умеет делать такие метки-адреса, но для их получения можно «прокатать» программу через Dizas.

Запрос компьютера «Имя файла:» приглашает нас ввести имя, под которым полученный ассемблерный текст будет записан на диск:

- D1 «ВВОД» — записываем программу под именем «D1»;
- «СТОП» MON «ВВОД» T «ВВОД» — выходим в ANDOS;
- Turbo «ВВОД» — запускаем ассемблер;
- LO D1 — загружаем программу, полученную в Dizas;
- SC — видим загруженную программу с доработкой, сделанной в отладчике;
- «ВС», «ТАБ», CO RU — проверяем программу в работе (далее ее можно дорабатывать уже в Turbo).

«Русификация» программ

Во многих системных программах сообщения нередко выдаются по-английски. Это можно объяснить обыкновенным пижонством их авторов, их неуважением к российским читателям. Такие авторы рассуждают, наверно, так: «Вот какой Я деловой — могу и по-иностранному! А ты — салага — учи английский». Например, раньше Dizas выдавал «File name:» или что-то подобное — на чужом языке и не нашенскими буквами*. А с помощью отладчика можно отдельные фразы заменить на русские так, что они будут понятны без словаря и доп кольнику.

* Причины использования английских фраз в сообщениях, выдаваемых теми или иными программами, уже упоминались в одном из предыдущих выпусков журнала, и думается, нелишне их повторить хотя бы кратко. Дело здесь в большинстве случаев заключается вовсе не в «пижонстве» авторов, а имеет более веские причины (хотя иногда, действительно, это только небрежность и непродуманность интерфейса). Так, на IBM-совместимых ЭВМ, где существует несколько различных стандартов кодировки русских букв, «англизированные» сообщения — просто вынужденная мера обеспечения универсальности программ.

На БК, конечно же, такой проблемы нет, благо имеется изначальная и неизменная на всех моделях таблица кириллицы, но и здесь возможны сложности. Например, в ANDOS 3.1 А. М. Надежин перешел от русскоязычных сообщений к английским фразам, чтобы они целиком умещались в одной 32-значной строке (а заодно, чтобы сэкономить немного памяти): использованные в сообщениях английские слова оказались короче аналогичных по смыслу русских.

Конечно, приведенное здесь мнение редактора может не совпадать с воззрениями отдельных читателей, как не совпадает оно с мнением автора этой статьи (не зря же говорят, что сколько людей, столько и мнений). Но все же думается, не следует обижать разработчиков ПО огульным обвинением в неуважении к пользователям, не разобравшись вначале как следует в конкретной ситуации. — *Прим. ред.*

Для этого загрузите в OTL16 (но не в MIRAGE) программу, которую вы хотите русифицировать. «Пройдитесь» по листингу и найдите ячейки, текст в которых надо перевести на русский язык или просто изменить по своему усмотрению. Запишите на бумагу, чтобы не забыть, восьмеричный код байта, стоящего в листинге сразу после фразы, которую вы хотите заменить. Подсчитайте, из скольких символов состоит вся фраза, и составьте новый текст не длиннее прежнего. Если строка получится короче, лишние знакоместа можно заполнить пробелами или чем-нибудь еще, в зависимости от ситуации. Далее:

- наберите начальный адрес фразы и нажмите «ВВОД»;
- подайте команду «Точка с запятой», наберите нужный текст и нажмите клавишу «СТОП» (но не «ВВОД»!);
- снова наберите начальный адрес фразы, «ВВОД», «Запятая», «Запятая» и т. д.

Просмотрите, что у вас получилось. Главное, обратите внимание, чтобы длина фразы не оказалась больше прежней, а следующий после нее байт (в начале работы переписанный вами на бумагу) остался неизменным. Иначе ваши старания могут только испортить программу.

В MIRAGE для той же цели можно применять команду S <адрес>/B, но это гораздо менее удобно.

Машинная арифметика

Не секрет, что БК способен решать арифметические задачи любой трудности, если постараться составить соответствующую программу. Вот пример сложения и вычитания чисел (в MIRAGE):

```
MOV #2,R2        ; записать число 2 в R2 (результат в R2)
ADD #3,R2        ; прибавить к нему число 3
SUB #6,R2        ; отнять 6
ADD #1,R2        ; прибавить 1
MOV #100000,R5   ; записать 100000 в R5
ADD #100000,R5   ; прибавить 100000
SUB #10,R5       ; отнять 10
ADD #11,R5       ; прибавить 11
```

Сделайте пошаговый прогон. Вы, конечно, заметили, что некоторые результаты, мягко говоря, не совпадают с вашими ожиданиями. Это происходит от того, что вы знаете еще не все секреты своей ЭВМ. Тогда продолжим «грызение гранита науки».

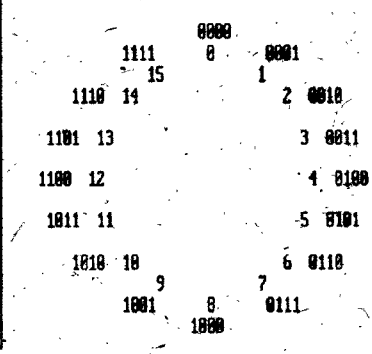
Как нам уже известно, БК работает с двоичными 16-битовыми числами — словами. Для изучения правил машинной арифметики условно примем длину чисел — слов равной всего четырем битам вместо 16, чтобы не рисовать громоздкие 16-битовые цепочки нулей и единиц. Но законы обработки тех и других одинаковы.

Итак, пусть каждое слово состоит из четырех ячеек-битов, занумерованных справа налево от 0 до 3:

```
  - - - -
  3 2 1 0  <- битовая позиция (номер)
  ↑     ↑
Самый старший бит  Самый младший бит
```

Рассмотрим нулевое четырехбитовое слово 0000. Если его увеличить на 1, то получим число 0000+1=0001. Продолжая увеличивать его последовательно до заполнения всех битов, получаем ряд чисел, представленный на рисунке в таблице слева (наибольшее возможное четырехбитовое число равно 1111 или 15 в десятичной системе счисления).

Содержимое	
'2	'10
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	10
1011	11
1100	12
1101	13
1110	14
1111	15



Если к последнему числу прибавить еще 1, то получится 1111+1=10000 (в десятичном виде — 15+1=16), для отображения которого требуется уже пять битов. Но слова в нашей «условной» ЭВМ, как мы договорились, имеют только по четыре бита, поэтому реально мы получим число 0000. А куда же делась единица из пятого, несуществующего бита? Ведь уравнение 15+1=0 — это явная ерунда!

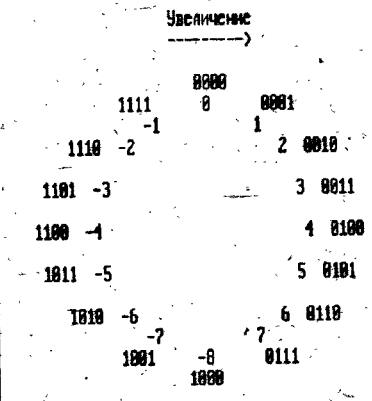
(Разработчики микропроцессоров хорошо знакомы с данной проблемой и предусматривают на этот случай особые флаги ССП — C и V, о существовании которых уже упоминалось раньше. Так, флаг C устанавливается в «1», если при выполнении операций сложения беззнаковых чисел процессору не хватает количества отпущенных под число-результат 16 битов. Далее в программе можно предусмотреть проверку значения бита C с помощью оператора условного перехода (например, для вывода сообщения о некорректности операции) или, поскольку бит C, по сути, выполняет функции дополнительного 17-го бита результата, можно учесть его в дальнейших вычислениях, формируя 32-битовый результат «двойной длины». Более подробное обсуждение работы с флагами C и V приводится чуть ниже. — *Прим. ред.*)

Заметим и такой факт: прибавив к максимально возможному числу единицу, мы снова вернулись к нулю, поэтому можно представить непрерывное увеличение чисел в виде замкнутого кольца (как это и сделано на рисунке).

Однако реально в компьютерных вычислениях требуются не только положительные, но и отрицательные числа, поэтому была придумана система, когда левый крайний бит (в нашем примере — третий) понимается как признак отрицательности числа. Если он равен нулю, то число считают положительным, а при единице — отрицательным*. Теперь половина наших четырехбитовых слов будет положительной, а другая половина — отрицательной. Количество разрядов, отводимых для отображения чисел, при этом уменьшилось до трех, что позволяет кодировать числа от −8 до +7, но зато со знаком!

Содержимое	
'2	'10
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	-8
1001	-7
1010	-6
1011	-5
1100	-4
1101	-3
1110	-2
1111	-1

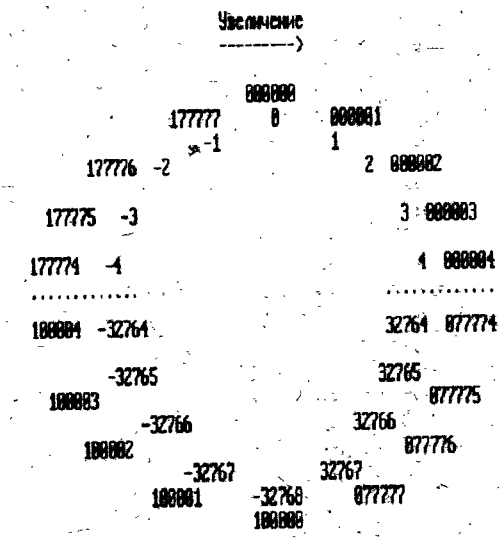
Кольцо четырехбитовых чисел (с учетом знака)



Эта числовая система также конечна и циклична, но здесь разрыв непрерывности происходит между 0111 и 1000 (между десятичными +7 и −8), а также, как и в предыдущем случае, между 1111 и 0000 (десятичными −1 и 0).

Перейдем теперь к реальным 16-битовым числам. Аналогично первому из уже рассмотренных замкнутому кольцу чисел (без знака) в БК изменяются адреса памяти от 0 до 1 111 111 111 111 111 или, в восьмеричном отображении, от 0 до 177777. По схеме же второго кольца изменяются числа со знаком — аргументы арифметических операций: от 1 000 000 000 000 000 до 0 111 111 111 111 111, в восьмеричном виде — от 100000 до 077777, а в десятичном — от −32768 до 32767.

Кольцо 16-битовых чисел с учетом знака (упрощенное изображение)



«Критические точки» в этом кольце остались на тех же местах, как и в нарисованной ранее схеме знаковых четырехбитовых чисел. Первая из этих точек расположена на переходе от 77777 к 100000 (77777+1=100000), когда к положительному числу 77777 прибавляется единица, а результат получается отрицательный. Такого в «нормальной» арифметике не бывает. В машинных же вычислениях это явление названо «переполнением разрядной сетки», а его признаком является установка флага V (переполнение) в слове состояния процессора.

Продолжение на с. 76

* Флаг N в ССП фактически является копией «знакового» 15-го бита результата предыдущей операции. — *Прим. ред.*

Памяти Е. М. Рассказова

Россия! Прекрасная страна! Но почему время, отпущенное твоим самым лучшим сыновьям, так коротко?

31 августа (или 1 сентября — установить это сейчас невозможно) 1992 года погиб в горах Евгений Михайлович Рассказов. Человек неординарный, талантливый изобретатель, один из первых создателей и пропагандистов периферийных устройств для БК. Многим знакомы распространявшиеся через «Геоком ЛТД» его разработки: «ОЗУ-32», «ОЗУ-128», «Оптический телетайп», «Детектор лжи», «Автономный накопитель информации», «БК+телефон» и многие другие. Но для нас, его знакомых и коллег, Женя был прежде всего другом. Прошло уже много времени, как он ушел от нас, но до сих пор нам не хватает его оптимизма и жизнелюбия, его юмора и искренности. Это не дежурные слова — о мертвых только хорошее. Его действительно любили и ценили все, кто его знал. И до сих пор говорить о Нем в прошедшем времени кощунственно. Пусть публикация одной из последних его разработок — «БК+телефон» — послужит доброй памятью об этом прекрасном человеке.

Сотрудники Кроноцкого заповедника

Клиенты фирмы «Геоком ЛТД», перечислившие деньги и не получившие ответа от Е. М. Рассказова, по всем вопросам могут обращаться по адресу: 684010, Камчатская область, г. Елизово, ул. Рябикова, 48, Кроноцкий заповедник. Зыкову Владимиру Васильевичу.

С разрешения вдовы погибшего в горах Евгения Михайловича Рассказова публикуем одну из его последних разработок. Телефонная приставка была в свое время изготовлена, успешно работала как автоответчик и сейчас используется в Кроноцком государственном заповеднике для сбора и обработки информации, поступающей по телефонам (в частности, сообщений о землетрясениях).

Е. М. Рассказов,

г. Петропавловск-Камчатский

Телефонная приставка к БК-0010(.01)

Предлагаемая приставка к БК-0010(.01) позволяет регистрировать телефонный звонок, различать длинные и короткие гудки, осуществлять поднятие трубки, набор номера, передачу на линию сообщений с магнитофона и программного синтезатора речи, а также записывать информацию, принимаемую с телефонной линии. Кроме того, приставка «БК+телефон» (БК-ТЕЛ) содержит генератор секундных импульсов, позволяющий реализовать функцию часов. Использование для коммутации сигналов электромагнитных реле не только упрощает схему и обеспечивает абсолютную гальваническую развязку, но и не нарушает работу телефона при отключении БК. Программное управление приставкой дает широчайшие воз-

можности для создания телефонов, автоответчиков, АОНов, оповестителей и т. п. При этом приставка БК-ТЕЛ не мешает подключению к порту принтера и мыши.

Так как порт УП БК использует инверсные коды, при записи в соответствующий разряд регистра 177714 «лог. 1» на выходе порта будет установлен низкий уровень. Сразу же после включения компьютера на всех разрядах порта устанавливается «лог. 0», т. е. высокий уровень на всех выводах разъема УП. При этом транзисторы Т1—Т3 и Т5 будут заперты, а Т4 открыт (см. принципиальную схему БК-ТЕЛ на рис. 1). Контакты реле К1.1, К3.1 и К4.1 будут разомкнуты, а К2.1 — замкнут. Это состояние является дежурным: сохраняется «ручное» управление телефоном, а двигатель МГ-ЗАП выключен.

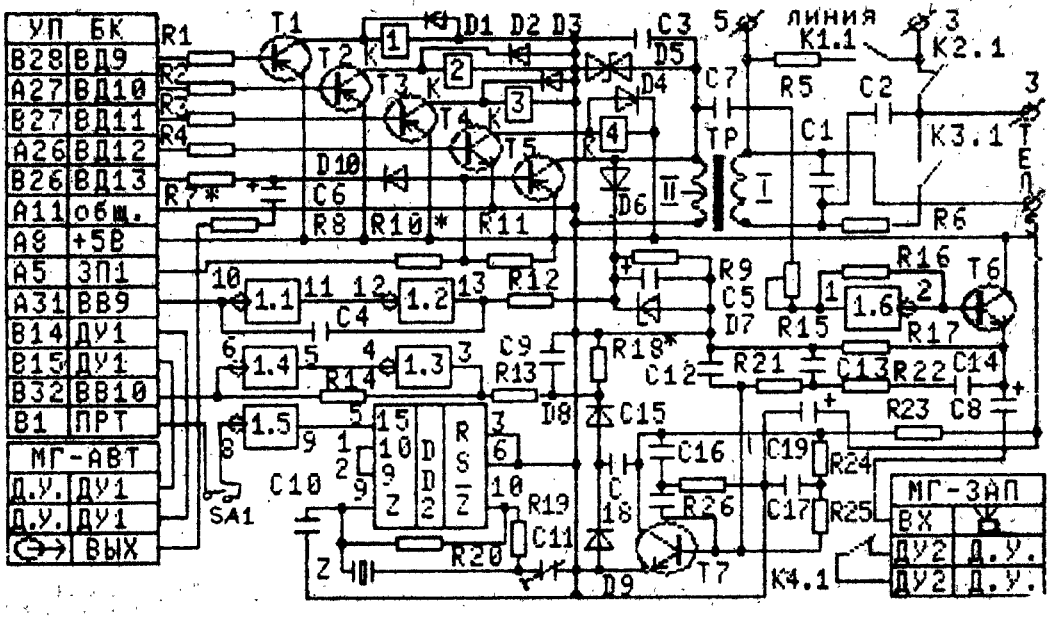


Рис. 1. Принципиальная схема приставки БК-ТЕЛ

Используемые радиоэлементы:

- резисторы: R1, R2, R3, R4, R23 — 3кОм; R5, R6 — 100 Ом; R7, R9, R10 — 30 кОм; R8 — 1.5 кОм; R11 — 20 кОм; R12, R13, R16 — 750 кОм; R14 — 1.5 МОм; R15, R24, R25 — 100 кОм; R17 — 1 кОм; R18 — 39 кОм; R19 — 470 кОм; R20 — 10МОм; R21, R22 — 6.8 кОм; R26 — 50 кОм;
- конденсаторы: С1, С3 — 68 нФ; С2 — 1 мкФ; С4, С7, С12—С15 — 0.1 мкФ; С5, С6, С8, — 15 мкФ; С9 — 47 нФ; С10 — 50 пФ; С11 — 6/25 пФ; С16, С18 — 5000 пФ; С17 — 0.01 мкФ; С19 — 50 мкФ;
- транзисторы: Т1—Т3, Т5 — КТ209 (КТ502, КТ208); Т4 — КТ503 (КТ608, КТ3102); Т6, Т7 — КТ315 (КТ3102, КТ503);
- диоды: D1—D4, D6, D8—D10 — КД522 (Д223); D5 — КС168В (КС162А); D7 — КС13[illegible] (КС133);
- микросхемы: DD1 — К561АН2 (с учетом изменения цоколевки — К176ПУ2); DD2 — К176ИЕ12;
- Z — кварц на 32768 Гц;
- ТР — согласующий трансформатор от радиоприемника;
- К1—К4 — электромагнитные реле РЭС15 (паспорт РС4.591.005) или другие на напряжение 5 В и ток срабатывания не более 100 мА. Контакты К4.1 и К2.1 должны быть нормально замкнутыми, а К1.1 и К3.1 — нормально разомкнутыми.

При записи «лог. 1» в одиннадцатый разряд порта УП (регистр 177714) производится поднятие трубки. «Лог. 1», удерживаемая в течение 50 мс в девятом разряде порта и сменяемая на «лог. 0» в девятом и «лог. 1» в десятом разрядах на те же 50 мс, а затем вновь на «лог. 0» в десятом разряде, соответствует набору цифры «1». Повторяя эту последовательность с интервалом между сериями 200—300 мс, можно набрать любую цифру телефонного номера (количество серий равно значению цифры). При записи же «лог. 0» в одиннадцатый разряд на время более 400 мс линия освобождается.

«Лог. 1» в двенадцатом разряде включает магнитофон МГ-ЗАП для записи сообщения с линии. Транзистор Т5 при этом должен быть заперт, чтобы не блокировать сигнал с трансформатора. Для подачи информации в линию используется шестой разряд регистра ВУ (адрес 177716). Этот разряд используется синтезатором речи «ГОВОРУН», к нему же подключен и звукоизлучатель компьютера (сигнал ЗП1).

«Лог. 1» в тринадцатом разряде порта переводит транзистор Т5 в активный режим. Это позволяет передавать информацию в линию с магнитофона МГ-АВТ. Его двигатель включается подачей «лог. 0» в седьмой разряд регистра с адресом 177716.

Когда на телефонной линии присутствует напряжение вызова («звонок»), оно выпрямляется диодом D6, заряжает конденсатор С5, проходит через формирователь на элементах DD1.1, DD1.2 и поступает на девятый разряд регистра ввода порта УП. «Лог. 0» в этом разряде соответствует «звонку».

Напряжение с вторичной обмотки трансформатора ТР через конденсатор С7, регулятор уровня R15, усилитель DD1.6 и транзистор Т6 поступает на вход МГ-ЗАП и одновременно через фильтр С14, R22, С13, R21 и транзистор Т7 на выпрямитель D8, D9, C9, R18. Если это напряжение представляет собой НЧ-сигнал с частотой 400—600 Гц («гудок»), то срабатывает триггер Шмидта на элементах D1.3, D1.4. Высокий уровень напряжения, соответствующий в инверсной логике «лог. 0», поступает на десятый разряд регистра ввода порта УП. Длительность гудка определяется программно. Речевой сигнал с линии не переключает триггер (или переключает только кратковременно, что легко отфильтровывается программой), а сильный сигнал с МГ-АВТ проходит через триггер. Это позволяет останавливать двигатель МГ-АВТ после паузы в конце сообщения, записанного на «бесконечной» кассете.

Разряды 11—15 регистра ввода порта можно дополнительно использовать для охранной и пожарной сигнализации. Охранный шлейф с сопротивлением не более 300 Ом может быть подключен между входом соответствующего регистра УП и общим выводом. При разрыве шлейфа на входе регистра появляется «лог. 0».

На элементах DD2 и DD1.5 собран кварцевый генератор секундных импульсов, вызывающих прерывание по вектору 100 (в ФОКАЛе программная поддержка генератора реализована в виде стандартной функции FCLK()).

Конструкция БК-ТЕЛ

Блок монтируется навесным монтажом на макетной плате и помещается в корпус от блока МСТД. На узкой стороне платы закрепляется разъем СНП58-64/95 (или прилагавшийся в комплекте к БК при продаже), посредством которого блок БК-ТЕЛ подсоединяется к порту УП компьютера. К выводам 3 и 5 «ЛИНИЯ» присоединяется кабель телефонной линии, а к выводам 3 и 5 «ТЕЛ» — телефонный аппарат. При выключенном БК работоспособность телефона полностью сохраняется. Тумблер SA1 отключает генератор часов.

Изменения на плате БК

Необходимо соединить навесным монтажом:

- вывод 12 микросхемы D2.6 (К561ПУ4) с выводом А5 порта УП (сигнал ЗП1);
- сигнал УПР1 (вывод 1 разъема ХТ4) с выводом В14 порта УП;
- сигнал УПР2 (вывод 4 разъема ХТ4) с выводом В15 порта УП.

Требования к магнитофонам

Для подключения к приставке БК-ТЕЛ можно использовать магнитофоны любого типа с дистанционным управлением двигателя (при замыкании контактов реле К4.1 и штатного реле «пуск двигателя» в БК). Для записи сообщений используется вход магнитофона «радиоприемник», а при его отсутствии — вход «микрофон» с последовательно включенным резистором с сопротивлением 51 кОм. Для выдачи сообщения автоответчика (МГ-АВТ) используется стандартный линейный выход.

Наладка устройства

После сборки необходимо тщательно проверить монтаж, и только после этого можно подключать БК-ТЕЛ к компьютеру, телефону и магнитофонам. Регулятор R15 поставьте в среднее положение. Если триггер регистрации гудка не срабатывает от сигнала гудка в линии, нужно уменьшить R15, а если срабатывает даже от голоса звонящего абонента — увеличить. Если это не помогает, нужно подобрать R18. Иногда требуется подобрать и резистор R7 по минимуму искажений при передаче сообщений с МГ-АВТ (при наличии «лог. 1» в тринадцатом разряде). Затем с помощью конденсатора C11 нужно установить частоту на выводе 10 микросхемы DD2 равной 32768 Гц.

Схема «бесконечной» кассеты

Рис. 2. Изготовление «бесконечной» кассеты

Описание программы BKTEL

Перед загрузкой программы BKTEL необходимо разомкнуть выключатель SA1, выйти в МСТД, загрузить синтезатор речи «ГОВОРУН2», запустить его, а затем перейти из него в ФОКАЛ. После этого можно загрузить и запустить программу BKTEL, а затем замкнуть SA1. Без синтезатора речи некоторые функции программы (запись времени телефонного звонка на МГ, передача сообщений) не выполняются, но все остальное работать будет.

Назначение групп строк программы:

- 1 — меню (главная программа);
- 2, 3, 5, 6, 7, 8, 9 — «кнопочный телефон» с памятью до 256 номеров;
- 19 — сброс некоторых разрядов порта;
- 20 — детектор звонка, выход в меню — по клавише «ВВОД»;
- 21 — детектор коротких гудков (длина в строке 21.3, количество гудков — в строке 21.35);
- 22 — детектор паузы на бесконечной кассете (длина в 22.3);
- 25 — запись данных за сутки (в строке 22.15 — время записи);
- 30 — часы (CS — секунды, CM — минуты, CC — часы, CD — дата);
- 33 — счетчики количества звонков: I1(CD) — информативных (т. е. с записью сообщения) за сутки, I2(CD) — неинформативных за сутки, I1Z — информативных со времени включения, I1(CC) — информативных за час;
- 40 — распечатка номеров телефонов из памяти, ввод критерия информативный/неинформативный (CZ);
- 42, 43 — поддержка режима охраны;
- 50 — диаграммы работы автоответчика;
- 55, 56 — печать диаграммы;
- 60, 10, 11 — цикл оповещения по телефону;
- 61 — определение паузы между гудками;
- 63 — текст сообщения 1;
- 64 — текст сообщения 2.

Примечание. Переменные A1—A4 и некоторые из Bn (n — цифра) используются в разных модулях многократно. Вместо МГ-АВТ можно задействовать сообщение, выдаваемое синтезатором речи. В варианте автора сообщение автоответчика, записанное на бесконечную кассету, звучит так:

«Здравствуйте, вы набрали номер 66-66-666. Сейчас у телефона никого нет, но после звукового сигнала вы можете оставить свое сообщение на автоответчике».

Затем идет пауза 1.5—2 с, которая определяется в строках 22, после чего МГ-АВТ останавливается.

Модули 25, 33, 50—56 при их ненадобности можно удалить, а их вызовы из главной программы убрать.

```
      C:  ФОКАЛ-БК0010
01.01 C Программная поддержка приставки БК-ТЕЛ
01.02 C Автор: Е.Рассказов, 1991г.
01.03 D 19;X FCHR(12,155,156);T "-REM-";X FCHR(156);T !
01.05 T !!,"              БК-ТЕЛ",!!;
01.10 T ! "           1 - АВТООТВЕТЧИК"
01.14 T ! "           2 - ГРАФИК"
01.16 T ! "           3 - СПИСОК НОМЕРОВ"
01.18 T ! "           4 - ТЕЛЕФОН"
01.20 T ! "           5 - ЧАСЫ"
01.21 T ! "           6 - ВКЛ. ОПОВЕЩЕНИЯ"
01.22 X FCHR(155);T !!,"нажмите нужную клавишу",!,"возврат в меню - <ВВОД>",!
01.25 S A2=FCHR(-1);X FCHR(12);I(A2-49)1.03,1.6
01.30 I(A2-50)1.03,1.7
01.35 I(A2-51)1.03,1.75
01.40 I(A2-52)1.03,1.8
01.45 I(A2-53)1.03,1.85
01.50 I(A2-54)1.03,1.9,1.03
01.60 D 25;D 4;T."АВТООТВЕТЧИК  "; G 20.02
01.62 D 21;D 19;X FCHR(1);D 30.3;D 30.35;S CK=CM;D 30.4;X FCHR(4);X FP(1,10000)
01.63 D 33;G 1.6
01.70 S HZ=HZ-1;S CN=0;S CK=0;S H(CD)=H(CD)-1;S H1(CC)=H1(CC)-1;D 33;X FCHR(-1)
01.72 G 1.03
01.75 D 40;G 1.03
01.80 G 2.1
01.85 D 30;G 1.03
01.90 S OP=42; G 1.03
02.10 D 7;  X FP(2,1000); X FP(1,1000);S B1=1
02.15 T !,"(набор ручной ;)",!,"(из  памяти  -)",!,"(выход  <ввод>)",!!
02.40 S B5=FCHR(-1); X FCHR(B5)
02.46 I(B5-59)2.47,5.1,2.47
02.47 I(B5-45)2.49,6.1,2.49
02.49 I(B5-13)2.5,1.03,2.5
02.50 S B(B1,D)=B5-48; I(B(B1,D)-10)2.7,2.4,2.4
02.70 I(B(B1,D))2.4,2.75,2.8
02.75 S B(B1,D)=10
02.80 X FP(1,2000); F B3=1,B(B1,D); D 3
02.90 X FP(1,1000); X FP(1,2000); S B1=B1+1;S B1(D)=B1;G 2.4
03.10 X FP(2,1000); X FP(2,2000); F B2=1,2;D 9
03.20 X FP(1,1000); X FP(2,2000); F B2=1,2;D 9
04.10 X FK(0,23)
05.10 S B1=1; X FP(2,2000);T !,"номер телефона",!; F B2=1,200;D 9
05.30 X FP(1,2000); G 2.4
06.10  T !; X FP(2,2000);F B2=1,200;D 9
06.20 X FP(1,2000); F B4=1,(B1(D)-1); D 8; X FP(1,1000);X FP(1,2000)
06.30 G 2.4
07.10 A "номер записи ",D
08.01 T %1.00;I(B(B4,D)-10)8.03,8.05,8.03
08.03 T B(B4,D);G 8.1
08.05 S B2=0;T B2
08.10 I(B4-2)8.15,8.12,8.15
08.12 F B2=1,70; D 9
08.15 F B2=,40; D 9
```

```
08.20 F B3=1,B(B4,D); D 3
09.10
10.10 X FP(2,4000);D 8.15;D 6.1;D 6.2
11.10 X FP(1,4000)
19.10 X FP(1,4000);X FP(1,1000);X FP(2,2000);D 8.12;X FP(1,2000)
20.01 C ОПРОС ПОРТА BB9 - 5V=ЗВОНОК (ЛОГ.0)
20.02 D OP; S A3=0; I(FX(1,177662)-10)20.05,20.03,20.05
20.03 D 30.3;D 30.35;G 1.25
20.05 I(FP(0,1000))20.02,20.1,20.02
20.10 S A3=A3+1
20.15 I(A3-2)20.05
20.20 D 4;T "1. звонок     ";X FP(2,4000)
20.25 D 22
20.30 G 1.62
21.01 C ОПРОС BB10 - 5V=ГУДОК (ЛОГ.0)
21.02 D 4;T "4. начало записи";D 30.3;D 30.35;S CN=CM;X FP(2,10000)
21.03 S A4=0
21.05 I(FP(0,2000))21.05,21.1,21.05
21.10 S A3=0
21.15 I(FP(0,2000))21.2,21.2,21.3
21.20 S A3=A3+1; G 21.15
21.30 I(A3-5)21.03,21.35,21.35
21.35 S A4=A4+1;I(A4-3)21.05,21.4,21.4
21.40 D 4;T "5. запись времени"
22.01 C ОПР. BB10-ОПРЕД. ПАУЗЫ МГ-АВТ.(ПАУЗА-ЛОГ.1)
22.02 D 4;T "2. пуск МГ-АВТ";S A1=0;F A2=,2;D 8.15;L R;D 8.15;L M
22.05 X FP(2,20000);F A2=,1000;D 9
22.10 I(FP(0,2000))22.2,22.15,22.2
22.15 S A1=0
22.20 I(FP(0,2000))22.2,22.2,22.25
22.25 S A1=A1+1
22.30 I(A1-9)22.1,22.35,22.35
22.35 L R;X FP(1,20000);D 4;T "3. стоп МГ-АВТ"
22.40 X FCHR(7,7,7)
25.10 C СОХР.ДАННЫХ - В 00 ЧАС
25.15 I (CC)25.9,25.2,25.9
25.20 I(CM)25.9,25.25,25.9
25.25 X FP(2,10000);X FCHR(1);T "ДАННЫЕ ЗА",CD,"ЧИСЛО, ИНФОРМАТИВНЫХ,ЗВОНКОВ,",H(CD)
25.35 T !,",ВСЕГО, ЗВОНКОВ,",H(CD)+H2(CD);X FCHR(4);X FP(1,10000)
25.90
30.01 C ЧАСЫ
30.10 X FCHR(12);T !,"                    1-индикация",!
30.11 T !,"              2-установка",!
30.15 S A2=FCHR(-1);I(A2-49)30.25,30.3,30.2
30.20 I(A2-50)30.25,30.7,30.25
30.25 R
30.30 S C1=FCLK();S C=C1-C0;S C2=FITR(C/60);S CS=C-C2*60;S C3=FITR(C2/60)
30.35 S CM=C2-C3*60;S CD=FITR(C3/24);S CC=C3-CD*24
30.40 T %2.00,"ДЕНЬ "CD," ,ЧАС " ,CC,",МИНУТ " ,CM; F A2=,150;D 9
30.45 X FCHR(12,18);G 30.95
30.70  A "число",CD,"час",CC,"мин",CM;S CU=CM*60+CC*3600+(CD)*24*3600
30.75 S C0=C1-CU; S D0=CD
30.95
33.01 C СЧЕТЧИКИ
33.05 I(CK-CN)33.1,33.15,33.15
33.10 S CF=CK+60-CN;G 33.2
33.15 S CF=CK-CN;G 33.2
33.20 I(CF-CZ)33.25,33.3,33.3
33.25 S H2(CD)=H2(CD)+1; R; C K-ВО НЕИНФОРМ.
33.30 S H(CD)=H(CD)+1; S HZ=HZ+1
33.40 S H1(CC)=H1(CC)+1; D 50
40.10 X FCHR(12); T !"   ТЕКУЩИЕ НОМЕРА",!!,"(просмотр <пробел>)",!
40.15 S A2=1
40.20 F D=A2,20+A2; T %3.00,!,D,"-";D 41
40.25 S A2=A2+21;I(FCHR(-1)-32)40.3,40.2
```

```
40.30 T %3.00,!!  "критерий",!, "CZ=",CZ,!;C  "O=",O,!
40.40  T ! "новое значение",!;A "CZ",CZ,; C "O",O
41.10 F B4=1,(B1(D)-1); T %1.00,B(B4,D)
42.01
42.10 I(FP(0,4000))42.2,42.5,42.2
42.20 I(FP(0,10000))42.9,42.3,42.9
42.30 S TX=64;D 60;R
42.50 S TX=63;D 60
42.90
43.10
50.01 C ГРАФИК
50.05 T %2.00;X FCHR(12)
50.10 F A2=1,2,48;X FK(A2+14,21);T "┼──┤"
50.15 F A2=0,4,48;X FK(A2+12,22);T A2/2
50.20 F A2=2,21;X FK(14,A2);T "┼";X FK(10,A2);T 21-A2
50.22 T %3.00
50.25 X FK(32,23);T "время суток";X FCHR(18);T "ЗА СУТКИ: неинф.=",H2(CD),!
50.27 T ! "за час >"
50.30 X FK(24,0);T "инф.=",H(CD);X FK(38,0);T "ВСЕГО=",HZ,"  за"
50.32 T CD-D0,"  дней",%2.00
50.35 T !!!!!!!!," ТЕКУЩЕЕ:",!!,"число=",CD,!!,"час  =",CC,!!,"мин.  =",CM
50.40 T !!, "к-во звон-",!,"ков  =",H1(CC);D 55
55.10 F A4=1,CC; D 56; C ВОССТАНОВЛ. ГРАФИКА.
56.10 F A2=0,H1(A4); X FK((A4*2)+15,21-A2); T "▒"
60.01 C ОПОВЕЩЕНИЕ ПО ТЕЛЕФОНУ
60.10 S OP=43;F D=1,1;D 10; S A3=0;D 61;D 19
61.01 C ИЗМ. ПАУЗЫ МЕЖДУ ГУДКАМИ
61.10 S A3=A3+1; I(A3-18)61.2
61.15 R
61.20 I(FP(0,2000))61.25,61.6
61.25 S A1=0
61.30 I(FP(0,2000))61.35,61.1
61.35 S A1=A1+1; I(A1-60)61.3
61.40 X FCHR(1);D TX;X FCHR(4);R
61.60 I(FP(0,2000))61.1,61.6,61.1
63.10 T " ТЕКСТ 1 "
64.10 T "ТЕКСТ 2"
```

В различных изданиях часто публикуются советы пользователям БК. Предлагаемые рекомендации затрагивают исключительно hardware и основаны на многолетнем опыте эксплуатации БК.

Е. М. Рассказов,

г. Петропавловск-Камчатский

Советы пользователям БК-0010(.01)

Блок питания*

Правильно отрегулированный блок имеет на контрольной точке схемы КТ4 напряжение 5 В (устанавливается резистором R29), а на выходе (вывод 5 разъема XT2) — 4.8—5 В (устанавливается резистором R19). Если к блоку питания подключены дополнительные устройства (их суммарная мощность может достигать 5—7 Вт без перегрева трансформатора), необходимо увеличить ток срабатывания защиты от перегрузки до 3 А, включив параллельно R8 резистор на 200—300 Ом. Желательно также поставить на транзистор VT5 дополнительный радиатор площадью 10—15 $см^2$, а еще лучше — заменить этот транзистор на КТ819. Если блок питания

* Принципиальная схема блока питания БК опубликована в предыдущем выпуске журнала «Персональный компьютер БК-0010 — БК-0011М». — *Прим. ред.*

сильно «свистит», то между КТ5 и КТ6 надо включить резистор с сопротивлением 400—600 кОм. Сберечь же программы в условиях неожиданных перебоев в электропитании поможет буферный аккумулятор (лучше всего щелочной из 10—12 банок емкостью 7—55 ампер-часов), подключенный минусом к общему проводу, а плюсом — к аноду диода КД202, катод которого присоединен к точке КТ1. Для подзарядки аккумулятора параллельно диоду достаточно включить резистор на 20—50 Ом. Отключать аккумулятор можно общим выключателем, используя верхние (по схеме блока питания) контакты для сети, а нижние — для аккумулятора.

Клавиатура

Пленочную клавиатуру старой модели БК-0010 советую переделать следующим образом: разобрать ее (покровную пленку и резиновую прокладку можно смело выбросить), снять полиэтиленовые толкатели кнопок и напильником аккуратно выровнять (обточить) их лицевую часть. Затем вырезать изображения кнопок так, чтобы полученные квадратики были на 2—3 мм меньше толкателей, прозрачной эпоксидной смолой покрыть толкатели и наклеить на них квадратики с изображением клавиш. Когда смола достаточно затвердеет, нужно приготовить свежую порцию смолы и покрыть ею сверху клавиши тонким слоем. Вместо эпоксидной смолы можно использовать прозрачный лак.

Кнопочную клавиатуру БК-0010.01, чтобы не досаждал «дребезг» клавиш, необходимо почистить, отъюстировать и смазать. Для этого снимите кнопки (они сдергиваются вверх) и аккуратно подогните «усики» подвижных контактов вниз так, чтобы при легком нажатии на кнопку они «вонзались» в неподвижные контакты перпендикулярно, а не скользили по ним. Затем можно часовым или машинным маслом смазать трущиеся части кнопки, стараясь не попасть на контакты.

«Дребезг» клавиатуры можно также устранить, увеличив емкость конденсатора С4 до 0,47 мкФ и сопротивления резисторов R3 и R4 до 910 кОм (при С3=68 нФ). Проделав то же самое с С2, вы улучшите работу клавиши «СТОП».

Все манипуляции с клавиатурой следует проводить, предварительно дотронувшись до заземления, так как микросхема контроллера клавиатуры 1801ВП1-014 легко может быть пробита статическим электричеством.

Основной блок

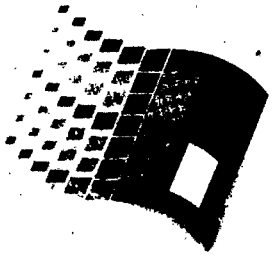
Практика показала, что к разъему МПИ можно подключать одновременно до трех внешних устройств без буферизации шины. Например, у автора к МПИ были подключены КНГМД, ОЗУ-128 и ЭОЗУ-8. Но в компьютерах БК-0010 желательно приклеить к корпусу микросхемы процессора К1801ВМ1 медную пластинку-радиатор площадью 10 $см^2$.

Надежность чтения с магнитофона повысится, если между выводами 2 и 3 разъема XT4 включить конденсатор емкостью 0.068—0.1 мкФ.

В компьютер БК-0010.01 желательно установить переключатель SA2 (в БК-0010 он уже имеется), который при зависании позволяет перезапустить процессор без потери содержимого ОЗУ. А чтобы избежать при перезапуске выхода в ФОКАЛ или БЕЙСИК, можно добавить выключатель ПЗУ DS18. Для этого надо перерезать печатный проводник, соединяющий вывод 23 (CE) этой микросхемы с общим проводом, и соединить его с «общим» через резистор сопротивлением 300—500 Ом. Теперь, установив в удобном месте корпуса выключатель, можно при работе в кодах отключать ПЗУ, подавая напряжение +5 В через выключатель и резистор 100 Ом на тот же вывод 23 (CE). При перезапуске вы будете попадать в монитор, а оттуда с помощью команды S<адрес> — вновь в программу. Такие же выключатели можно поставить на DS19 и DS20 при подключении дисковода или внешних модулей ОЗУ-ПЗУ.

Качество изображения на черно-белом дисплее регулируется резистором R57.

Керамический излучатель звука ЗП-1 подключен к выводу 12 микросхемы D2.6 (К561ПУ4). Качество звука улучшится, если ЗП-1 заменить на высокоомный телефон, включенный последовательно с конденсатором емкостью несколько микрофарад.



Среди многочисленных удобств, так привлекающих пользователей IBM-совместимых и других, не менее мощных компьютеров, не последнюю роль играет встроенный HELP — разнообразная справочная информация об используемой программе, правилах работы с ней, возникшей ошибке и т. п. Любой пользователь ПЭВМ, даже новичок, сразу же запоминает наизусть, что достаточно нажать F1 — и компьютер всегда подскажет, что и как нужно делать (если, конечно, программное обеспечение достаточно высокого класса).

Д. Ю. Усенков,
г. Москва

Резидентный HELP – драйвер для БЕЙСИКа БК-0010.01

Попытаемся реализовать HELP-интерфейс на БК-0010(.01), например, дополнить этой удобной функцией так популярный сегодня вильнюсский БЕЙСИК. (Одна из таких попыток была сделана В. В. Сапоговым в журнале «Вычислительная техника и ее применение», №8 за 1991 г. Предложенный им «HELP-драйвер» имел ряд крупных недостатков, но его идея легла в основу описываемой в данной статье разработки.) Предлагаемая программа на ассемблере «SCREW Super Help» обеспечивает вывод на экран заданного пользователем произвольного текста в виде «выпадающего» окна в режиме как 32, так и 64 символа в строке. HELP-драйвер является резидентным и отслеживает прерывание от клавиатуры (через системную ячейку @#260, содержащую адрес подпрограммы, вызываемой стандартным драйвером прерывания от клавиатуры), поэтому подсказку можно получить в любой момент работы с БЕЙСИК-программой и даже после выхода в диалоговый режим БЕЙСИКа (лишь бы не было изменено содержимое ячейки @#260 и не запрещено прерывание от клавиатуры). Прежнее содержимое экрана после вызова и отмены окна с подсказкой полностью сохраняется, как и все установленные перед этим режимы вывода на экран.

Текст подсказки может содержать любое количество строк произвольной длины — HELP-драйвер самостоятельно определяет, какая часть текста умещается в пределах окна заданного пользователем размера. Оставшееся можно просмотреть, нажимая клавиши-«стрелки» (в том числе вместе с «АР2»), при этом текст «за окном» перемещается вправо, влево, вверх или вниз (т. е. реализован полнооконный интерфейс, аналогичный применяемому на IBM PC).

При работе с HELP-драйвером пользователю предоставлены широкие возможности по изменению характеристик вывода подсказки. Так, можно по своему желанию изменить каждый из нижеперечисленных параметров:

- размеры окна (так как прежнее содержимое экрана при выводе подсказки сохраняется в буфере, окно нельзя делать слишком большим);
- положение окна на экране (окно при заданных размерах не должно выходить за границы экрана);
- цвет рамки окна, фона и текста.

При необходимости можно перезадать клавишу вызова и отмены HELP (или выделить для этого две разные клавиши), адрес буфера для сохранения содержимого экрана, а также коды «конца строки» и «конца текста».

Работа с программой

Программа «SCREW Super Help» перемещаема и может быть загружена в БЕЙСИК с любого адреса оператором **BLOAD "SSHELP",R,&O<адрес>** (не забудьте предварительно освободить нужный участок памяти с помощью оператора CLEAR). Наличие параметра R при чтении обязательно, так как для инсталляции HELP-драйвера требуется запустить его с начального адреса. (Впрочем, можно просто загрузить его в ОЗУ, а при необходимости активизировать парой команд: DEF USR=&O‹начальный адрес› и A%=USR(A%).) Драйвер, тексты (их может быть несколько — драйвер обеспечивает возможность их переключения) и буфер, объем которого равен длине спрайта размером с окно (т. е. около 20 байт на каждую его символьную позицию), лучше всего размещать в старших адресах ОЗУ, например в таком порядке: *драйвер—текст—буфер*.

Текст подсказки может вводиться пользователем с помощью операторов DATA (как это показано в демонстрационной БЕЙСИК-программе) или в каком-либо текстовом редакторе с последующей подгрузкой оператором BLOAD. В последнем случае коды «конец строки» и «конец текста» нужно переустановить с используемых для БЕЙСИКа «♥» и «♣» на принятые в конкретном редакторе. (Каждая строка должна оканчиваться кодом «конец строки», а в последней строке кроме него должен стоять и код «конец текста».) «Подключение» уже находящегося в ОЗУ текста подсказки (их может быть и несколько, тогда каждый должен оканчиваться кодом «конец текста») производится с помощью автоматически определяемой во время инсталляции HELP-драйвера функции USR0:

I%=USR0(&O<адрес>) — указывает начальный адрес текста подсказки (при задании другого значения новый HELP-текст замещает старый);

I%=USR0(0%) — отключает вывод подсказки, пока не будет задан новый начальный адрес текста, и дополнительно возвращает абсолютный (с учетом реального адреса загрузки HELP-драйвера) адрес блока параметров (см. далее).

Вызов и отмена окна подсказки производятся нажатием клавиши «КТ» (при желании можно задействовать любую другую или две разные клавиши). После этого в окне появится левый верхний участок текста, умещающийся в окне. Теперь клавиши «ВВЕРХ» и «ВНИЗ» перемещают текст построчно, «ВПРАВО» и «ВЛЕВО» — на один символ по горизонтали, «АР2»+«ВВЕРХ» и «АР2»+«ВНИЗ» позволяют выполнить постраничное (по высоте окна) листание, а «АР2»+«ВЛЕВО» и «АР2»+«ВПРАВО» — переходить в начало и конец текста соответственно.

Установка параметров вывода окна может производиться с помощью операторов POKE (адрес начала блока параметров возвращает функция USR0(0%), смещения относительно этого адреса указаны в таблице).

Таблица

Адрес (относительно начала блока параметров)	Название	Назначение	Значение по умолчанию
0	HELP	Код клавиши вызова HELP	#3 («КТ»)
2	ESC	Код клавиши отмены HELP	#3 («КТ»)
4	X0	Координаты левого верхнего угла окна (X0=1...30 или 62, в зависимости от режима 32/64 символа в строке, Y0=1...22)	#1
6	Y0		#1
10_8	X	Размеры окна (символов х строк)	#10
12_8	Y		#2
14_8	BUFADR	Адрес буфера для сохранения участка ОЗУ экрана	#33000
16_8	BORDER	Цвет рамки окна	#52525 (синий)
20_8	TXTCOL	Цвет текста в окне	#177777 (красный)
22_8	FONCOL	Цвет фона окна	#0 (черный)
24_8	TXTHLP	Адрес начала HELP-текста	Заполняется с помощью USR0
26_8	LINCAR	Код «конец строки»	#242 («черви»)
30_8	ENDTXT	Код «конец текста»	#264 («крести»)

Примечания к таблице.

1. Цвет фона можно задавать не черным только в режиме 32 символа в строке (при 64 символах любое ненулевое значение цвета дает красный или белый цвет, в зависимости от типа используемого дисплея). Цвет рамки от режима 32/64 символа не зависит.

2. По умолчанию адрес начала текста в блоке параметров равен нулю, т. е. просто резервируется ячейка памяти. Реальное значение заносится в нее с помощью USR0.

3. Объем буфера зависит от размеров окна (X*Y) и равен 20 байт на каждую символьную позицию плюс запас для рамки.

Более удобным представляется использование группового задания с помощью функции USR1. Общий вид возможной записи:

A%=FONCOL%=(TXTCOL%=(BORDER%=(BUFADR%=(Y%=(X%=(Y0%=(X0%= USR1(<1..8>%))))))))

Первые ... параметров можно не указывать, убрав соответствующее количество скобок и уменьшив значение аргумента USR1. Вместо FONCOL%, TXTCOL% и т. д. нужно подставлять требуемые целочисленные или восьмеричные значения (коды «перевода каретки» и «конца текста» можно задать только через POKE); A% — некоторая «фиктивная» переменная, не используемая в программе для других целей. (Подробно об этом необычном способе использования функций USR для группового ввода значений можно прочитать в журнале «Вычислительная техника и ее применение», 1992, №3, с. 35.)

Примеры задания параметров (см. также листинг демонстрационной программы на БЕЙСИКе):

POKE USR0(0%)+&O22,&O125252 — задать зеленый фон окна (так как USR0(0%) отключает драйвер, нужно вновь включить его с помощью USR0(&O<адрес текста>));

A%=&O177777=(&O125252=(&O167356=(&O30000=(5%=(W0%=(10%=(10%=USR1(8%)))))))) — установить окно высотой в пять строк и шириной, заданной в переменной W0%. Окно размещается начиная с позиции (10,10). Текст зеленого цвета на красном фоне с желтой рамкой. Буфер для сохранения экранного ОЗУ начинается с адреса &O30000;

A%=2%=(2%=USR1(2%)) — переместить окно в позицию (2,2). Прочие параметры не меняются;

A%=3%=(10%=(2%=(2%=USR1(4%)))) — установить новый размер окна 10*3 символов. Хотя положение окна менять не предполагалось, пришлось лишний раз повторить задание координат верхнего левого угла, чтобы не изменился порядок следования параметров. Это одно из неудобств группового способа задания, поэтому при вводе какого-либо одного значения можно рекомендовать использовать POKE.

Замечания к листингу

Нижеприведенный ассемблерный листинг программы «SCREW Super Help» соответствует стандарту M18, во многом аналогичному ассемблеру-Микро. Кроме него приводится кодовый листинг, который можно ввести, например, с помощью режима ТС. Трансляция ассемблерного листинга производится с начального адреса 310008, но при необходимости можно выбрать другое подходящее значение.

Программа состоит из нескольких блоков, первый из которых («стартовый») производит автоматическую инсталляцию HELP-драйвера (автоопределение функций USR0 и USR1 и звуковая индикация успешного завершения инсталляции). Затем следует «тело» функций USR0 и USR1, после которых размещается стандартная подпрограмма расчета абсолютного адреса метки по величине смещения до нее (макрокоманда .@<имя метки>) и далее сам HELP-драйвер.

Листинг на ассемблере:

```
;
; SCREW (c) SUPER HELP 1992
; HELP-драйвер для БЕЙСИКа БК-0010.01
; ==================================
;
START¤: JSR PC,ADR0
        .@USR0
        MOV R0,@#2100
; DEF USR0
        JSR PC,ADR0
        .@USR1
        MOV R0,@#2102
; DEF USR1
        MOV #3,R1
; троекратный звуковой сигнал
B3:     MOV #100,R2
        MOV #200,R3
        JSR PC,@#102062
        SOB R1,B3
        RTS PC
;
; USR0(<adr>) - задание начального
;  адреса нового Help-текста
; USR0(0%) - сбрасывает Help-драйвер
;  и возвращает адрес начала блока
;  параметров (для операторов POKE)
USR0:   JSR PC,ADR0
        .@TXTHLP
        TST @R5
        BEQ B1
        MOV @R5,@R0
        JSR PC,ADR0
        .@STDRV¤
        MOV R0,@#260
        RTS PC
B1:     CLR @#260
```

```
        JSR PC,ADR0
        .@HELP
        MOV R0,@R5
        RTS PC
;
; Задание параметров ([...] -
;  необязательный текст):
;   A%=X0%=[(Y0%=[(X%=[(Y%=[(BUFADR%=
;  [(BORDER%=[(TXTCOL%=[(FONCOL%=]]]]]]]
;  USR1(<кол-во параметров, задаваемых
;  в этой строке>%)[])))))))]
USR1:   JSR PC,ADR0
        .@X0
        MOV @R5,R1
        MOV SP,R2
        ADD #4,R2
B2:     MOV (R2)+,(R0)+
        SOB R1,B2
        RTS PC
;
; стандартная п/пр расчета
;  абсолютного адреса
ADR0:   MOV @SP,R0
        ADD @R0,R0
        ADD #2,@SP
        RTS PC
;
STDRVп: CMPB HELP,R0
; HELP - код клавиши вызова Help-окна
        BEQ QST
        RTS PC
QST:    EMT 6
        CLR TRIM
; вспомогательная переменная
        MOV #1,NSTR
; первая строка
; ----------------------
;   Отключение драйвера от
;    самого себя и сохранение в стеке
;    режимов работы дисплея
        MOV @#260,-(SP)
        CLR @#260
        MOV #40,R1
        MOV #20,R2
A1:     MOVB (R1)+,-(SP)
        SOB R2,A1
        MOV @#212,-(SP)
        MOV @#214,-(SP)
        EMT 26
        MOV R1,-(SP)
        MOV R2,-(SP)
; ----------------------
;  Звуковая индикация вызова Help
        MOV #127,R3
        MOV #120,R2
        JSR PC,@#102062
;  Погасить курсор
        TSTB @#56
        BNE STRT
        MOV #232,R0
        EMT 16
; ----------------------
;  (X0,Y0) - координаты верхнего
;   левого угла окна
STRT:   MOV X0,R1
        MOV Y0,R2
        TST Y0
        BEQ A2
        MOV #4,@#256
        DEC R2
;  Позиционирование окна и
;   определение адреса начала
;   сохраняемой зоны экрана
A4:     EMT 24
        MOV @#160,R0
        ADD #600,R0
        CMP #100000,R0
        BPL A3
        BIC #100000,R0
        ADD #40000,R0
        BR A3
A2:     CLR @#256
        BR A4
A3:     CLR SYM
        TST X0
        BEQ A5
        DEC R0
        INC SYM
A5:     MOV BUFADR,R1
; Адрес буфера для сохранения экрана
        MOV Y,R2
; Y-высота окна, текстовых строк
; ----------------------
;   Умножение R2 на 12 (восьм.)
        ASL R2
        MOV R2,R3
        ASL R2
        ASL R2
        ADD R3,R2
; ----------------------
        ADD #4,R2
        ADD @#256,R2
        MOV X,R3
; X - ширина окна в символах
        TSTB @#40
        BEQ A6
        ASL R3
A6:     ADD SYM,R3
        MOV R3,R4
        BIC #177400,R4
        BEQ A7
        INC R3
A7:     MOV R3,R4
A11:    TST NSTR
        BEQ A8
; Сохранение экрана
        MOVB @R0,(R1)+
        MOVB BORDER,(R0)+
; BORDER-цвет рамки окна (байт)
        BR A9
; Восстановление экрана
A8:     MOVB (R1)+,(R0)+
```

```
A9:     SOB R4,A11
        ADD #100,R0
        SUB R3,R0
        CMP #100000,R0
        BPL A10
        BIC #100000,R0
        ADD #40000,R0
A10:    SOB R2,A7
        TST NSTR
        BEQ QUIT
        MOV TXTCOL,@#214
        MOV FONCOL,@#212
; TXTCOL - цвет текста в окне
; FONCOL - цвет фона окна
        BR WORK
QUIT:   MOV (SP)+,R2
        MOV (SP)+,R1
        EMT 24
; Восстановление режимов работы
;  дисплея
        MOV (SP)+,@#214
        MOV (SP)+,@#212
        MOV #60,R1
        MOV #20,R2
A13:    MOVB (SP)+,-(R1)
        SOB R2,A13
; Подключение Help-драйвера
;  и выход
        MOV (SP)+,@#260
        RTS PC
;
WORK:   MOV TXTHLP,R5
; TXTHLP - адрес начала Help-текста
        CLR SYM
        MOV #1,R3
A14:    CMP R3,NSTR
        BEQ A16
A15:    CMPB LINCAR,(R5)+
; LINCAR - байт "конец строки"
        BNE A15
        CMPB ENDTXT,@R5
; ENDTXT - байт "конец текста"
        BEQ A17
        INC R3
        BR A14
A17:    DEC NSTR
        BR KOMMND
A16:    MOV #40,R0
        MOV Y0,R2
        MOV R2,R4
        ADD Y,R4
A18:    MOV X0,R1
        MOV X,R3
        EMT 24
A19:    EMT 16
        SOB R3,A19
        INC R2
        CMP R4,R2
        BNE A18
        MOV X0,R1
        MOV Y0,R2
A20:    EMT 24
        MOV TRIM,R4
        BEQ A23
A21:    CMPB LINCAR,(R5)+
        BEQ A22
        SOB R4,A21
A23:    CLR R4
A235:   MOVB (R5)+,R0
        BIC #177400,R0
        CMPB R0,LINCAR
        BNE A24
A22:    INC SYM
        CMP SYM,Y
        BGE KOMMND
        CMPB ENDTXT,@R5
        BEQ KOMMND
        INC R2
        BR A20
A24:    CMP R4,X
        BGE A236
        EMT 16
        INC R4
        BR A235
A236:   CMPB (R5)+,LINCAR
        BNE A236
        BR A22
KOMMND: CLR SYM
        EMT 6
; Ввод команды пользователя
        CMPB R0,ESC
; ESC-код клавиши выхода
        BNE A30
        CLR NSTR
        JMP STRT
A30:    CMPB R0,#10
        BNE A31
; "стрелка влево":
;  сдвиг текста в окне
;  на 1 символ влево
        DEC TRIM
        BGE A32
        INC TRIM
        BR KOMMND
A32:    JMP WORK
A31:    CMPB R0,#31
        BNE A33
; "стрелка вправо":
;  сдвиг текста в окне
;  на 1 символ вправо
        INC TRIM
        JMP WORK
A33:    CMPB R0,#32
        BNE A34
; "стрелка вверх":
;  на одну строку вверх
        DEC NSTR
        BNE A35
        INC NSTR
        BR KOMMND
A35:    JMP WORK
A34:    CMPB R0,#33
```

```
        BNE A36
; "стрелка вниз":
;  на одну строку вниз
        INC NSTR
        JMP WORK
A36:    CMPB R0,#232
        BNE A37
;  "AP2"+"стрелка вверх":
;  на одну страницу назад
        SUB Y,NSTR
        BGT A38
A42:    MOV #1,NSTR
A38:    JMP WORK
A37:    CMPB R0,#233
        BNE A39
;  "AP2"+"стрелка вниз":
;  на одну страницу вперед
A43:    JSR PC,FNDKT
; определение количества
;  строк текста
        ADD Y,NSTR
        SUB Y,R0
        INC R0
        CMP NSTR,R0
        BLE A40
        MOV R0,NSTR
A40:    JMP WORK
A39:    CMPB R0,#230
        BNE A41
;  "AP2"+"стрелка влево":
;  переход в начало текста
        BR A42
A41:    CMPB R0,#231
        BNE KOMMND
;  "AP2"+"стрелка вправо":
;  переход в конец текста
        MOV #77777,NSTR
; сделали так, чтобы условие:
;  "NSTR+Y больше кол-ва
;  строк в тексте" всегда
;  выполнялось
        SUB Y,NSTR
        DEC NSTR
        BR A43

; П/пр определения
;  кол-ва строк текста
FNDKT:  MOV TXTHLP,R5
        CLR R0
A45:    CMPB LINCAR,(R5)+
        BNE A45
        INC R0
        CMPB ENDTXT,@R5
        BNE A45
        RTS PC
;
;  Вспомогательные ячейки
SYM:    .#0
TRIM:   .#0
NSTR:   .#0
;
;  Блок параметров
; (значения по умолчанию)
HELP:   .#3
; вызов окна по "КТ"
ESC:    .#3
; отмена по "КТ"
X0:     .#1
Y0:     .#1
; по умолчанию - левый верхний
;  угол экрана
X:      .#10
Y:      .#2
; окно размером 10*2
BUFADR: .#33000
; буфер по адресам 33000-40000
BORDER: .#52525
; синяя рамка
TXTCOL: .#177777
; красный текст
FONCOL: .#0
; черный фон
TXTHLP: .#0
LINCAR: .#242
; "Черви" ("♥", "AP2"+"Б"), код 242
ENDTXT: .#264
; "Крести" ("♣", "AP2"+"Т"), код 264
; ------------------------------------
```

Листинг в машинных кодах:

Начальный адрес: 31000_8 Длина: 1432_8

```
004767  000136  000044  010037  002100  004767  000124  000100
010037  002102  012701  000003  012702  000100  012703  000200
004737  102062  077107  000207  004767  000066  001350  005715
001407  011510  004767  000052  000064  010037  000260  000207
005037  000260  004767  000032  001270  010015  000207  004767
000020  001262  011501  010602  062702  000004  012220  077102
000207  011600  061000  062716  000002  000207  126700  001220
001401  000207  104006  005067  001202  012767  000001  001176
013746  000260  005037  000260  012701  000040  012702  000020
112146  077202  013746  000212  013746  000214  104026  010146
010246  012703  000127  012702  000120  004737  102062  105737
```

```
000056  001003  012700  000232  104016  016701  001106  016702
001104  005767  001100  001421  012737  000004  000256  005302
104024  013700  000160  062700  000600  022700  100000  100010
042700  100000  062700  040000  000403  005037  000256  000760
005067  001006  005767  001014  001403  005300  005267  000772
016701  001010  016702  001002  006302  010203  006302  006302
060302  062702  000004  063702  000256  016703  000752  105737
000040  001401  006303  006703  000720  010304  042704  177400
001401  005203  010304  005767  000704  001404  111021  116720
000714  000401  112120  077411  062700  000100  160300  022700
100000  100004  042700  100000  062700  040000  077225  005767
000634  001407  016737  000650  000214  016737  000644  000212
000420  012602  012601  104024  012637  000214  012637  000212
012701  000060  012702  000020  112641  077202  012637  000260
000207  016705  000576  005067  000540  012703  000001  020367
000534  001413  126725  000556  001375  126715  000552  001402
005203  000765  005367  000506  000471  012700  000040  016702
000504  010204  066704  000502  016701  000470  016703  000470
104024  104016  077302  005202  020402  001366  016701  000444
016702  000442  104024  016704  000422  001404  126725  000446
001410  077404  005004  112500  042700  177400  120067  000426
001013  005267  000364  026767  000360  000376  002017  126715
000406  001414  005202  000746  020467  000354  002003  104016
005204  000751  122567  000356  001375  000753  005067  000312
104006  120067  000314  001004  005067  000302  000167  177172
120027  000010  001010  005367  000262  002003  005267  000254
000755  000167  177474  120027  000031  001004  005267  000234
000167  177456  120027  000032  001010  005367  000220  001003
005267  000212  000733  000167  177430  120027  000033  001004
005267  000172  000167  177412  120027  000232  001011  166767
000170  000152  003003  012767  000001  000142  000167  177362
120027  000233  001017  004767  000072  066767  000134  000116
166700  000126  005200  026700  000104  003402  010667  000076
000167  177316  120027  000230  001001  000745  120027  000231
001255  012767  077777  000046  166767  000056  000040  005367
000034  000741  016705  000054  005000  126725  000050  001375
005200  126715  000042  001371  000207  000000  000000  000000
000003  000003  000001  000001  000010  000002  033000  052525
177777  000000  000000  000242  000264  000000  000000  000000

Контрольная сумма:3
```

Демонстрационная программа на БЕЙСИКе:

```
1 ' Демонстрация работы SCREW Super Help
2 ' Не забудьте ввести CLEAR 200,&O30000
3 ' и BLOAD"SSHELP",R,&O31000 !
10 CLEAR 1000
20 CLS
30 ? AT(6,1);CHR¤(146);"Демонстрация
   работы";
35 ? AT(7,2);"SCREW Super Help.";CHR¤(145)
40 ?STRING¤(32,"-");
50 TA%=&O30000
60 READ A¤
70 FOR I%=1% TO LEN(A¤) STEP 2%
80 IF I%=LEN(A¤) THEN J#=0
   ELSE J#=ASC(MID¤(A¤,I%+1%,1%))
90 J#=J#*256%+ASC(MID¤(A¤,I%,1%))
100 IF J#>=32767 THEN J#=J#-65536
110 I2%=TA%+I%-1%
120 POKE I2%,CINT(J#)
130 NEXT I%
140 TA%=I2%+2%
150 IF MID¤(A¤,LEN(A¤),1%)<>"♣" THEN 60
160 TA%=&O30000
170 TA%=USR(TA%)
180 ? CHR¤(155);"Нажмите ВВОД для
    продолжения, КТ-HELP";
190 ? CHR¤(155)
200 IF INKEY¤<>CHR¤(10) GOTO 200
210 ?
220 ? "Позиция окна:"
230 INPUT "X0 <1...10> ";X0%
240 IF X0%<1 OR X0%>10 GOTO 230
250 INPUT "Y0 <1...15> ";Y0%
260 IF Y0%<1 OR Y0%>15 GOTO 250
270 ? "Размеры окна:"
```

```
280 INPUT "X  <5...20> ";X%
290 IF X%<5 OR X%>20 GOTO 280
300 INPUT "Y  <1...5>  ";Y%
310 IF Y%<1 OR Y%>5 GOTO 300
320 ? "Цвет <1,2,3,4>:"
330 INPUT "Цвет рамки ";BORDER%
340 IF BORDER%<1% OR BORDER%>4%
    GOTO 330
350 ? CHR$(144+BORDER%);
360 BORDER%=PEEK(&O214)
370 ? CHR$(145);
380 INPUT "Цвет текста";TXTCOL%
390 IF TXTCOL%<1% OR TXTCOL%>4%
    GOTO 380
400 ? CHR$(144+TXTCOL%);
410 TXTCOL%=PEEK(&O214)
420 ? CHR$(145);
430 INPUT "Цвет фона  ";FONCOL%
440 IF FONCOL%<1% OR FONCOL%>4%
    GOTO 430
450 ? CHR$(144+FONCOL%);
460 LET FONCOL%=PEEK(&O214)
470 ? CHR$(145);
480 BUFFER%=&O33000
490 TA%=FONCOL%=(TXTCOL%=
    (BORDER%=(BUFFER%=(Y%=(X%=
    (Y0%=(X0%=USR1(8%)))))))
500 TA%=&O30000
510 GOTO 180
1000 DATA "    Пример HELP-текста.♥"
1010 DATA " Вы можете перемещать текст♥"
1020 DATA "в окне с помощью клавиш: ♥ "
1030 DATA " - <влево> - <вправо>.♥"
1040 DATA " - <вверх> - <вниз>.♥"
1050 DATA " - AP2+<вверх>, AP2+<вниз>.♥"
1060 DATA " - AP2+<влево>, AP2+<вправо>.♥"
1070 DATA "♥ ♥ Для выхода из HELP♥"
1080 DATA " нажмите 'КТ' ♥"
1090 DATA "♥  Конец текста.♥♣ "
```

Примечания.

1. Из-за несовершенства транслятора БЕЙСИКа, не позволяющего в операторах POKE адресоваться к отдельным байтам, при вводе текста по упрощенному алгоритму «READ-DATA-POKE» (как это сделано в демонстрационной программе) строки нечетной длины заканчиваются нулевыми байтами. При просмотре HELP-текста это иногда приводит к нарушению выравнивания строк по левому краю. Если же подготавливать текст в каком-либо текстовом редакторе (или с помощью отладчика типа DEBU10 либо аналогичного), записывать в файл <имя>.BIN, а затем загружать в БЕЙСИК оператором BLOAD, нарушений выравнивания не происходит. (Правильно подготовленный текст не должен содержать в себе нулевые байты, за исключением случаев, когда они используются в качестве кодов «перевода строк».)

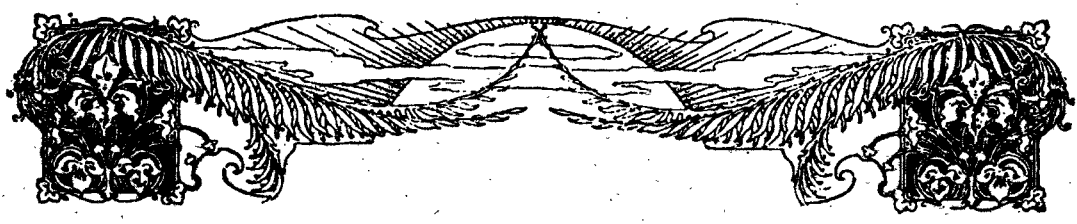
2. При желании можно использовать программу Screw Super HELP и в ассемблерных разработках. В этом случае нужно убрать из листинга SSHELP несколько строк от начала до метки ADR0 (не включая строку с этой меткой), а подключение HELP к основной программе (как и в БЕЙСИКе, SSHELP автоматически отслеживает нажатие клавиши вызова подсказки) производить следующими командами:

```
MOV #<адрес текста>,TXTHLP
MOV #STDRV0,@#260
```

Отключение HELP-драйвера производится командой CLR @#260.

При необходимости можно в процессе работы основной программы менять текст подсказки (чтобы сделать ее «контекстной»), записывая требуемый начальный адрес текста в ячейку TXTHLP. Кроме того, можно произвольно менять характеристики вывода окна, обращаясь к соответствующим ячейкам блока параметров.

При работе в ассемблере необходимо, как и в БЕЙСИКе, учитывать упомянутые в статье ограничения, в частности, разрешение прерывания от клавиатуры и необходимость наличия буфера для сохранения участка видеоОЗУ под выводимым окном.



СПРАВОЧНЫЙ ЛИСТОК

А. И. Кузнецов,
Москва

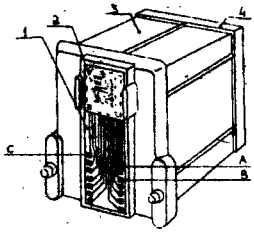
Термоструйные печатающие головки серии «Электроника МС-6902» (для принтеров типа 6312)

В 1989—1990 гг. московским заводом «Эра» (АО НПП «Радий») было начато серийное производство малогабаритных струйных принтеров серии 6312 и сменных термоструйных печатающих головок (ТСПГ) для них.

Печатающие головки «Электроника МС-6902.02» и «Электроника МС-6902.03», выпускающиеся в настоящее время, являются усовершенствованной модификацией более старых типов «Электроника МС-6902» и «Электроника МС-6902.01».

В первоначальной модели ТСПГ «Электроника МС-6902», в отличие от последующих модификаций, применялись более толстые прижимные площадки, поэтому при переходе от «МС-6902» к «МС-6902.01», «МС-6902.02» или «МС-6902.03» желательно использовать другой «замок каретки» (обычно он продается в комплекте с печатающей головкой). По габаритным и посадочным размерам ТСПГ «Электроника МС-6902.01», «МС-6902.02» и «МС-6902.03» соответствуют аналогичным изделиям фирмы «Hewlett Packard» (N 51604A; 51605R, G, B; 51616A; 92261A и др.), что позволяет использовать последние в принтерах серии 6312 (при наличии в принтере напряжения U=22 В). Печатающие головки «Электроника МС-6902», «МС-6902.01» и «МС-6902.02» рассчитаны на рабочее напряжение 16.2—17.0 В, а модель «Электроника МС-6902.03» — на напряжение 22.2—23.0 В. В настоящее время ТСПГ типов «Электроника МС-6902» и «МС-6902.01» сняты с производства как устаревшие модели.

Конструктивно ТСПГ «Электроника МС-6902.01», «МС-6902.02» и «МС-6902.03» состоят из резистивной матрицы с камерообразующей пластиной, корпуса, задней крышки, губчатого наполнителя — емкости для чернил и системы фильтров (см. рисунок). Наполнитель и фильтры (на рисунке не показаны) находятся внутри корпуса, который неразъемно соединен с крышкой посредством клеевого шва. На лицевой (рабочей) поверхности корпуса размещается резистивная матрица в сборе, зафиксированная и герметизированная клеевым соединением.



Общий вид ТСПГ МС-6902.

1 — резистивная матрица; 2 — камерообразующая пластина; 3 — корпус; 4 — крышка; A, B, C — контактные площадки: A — вывод общей шины, B и C — выводы первого и двенадцатого резистивных элементов соответственно (нумерация контактов ведется по часовой стрелке).

Печать изображения (графики и символов) в виде отдельных точек производится за счет «выстреливания» микрокапель чернил из сопел камерообразующей пластины в определенной последовательности на поверхность бумаги. «Выстреливание» же микрокапель происходит при нагревании соответствующих резистивных элементов матрицы, что вызывает вскипание порции чернил над ними и образование избыточного давления между пластиной и матрицей. Для нагревания нужного резистивного элемента через контактные площадки подаются импульсы напряжения от схемы управления принтера. Всего в матрице имеется двенадцать резистивных элементов.

Рабочие параметры и режимы эксплуатации ТСПГ «МС-6902.02» и «МС-6902.03»

- Температура окружающего воздуха (°С): от +10 до +55.
- Относительная влажность воздуха (%): от 50 до 80.
- Атмосферное давление (кПа): 84—106 (630—800 мм рт. ст.).

- Отклонение центров отдельных точек от вертикали и горизонтали (мм): не более 0.15.
- Разрешающая способность при печати (точек на дюйм): 96.

Примечание. Выполнение последних двух требований обеспечивается при расстоянии между соплами печатающей головки и поверхностью бумаги не более 1.5 мм.

- Энергия, выделяемая в одном резистивном нагревательном элементе (РНЭ) за один рабочий импульс (Вт*мкс): от 41.0 до 56.0 («МС-6902.02») и от 63.3 до 75.0 («МС-6902.03»).
- Величина сопротивления РНЭ (Ом): от 28.0 до 34.0 («МС-6902.02») и от 43.0 до 46.0 («МС-6902.03»).
- Напряжение рабочего импульса (В): от 16.2 до 17.0 («МС-6902.02») и от 22.2 до 23.0 («МС-6902.03»).
- Длительность рабочего импульса по уровню 0.5 (мкс): от 5.8 до 6.2 («МС-6902.02») и от 5.8 до 6.2 («МС-6902.03»).
- Частота следования рабочего импульса (Гц): не более 2500.
- Масса головки в сборе, заправленной чернилами (г): от 14.5 до 15.5.

Печатающая головка должна обеспечивать вывод через два интервала не менее 600 страниц формата А4 полного текста Шеннона (ГОСТ 21776), что соответствует 2 000 000 чернильным каплям на каждый РНЭ резистивной матрицы.

Критерии отказа ТСПГ

Критерием отказа печатающей головки считается отсутствие выброса микрокапель чернил в любом из 12 сопел головки, определяемое визуально в виде непрерывного пробела при выводе текстовой или графической информации. Отказом не считается временное отсутствие выброса микрокапель чернил, если в результате промывки ТСПГ (см. «Указания по эксплуатации») ее функционирование полностью восстановилось, а также израсходование чернил, заправленных на заводе-изготовителе, в течение гарантийного срока.

Гарантийному обслуживанию не подлежат ТСПГ после повторной заправки или дозаправки чернил, при механических и электрических (перегорание более одного терморезистора РНЭ) повреждениях.

Примечание. Завод-изготовитель замену израсходованных печатающих головок не производит.

Указания по эксплуатации ТСПГ

1. Протрите смоченной водой тканью желтые площадки на печатающей головке и на каретке принтера.
2. Поверните ТСПГ металлической пластинкой (лицевой поверхностью) вниз.
3. Прижмите без усилий смоченную водой ткань к металлической пластинке и подержите ее несколько секунд до появления на ткани чернильного пятна (использование других жидкостей для увлажнения ткани запрещается).
4. Установите ТСПГ в принтер в соответствии с инструкцией по эксплуатации последнего.
5. В перерывах между сеансами печати храните головку в плотно закрытой таре (заводской упаковке) в прохладном месте.

Возможные неисправности ТСПГ

Механические повреждения корпуса ТСПГ (3) или камерообразующей пластины (1). Чаще всего оказывается повреждена керамическая камерообразующая пластина (1) при установке ТСПГ в принтер из-за чрезмерного ее прижима замком каретки (здесь и далее цифры и буквы в скобках соответствуют обозначениям на рисунке).

Электрические повреждения резистивных элементов матрицы чаще всего возникают при подаче на них повышенного напряжения или из-за неисправности самого принтера. Работоспособность резистивных элементов можно проверить измерением сопротивления между их выводами (В—С) и общей шиной (А).

Разгерметизация клеевых соединений проявляется в виде трещин в клеевых швах крышки, камерообразующей пластины или резистивной матрицы.

Засорение резистивных элементов приводит к отсутствию или неравномерности выброса микрокапель чернил резистивными элементами головки при печати информации, если их работоспособность после выполнения указаний по эксплуатации ТСПГ не восстанавливается.

Гарантийный срок использования печатающей головки установлен равным 10 месяцам со дня изготовления.

Возможность повторной заправки ТСПГ

Печатающая головка согласно техническим условиям рассчитана на одну (заводскую) заправку специальными чернилами для термоструйной печати, но реально каждый РНЭ может выдержать выброс 6 000 000 чернильных микрокапель (втрое больше, чем при однократной заправке). При выполнении нижеследующих рекомендаций можно попытаться продлить срок службы ТСПГ. (**Примечание.** Завод-изготовитель продажу чернил и повторную заправку ТСПГ не производит.)

1. Прежде всего следует убедиться в исправности печатающей головки, используемой для повторной заправки.

2. Лучше всего производить периодическую дозаправку ТСПГ по мере расходования чернил порциями приблизительно по 2—3 $см^3$ (200—300 страниц формата А4 текста Шеннона ГОСТ 21776, количество страниц графического изображения зависит от степени «черноты» рисунка).

3. Чернила, предназначенные для повторной заправки, должны по химическому составу соответствовать заводским. Хранить их необходимо в прохладном месте в плотно закрытой таре, не допуская прямого попадания на них солнечных лучей. Применение старых чернил нежелательно. При обнаружении осадка чернила следует не перемешивать, а профильтровать, чтобы отделить осадок.

4. Качество печати старых ТСПГ после их длительной эксплуатации при повторной заправке может оказаться значительно худшим по сравнению с новыми головками, так как РНЭ в процессе работы постепенно теряют свои качества (в них образуется нагар, что приводит к их засорению). Длительное использование ТСПГ с непропечаткой отдельных РНЭ также приводит к их подгоранию и ухудшению качества печати.

5. Заправка производится при помощи медицинского шприца с диаметром иглы 0.4—0.6 мм и рабочим объемом не более 5 $см^3$ путем прокола наполнителя до упора через технологическое отверстие в крышке ТСПГ. Плавно надавливая на поршень шприца, впрыскивайте чернила из шприца в головку. Попадание при заправке воздуха вместе с чернилами в шприц и в ТСПГ не допускается.

Примечание. При вытекании чернил из заправочного отверстия до полного опорожнения шприца заправку нужно прекратить.

Возможен и другой, более эффективный, но длительный способ заправки ТСПГ: чернила наливаются в любую емкость — поддон и головка лицевой (рабочей) частью опускается в чернила. У заправочного технологического отверстия в крышке ТСПГ создается разрежение (например, при помощи медицинской резиновой груши), за счет чего чернила из поддона всасываются в головку через камерообразующую пластину и резистивную матрицу. Преимущество этого способа заключается в невозможности проникновения пузырьков воздуха в ТСПГ и в попутной промывке сопел камерообразующей пластины потоком проходящих через них чернил.

6. Вес повторно заправленной ТСПГ не должен превышать 15.5 г.

7. Запрещается производить чистку ТСПГ спиртом, ацетоном и другими растворителями.

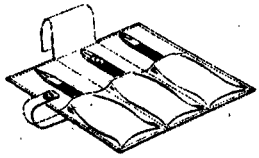
Примечание. Изменение номинала рабочего напряжения современных ТСПГ «МС-6902.03» по сравнению со старыми типами головок могут привести к непропечатке вертикальных линий на принтерах ранней модификации МС-6312. Несложной доработкой принтера этот недостаток можно устранить, как и реализовать подачу рабочего напряжения для ТСПГ фирмы «Hewlett Packard», так как это технологически предусмотрено в любой модификации принтера: «МС-6312», «MS(MP)-6312» или «MS-6312M»*.

Покупаю, продаю, ремонтирую и дорабатываю принтеры серии 6312.

Продаю печатающие головки «МС-6902» для принтеров 6312, кассовых аппаратов «ЭКР 3102» (г. Курск), «ИВКО» (Москва) и специальные чернила промышленного изготовления для термоструйной печати для повторной заправки головок.

Обращайтесь по адресу:
113447, Москва, а/я 5, Кузнецову Александру Ивановичу.
Телефон: (095) 129-58-85.

* Подобная доработка описана в журнале «Персональный компьютер БК-0010 — БК-0011М» №4 за 1995 г. — *Прим. ред.*



За последние два-три года дисковод стал, пожалуй, неотъемлемой частью большинства домашних вычислительных комплексов на основе БК. Учитывая значительный интерес читателей к этой теме, публикуем рекомендации по установке контроллера дисковода внутрь корпуса БК (что делает работу с компьютером более удобной), а также описание способа устранения ошибки, допущенной в схеме КНГМД заводом-изготовителем.

С. М. Неробеев, А. В. Сорокин,
Москва

Доработки контроллера дисковода

Доработка КНГМД от зависания

Контроллер НГМД БК-0011(М) разрабатывался как внешнее устройство и имеет собственный тактовый генератор, что делает его независимым от тактовой частоты компьютера (именно благодаря этому он без проблем подключается к БК-0010(.01)). Однако вопрос синхронизации вычислительной машины с КНГМД был продуман разработчиками недостаточно хорошо. Так, на БК-0011(М), где компьютер и контроллер работают на одинаковых частотах, при обращении к дисководу ЭВМ иногда «зависает», так как процессор не успевает принять сигнал ответа от контроллера.

Одно из ранее предлагавшихся «радикальных» решений состоит в следующем: внутренний генератор контроллера отключается совсем, а тактовые импульсы подаются с компьютера, благо на разъеме МПИ БК-0011(М) имеется выход сигнала тактовой частоты. Такой способ наиболее прост, но, к сожалению, не универсален: доработанный контроллер нельзя подключать к БК-0010(.01), так как на системном разъеме десятой модели нет выхода тактовой частоты. Конечно, можно снять этот сигнал непосредственно с платы вычислителя, но тогда контроллер окажется привязан к конкретной ЭВМ и появится зависимость КНГМД от ее тактовой частоты.

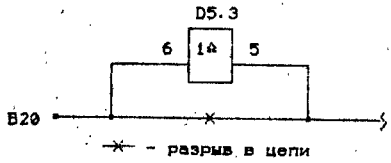


Рис. 1

Нами предлагается схемотехнически более грамотное решение данной проблемы (схема доработки приведена на рис. 1). Для предотвращения зависания вычислительной машины нужно задержать сигнал ответа RPLY. Цепь задержки реализуется на одном из свободных элементов микросхемы К155ЛП9 (D5), т. е. никаких дополнительных деталей на плату КНГМД устанавливать не требуется.. Дорожку на печатной плате следует перерезать непосредственно вблизи контакта B20.

Установка КНГМД внутрь корпуса БК-0010/11

Поскольку БК имеет единственный разъем МПИ, который к тому же при работе с дисководом занят КНГМД, подключение к системной шине дополнительных внешних устройств представляет определенную сложность. Между тем подобные устройства уже существуют, например модем, который стыкуется с БК через блок ИРПС.

Известно несколько способов решения этой задачи. Чаще всего для этого используются всевозможные разветвители и переходники, подключаемые к разъему МПИ и имеющие на выходе целую линейку запараллеленных разъемов. Но, не говоря уже об увеличении габаритов вычислительной машины и повышенной механической нагрузке на разъем МПИ, при этом неизбежны конфликты между различными внешними устройствами, занимающими одно и то же адресное пространство.

Другой способ — установка дополнительного внешнего устройства в корпус вычислительной машины с возможностью его отключения, что неоднократно опробовано пользователями БК-0010.01 для блока МСТД. Недостатки такого решения очевидны: оно возможно только для небольшого по размерам внешнего устройства, причем последнее (как правило, достаточно

редко используемое) постоянно подключено к источнику питания БК, а для полного отключения устройства требуются крупногабаритные переключатели.

По мнению авторов, наиболее рациональным способом является установка КНГМД внутрь корпуса БК, поскольку контроллер дисковода является постоянно действующим внешним устройством и вычислительная машина без него практически не используется. При этом значительно уменьшаются габариты всей конструкции в целом, а освободившийся разъем МПИ можно использовать для подключения других внешних устройств.

Сама по себе установка контроллера в корпус вычислительной машины не представляет сложности. Для этого надо предварительно выпаять из КНГМД разъем, посредством которого он стыкуется с системной шиной, и подключить контроллер непосредственно к цепям на плате БК. Все сигналы (за исключением DOUT, INIT и «Выбор устройства») снимаются с микросхемы DS17.

Далее необходимо обеспечить совместимость КНГМД с другими внешними модулями, для чего следует предусмотреть возможность отключения контроллера, если подстыковываемое к разъему МПИ устройство занимает адресное пространство 160000—174000 или устанавливает свои регистры по адресам 177730, 177132 (таким устройством может быть другой контроллер дисковода или блок МСТД). Соответствующая доработка показана на рис. 2. Микросхема DD1 устанавливается на плате КНГМД поверх микросхемы D8 (К555ТЛ2), все ее выводы, кроме выводов питания, отгибаются в стороны. Микросхема D11 (КР1801ПЕ2-326) находится на плате контроллера, а DS17 (КР1801ПЕ2) — на плате БК-0010.01 или БК-0011М (расположение DS17 не зависит от модели ЭВМ). Перед сборкой следует отсоединить контакт 23 микросхемы ПЗУ КНГМД компьютера и в разрыв впаять резистор с сопротивлением приблизительно 1 кОм.

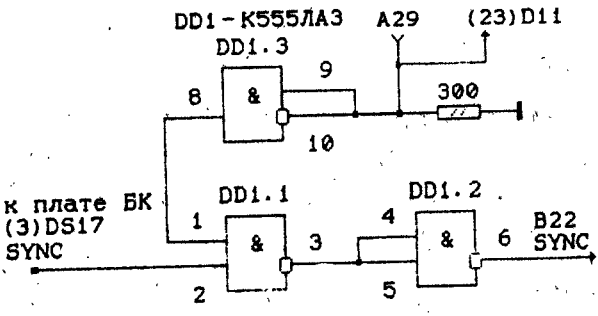


Рис. 2

Чтобы при подключении внешних устройств происходило отключение встроенного в корпус контроллера, нужно сделать еще одну доработку, теперь уже на внешнем устройстве, которое может войти в конфликт с контроллером дисковода. Требуется подключить резистор с сопротивлением примерно 330 Ом между контактами В12 и А29 разъема внешнего устройства. (Повторяем, что это следует делать только в том случае, если внешнее устройство занимает одинаковые с контроллером дисковода адреса.)

Размещение платы контроллера в корпусе вычислительной машины показано на рис. 3. Над разъемом МПИ в корпусе нужно любым способом (например, с помощью прибора для выжигания или лобзика с последующей зачисткой краев) сделать прямоугольное отверстие под выходной разъем контроллера для подключения шлейфа дисковода. При установке КНГМД над платой БК особенно внимательно следите за тем, чтобы не нарушился тепловой режим микросхем ОЗУ.

В заключение отметим, что устанавливать контроллер дисковода в корпус БК с «жесткой» клавиатурой (старого типа) не рекомендуется, так как пространство между печатной платой и нижним краем клавиатуры слишком мало.

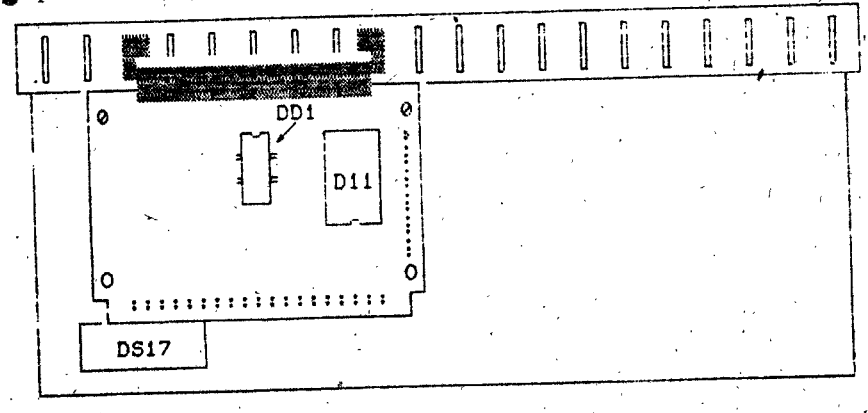


Рис. 3

Похоже, программисты-БКманы явно сговорились отметить десятилетний юбилей БК-0010 наиболее знаменательными достижениями в своем интеллектуальном творчестве — впервые на БК-0011М реализована полноценная издательская система со всеми ее атрибутами: WYSIWYG, совместная верстка текста и графики, загружаемые шрифты различного начертания, возможность подготовки макета страницы формата А4, ее просмотра и печати на принтере. Конечно, это только первый опыт разработки на БК программных средств такого класса, поэтому будем снисходительны к некоторым недостаткам программы БК-PageMaker (очень небольшими по сравнению с предоставляемыми возможностями). А в будущем — как знать? — может быть, БКшка стараниями энтузиастов сможет заметно потеснить «Эпплы» и «Айбиэмы» на поприще издательской деятельности.

М. И. Кондратович,

г. Минск

Малая издательская система БК-PageMaker

В течение всего 1995 года автора этой статьи не оставляло желание создать полноценную издательскую систему для БК. Под этим понятием правильнее всего было бы понимать как можно более полную реализацию принципа WYSIWYG («видеть на экране все именно так, как позднее будет напечатано на принтере»), возможность использования графических шрифтов различного начертания, вставки в текст иллюстраций и т. д. Отметим, что на БК до сих пор за подобный проект никто не брался. (Даже всем известный VorteX 4.0 — это всего только мощный текстовый редактор, ведь он не позволяет вставлять в текст даже простейшие иллюстрации.)

Потратив несколько месяцев, автор смог в достаточной мере приблизиться к желаемой цели, насколько близко — судить пользователям.

Сразу оговоримся, что данная программа ПОКА не претендует на звание «профессиональной». При ее создании автор не ставил своей целью реализовать на БК все возможности одноименной системы с IBM PC. Прототипом БК-PageMaker, скорее, стала издательская система POLISS (см. ее описание в журнале «Информатика и образование», №2 за 1992 г., с. 83, 85) для УКНЦ. Само слово «Pagemaker» означает в дословном переводе с английского «создатель страниц» и этому названию описываемая программа отвечает как нельзя лучше: ведь ее основное предназначение — распечатывать на принтере полностью сверстанную страницу формата А4.

Издательская система БК-PageMaker (далее — BPM) может быть использована при подготовке разного рода документов методического, рекламного, технического и учебного характера, удобна она для создания и тиражирования различных бланков (счетов, доверенностей, накладных), а также любых других печатных материалов, включающих в себя текст и графику (см. рисунок — календарь на 1996 г.).

Для функционирования программы BPM необходимы следующие аппаратные средства:

- микроЭВМ БК-0011М,
- операционная система ANDOS версии 3.1,
- принтер, совместимый по набору ESC-команд со стандартом EPSON (Epson FX, Robotron D100M, MC-6313, MC-6312 и т. д.).

Издательская система состоит из загрузочного модуля BPM и оверлеев BPM.OVL (основной) и HELP_BPM.OVL (встроенная подсказка). Все они должны находиться на рабочем диске БК (кроме виртуального диска E: !).

Запуск BPM осуществляется из оболочки Disk Master или из монитора, при этом виртуальный диск E: отключается. Чтобы пропустить вывод мультипликационной заставки, можно нажать клавишу «КТ», после чего подгружается файл BPM.OVL и система готова к работе.

Функционирование BPM сводится к подготовке листа формата А4, который хранится в графическом файле (один лист — один файл) объемом 96 Кб (960x768 графических точек). Перемещая его относительно экрана, можно просматривать и редактировать те или иные участки страницы: вставлять копии экрана БК и спрайты, набирать фрагменты текста

Пн Вт Ср Чт Пт Сб Вс

ЯНВАРЬ

1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				

ФЕВРАЛЬ

			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29			

МАРТ

				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

АПРЕЛЬ

1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30					

МАЙ

		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31		

ИЮНЬ

					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30

Пн Вт Ср Чт Пт Сб Вс

Пн Вт Ср Чт Пт Сб Вс

ИЮЛЬ

1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				

АВГУСТ

		1	2	3	4	
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	

СЕНТЯБРЬ

						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30						

ОКТЯБРЬ

	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31			

НОЯБРЬ

				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	

ДЕКАБРЬ

						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

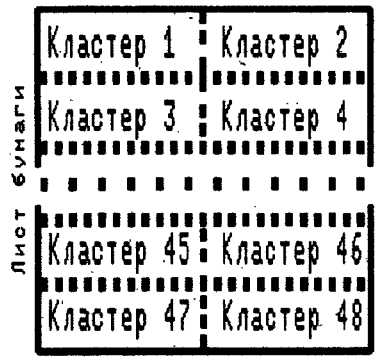
Пн Вт Ср Чт Пт Сб Вс

1996

(C) MIKsoft

Издательская система БК-PageMaker

Строение графического файла (размещение информации на диске) наглядно показано на рисунке:



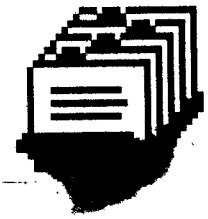
Перечислим кратко основные преимущества системы БК-PageMaker:

- удобный оконный интерфейс;
- работа с графическими изображениями большого размера (до 960x768 графических точек, что соответствует целому листу формата А4);
- использование графических шрифтов в формате COMFORT;
- импорт текстов в формате VorteX с их автоматическим преобразованием в графический вид шрифта;
- импорт/экспорт блоков спрайтов в формате Techno-Art. Вообще, программа БК-PadeMaker максимально ориентирована на совместную работу с этим графическим редактором, что позвслило автору не «изобретать велосипед» и отказаться от добавления в систему BPM собственного графического редактора (передача изображений производится через первую страницу видеоОЗУ);
- наличие в комплекте поставки библиотеки спрайтов-виньеток (более 400 образцов), заимствованных из широкоизвестной программы «NEWS» («Журналист») для IBM PC, которые окажутся полезными при создании объявлений или рекламных проспектов;
- импорт/экспорт копий экрана БК;
- возможность просмотра уменьшенной полной копии листа на экране (PreView);
- встроенный графический мини-редактор, позволяющий рисовать или редактировать изображение по точкам, очищать или инвертировать выбранные участки экрана.

Являясь первым опытом создания издательской системы на БК, программа BPM не свободна и от некоторых недостатков. Так, во избежание порчи записанной информации не следует заменять при работе с системой диск с обрабатываемым файлом без его закрытия и повторного открытия, так как данная версия BPM смену диска автоматически не опознает. Нельзя также заклеивать рабочий диск, так как издательская система при перемещении изображения страницы относительно экрана постоянно читает и записывает кластеры. Существенным, но все же не «фатальным» недостатком является то, что для нескольких страниц приходится создавать столько же отдельных файлов, и отсутствие упаковки графической информации, приводящее к быстрому расходованию дискового пространства (правда, упаковка/распаковка файлов заметно снизила бы скорость работы с программой). И наконец, следует обратить внимание на то, что изображение при выводе на принтер всегда инвертируется (соответствующий запрос пользователю пока не реализован). При распечатывании текста (который на экране отображается белым на черном фоне) и простейших двухцветных рисунков-виньеток это удобно, но полутоновые иллюстрации перед их загрузкой на создаваемую страницу, возможно, потребуется также проинвертировать (чтобы при печати не получить «негатив» вместо «позитива»).

Данная версия БК-PageMaker ориентирована на работу с ОС ANDOS v3.1 и, будем надеяться, выше. Эта операционная система кажется автору статьи наиболее удобной, во всяком случае ему не известен ни один (!) пользователь в Беларуси, который серьезно работал бы с другими ОС. Однако при соответствующем спонсировании, возможно, появятся версии под MKDOS, CSI-DOS и др. В будущих версиях также планируется использование векторных шрифтов Borland, а может быть, даже True-Type шрифтов MS Windows, реализация возможности перекачки больших картинок с IBM, работы с файлами .PCX и с многостраничными документами.

По вопросам приобретения издательской системы БК-PageMaker можно обращаться в редакцию журнала по адресу 125315, Москва, а/я 17 и E-Mail: mail@infoobr.msk.su



Обзор баз данных для БК, опубликованный в четвертом выпуске нашего журнала за 1995 г., вызвал значительный интерес как среди пользователей, так и среди разработчиков подобных программных средств. «Сколько людей, столько и мнений» — гласит известная поговорка, причем мнения эти далеко не всегда совпадают. Ниже мы публикуем отзыв автора программы NORD-BASE, дополняющий ранее напечатанный обзор новыми сведениями. Редакция надеется получить аналогичный материал и от разработчика базы данных С. Камнева, чтобы познакомить читателей и с его точкой зрения на обсуждаемую проблему.

А. Г. Прудковский,
Москва

NORD-BASE v2.3: база данных для системы NORD на БК-0011(М)

(отзыв автора NORD-BASE на статью П. П. Животовского в журнале «Персональный компьютер БК-0010 — БК-0011М», №4 за 1995 г.)

База данных NORD-BASE создавалась на основе упрощенной идеологии распространенного в настоящее время стиля «программирования без программирования», свойственного большинству современных языков управления базами данных (примерами являются CLIPPER, CLARION, FoxPro, PARADOX). Пользователю в этом случае не требуется составлять собственную программу в виде набора операторов, достаточно лишь задать ту или иную систему меню, экранных форм, бланков отчетов и определить их связи в стандартной оболочке СУБД.

Из-за небольшого объема памяти БК программа NORD-BASE включает в себя только самое необходимое: меню формирования баз данных, поиск, сортировку, выдачу информации в текстовый файл и на принтер, считывание из текстового файла, четыре действия арифметики с возможностью их простейшего программирования с помощью утилиты «МАКРОС» (как в программируемом микрокалькуляторе). Двухмодельная система поиска позволяет искать в текстовых полях строки, удовлетворяющие первой модели, за исключением строк, удовлетворяющих второй модели; в числовых полях — числа, принадлежащие интервалу [*модель1, модель2*], если *модель1* $\prec$ *модель2*, или интервалам $]-\infty$, *модель2*] и [*модель1*, $+\infty$[, если *модель1* > *модель2*. Построение баз данных отлаживалось на примере бухгалтерских работ (расчет зарплаты, инвентаризация содержимого складов и т. д.).

Автора давно интересовал вопрос о сравнительных характеристиках его разработки и других аналогичных программ. На него достаточно квалифицированно отвечает статья П. П. Животовского. Однако хотелось бы отметить, что он описывает устаревшие версии системы NORD 2.17 и базы данных NORD-BASE v2.1. В отличие от них современные версии системы NORD 3.4 и 3.5 легко копируются, обладают повышенными скоростями файлового обмена с диском, а NORD 3.5 может работать и с винчестером. В последней версии базы данных NORD-BASE v2.3 исправлены многие мелкие недоработки, а главное — ликвидирована ошибка при умножении и делении отрицательных чисел.

Неточность статьи П. П. Животовского состоит и в том, что, как он утверждает, NORD-BASE не умеет осуществлять поиск до первого удовлетворяющего заданным условиям элемента базы. Но это вовсе не так: в NORD-BASE существует механизм задания в модели отрицательных порядковых номеров, когда они воспринимаются по абсолютному значению, причем первое число указывает количество пропускаемых строк базы данных, удовлетворяющих заданной модели, а второе — количество подходящих по модели строк, которые достаточно найти для окончания поиска.

Не обсуждались в статье П. П. Животовского и вопросы сравнения функций арифметики в обеих базах данных. Система NORD-BASE может оперировать числами с фиксированной точкой (47 знаков до десятичной точки и до 9 после нее), а BASIS ?

Имеются в NORD-BASE и интересные возможности для взаимодействия различных баз данных, а также взаимодействия баз и текстовых файлов. Так, допускается ввод данных из файла, возможен ввод «по ключевому полю», что позволяет создавать сводные отчеты по информации из нескольких баз данных. (При обработке текстовых файлов, генерируемых программой NORD-BASE, хорошо зарекомендовал себя редактор VorteX.)

Следует отметить, что автор не считает нужным увеличение количества строк в каждой базе данных с 10 000 до 100 000, так как это делает ее излишне громоздкой, а время поиска увеличивается до недопустимых пределов и часто требуется многократная смена дискет в дисководе. Винчестер же, в свою очередь, накладывает на работу с СУБД свои ограничения:

- перед запуском программы NORD-BASE необходимо загрузить утилиту ABCD.COM, чтобы переобозначить рабочий диск винчестера (с меткой A: или B:, с которыми только и работает NORD-BASE, должен оказаться «винчестерным»);
- как известно, база NORD-BASE может работать только в «материнском» каталоге выбранного диска и записывать в него, в соответствии со стандартами системы NORD, не более 198 (т. е. 200–2) информационных файлов;
- учитывая, что каждый информационный файл может иметь длину не более 15 кб, получаем, что весь диск не имеет смысла делать больше 15*198=2970 кб, или 5940 блоков);
- как нетрудно подсчитать, чтобы в базе данных объемом 2970 кб уместилось 10000 строк, необходимо, чтобы длина строки не превышала 297 символов, а как известно, максимально допустимая длина строки в базе NORD-BASE равна 1023 символам. Если бы строк было в десять раз больше, то и длину каждой строки пришлось бы уменьшить до 29 символов, что уже совсем неудобно при работе.

Вообще же, по мнению автора, создание NORD-BASE и BASIS показало, что компьютер БК вполне пригоден даже для таких работ, которые традиционно считались прерогативой более мощной вычислительной техники. А потому можно считать, что реализация на БК какого-либо стандартного языка управления базами данных (скажем, того же CLIPPERa) — вещь вполне реальная, если найдется энтузиаст, готовый затратить на это год работы.

А. Г. Прудковский,
Москва

Операционная система NORD: версия для работы с винчестером

С появлением винчестера на БК для ОС NORD настало время продемонстрировать все изначально заложенные в нее возможности. В отличие от других систем, таких, как ANDOS или MKDOS, обрабатывающих только диски стандартной емкости, NORD даже старых версий поддерживала диски объемом до 16 Мб, а созданная недавно специально для винчестера версия NORD 3.5 обслуживает диски емкостью до 32 Мб. При этом обеспечивается не только возможность работы системы на таком диске, но и полное использование всего объема диска за счет создания на каждом физическом устройстве (A:, B:, C:, D:) до 20 логических устройств с именами F:,G:,...L:, N:,...Y:, Z:, каждое из которых может хранить до 200 файлов (стандартная емкость каталога). Если учесть также 200 файлов в «материнском» (корневом) каталоге и вычесть из полученного числа 20 имен логических дисков, записи о которых занимают в «материнском» каталоге столько же места, сколько и файлы, то в итоге получается, что система NORD способна хранить до 40180 файлов на каждом физическом устройстве.

Кажущимся ограничением ОС NORD является то, что на БК-0011(М) она поддерживает только четыре физических устройства (A:, B:, C:, D:), а на БК-0010(.01) — еще E: (на одиннадцатой модели E: зарезервировано за виртуальным диском), однако на самом деле это не так. Благодаря квалифицированно написанному драйверу винчестера в служебное ОЗУ по адресам 170000—177000 можно загрузить драйвер ABCD.COM, «подсоединяющий» к физическим устройствам A:, B:, C: и D: любые из 22 устройств E:, F:,... Z:, причем двойная идентификация дисков в HOPTOH-таблицах позволяет отслеживать все произведенные замены. Таким образом, система NORD может обслуживать винчестер с максимальным объемом до 768 Мб, содержащим до 964320 имен файлов (включая имена каталогов).

Следует отметить, что система NORD, с точки зрения автора, достигла своего «потолка», т. е. не способна к дальнейшему существенному развитию, что связано прежде всего с ее жесткой структурой, не предусматривающей модификаций с помощью дополнительных драйверов, как, например, в AO-DOS. Но «потолок» этот оказался достаточно высоким и, думается, не скоро будет достигнут прочими системами.

Итак, ОС NORD к настоящему времени имеет три модификации: NORD 3.2-БК10, NORD 3.4-БК11 и NORD 3.5 — «винчестерный» вариант, функционирующий и на БК-0010(.01), и на БК-0011(М). «Винчестерный» вариант системы может работать и на дискетах, но требует использования качественных дисководов, так как, в отличие от «дисководных» версий, поддерживает только одну копию каталога и системных дорожек. (Для создания второй копии каталога в версии NORD 3.5 есть специальная команда «АР2»+«Щ», а для ее чтения — «АР2»+«Ч», причем можно создать дополнительную копию не только для «материнского» каталога, но и для каталогов логических дисков.) «Дисководные» же версии поддерживают всегда две копии «материнского» каталога (и одну копию каталога логического диска), а по команде «АР2»+«Ч» могут записывать или считывать «материнский» каталог с последней (80-й или 40-й) дорожки дискеты. На БК-0010(.01) «винчестерная» версия NORD работает только при наличии 16 кб ДОЗУ, а NORD 3.2 поддерживает и 32 кб ДОЗУ конструкции П. В. Петрова (см. №4 за 1994 г., с. 29). В остальном же все три версии системы NORD идентичны. Все они запоминают текущее положение курсора на файловых панелях и восстанавливают его после выхода из программ; скорость движения курсора не меньше, чем в ОС ANDOS или MKDOS; поддерживается совершенный механизм открытия/закрытия файлов (см. №2 за 1994 г., с. 48), что позволило Б. Н. Морозову создать NORD-версию языка PASCAL; с системой NORD нормально работают БЕЙСИК и ФОКАЛ, текстовый редактор VorteX и графический Animatic, причем в отличие от других систем язык ФОКАЛ снабжен компилятором, а БЕЙСИК — возможностью работы с текстовыми файлами в ASCII-кодах*. На БК-0011(М) система комплектуется СУБД NORD-BASE v2.3 (см. №4 за 1995 г., с. 51), позволяющей хранить до 10000 строк в каждой базе данных и производить расчеты по четырем действиям арифметики с точностью до 9 знаков после десятичной точки. В современной версии NORD добавлены новые ключи «FIND» и «MOVE», первый из которых позволяет искать файл по модели его имени, а второй — перемещать файлы (если перемещение выполняется между подкаталогами одного и того же устройства, то перемещается только имя файла без его реального переписывания). Для выхода из подкаталога теперь используется клавиша «ИНД СУ». Упраздненный ключ «RDIR» заменен внешней утилитой с тем же названием RDIR, не только восстанавливающей каталоги по двум их копиям, но и грамотно преобразующей стандарт MKDOS в NORD. Функция поддержки магнитофона также оставлена только в виде добавочной утилиты, а по ключу «АР2»+«М» теперь производится задание модели распечатки каталога по первой букве; если же в качестве модели задан символ «*», то выводятся все файлы подряд без учета разбивки на подкаталоги. Новый командный язык системы, поддерживаемый файлом COMMAND.COM, теперь позволяет выполнять простейшие действия с целыми числами, что увеличивает возможности создания на его основе оверлейных командных систем. Запуск программ производится как по клавише «ВВОД» (после наведения курсора на имя файла), так и с помощью командной строки, в которую по нажатию «АР2»+«ВВОД» можно «сбросить» сразу несколько имен обрабатываемых файлов (пример организации такой обработки дан в отладчике TRACER-NORD).

В заключение следует отметить ошибки, выявленные в прежних версиях системы (в настоящее время все они исправлены). Так, программа СКОРОСТЬ_ДИСК, предназначенная для задания задержек дисководов A: и B: в 36-й ячейке системного диска, оказалась способной «коварно» переставлять каталоги с одного диска на другой без разрешения пользователя (сейчас она заменена исправленной версией СКОРОСТЬ_ДИСК3). В базе данных NORD-BASE v2.2 обнаружена ошибка при умножении и делении отрицательных чисел, она исправлена в новой версии NORD-BASE v2.3. Обнаружена и исправлена также ошибка форматирования диска в программах, обслуживающих формат IBM на БК-0011(М).

Исправленные версии утилит можно получить в редакции журнала «Персональный компьютер БК-0010 — БК-0011М» или непосредственно у автора данной статьи (телефон имеется в документации к NORD).

* В редакции имеется программа ZIPPER, обеспечивающая генерацию и запись на дискету листинга в формате .ASC в виде единого файла для «дискового БЕЙСИКа» DBASIC (ОС ANDOS). — *Прим. ред.*

Д. Ю. Усенков,
Москва

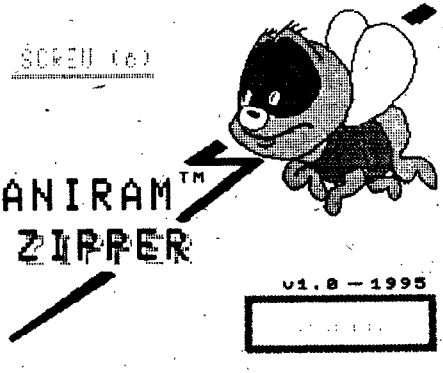
ANIRAM ZIPPER: утилита для записи ASCII-листинга (дисковый БЕЙСИК) в единый файл

Подключив к БК-0010.01 контроллер дисковода, оснащенный дисковым БЕЙСИКом (описание соответствующей аппаратной доработки КНГМД приведено в N2 за 1995 г.), и запустив специальный драйвер (чаще всего это DBASIC из комплекта ANDOS), вы получаете возможность работы в стандартном вильнюсском БЕЙСИКе с дисководом. Но удобство чтения-записи программ сохраняется только при использовании команд CLOAD или BLOAD. Если же попытаться сохранить на диске листинг БЕЙСИК-программы в виде ASCII-кодов (команда LOAD), то на дискете формируется целый набор его коротких фрагментов (как и в магнитофонном варианте). При этом дисковое пространство используется крайне нерационально (каждый блок объемом 400$_8$ байт занимает целый кластер), каталог засоряется именами фрагментов (количество же файлов в каталоге ANDOS ограничено), да и многократное включение-выключение мотора дисковода и удары магнитной головки о поверхность дискеты (если НГМД оснащен электромагнитом опускания головки) тоже вовсе не повышают надежность хранения информации. Можно, конечно, всегда сохранять свои программы только во внутреннем формате (по команде CLOAD), но как быть, если нужен файл листинга именно в виде ASCII-текста?

Один из вариантов: заранее подготовив свободную («буферную») дискету, записать на нее набор блоков по команде LOAD, а затем объединить их в единый текст, подгружая один за другим в текстовый редактор. Но при этом все упомянутые выше неудобства предстают «во всей красе». Чтобы избавить от них пользователей, была разработана загружаемая утилита ANIRAM ZIPPER, которая записывает ASCII-листинг на диск в виде единого файла.

Комплект поставки состоит из документации пользователя, считываемого в область стека загрузчика, графической заставки и собственно утилиты, считываемой загрузчиком в начало экранного ОЗУ (при включенном режиме РП). Подобная модульная структура выбрана по следующей причине: в стеке для утилиты остается слишком мало свободного места (так как там уже располагается драйвер дискового БЕЙСИКа). А загружать ZIPPER сразу в экранное ОЗУ командой BLOAD не совсем удобно — нужно перед загрузкой вручную переходить в режим РП или следить за рулонным смещением экрана и вертикальной позицией строки с командой BLOAD, иначе только что считанная утилита может быть затерта. Впрочем, при наличии дисковода модульная структура программы — не слишком заметное неудобство.

Работа с ANIRAM ZIPPER очень проста. Когда дисковый БЕЙСИК (программа DBASIC из комплекта ANDOS 3.1, КНГМД «АльтПро»; другие варианты не проверялись) запущен, а БЕЙСИК-программа находится в ОЗУ пользователя, введите команду **BLOAD "ZIPPER",R** (обязательно с R !). После появления заставки нажмите любую клавишу (кроме, конечно же, «СТОП») и введите имя файла, в который вы хотите записать листинг (до 6 символов без расширения). Расширение .BAS добавляется автоматически. После вывода на экран сообщения об успешной записи для выхода в диалоговый режим БЕЙСИКа нажмите любую клавишу. Текст БЕЙСИК-программы в ОЗУ при работе ZIPPERa полностью сохраняется. Максимально допустимая длина генерируемого листинга в данной версии равна 260008 байт, чего в большинстве случаев вполне достаточно.



Следует обратить внимание пользователей на то, что обратная загрузка созданного утилитой листинга в дисковый БЕЙСИК невозможна, следовательно, нужно дополнительно сохранять программу во внутреннем формате командой CLOAD.

VorteX 4.0: похвала и немного критики

Название «VorteX» (автор Д. Романов) хорошо известно многим БКманам: за последние три года этот текстовый редактор справедливо считался лучшим среди программ данного типа на БК. И если не считать ряда мелких недоработок, на которые чаще всего можно было не обращать внимания, единственным крупным недостатком VorteX оставалась ограниченная длина обрабатываемого текста. Впрочем, не стоит обвинять в этом автора программы: для БК дефицит пользовательского ОЗУ давно уже стал привычной вещью. (Д. Романов и так сделал все возможное, чтобы хоть как-то облегчить неудобства, связанные с ограничением объема текстов,— один только механизм постепенного уменьшения размеров рабочей части экрана представляет собой замечательную алгоритмическую находку.)

Но мысль программистов не стоит на месте. И вот как бы в качестве подарка к десятилетнему юбилею компьютера БК-0010 его пользователи получили возможность работы с новой версией VorteX 4.0. Впрочем, это даже не столько «обновленная версия», сколько новая программа, автором которой является С. Камнев (хорошо известный БКманам по таким мощным программным разработкам, как файловая оболочка DiskMaster для ОС ANDOS, база данных BASIS и др.). От прежнего текстового редактора остались, пожалуй, лишь основные идеи, сама же программа практически была написана заново. Значительно более удобным стал интерфейс пользователя (выпадающее двухуровневое меню, окна с подсказками, встроенный перекодировщик текста из формата БК в IBM и обратно), но самое главное — VorteX 4.0 позволяет обрабатывать тексты любой длины, лишь бы файл умещался на дискете! При этом скорость работы достаточно велика: текст объемом 11143 строки (196 кластеров, или почти 400 Кб) открывается за 2 с, преобразуется в формат IBM за 2 мин 40 с, а переход от начала текста к его концу («ВС»+«К») требует 1 мин 20 с. Если же работать не с дисководом, а с винчестером, эти цифры еще существенно уменьшатся.

Однако, как и любая новая программа, VorteX 4.0 не безгрешен в отношении недоработок. Ниже перечислены некоторые из них, отмеченные редакцией и читателями. Надеемся, что автор VorteX 4.0 не обидится на критику, а постарается устранить недостатки в следующих версиях.

1. При соединении фрагментов текста или его преобразовании в формат IBM старый файл .VXT не сохраняется. Значит, для последующей работы с ним нужно производить обратное преобразование из .TXT в .VXT, что не всегда удобно. Конечно, разместить дубль размером 400 Кб вместе с оригиналом на одной дискете невозможно, но в расчете на использование винчестера (для повышения надежности) неплохо было бы предусмотреть возможность создания BAK-копий (например, при установке специального ключа в начальных установках SETUP). Пока же, чтобы получить на диске оба файла (и в БКшной кодировке текста, и в IBMовской), приходится выходить из редактора, копировать файл средствами DiskMaster (на обеих панелях открываем каталог одного и того же диска, вводим «АР2»+«5», вместо смены диска нажимаем «ВВОД», даем команду «Double» на соответствующий запрос БК) и, снова запустив VorteX, перекодировать .BAK копию. Другой способ: не выходя из VorteX, открыть в главном меню опцию «Блок»/«Файл», переименовать буфер в <имя>.BAK, выделить весь текст, скопировать его в буфер («ВС»+«Д»), опять вызвать опцию «Файл», сразу нажать клавишу «%» и «ВВОД». После этого восстановится режим «%» для буфера и имя основного файла с расширением .VXT. Вообще же вместо возни с переименованием файла буфера лучше всего было бы предусмотреть в списке буферных операций команду «Save As» — запись выделенного фрагмента в файл с указанным пользователем именем (если такой файл уже существует, нужно предоставить пользователю выбор: «Overwrite», «Double» или «Append»).

2. Желательно предусмотреть копирование текстовых файлов средствами VorteX с возможностью изменения имени. Можно, например, дополнить запросом нового имени (с выводом на экран старого и его подтверждением при нажатии на «ВВОД») имеющуюся в меню опцию «Копия».

3. Как сказано в документации, VorteX 4.0 автоматически сохраняет все изменения в тексте, однако реально это происходит только при переходе курсора из одного кластера файла в другой. Учитывая хотя и редкую, но возможную ситуацию потери информации из-за сбоя питания, хотелось бы иметь опцию «принудительного» сохранения текста в виде команды меню, а еще лучше — «горячей» клавиши. На роль последней вполне подходит клавиша «СТОП» в версии

4.0 она, как и «КТ», выполняет сохранение текста (правда, в документации это явно не отмечено). Остается только отменить вызов главного меню после нажатия на «СТОП», оставив лишь запись в файл, и добавить соответствующий пункт в документацию.

4. К сожалению, исчезла возможность запуска VorteX из USER-меню с одновременной загрузкой выделенного файла.

5. Нумерацию позиций курсора в строке лучше было бы начинать с 1, а не с 0.

6. Для замены строчных символов на заглавные и обратно лучше оставить клавишу «ИНД СУ», как в прежнем VorteX! 1.05. Вызов же подсказки требуется не так часто, и его можно оформить только в виде пункта меню или производить клавишами «АР2»+«КТ». Неплохо было бы также добавить клавишу «Del» (аналогично используемой на IBM одноименной кнопке), которая удаляла бы символ под курсором (в режиме «Замена») или справа от него (в режиме «Вставка»). Это заметно ускорит работу, например, когда правые края строк выходят за пределы экрана.

7. Желательно все команды главного меню продублировать «горячими» клавишами (типа «АР2»+«‹первая или выделенная цветом средняя буква команды›»). Еще лучше дополнить редактор возможностью работы с мышью (вызов команд меню, перемещение курсора, выделение строчных и прямоугольных блоков и т. п.).

8. При перекодировании в формат IBM не всё расширение файла заменяется на «.TXT» (как это указано в документации), а только его первая буква — на «T». Последнее обеспечивает получение правильного «.TXT», лишь если исходный файл имел расширение .VXT. В других случаях (что вполне допускается в VorteX 4.0) результат может оказаться довольно неожиданным (из «.BAK» получается «.TAK» и т. д.), а файл приходится переименовывать после выхода в DiskMaster. Лучше уж пожертвовать парой байт под оставшиеся буквы «XT», но зато сделать работу с перекодировщиком форматов более удобной. (Кстати, при обратном перекодировании из формата IBM расширение во всех случаях заменяется на «.VXT».)

9. При ошибке обмена с диском выдается сообщение об этом, но зачастую нет никаких возможностей исправить ошибку: программа упорно пытается записать или считать содержимое сбойного кластера (даже при простом листании текста производится постоянный обмен с диском), а сменить диск нельзя. Следовало бы при записи очередного кластера проверять качество выполнения операции (сохраняя информацию в ОЗУ, пока она не будет надежно записана), а в случае повреждения данного участка дискеты помечать кластер как «BAD» и продолжать запись со следующего. (Кстати, аналогичный недостаток имеется в программе Altec Disk Doctor. При восстановлении информации на дискете «доктор», считав из поврежденного сектора все, что удастся, сохраняет спасенные данные на том же месте и при физическом дефекте магнитного слоя (не устранимом повторным форматированием) попросту зацикливается. Хотелось бы посоветовать разработчикам «диск-доктора» взять за образец работу Norton Disk Doctor на IBM-совместимых ПК, где вычитанные из сбойного кластера данные всегда помещаются В ДРУГОЕ МЕСТО ДИСКА, а прежний кластер помечается как «BAD».

10. Неправильно индицируется при включении/выключении режима рисования линии («ВС»+«Л») символ «⊞». (Аналогичная ошибка индикации характерна и для режима «автопробел».) Бывают также случаи, когда рисование линий продолжается после нажатия других клавиш.

11. При рисовании часто приходится поднимать перо, чтобы перевести курсор в соседнюю позицию, не оставляя следа на экране. Хорошо бы дополнить возможности данного режима функцией кратковременного поднятия пера, например, по клавише «ПРОБЕЛ».

12. В документации указано, что после очистки окна «Открыть» клавишей «ПРОБЕЛ» можно загрузить любой файл. Это не совсем точно, так как в файлере выводятся только файлы с расширением .VXT. Чтобы получить доступ ко всем файлам, нужно ввести в качестве «шаблона» имени комбинацию «ПРОБЕЛ»+«точка».

13. К сожалению, из редактора невозможна печать машинописным шрифтом (или любым другим нестандартного вида, загружаемым извне). Такая функция может быть реализована в виде загружаемого модуля.

14. Одним из преимуществ новой версии VorteX является широкий набор используемых принтеров. Но как быть тем, у кого дома принтера нет, но он имеется на работе (скажем, подключен к IBM) или у кого-то из знакомых? Необходимо предусмотреть в программе (или в загружаемом дополнительном модуле) функцию «печать в файл», когда все байты (символы и управляющие коды), при обычной печати передаваемые в принтер, записываются в отдельный файл. Последний затем можно будет перенести на дискете на другую машину (в том числе на IBM) и распечатать, просто скопировав его на принтер (на IBM — командой COPY /B <имя_файла> LPT1 , а на БК с помощью несложной программы). Тогда работа с текстовым редактором станет полноценной для всех пользователей.

15. В подменю «Печать» желательно выводить все установки режима печати и давать возможность оперативно их изменять, не выходя из этого режима. Кроме того, желательно предусмотреть функцию «Печать буфера», чтобы не выполнять перезагрузку буфера как текста. Впрочем, недостатки реализации драйвера печати на принтер этим далеко не исчерпываются. Так, нет возможности печати начального номера первого листа (например, когда нумерация начинается не с единицы). А при печати выделенного фрагмента нумерация страниц, увы, и вовсе не производится — это явный непорядок. Обязательно нужно предусмотреть и пропуск печати текущего листа с переходом сразу к следующему (например, по клавише «ПРОБЕЛ», как в ЛЕКСИКОНе на IBM).

16. К сожалению, «локальные» установки интервала между строками ESC-последовательностями (♠ 27...) в самом тексте подавляются «глобальной» установкой в меню «Печать», хотя все должно быть наоборот. Кроме того, при печати текста с различными интервалами между строками на одном и том же листе нужен перерасчет длины страницы. Например, если набрано 20 строк (N1) с интервалом 44 (ESC 3 44), затем еще 27 строк (N2) с интервалом 1.0 (ESC 3 36), скажем, для таблицы, и потом 7 строк (N3) с интервалом 44, а высота листа (с учетом отступов снизу и сверху) равна 10 дюймов (или 60 строк), то формула расчета длины страниц может быть такой: **n1/216 * N1 + n2/216 * N2 + ... + ... = 44/216 * 20 + 36/216*27 + 44/216 * 7 = 10 дюймов**. Последующие строки должны переноситься на другой лист.

Если же, наоборот, требуется обеспечить вывод на листе определенного количества строк, это можно сделать с помощью следующего перечня ESC-последовательностей (лист формата А4 с учетом расстановки номеров страниц):

Код	Интервал, устанавливаемый в меню	Количество строк
♠ 27 ♠ 51 ♠ 36	36	60
. . . 39	39	55
. . . 42	42	50
. . . 46	46	45
. . . 52	52	40
. . . 60	60	35
. . . 68	68	30
. . . 72	72	28

Так как для первой страницы не учитывается пропуск первых двух строк (номер страницы и пустая строка после него), целесообразно нулевую строку в начале текста сделать пустой (нажать «ВВОД» и «ВС»+«Ч»), а на принтере три раза нажать кнопку «ПС». Тогда пустая строка станет «нулевой страницей», а число, указываемое при установке длины страницы («СУ»+«Г»), будет соответствовать номеру ее последней строки.

Вообще же в меню лучше всего устанавливать только количество строк («СУ»+«Г»), а не интервал между ними, но тогда потребуется предусмотреть соответствующий пересчет в программе. В этом случае на экране страница будет видна в том виде, в каком она окажется напечатана (режим «WYSIWYG»). При печати на бумажном рулоне количество строк можно устанавливать равным нулю, что для программы должно являться признаком блокировки приостанова печати в конце страницы.

17. Как выяснилось, в VorteX 4.0 можно печатать заголовки с тенями, хотя в документации об этом не говорится ни слова. Вот пример печати заголовка высоким шрифтом:

```
↑27↑119↑1↑27↑69↑27↑32↑8ПРИМЕР ПЕЧАТИ↑27↑32↑0↑27↑70'
↑27↑32↑8↑27↑106↑40↑27↑92↑4↑0ПРИМЕР ПЕЧАТИ↑27↑32↑0↑27↑119↑0'
```

ПРИМЕР ПЕЧАТИ

Здесь строки со значками «пики» содержат следующие ESC-команды:

- 1-я строка: 27,119,1 — высокий шрифт; 27,69 — жирный; 27,32,8 — интервал=8; 27,32,0 — отмена интервала; 27,70 — отмена жирного шрифта;
- 2-я строка: 27,32,8 — снова интервал=8; 27,106,40 — вверх на 28 точек; 27,92,4,0 — вправо на 4 точки; 27,32,0 — отмена интервала; 27,119,0 — отмена высокого шрифта.

В последовательности 27,106,nn значения nn приблизительно равны (NN–8...10), где NN — значение интервала, устанавливаемого в меню «Печать».

Еще один пример:

```
↑27↑33↑104↑27↑120↑1↑27↑32↑4РЕДАКТОР VX↑27↑32↑0↑27↑120↑0↑27↑33↑0
↑27↑33↑96↑27↑32↑4↑27↑106↑38↑27↑92↑4↑0РЕДАКТОР VX↑27↑32↑0↑27↑33↑0
```

РЕДАКТОР VorteX

Строки с «пиками» означают:

- 1-я строка: 27,33,104 — широкий жирный курсив; 27,120,1 — качественный шрифт; 27,32,4 — интервал=4; 27,32,0 — отмена интервала; 27,120,0 — отмена качественного шрифта; 27,33,0 — отмена жирного курсива;
- 2-я строка: 27,33,96 — широкий курсив; остальные команды — как в предыдущем примере.

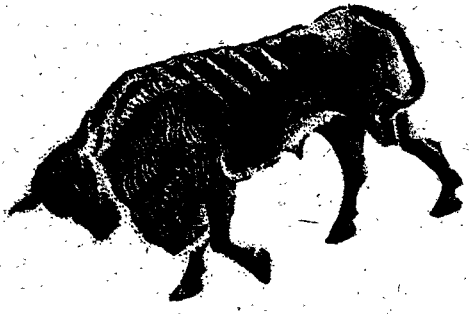
Наиболее часто применяются команды включения широкого шрифта — 27,33,32 и широкого жирного — 27,33,40 (все ESC-последовательности даны для принтера МС6313).

18. Желательно предусмотреть два режима форматирования абзацев: всего текста или фрагмента от курсора до метки (например, чтобы не испортить таблицу).

19. На БК-0011(М) неплохо реализовать работу с двумя и более окнами (ведь при работе с файлом непосредственно на диске, да еще с буферизованным доступом, нет принципиальной разницы в количестве открываемых окон). Дополнит редактор полезной возможностью и модуль загрузки рисунков во вторую экранную страницу (для последующей их печати вместе с текстом). Управлять печатью картинок можно вручную (например, клавишей «ШАГ», как в GPS4), а если бы это осуществлялось программно, скажем, по специальному коду типа ESC, это был бы поистине «высший пилотаж»!

20. После точки помимо автопробела, хотелось бы иметь автоматический ввод заглавных букв (как в Word 6.0) с возможностью включения/выключения этой функции с клавиатуры. Желательно также предусмотреть автопробел после символа «точка с запятой». Ко всему этому неплохо было бы иметь и «калькулятор» или «формульный ввод» с вставкой результата вычислений в текст. Вообще же в будущем, если это возможно, полезной оказалась бы программная совместимость VorteX с BASIS, электронной таблицей и каким-либо достаточно мощным графическим редактором. Тогда VorteX во многом превзойдет IBMовские ЛЕКСИКОН и даже Word 6.0 for Windows.

Редакция благодарит за активное участие в подготовке данного материала читателя Ю. Крыжановского (Москва).



ОБМЕН ОПЫТОМ

Главным недостатком (во всем прочем весьма удобного в пользовании) вильнюсского БЕЙСИКа БК-0010.01 является невозможность генерации исполняемых модулей в кодах (т. е. для работы программы обязательно наличие в памяти ее исходного листинга и транслятора БЕЙСИКа). Изменить что-либо в прошивке ПЗУ, разумеется, невозможно без замены микросхем, но тем не менее удается реализовать автономную работу «шитых кодов» (с сохранением удобства отладки листинга), если воспользоваться разработкой, предлагаемой в данной статье.

Д. Вальяно,

Москва

БЕЙСИК БК-0010(.01) + ДОЗУ = новые возможности

Многие пользователи БК-0010.01 уже имеют дисководы, благодаря чему у них появилась возможность писать программы на таких языках высокого уровня, как, например, ПАСКАЛЬ (сегодня в продаже имеются как минимум две различные версии трансляторов с этого языка). Их преимущества перед стандартным (ПЗУшным) БЕЙСИКом выражаются, кроме всего прочего, и в том, что они транслируют программы в автономно запускаемые машинные коды. А значит, можно написать большую по объему программу с высоким быстродействием. Но и БЕЙСИК тоже прочно занял свою «экологическую нишу»: его преимущество — быстрота и удобство отладки листингов, на нем проще написать программу, реализующую сложные вычисления. К тому же за долгие годы своего существования БЕЙСИК-БК обрел большое число поклонников. Что же касается недостатков этого БЕЙСИКа, некоторые из них можно преодолеть. Уже предложено множество различных способов расширения его возможностей. Данный материал является продолжением идеи И. В. Канивца, изложенной в статье «Укорочение БЕЙСИК-программ при трансляции» (Персональный компьютер БК-0010 — БК-0011М. 1995. №2).

Коротко поясню суть метода, реализованного в программе UKOR. Основная идея состоит в том, чтобы перед трансляцией «выкинуть» из ОЗУ строки текста, не нужные для нормальной работы объектного кода (не содержащие текстовых констант или операторов DATA). Эти строки сначала перемещаются в экранное ОЗУ (с соответствующим изменением таблицы номеров и адресов строк), а после трансляции они больше не нужны и стираются вовсе. В итоге достигается значительная экономия памяти (до 3—4 кб).

Не секрет, что почитатели вильнюсского БЕЙСИКа значительно облегчили себе жизнь, приобретя дисковод. В принципе, немного измененную программу UKOR можно было бы использовать и совместно с дисковым БЕЙСИКом (правда, некоторые его версии придется слегка доработать, о чем будет сказано ниже), тогда скорость отладки большой программы значительно возрастет. Но если у вас есть ДОЗУ на 8, а еще лучше — на 16 кб, то можно «копнуть глубже». Речь идет об использовании части ДОЗУ, не занятой операционной системой, для хранения текста программы и данных, что до этого приходилось бы периодически сбрасывать на диск. Под операционной системой подразумевается в первую очередь ANDOS, так как ее ядро (основная часть, эмулирующая EMT 36) занимает чуть менее 3.5 кб, а всю остальную дополнительную память можно использовать для своих целей (испортив, конечно, при этом файловую оболочку, но она при работе с БЕЙСИКом и не нужна). А что же необходимо сохранять в ДОЗУ? Ответ ясен: текст программы и таблицу адресов строк (от адреса в ячейке 2024 до адреса в ячейке 2026), кроме того, дополнительно сохраняется ячейка 2002 (адрес

конца текста). Если у вас имеется ДОЗУ на 16 кб, то свободных 12.5 кб для этих целей вполне достаточно (да еще, возможно, останется место и для массивов данных), а если ДОЗУ имеет емкость всего 8 кб, то придется изыскивать дополнительные ресурсы памяти (например, частично использовать ОЗУ экрана). Но, как говорилось выше, и этим объемом ДОЗУ можно обойтись, если использовать освободившуюся от текста БЕЙСИК-программы память не под дополнительные строки листинга, а, например, под машинные подпрограммы.

Итак, впереди уже вырисовываются очертания будущего драйвера, но все еще остается много неясного: понятно, что подпрограмма сохранения текста в ДОЗУ должна вызываться перед его трансляцией, но когда и каким образом должна вызываться подпрограмма восстановления текста для его дальнейшей отладки? Можно, конечно, выделить под эту подпрограмму отдельную USR-функцию и вызывать ее по мере необходимости вручную. Но предлагаемый мною метод позволяет полностью автоматизировать весь процесс трансляции и последующего восстановления текста (хотя сохранен и ручной режим). Вам достаточно один раз загрузить драйвер с диска, а вся дальнейшая работа в среде БЕЙСИКа будет практически неотличима от прежней как по удобству, так и по времени. И притом появится возможность создавать программы увеличенного объема!

Лучше всего, по моему мнению, чтобы подпрограмма восстановления текста вызывалась всякий раз после останова БЕЙСИК-программы или при прекращении трансляции после обнаружения ошибки. Будем считать, что корректный останов программы (или трансляции) возможен по трем причинам:

- при обнаружении ошибки (как во время трансляции, так и во время выполнения БЕЙСИК-программы);
- «по естественным причинам» (после выполнения последней строки или после операторов END и STOP);
- после нажатия клавиши «СТОП».

Значит, нужно найти что-то общее для всех этих случаев. А общим является извечное бейсиковское «Ok»! Метод «отлова» этого сообщения похож на аналогичный, примененный в программе UKOR. Отслеживается команда EMT 20 (через которую выводятся практически все сообщения БЕЙСИКа). После обнаружения этой команды проверяется содержимое регистра R1 (адрес строки символов): если оно равно 132730 (сообщение «Ok»), то управление передается подпрограмме восстановления текста, в противном случае обработка EMT «поручается» дисковому БЕЙСИКу. Конечно, восстановление листинга должно производиться только при первой после останова подаче этого сообщения (иначе редактирование программы станет невозможным). Для этого предусмотрим специальную ячейку-признак PR1, куда после копирования строк в ДОЗУ записывается единица, а после восстановления текста — нуль, что блокирует повторный вызов подпрограммы восстановления (эта ячейка может быть использована для реализации ручного режима, о чем будет сказано ниже). Признаком того, что восстановление текста произведено, будет знак «!» перед сообщением «Ok».

При реализации данного метода проблемы могут возникнуть только с двумя командами и одним оператором БЕЙСИКа (других проблем автор при испытании не обнаружил). Трудности с CONT и GOTO (в непосредственном режиме) объясняются тем, что объектный код частично портится восстановленным после останова текстом, т. е. в этом случае данные команды нормально работать не будут (в программе предусмотрено отключение команды CONT, а оператор GOTO в непосредственном режиме лучше не использовать!). Что же касается команды BSAVE, то в автоматическом режиме драйвера вы не сможете только записать на диск объектный код (по тем же причинам, что и выше), но никаких поводов для неправильной работы этой команды во всех прочих случаях нет.

Теперь пора, наконец, поговорить о реализации ручного режима, благодаря которому можно устранить и указанные выше проблемы. Для этого нам нужно отключить автоматический вызов подпрограммы восстановления текста, чтобы можно было работать с объектным кодом в непосредственном режиме, и вызывать эту подпрограмму вручную при необходимости (для отладки текста). Первый способ сделать это заключается в добавлении в БЕЙСИК-программу оператора обнуления ячейки-признака (PR1). Например: **10 POKE <адрес ячейки PR1>,0%**. Перед отладкой же листинга надо в непосредственном режиме выполнить обратную операцию: **POKE <адрес ячейки PR1>,1%**. После этого текст автоматически восстановится.

Второй способ более удобен, так как не требует записи специальной строки в БЕЙСИК-программу. Вместо этого перед запуском по команде RUN надо изменить в драйвере UKOR2 код 1005 (BNE VP) после метки RR на 405 (BR VP), в результате отлов сообщения «Ok» не будет производиться. Для нормального функционирования драйвера надо восстановить прежнее состояние.

Кстати, временное хранение текста и адресов строк в ДОЗУ позволяет во многих случаях восстановить БЕЙСИК-программу даже после различных аварийных ситуаций (переинициализации системы по какой-то причине и др.). Если не произошло серьезных сбоев типа аварии питания, то информация в ДОЗУ должна сохраниться. В таком случае прежде всего надо как-то восстановить прежнее (до «вылета» из БЕЙСИК-системы) содержимое ячеек с адресами 2024 и PR2 , если они не сохранились (вообще же, в драйвере можно предусмотреть автоматическое сохранение этих ячеек в ДОЗУ). После этого перезагрузите (при необходимости) дисковый БЕЙСИК и драйвер, установите прежний размер памяти под программу пользователя (CLEAR ... , ...) и запишите восстановленную информацию в вышеуказанные ячейки. Затем установите ячейку PR1: **POKE <адрес яч. PR1>,1%**. Далее необходимо активизировать драйвер, и тогда текст должен автоматически восстановиться.

Теперь пришло время поговорить о различных дисковых БЕЙСИКах (драйверах, активизирующих стандартный вильнюсский БЕЙСИК в ПЗУ и обеспечивающих работу в нем с дисководом), об их «взаимоотношениях» с драйвером UKOR2, а также об использовании зашитого в ПЗУ отладчика (ANDEBUG) для отладки кодовой части БЕЙСИК-программы пользователя.

Разных версий дисковых БЕЙСИКов существует много, они различаются по возможностям, но, в принципе, все их можно приспособить для совместной работы с драйвером UKOR2. При выборе версии (если такая возможность есть) надо руководствоваться не только возможностью более полной эмуляции всех магнитофонных команд (LOAD, SAVE и т. д.). Например, не самая мощная версия BASIC10 предоставляет возможность использования зашитого в ПЗУ (или размещенного в ДОЗУ) отладчика. Дело в том, что после выхода в монитор из этого дискового БЕЙСИКа имеется возможность перехода туда и обратно без потери БЕЙСИК-программы после команды перехода по адресу 1000 (если, конечно, не были испорчены системные ячейки и стек БЕЙСИКа).

Для работы с нашим драйвером эту версию дискового БЕЙСИКа придется немного доработать. Изменения сводятся к тому, чтобы все константы 1104 в программе BASIC10 (адрес обработки EMT) заменить на аналогичный адрес драйвера UKOR2 (т. е. на адрес метки LEMT). После этого можно запускать драйвер.

Если вы работаете с версией DBASIC, то после его запуска необходимо активизировать предварительно загруженный драйвер UKOR2, т. е. записать в ячейку 30 адрес метки LEMT. Этот дисковый БЕЙСИК не предоставляет возможность выхода в монитор и возврата обратно без потери БЕЙСИК-программы (как и указано в описании к нему), но данный недостаток можно обойти. Следует отметить, что приведенный ниже листинг драйвера рассчитан на работу совместно с программой BASIC10, при использовании других версий дисковых БЕЙСИКов его придется изменить (число 1104 в команде JMP @#1104, являющееся начальным адресом EMT-диспетчера BASIC10, нужно заменить на аналогичный адрес в используемом вами дисковом БЕЙСИКе).

Где же разместить драйвер UKOR2? Так как он (в приведенном ниже варианте) занимает 6208 байта, в стеке его целиком разместить не удастся. Можно, во-первых, выделить под него место командой CLEAR. Во-вторых, разместить фрагментами в различных неиспользуемых участках ОЗУ. Можно, наконец, разместить его в ДОЗУ (если там хватает места). В последнем случае где-то в ОЗУ должна остаться маленькая программа (начальная часть EMT-драйвера), которая будет подключать ДОЗУ и передавать управление основной части драйвера UKOR2 или подключать ПЗУ и передавать управление БЕЙСИК-системе после окончания работы основной части драйвера в ДОЗУ. Естественно, в двух последних случаях потребуется внести некоторые изменения в листинг драйвера.

После окончательной отладки и записи на диск объектного кода его дальнейшая работа не требует наличия ни драйвера UKOR2, ни ДОЗУ. Освободившиеся ресурсы памяти можно использовать по своему усмотрению (в принципе, готовая БЕЙСИК-программа может задействовать ДОЗУ, например для хранения графической заставки и т. д., однако реализация этой возможности не очень проста и требует определенного умения).

Список литературы

1. *Котов Ю. В.* Совместное использование БЕЙСИКа и машинных кодов //Вычислительная техника и ее применение. 1992. №3. С. 25

2. *Канивец И. В.* Укорочение БЕЙСИК-программ при трансляции //Персональный компьютер БК-0010 — БК-0011М. 1995. №2. С. 45

3. Подключение кодовых подпрограмм к Бейсик-программам на этапе трансляции //Персональный компьютер БК-0010 — БК-0011М. 1995. №2. С. 49

```
;Программа UKOR2
;(c) 1995 Д. ВАЛЬЯНО & И. В. КАНИВЕЦ
        MOV PC,R0
;начальный загрузчик
        .#62700
        .@LEMT+2
        MOV R0,@#30
        JMP @#1000
;переход в дисковый БЕЙСИК
LEMT:   CMP 4(SP),#135312
;отлов EMT. Начало трансляции?
        BEQ OTR
;если да, то то переход к п/пр UKOR
        MOV R5,-(SP)
;иначе продолжаем исследование
        MOV 2(SP),R5
        MOV -(R5),R5
        CMP R5,#104020
;текущая команда EMT 20?
RR:     BNE VP
;если нет, переход к нормальной
; обработке EMT
        CMP R1,#132730
;выводится  сообщение "Ok"?
        BNE VP
;если нет - переход к нормальной
; обработке EMT
        JMP VOS
;иначе переход к подпрограмме
; восстановления
VP:     MOV (SP)+,R5
;нормальная обработка EMT
; "поручается" дисковому  БЕЙСИКу
        JMP @#1104
OTR:    JSR PC,ZAT
;начало подпрограммы UKOR.
; Обращение к подпрограмме
; запоминания текста в ДОЗУ
        MOV #2036,@#2040
        CLR @#2052
;подпрограмма  "UKOR"
        CLR @#2036
M10:    INC @#2032
        BEQ M10
        INC @#2000
        MOV R4,@#2046
        MOV @#2024,@#2016
        MOV @#2024,@#2020
        MOV @#2024,@#2022
        CLR @#2004
        MOV R0,-(SP)
        CLR R4
        MOV #3052,R1
        MOV #40000,R0
CYCLE:  MOV #177777,R3
        MOV @#2026,R2
        CMP -(R2),-(R2)
M0:     CMP (R2),#40000
        BHIS M1
        CMP (R2),R4
        BLOS M1
        MOV (R2),R3
        MOV R2,R5
M1:     CMP -(R2),-(R2)
        CMP R2,@#2024
        BCS MF
        CMP -(R2),R3
        BCS M0
        BR M1
MF:     CMP R3,#177777
        BEQ END
        MOV R3,R2
        CMPB (R3),#100
        BEQ L0
        CMPB (R3),#2
        BEQ L0
        CMPB (R3),#50
        BEQ HI
        TSTB (R3)+
M2:     CMPB (R3),#42
        BEQ L0
        CMPB (R3)+,#12
        BNE M2
HI:     MOV R0,(R5)
        MOV R2,R4
        MOVB (R2)+,(R0)+
M3:     MOVB (R2)+,(R0)
        CMPB (R0)+,#12
        BNE M3
        BR CYCLE
L0:     MOV R1,(R5)
        MOV R2,R4
        MOVB (R2)+,(R1)+
M4:     MOVB (R2)+,(R1)
        CMPB (R1)+,#12
        BNE M4
        BR CYCLE
END:    MOV (SP)+,R0
        MOV R1,@#2002
        INC R1
        BIC #1,R1
        MOV R1,@#2014
```

```
        MTPS #0
;окончание работы п/пр UKOR
        JMP @#135400
;возврат к трансляции
ZAT:    CLR @#177130
;подпрограмма запоминания текста,
; отключение ПЗУ
        MOV #126500,R2
;R2 - начальный адрес
; используемой части ДОЗУ
        MOV #3052,R0
;R0 - начальный адрес
; листинга БЕЙСИК-программы
01:     MOVB (R0)+,(R2)+
        CMP R0,@#2002
;в ячейке 2002 содержится
; адрес конца листинга
        BLE 01
;скопировали текст в ДОЗУ
        MOV @#2024,R0
;R0 - адрес начала таблицы строк
02:     MOVB (R0)+,(R2)+
        CMP R0,@#2026
;в ячейке 2026 содержится
; адрес конца таблицы строк
        BNE 02
;скопировали таблицу строк
; в ДОЗУ вслед за листингом
        MOV #14,@#177130
;подключаем ПЗУ
        MOV @#2002,PR2
;запоминаем адрес конца листинга
        INC PR1
;устанавливаем ячейку-признак
        RTS PC
;выход из подпрограммы
VOS:    MOV (SP)+,R5
;начало подпрограммы
; восстановления текста
        TST PR1
;проверяем ячейку-признак
        BEQ RV
;если текст уже
; восстанавливался - на выход
        MOV R0,-(SP)
;запоминаем регистры в стеке
        MOV R1,-(SP)
        MOV R2,-(SP)
        MOV #40200,@#177130
;отключаем ПЗУ
        MOV #126500,R2
;R2 - начало использованной
; части ДОЗУ
        MOV PR2,@#2002
;восстановление ячейки
; конца текста
        MOV #3052,R0
;R0 - начальный адрес текста
V1:     MOVB (R2)+,(R0)+
;восстановление текста
        CMP R0,PR2
        BLE V1
        MOV @#2024,R0
;R0 - адрес начала
; таблицы строк
        MOV R0,R1
;проверяем R0 на четность
        ROR R1
        BHIS V2
        DEC R0
;если нечетное,
; вычитаем единицу
V2:     MOVB (R2)+,(R0)+
;восстанавливаем таблицу строк
        CMP R0,@#2026
;в ячейке 2026 -
; адрес конца таблицы
        BNE V2
        MOV #14,@#177130
;подключаем ПЗУ
        CLR PR1
;обнуляем ячейку-признак
        MOV '!',R0
;выводим на экран знак "!"
        EMT 16
        MOV (SP)+,R2
;восстанавливаем регистры
        MOV (SP)+,R1
        MOV (SP)+,R0
        CLR @#2040
;отключаем команду CONT
RV:     JSR PC,@#107050
;выводим сообщение "Ok"
        RTI
;возврат из прерывания
PR1: .#0
;ячейка-признак
PR2: .#0
;адрес конца листинга
        END
```

С. П. Рожков,

г. Кишинев

«Волшебные ячейки» ключей

В журнале «Информатика и образование», №3 за 1991 г. была опубликована статья «Волшебная сила двух ячеек», где описывалась программа в кодах, позволяющая имитировать программный режим для управляющих команд БЕЙСИКа, обычно доступных только в непосредственном режиме. Эту программу можно сократить всего до трех команд ассемблера (пяти машинных слов).

```
        MOV (R5)+,@#122   ; &012537 ;  Передача длины программируемой
                          ; &0122   ;   строки в ячейку 122
        MOV @R5,@#124     ; &011537 ;  Передача адреса программируемой
                          ; &0124   ;   строки в ячейку 124
        RTS PC            ; &0207   ;  Возврат из подпрограммы.
```

Вызов из программы на БЕЙСИКе производится с помощью функции USR:

```
100 A=USR("<текст ключа>")
110 STOP
```

В данном варианте, в отличие от ранее опубликованного, текст программируемой строки не переписывается в стековую область, но за это приходится платить тем, что теряется возможность при реализации оверлейных процессов передавать данные из работающей программы в вызываемую и записывать в программируемой строке какие-либо команды после операторов LOAD или CLOAD. Это связано с тем, что загружаемая программа стирает старую, а следовательно затирает и запрограммированную строку — последовательность команд, параметры которой мы переслали в системные ячейки. Остальные же операторы БЕЙСИКа (в том числе NEW и DELETE) можно использовать в любой комбинации.

Ряд других неудобств при использовании этой программы связан с тем, что и запрограммированные команды, и сообщения системы выводятся на экран (точно так же, как и в непосредственном режиме), а это приводит к рулонному сдвигу (не всегда желательному) или к необходимости отводить место на экране для вывода сообщений. Пытаться же сделать сообщения системы «невидимыми» с помощью команды COLOR бесполезно, так как при этом все равно будет происходить рулонный сдвиг и возможно затирание полезной информации на экране. Лучше всего запретить вывод на экран вообще, что и призвана сделать программа UPRKOM.BIN:

```
        MOV (R5)+,@#122   ; &012537   ; Передача
                          ; &0122     ;  длины строки
        MOV @R5,@#124     ; &011537   ;  и ее адреса в системные
                          ; &0124     ;  ячейки
        MOV R7,@#30       ; &010737   ; Изменение содержимого вектора
                          ; &030      ;  EMT-прерывания
        ADD #12,@#30      ; &062737   ;  на адрес метки D
                          ; &012      ;
                          ; &030      ;
        RTS R7            ; &0207     ; Возврат из подпрограммы
D:      MOV R5,-(R6)      ; &010546   ; Запоминание в стеке R5 и
        MOV 2(R6),R5      ; &016605   ;  извлечение кода
                          ; &02       ;  EMT-команды, вызвавшей
        MOV -(R5),R5      ; &014505   ;  прерывание
        CMP R5,#104016    ; &020527   ; Перехват EMT16
                          ; &0104016  ;  и переход
        BEQ E             ; &01405    ;  на метку E
        CMP R5,#104020    ; &020527   ; Перехват EMT20
                          ; &0104020  ;  и переход
        BEQ E             ; &01402    ;  на метку E
        JMP @#100122      ; &0137     ; Для прочих EMT переход на
                          ; &0100122  ;  стандартную обработку
E:      MOV (R6)+,R5      ; &012605   ; Восстановление R5 из стека
        RTI               ; &02       ; Выход из прерывания
```

Программа UPRKOM.BIN перемещаема и может быть размещена в любой свободной области ОЗУ. Ее вызов из БЕЙСИК-программы производится операторами:

```
100 A=USR("<текст ключа>")
110 STOP
```

Примечание редактора

Предлагаемая С. П. Рожковым программа, как и многие другие известные пользователям БК аналоги, использует аппарат программируемых ключей. Вся «волшебная сила» ячеек @#122 и @#124 состоит в том, что они являются рабочими переменными при «штатной» отработке ключа после нажатия клавиш АР2+<цифра>.

Вот как это происходит. Программируя ключи с помощью функции ассемблера ЕМТ12 или в БЕЙСИКе с помощью вызывающего ту же функцию оператора KEY, мы формируем в памяти ряд строк ключей, адреса которых становятся известными БК (заносятся в ячейки @#126 — @#150), а длины указаны в первом байте каждой строки. Когда мы нажимаем АР2+<цифра>, программа обработки прерывания от клавиатуры (вектор @#274) заносит адрес первого символа соответствующего ключа (указатель на строку ключа) в ячейку @#124, а длину этой строки — в ячейку @#122.

При вызове ЕМТ6 зашитая в ПЗУ подпрограмма обработки этой функции, прежде чем считать из регистра клавиатуры находящийся там код очередного символа, проверяет три условия:

Не был ли задан программируемый ключ? Тестируется ячейка @#122. Если в ней содержится не нуль, в качестве очередного символа, функция ЕМТ 6, игнорируя код, находящийся в регистре клавиатуры (если он там есть), возвращает символ заданной строки ключа, адрес которого хранится в ячейке @#124. Затем содержимое ячейки @#122 уменьшается на 1, а содержимое @#124 увеличивается на 1 (указатель смещается на следующий символ ключа). Когда весь ключ выведен, а новый не включен очередным нажатием «АР2»+«<цифра>», содержимое @#122 равно нулю и подпрограмма ЕМТ6 переходит к следующему шагу.

Не была ли нажата клавиша «ТАБ»? При нажатии этой клавиши подпрограмма обработки прерывания клавиатуры (вектор #60) вычисляет и заносит в ячейку @#111 количество выводимых по «ТАБ» пробелов. ЕМТ6 проверяет значение ячейки @#111 и, если она содержит не нуль, возвращает в качестве очередного символа пробел, а содержимое ячейки @#111 уменьшает на 1. Если же ячейка @#111 содержит нуль (все пробелы по табуляции выведены, а снова клавиша «ТАБ» не нажималась), то ЕМТ6 переходит к последнему шагу.

Не нажата ли клавиша «ПОВТ»? Если флаг нажатия «ПОВТ» (ячейка @#110) установлен и бит 6 регистра @#177716 сброшен в нуль (клавиша «ПОВТ» еще не отпущена), отрабатывается цикл, задержка которого записана в ячейке @#106, и в качестве очередного символа возвращается последний введенный перед «ПОВТ», код которого ранее запомнен в буфере.

И только в том случае, если все эти этапы пройдены, ЕМТ6 обращается к коду символа в буфере @#104 — @#105, который помещен туда программой обработки прерывания от клавиатуры. Если же он там отсутствует, все, начиная с проверки на программируемые ключи, повторяется сначала — ЕМТ6 «ждет», пока будет нажата клавиша, сработает прерывание, код появится в буфере ЕМТ6 и т. д.

Учитывая, что функция ЕМТ10 (ввод строки) программно реализована путем циклического вызова ЕМТ6, получаем, что ряд последовательных обращений к ЕМТ6 или ЕМТ10 либо вызов их в цикле передает в программу на ассемблере символ за символом всю строку ключа. А так как трансляторы ФОКАЛа и БЕЙСИКа сами по себе являются программами на ассемблере (отранслированными в машинные коды и прошитыми в ПЗУ), то реализации соответствующих операторов чтения символов с клавиатуры (кроме INKEY$, который реализован не через ЕМТ6, и его циклический вызов дает строку ключа после «АР2»+«<цифра>» только при нажатии еще одной «инициирующей» клавиши) работают точно так же и в этих языках программирования. Правда, стандартный аппарат KEY в БЕЙСИКе неудобен тем, что длина строк ключей ограничена 16 символами и что для «задействования» ключа нужно нажать «АР2»+«<цифра>». Но, зная, как при стандартной отработке ключа используются рабочие ячейки @#122 и @#124, мы можем сами записать в них адрес начала и длину нужной нам строки текста, хранящейся в ОЗУ, и ЕМТ6 немедленно отреагирует на это. Именно на данном принципе и построены все программы реализации «командного режима» на БК, в том числе и предложенная в данной статье. Однако, как справедливо замечено автором, одно неудобство при этом остается — все строки, заданные в качестве ключей, выводятся на экран, «засоряя» его. Как быть? И автор предлагает простое и изящное решение — «отловить» функции вывода символов на экран ЕМТ16 и ЕМТ20, перехватывая вектор ЕМТ и проверяя, какая функция ЕМТ вызвана, а «отловив», сразу произвести выход из прерывания, не передавая управления подпрограммам вывода на экран. Остальные же функции обрабатываются путем передачи управления на стандартный ЕМТ-диспетчер в ПЗУ. Но из-за этого при отработке «командного режима» с помощью программы UPRKOM.BIN невозможно использование каких-либо команд, выводящих на экран полезную информацию (например, LIST). По той же причине первым оператором, который выполняется сразу после завершения отработки строки ключа (либо последним оператором, запрограммированным в этой строке), должен быть POKE &O30,&O100112 , иначе после отработки ключа компьютер «оглохнет и ослепнет» — на экран не будет выводиться вообще ничего и создастся впечатление, что ЭВМ «зависла».

Нам пишут

А. Глускер,
Москва

Динамическая индикация вычислительных процессов

Одним из удобнейших средств демонстрации хода процесса вычислений, обмена с диском, печати на принтере и т. д. является динамический индикатор, в котором по мере выполнения операции постепенно «закрашивается» горизонтальная полоска. В большинстве случаев индикатор разбивают на 16 частей, так как это число «круглое» в двоичной системе счисления.

Пусть X — объем всего процесса. Тогда при истечении 1/16 части от X нужно дозаполнить индикатор на одно деление. Но на языке ассемблера возможна работа только с целыми числами, поэтому используем прием, аналогичный заложенному в алгоритм рисования окружностей (см. №3 за 1995 г., с.14).

Если Y = 1/16 части объема процесса, то X/16=Y и X=16*Y. Введем переменную Z, которая будет обозначать соответствие между выполненной работой и «позицией индикатора». При увеличении позиции индикатора на одно деление Z уменьшается на X. С ростом объема выполненной работы Z увеличивается на число, соответствующее этому приращению, умноженному на 16. Увеличивать позицию индикатора мы будем, когда Z≥X. (Следует учесть, что за один прием может понадобиться увеличить позицию индикатора больше чем на одно деление.)

Так как на ассемблере меньше места и времени занимает команда сравнения с нулем (TST), а не с каким-либо числом (CMP), начальное значение Z мы установим равным −X, а увеличивать позицию индикатора будем при условии Z≥0.

В качестве примера рассмотрим листинг программы проверки дискеты на плохие секторы.

```
;TEST.ASM
;(C) Reiter software corporation 1995
;Очистка экрана, установка режима 32 символа
;в строке и гашение курсора

READDC=160006                          ; Символические имена
INITDC=160010                  ; для п/пр прошивки ПЗУ КНГМД

        MOV #1000,SP
        MTPS SP
        EMT 14
        RESET           ;Отключить прерывание от клавиатуры
        MOV (PC)+,R0
        .BYTE 233,232
        EMT 16
        SWAB R0
        EMT 16
        CALL CLS
        JSR R5,TYPE2
        .BYTE 2.
        .ASCIZ /(C) Reiter corporation 1995/
        .EVEN
        JSR R5,TYPE1
        .BYTE 2.,3.
        .ASCII /Проверка диска на плохие блоки/<12>
        .ASCIZ /Имя дисковода(A-D):/
        .EVEN
```

```
        MOV #RAB,R3                 ;Рабочая область в R3
        CALL @#INITDC     ;Инициализируем ее, в R3 - адрес
                                    ; рабочей области
1:      CALL READKLA
        SUB #101,R0                 ;Код меньше "A"?
        BMI 1
        BIC #40,R0                  ;Приводим к "ЗАГЛ"
        CMPB R0,#'D-101             ;Код больше "D"?
        BHI 1
        MOVB R0,34(R3)        ;Установили номер дисковода
        ADD #101,R0           ;Печатаем название дисковода
        EMT 16
                        ;Для демонстрации - режимы работы
              ;"одна сторона" и "без пропуска дорожек"
        JSR R5,TYPE1
        .BYTE 2.,5.
        .ASCIZ /Номер  первой дорожки:/
        .EVEN
        CALL READWO
        MOV R1,R2
        JSR R5,TYPE1
        .BYTE 2.,6.
        .ASCIZ /Номер последней дорожки:/
        .EVEN
        CALL READWO
        SUB R2,R1
                ;В R1 - кол-во рабочих дорожек минус 1
        JSR R5,TYPE1
        .BYTE 10.,0.
        .ASCII /+----------------+/<12>
        .ASCII /!                !/<12>
        .ASCIZ /+----------------+/
        .EVEN
        JSR R5,TYPE1
        .BYTE 0.,7.
        .ASCII /Нажмите любую клавишу для начала/<12>
        .ASCIZ /работы/
        .EVEN
        MOV #NAC,@#4
        CALL READKLA
NAC:    MOV #1000,SP
        MOV #STOP,@#4
        MOV R2,DOR                          ;Текущая дорожка
        MOV R1,NEXT         ;Кол-во необработанных дорожек
        INC R1
        CALL MUL10
        ASL R1                        ;R1 умножили на 20.
        MOV R1,X                                    ;X
        NEG R1
        MOV R1,Z                    ;Начальное значение Z
        MOV #11.,POZ                ;Позиция в индикаторе
4:      MOV #1,STORONA
3:      MOVB STORONA,32(R3)                ;Сторона диска
        MOV #BUF,26(R3)                 ;Установка адреса
        MOV #5000,30(R3)                           ;Длина
        MOVB DOR,33(R3)                   ;Номер дорожки
        MOVB #1,35(R3)                    ;Номер сектора
        CALL @#READDC                             ;Чтение
        BCC 1                              ;Ошибка есть?
        CMPB @#52,#7                ;Прерывание по "СТОП"
        BEQ STOP
        JSR R5,TYPE1
        .BYTE 2.,11.
        .ASCIZ /Error #/
        .EVEN
        MOVB @#52,R0                           ;Номер ошибки
        ADD #60,R0
```

```
        EMT 16
        JSR R5,TYPE
        .ASCIZ / in track #/
        .EVEN
        MOV DOR,R1
        CALL TYPE10
        PUSH R3
        MOV #60,R3                              ;Звук
        MOV #20,R2
        CALL @#102062
        CLR R0                                  ;Пауза
        SOB R0,.
        POP R3
1:      ADD #160.,Z                             ;10*16
10:     TST Z
        BMI 2
        SUB X,Z
        MOV POZ,R1                          ;Печать метки
        MOV #1.,R2
        EMT 24
        INC POZ
        MOVB #' ,R0
        EMT 16
        BR 10
2:      DEC STORONA                     ;Следующая сторона
        BPL 3
        INC DOR                         ;Следующая дорожка
        DEC NEXT
        BPL 4
        CLR @#177130                            ;Все
EXIT2:  JSR R5,TYPE2
        .BYTE 0.
        .ASCIZ /Для выхода нажмите любую клавишу/
        .EVEN
        BR EXIT3
STOP:   CLR @#177130
        MOV #1000,SP
        JSR R5,TYPE2
        .BYTE 2.
        .ASCIZ /Прерывание оператора        /   ;7 пробелов
        .EVEN
EXIT3:  CALL READKLA
EXIT:   JMP @#100000

;Чтение кода с клавиатуры
READKLA:TST @#177662                ;Сброс предыдущего кода
1:      BIT #200,@#177660     ;Ожидание нажатия на клавишу
        BEQ 1
        MOV @#177662,R0                     ;Чтение кода
        CMPB R0,#3
        BEQ EXIT
        RET

;П/п печати строк, начиная с указанной позиции
TYPE:   PUSH R2 PUSH R1 PUSH R0
        EMT 26
        BR TYPE3
TYPE1:  PUSH R0 PUSH R1 PUSH R2           ;Регистры в стек
        MOVB (R5)+,R1                             ;Позиция
        MOVB (R5)+,R2
TYPE3:  PUSH R1 PUSH R2             ;Может понадобиться...
        EMT 24
2:      MOVB (R5)+,R0                             ;Символ
        EMT 16                              ;Вывод на экран
        BEQ 1                                 ;Весь текст?
        CMPB R0,#12                           ;Конец строки?
        BNE 2
```

```
        POP R2 POP R1                                   ;Да -
        INC R2                                  ;следующая строка
        BR TYPE3                                        ;В начало
1:      POP R2 POP R1 POP R2    ;Восстанавливаем регистры
        BR RETTYPE                                      ;На выход
;Показан простой способ уменьшения длины программы,
;но возможно ухудшение читаемости из-за отсутствия
;структурности (выход из всех п/пр через одни и те же
;команды). Если при вызове кодировать: бит C=1 - одна
;программа, а C=0 - другая, то появится структурность

;П/п печати строки в служебной строке
TYPE2:  PUSH R1 PUSH R0                         ;Засылка в стек
        MOVB (R5)+,R1                           ;Номер позиции
2:      MOVB (R5)+,R0                           ;Код символа
        BEQ RETTYPE
        EMT 22                                  ;не 0 - печатаем
        INC R1                                  ;Следующая позиция
        BR 2
RETTYPE:POP R0 POP R1           ;Восстанавливаем регистры
        INC R5                                  ;Делаем R5 четным
        BIC #1,R5
        RTS R5

;П/п ввода десятичного числа в R1
READW0: PUSH R0                                 ;Сохраняем R0
        CLR R1                          ;Готовим регистр для числа
0:      CALL READKLA                                    ;Читаем код
        CMPB R0,#12                                     ;"ВВОД"?
        BEQ 1
        SUB #60,R0                              ;Код меньше "0"?
        BMI 0
        CMPB R0,#10.                            ;Код больше "9"?
        BHIS 0
        CALL MUL10
        ADD R0,R1                                       ;Получили число
        ADD #60,R0                      ;Печатаем код клавиши
        EMT 16
        BR 0
1:      POP R0                                  ;Восстановим регистр
        RET

;Умножить R1 на 10.
MUL10:  PUSH R2                                         ;R1=R1*10.
        ASL R1
        MOV R1,R2
        ASL R1
        ASL R1
        ADD R2,R1
        POP R2
        RET
CLS:    PUSH R0
        MOV #40000,R0
        CLR (R0)+                               ;Очистка экрана
        TST R0
        BPL CLS2
        POP R0
        RET

;Печать двухзначного десятичного числа
TYPE10: PUSH R1 PUSH R0
        MOV #57,R0
1:      INC R0
        SUB #10.,R1
        BPL 1
        EMT 16
        MOV R1,R0
```

```
        ADD #68+10.,R0
        EMT 16
        POP R0 POP R1
        RET

X:      .BLKW 1
Z:      .BLKW 1
NEXT:   .BLKW 1
DOR:    .BLKW 1
POZ:    .BLKW 1
STORONA:.BLKW 1
RAB:    .BLKB 66        ;Рабочая область драйвера дисковода
BUF:    .END
```

Примечание редактора

Программа тестирования дисков, приведенная в качестве примера использования динамического индикатора, написана на довольно редком «диалекте» ассемблера БК, в котором для удобства пользователя введены «укороченные» варианты наиболее часто применяемых команд (подобно знакомой всем БКманам записи CALL вместо JSR PC). Формат ассемблера здесь почти полностью соответствует стандарту Turbo с учетом трех основных отличий:

- PUSH <регистр> аналогично MOV <регистр>,-(SP) ;
- POP <регистр> аналогично MOV (SP)+,<регистр> ;
- точка в команде SOB означает переход на ту же самую строку (т. е. SOB R0,. аналогично 000: SOB R0,000 , где 000 — неиспользуемая в других строках локальная метка).

Отметим также размещение нескольких команд POP или PUSH в одной строке. В трансляторах Turbo это допустимо в большинстве случаев, а в других трансляторах нужно разбить такие строки на несколько отдельных.

Ю. В. Котов,
Москва

Использование TRAP-прерываний БЕЙСИКа в кодовых подпрограммах

Во втором выпуске журнала «Персональный компьютер БК-0010 — БК-0011М» за 1995 г. была опубликована статья «Подключение кодовых подпрограмм к БЕЙСИК-программам на этапе трансляции» с предложением использовать TRAP-прерывания обрабатывающей системы БЕЙСИКа. Однако эти прерывания при недостаточно корректном применении иногда дают сбои, пропускаются некоторые команды пользователя и т. д., поэтому имеет смысл дать к упомянутой статье несколько дополнительных замечаний.

В свое время программисты, особенно «системные», старались максимально экономить память еще не очень мощных машин и изощрялись в рационализации своих творений, благо система команд DEC предоставляет для этого множество возможностей.

Драйвер TRAP-прерываний системы БЕЙСИКа перед вызовом обрабатывающей подпрограммы или выдачей сообщения об ошибке временно запоминает в стеке содержимое регистров R0 и R2, занося из стека в R0 адрес возврата. После этого уже в R2 выбираются младшие разряды команды прерывания, и если разряд 6 нулевой, производится вывод сообщения об ошибке, иначе отрабатывается команда CALL @ATRAP(R2) (коды 4772, 146716), где ATRAP — адрес таблицы адресов подпрограмм. (Кстати, это хороший пример реализации «вычисляемого» вызова подпрограмм, что в БЕЙСИКе соответствует оператору ON XX GOSUB P1,P2,...)

После отработки подпрограммы содержимое регистров переносится в стек на место адреса возврата, а прежние значения R0 и R2 восстанавливаются. Это дает возможность в некоторых подпрограммах смещать точку возврата, изменяя нулевой регистр. В результате получается эффект «вычисляемого возврата», т. е. после выхода из прерывания по RTI следующей

выполняемой командой будет не обязательно непосредственно расположенная за командой прерывания. Благодаря этому несколько машинных слов после команды прерывания могут быть использованы для размещения команд перехода к разным ветвям программы — способ весьма экономный. Тем же способом в подпрограмму из вызывающей программы реализуется передача параметров — адресов, чисел или кодов, записанных после команды прерывания. Поскольку мы имеем дело с ПЗУ, они не могут быть переменными значениями, но при вызовах прерывания из разных мест программы параметры различны. (Впрочем, при написании USR-подпрограммы в оперативной памяти это ограничение снимается.) Так, подпрограмма TRAP 101 следующее слово после команды прерывания использует как адрес цепочки байтов, с которыми сравнивается текущий байт, адресуемый в регистре R3. Подпрограмма TRAP 102 анализирует адресуемый в R3 символ и по полученному результату производит возврат на следующую после вызова прерывания ячейку, если это специальный символ; на вторую ячейку — для латинской буквы (строчные переводятся в заглавные) и в третью — для цифры. Несколько выходов имеют также подпрограммы TRAP 105, TRAP 115, TRAP 116 и др. Примером же передачи параметров являются подпрограммы TRAP 101 и TRAP 113.

«Хитрые» способы передачи управления и возврата достигаются и другими методами, например в команде JSR R4,ADR1, после которой идет одна или несколько групп адресов подпрограмм «объектного кода» с завершающим каждую группу адресом продолжения основной программы в машинных кодах. При таком вызове «старое» значение R4 записывается в стек, а вместо него в R4 попадает адрес первого из слов с адресом «объектной подпрограммы». Затем осуществляется переход к адресу ADR1, где может быть изменен адрес в R4. Завершается эта подпрограмма командой JMP @(R4)+, после чего управление передается по адресу, указанному в группе слов после команды JSR R4,ADR1 — в первом, втором, третьем и т. д. слове (или в первой, второй, третьей и т. д. группе), в зависимости от того, наращивалось ли и насколько значение в R4. Подпрограммы, названные здесь «объектными», отличаются от обычных только тем, что завершаются также командой 134 и передают управление следующей такой же подпрограмме, указанной в списке после JSR. Список отрабатываемых таким образом подпрограмм заканчивается адресом продолжения обычной программы или подпрограммы, завершающейся командой RETURN. Так, в подпрограмме GET (обработка оператора БЕЙСИКа INPUT) имеется фрагмент:

```
JSR R4,TODBL
.WORD SD,DXMOV,G33
.WORD ID,DXMOV,G33
...
```

В подпрограмме TODBL значение R4 может быть увеличено до 6 байтов. Отработав подпрограмму TODBL (кончающуюся командой JMP @(R4)+), машина переходит к «объектной» подпрограмме SD или к ID из набора «объектных» блоков (перевод числа из «короткого» в «длинное» или из «целого» в «длинное»). Дальше в обоих случаях отрабатывается DXMOV — пересылка числа из стека по адресу, также заданному в определенном месте стека, и наконец, переход к продолжению GET в точке G33.

От редакции

О нумерации строк ФОКАЛ-программ

В прилагаемой к БК-0010(.01) при продаже документации по ФОКАЛу сказано, что для нумерации программных строк допускается использование чисел в диапазонах от 1.01 до 99.99 с шагом 0.01 и от 100.1 до 127.9 с шагом 0.1 (кроме чисел, заканчивающихся нулем).

В результате проведенных экспериментов выяснилось, что ФОКАЛ БК-0010 допускает использование номеров строк из диапазона 100.1 — 127.9 с шагом 0.01. Хотя при выводе листинга номера округляются по первой цифре после точки (например, выводится несколько строк с номерами 100.1), это все же РАЗНЫЕ строки, функционально ничем не отличающиеся от занумерованных по всем правилам (в чем можно убедиться, попытавшись вызвать одну из них на редактирование командой M). В качестве примера приведем небольшой демонстраци-

онный листинг. Наберите его, выведите на экран командой W, запустите на исполнение, чтобы проследить, как БК обрабатывает строки с нестандартными номерами.

```
C: ФОКАЛ-БК0010
10.01 C Демонстрационный листинг
10.10 A !!"Введите номер строки (100.01 - 100.1)",N
10.20 T !
10.30 D N ; C Выполнить строку с заданным номером
10.40 T !
10.50 G 10.10
99.99 C Далее следует ряд "нестандартных" строк
100.01T "Строка 100.01"
100.02T "Строка 100.02"
100.03T "Строка 100.03"
100.04T "Строка 100.04"
100.05T "Строка 100.05"
100.06T "Строка 100.06"
100.07T "Строка 100.07"
100.08T "Строка 100.08"
100.09T "Строка 100.09"
100.1 T "Строка 100.1"
100.16 C Конец списка нестандартных строк
```

Работа с файлером в дисковом БЕЙСИКе

Одним из значительных недостатков дискового БЕЙСИКа во всех прежних версиях ANDOS оставалась необходимость вводить имена загружаемых файлов по памяти. Просмотр же каталога диска по команде FIND тоже сопряжен с определенными неудобствами, так как имена файлов выводятся неотсортированными и без сохранения структуры подкаталогов. Но теперь, когда появилась версия ANDOS 3.1 с встроенным файлером, обо всех этих неприятностях можно забыть.

Чтобы воспользоваться встроенным файлером ANDOSа в дисковом БЕЙСИКе, нужно задать в команде CLOAD, BLOAD или LOAD имя, состоящее из одного символа пробела (наличие кавычек с обеих сторон имени здесь обязательно, иначе выдается сообщение об ошибке 56). При этом в файлере автоматически устанавливается соответствующее расширение имен (для команды CLOAD выводятся только файлы с расширением .COD, для BLOAD — с расширением .BIN, для LOAD — .000). После выбора нужного файла и нажатия клавиши «ВВОД» дальнейшая загрузка производится как и при явном указании имени.

(Данный факт проверен в дисковом БЕЙСИКе DBASIC из комплекта ANDOS 3.1. КНГМД стандарта «АльтПро».)

Подгружаемый DiskMaster для ANDOS 3.1

Появившаяся в продаже осенью 1994 г. новая версия дисковой операционной системы ANDOS (v3.1) при многочисленных преимуществах обладает и некоторыми недостатками*. Один из них, довольно неприятный, — «сброс» оболочки DiskMaster после использования для

* В этой статье имеется в виду весь комплект операционной системы, а не собственно дисковый монитор ANDOS, претензий к которому не имеется.

своих нужд области дополнительного ОЗУ 140000—157777. Ранее (в версиях ANDOS 2.50 и ниже) получили широкое распространение программы, размещающиеся в этой области или занимающие ее для своих подпрограмм и данных, — различные копировщики длинных файлов, перекодировщики текстов и т. д. Да и всем известный VorteX! 1.05 позволяет существенно увеличить объем обрабатываемого текста за счет размещения своих модулей в ДОЗУ. Теперь же (в версии ANDOS 3.1) после выхода из этих программ или перезапуска по кнопке «RESET» в распоряжении пользователя остается лишь монитор ANDOS и встроенный в него файлер.

Этот-то файлер и виноват (правда, косвенно) во всех бедах: именно он вытеснил оболочку DiskMaster в область ДОЗУ с адреса 140000. В прежних версиях основная часть оболочки располагалась вместо файлера, а дополнительные функции («View», «Rename» и др.) были реализованы в подгружаемом (оверлейном) модуле, размещаемом с адреса 140000. А если этот модуль оказывался испорчен, основная часть DiskMaster'а автоматически подгружала оверлей, если находила его на диске. (Впрочем, худо-бедно работать с оболочкой можно было и без него.) А в версии 3.1 вся файловая оболочка составляет единое целое с монитором ANDOS, совместно занимая все ДОЗУ. И если DiskMaster «сброшен», восстановить его можно только одним способом — заново перезагрузившись с системной дискеты.

Указанный недостаток нетрудно устранить самостоятельно. При работающей оболочке DiskMaster запустите какой-либо отладчик, загрузив его в основное ОЗУ (например, DEBU10 с адреса 1000). Затем средствами этого отладчика запишите на диск в виде файла содержимое области ДОЗУ с адреса 140000 длиной 20000. Отдельный оверлей оболочки DiskMaster получен, можно скопировать его на все рабочие дискеты (обычным способом, с помощью команды «Copy»). Теперь если после выхода из программы, работающей в области 140000—157777, или VorteX!'а DiskMaster окажется «сброшен», загрузите с помощью встроенного файлера полученный вами оверлей, нажмите «АР2»+«2», и файловая оболочка вновь к вашим услугам, а системную дискету доставать не понадобится.

Д. Ю. Усенков,
Москва

Есть ли файл на диске?

При разработке «коммерческих» программ для IBM-совместимых компьютеров часто используется несложный, но очень удобный прием: прежде чем пытаться загрузить файл (оверлей, заставку или данные), проверяется его наличие на диске. Такая возможность есть, например, в наборе команд batch-файлов для IBM: оператор IF EXIST обеспечивает отработку одной из двух алгоритмических цепочек в зависимости от того, найден ли файл с указанным именем на диске.

В программах же для БК подобная возможность, к сожалению, практически никогда не используется, из-за чего пользователь обязан сам следить за наличием на диске всех файлов программного пакета (хотя, может быть, какие-то из них необязательны: например, при отсутствии файла-заставки можно переходить сразу к рабочей части программы) и за правильностью ввода имен. В противном случае на экране появляется сообщение типа «File not found», да и то если оно предусмотрено в данной дисковой ОС. (В худшем же случае программа раз за разом слепо «тычется в дискету», ничего не сообщая пользователю, как в некоторых адаптациях многофайловых магнитофонных игр к дисковой ОС Нортон.)

Впрочем, пора от критики перейти к делу. В операционной системе ANDOS имеется богатый набор «служебных» подпрограмм (предназначенных для нужд самой системы), которые можно использовать и в работающих в среде этой ОС прикладных программах (спасибо автору ANDOS А. М. Надежину за подробное техническое описание, прилагаемое к каждой версии!). Есть среди них и функция поиска на диске файла с заданным именем. Осталось только написать на ее основе короткую подпрограмму. (Приведенный ниже листинг соответствует стандарту ассемблера М18. Использование же обладающего более богатыми возможностями транслятора Turbo позволит несколько оптимизировать данную подпрограмму.)

```
;  Определение наличия файла на диске
;  Входные данные:  R1 - адр. блока параметров для EMT 36
;  Выходные данные: R0=0   - файл найден;
;                   R0=1   - файл не найден;
;                   R0=-1. - диск не в формате ANDOS
; ---------------------------------------------------------
;
EXIST:   MOV R5,-(SP)       ; сохранить
         MOV R2,-(SP)       ;  регистры в стеке
         MOV R1,-(SP)
         MOV @#120204,R0    ; извлечь адрес INIDRV
         JSR PC,@R0         ; инициализация диска
         BCS 1              ; переход, если не формат ANDOS
         MOV @R1,-(SP)      ; прежнюю команду для EMT 36 - в стек
         CLR @R1            ; дать команду "нуль"
         EMT 36             ; перепаковка имени в NAMBUF
         MOV (SP)+,@R1      ; прежнюю команду EMT 36 - на место
         MOV @#120260,R0    ; извлечь адрес DIRFI2
         JSR PC,@R0         ; поиск файла с именем в NAMBUF
         BCS 2              ; переход, если не найден
         CLR R0             ; признак "файл найден"
         BR 3               ;  и на выход
2:       MOV #1,R0          ; признак "файл не найден"
         BR 3               ;  и на выход
1:       MOV #177777,R0     ; признак "не формат ANDOS"
3:       CLR @#177130       ; остановить мотор дисковода
         MOV (SP)+,R1       ; регистры -
         MOV (SP)+,R2       ;  из стека
         MOV (SP)+,R5
         RTS PC             ; возврат
; ---------------------------------------------------------
```

Использованные в комментариях обозначения INIDRV и DIRFI2 соответствуют именам «служебных» подпрограмм в технической документации к ANDOS 3.1 (в прежних версиях эти имена и соответствующие им адреса могут быть иными).

Работа с подпрограммой EXIST не сложна. Перед ее вызовом нужно занести в регистр R1 адрес начала блока параметров для EMT 36, а после возврата — проверить содержимое R0: нуль означает, что файл найден, единица — отсутствует, число 177777_8 (-1_{10}) — проверяемый диск не в формате ANDOS. Следующий листинг является более чем наглядной иллюстрацией сказанного.

```
         . . . . .  ; подготовка блока параметров для EMT 36,
         . . . . .  ;  например для чтения заставки: команда
         . . . . .  ;  3 (загрузка), нач. адрес = 40000, длина
         . . . . .  ;  нулевая, имя введено с клавиатуры
         . . . . .  ;  с помощью отдельной подпрограммы
0:  MOV #<A>,R1     ; <A> - адрес начала блока параметров
    JSR PC,EXIST    ; вызов подпрограммы
    TST R0          ; проверка равенства R0 нулю
    BEQ 1           ; переход, если файл найден
    DEC R0          ; (или CMP) - проверка равенства R0 единице
    BEQ 2           ; переход, если файл не найден
         . . . . .  ; иначе ясно, что возвращено R0=177777
         . . . . .  ;  (диск не в формате ANDOS) и выполняются
         . . . . .  ;  соответствующие действия. Например, вывод
         . . . . .  ;  запроса "Вставьте другой диск" и переход
         . . . . .  ;  на метку 0 для повторного поиска
2:       . . . . .  ; действия, если файл не найден (например, вывод
         . . . . .  ;  сообщения и повторный поиск (метка 0)
         . . . . .  ;  или переход сразу к рабочему модулю)
1:  EMT 36          ; файл найден, и его можно тут же загрузить,
         . . . . .  ;  благо подпрограмма EXIST сохраняет значение
         . . . . .  ;  регистра R1 и содержимое блока параметров
```

М. Серегин,
Москва

Печать из БЕЙСИКа БК-0010.01 на принтере МС6313 без аппаратных доработок

Как известно, богатые возможности владельцам компьютера БК-0010.01 предоставляет подключение принтера. В настоящее время большое распространение получил принтер «Электроника МС6313», для которого создано множество программ. Однако, к сожалению, подавляющее большинство этих программ позволяют получать на бумаге «твердые копии» только готовых текстовых и графических документов. А как быть, если необходимо быстро и красиво распечатать результаты каких-либо расчетов с постоянными формулами при изменении исходных данных? Можно, конечно, набрать в текстовом редакторе «бланк» выводимых расчетов, потом, пересчитывая результаты на калькуляторе, вписать их в текст и вывести его на принтер, затем все пересчитать снова и т. д. Но всем ясно, что такой метод отнюдь не лучший из возможных.

Проще всего решить данную проблему позволяет БЕЙСИК БК-0010.01, в котором имеется оператор LPRINT, обеспечивающий вывод на принтер любых текстовых данных. Но увы, этот оператор не поддерживает интерфейс принтера МС6313, и в настоящий момент известен только один способ заставить БЕЙСИК БК-0010.01 работать с этим принтером, для чего требуется по особому способу распаять шлейф, соединяющий принтер с компьютером (см.: Информатика и образование. 1992. №1. С. 78).

Но нельзя ли печатать из «десяточного» БЕЙСИКа без аппаратных доработок? Несложная программа на БЕЙСИКе в какой-то степени отвечает на этот вопрос.

```
10 N$="Привет всем"+CHR$(10)+CHR$(13)+"БКшечникам!"
20 FOR I%=1% TO LEN(N$)
30 I$=MID$(N$,I%,1%)
40 POKE -52%,ASC(I$)
50 POKE -52%,ASC(I$)+&O400
60 POKE -52%,ASC(I$)
70 NEXT I%
```

Пользователи, хотя бы немного знающие БЕЙСИК, легко поймут принцип ее работы, остается только рассказать, как установить переключатели в принтере:

```
10001001.00000111.00000001
```

Здесь «1» — «клювик» переключателя повернут вверх (от себя), «0» — вниз (к себе).

**Мечтая, будь осторожен!
Мечты иногда сбываются...**

Морин Хэрли

«Пьяница»

(гротескная игра с звуко-речевыми эффектами)

Вы идете по улице мимо заборов, урн, фонарных столбов и крупных булыжников. Но вот вдруг появляется полная рюмка вина, за ней вторая... Вы человек практичный и знаете, что делать в таких случаях — надо подойти и выпить. Но, как только вы выпьете одну-две рюмки, вы чувствуете, что ноги начинают отниматься. Вас уже шатает, не наткнуться бы на столб или камень, не говоря уже о том, чтобы добраться до вожделенной рюмочки...

Ваша задача: «борясь с законом всемирного тяготения», выпить как можно больше. Игра заканчивается, если вы на что-нибудь наткнетесь (компьютер при этом довольно внятно чертыхается).

```
5 DIM P(6,2),M(32),ML(30),PR(6,49),
   Z1(12),Z2(6),Z3(16)
10 CLS
15 GOSUB 940
20 ? AT(12,7);CHR$(147);"ИГРА";
   AT(10,12);CHR$(146); "ПЬЯНИЦА.";
30 ? AT(5,20);CHR$(145);"НАЖМИТЕ
   ЛЮБУЮ КЛАВИШУ."
40 IF INKEY$=""GOTO 40
50 XM=15
60 LET ST=0
65 CLS
66 RESTORE 140
70 FOR I=1TO6
80 READ P(I,2)
90 P(I,1)=INT(RND(1)*27+2.5)
100 IF P(I,1)=15GOTO90
110 P(I,0)=INT(RND(1)*4+1.5)
120 GOSUB 890
130 NEXT
140 DATA 2,6,9,12,16,20
150 GOSUB 790
160 ? AT(8,0);CHR$(146);"PRESS ANY KEY,
    PLEASE !";CHR$(145)
170 IF INKEY$=""GOTO170
175 ? AT(8,0);"                    "
180 CH$=INKEY$
190 IF CH$=""GOTO280
195 BEEP
200 IF ASC(CH$)<>8GOTO240
210 ? AT(XM,12);" ";
220 IF XM>1THENXM=XM-1
230 GOTO 290
240 IF ASC(CH$)<>25GOTO280
250 ? AT(XM,12);" ";
260 IF XM<30THENXM=XM+1
270 GOTO 290
280 ? AT(XM,12);" ";
290 XM=XM+SGN(SM)
300 IF SM<-1ANDSM>1THENSM=SM-SGN(SM)
    ELSESM=0
310 GOSUB 790
320 FOR I=1TO6
330 ? AT(P(I,1),P(I,2));" ";
340 IF P(I,2)=22GOTO380
350 P(I,2)=P(I,2)+1
360 GOSUB 890
370 GOTO 430
380 P(I,1)=INT(RND(1)*27+2.5)
390 IF P(I,1)=XM GOTO380
400 P(I,0)=INT(RND(1)*4+1.5)
410 P(I,2)=1
420 GOTO 360
430 NEXT
440 FOR I=1TO6
450 IF XM=P(I,1)ANDP(I,2)=12THEN460
    ELSE760
460 IF P(I,0)=3GOTO690
```

```
470 ? AT(XM,12);" ";AT(P(I,1),P(I,2));" ";
480 P(I,2)=P(I,2)-1
490 GOSUB 890
500 GOSUB 840
510 ? AT(12,0);CHR¤(146);
    "Ч-ЧЕРТ ПОБЕРИ !";CHR¤(145)
520 GOSUB 1160
530 FOR I=1TO1000
540 NEXT
550 CLS
560 ? AT(3,3);CHR¤(145);"СЭР!
    ВЫ РАЗБИЛИ СЕБЕ НОС!"
570 IF ST=0GOTO650
580 IF ST<5GOTO620
590 ? AT(1,5);CHR¤(146);ST;
    "СТАКАНОВ-ПРИЛИЧНАЯ ДОЗА."
600 ? AT(5,7);CHR¤(147);
    "ВЫ УМЕЕТЕ ПИТЬ!";CHR¤(145)
610 GOTO 660
620 ? AT(1,5);CHR¤(146);
    "И ЭТО С";ST;"СТАКАНОВ ?"
630 ? AT(2,7);CHR¤(147);"СЭР!ВЫ,
    ВЕРОЯТНО,БЫВШИЙ";AT(4,8);
    "ТРЕЗВЕННИК ?";CHR¤(145)
640 GOTO 660
650 ? AT(1,5);CHR¤(146);
    "ЕСЛИ ВЫ ТРЕЗВЫЙ НА ВСЕ";
    AT(0,6); "НАТЫКАЕТЕСЬ,
    ТО ЧТО БУДЕТ,";AT(0,7);
    "КОГДА ВЫ НАПЬЕТЕСЬ ?"; CHR¤(145)
660 ? AT(10,10);"GOOD-BYE !"
670 GOSUB 1220
680 END
690 ? AT(P(I,1),P(I,2));" ";
700 LET ST=ST+1
705 ? AT(27,0);CHR¤(145);CHR¤(156);" ";
    ST;CHR¤(156);
710 ? AT(12,0);CHR¤(146);
    "БУЛЬ-БУЛЬ-БУЛЬ!";CHR¤(145);
720 GOSUB 1280
730 P(I,0)=6
740 GOSUB 790
750 ? AT(12,0);"
760 NEXT
765 IF SM<>0GOTO180
770 SM=INT(ST*2*RND(1)+.5)-ST
785 GOTO 180
790 ? AT(XM,12);
800 FOR J=1TO32
810 ? CHR¤(M(J));
820 NEXT
830 RETURN
840 ? AT(XM,12);
850 FOR J=1TO30
860 ? CHR¤(ML(J));
870 NEXT
880 RETURN
890 ? AT(P(I,1),P(I,2));
900 FOR J=1TOPR(P(I,0),0)
910 ? CHR¤(PR(P(I,0),J));
920 NEXT
930 RETURN
940 RESTORE 3000
950 FOR I=1TO32
960 READ M(I)
970 NEXT
980 FOR I=1TO30
990 READ ML(I)
1000 NEXT
1010 FOR I=1TO6
1020 READ PR(I,0)
1030 FOR J=1TOPR(I,0)
1040 READ PR(I,J)
1050 NEXT J,I
1060 FOR I=1TO12
1070 READ Z1(I)
1080 NEXT
1090 FOR I=1TO6
1100 READ Z2(I)
1110 NEXT
1120 FOR I=1TO16
1130 READ Z3(I)
1140 NEXT
1150 RETURN
1160 FOR J=1TO12ST2
1170 T=Z1(J)
1180 S=Z1(J+1)
1190 GOSUB 1340
1200 NEXT
1210 RETURN
1220 FOR J=1TO6ST2
1230 T=Z2(J)
1240 S=Z2(J+1)
1250 GOSUB 1340
1260 NEXT
1270 RETURN
1280 FOR J=1TO16ST2
1290 T=Z3(J)
1300 S=Z3(J+1)
1310 GOSUB 1340
1320 NEXT
```

Рисунок И. Новикова («Крокодил». 1982. №12)

```
1330 RETURN
1340 FOR II=1TOT
1350 OUT &O177716,64,1
1360 S0=S
1370 S0=S0-1
1380 IF S0>0GOTO1370
1390 OUT &O177716,64,0
1400 S0=S
1410 S0=S0-1
1420 IF S0>0GOTO1410
1430 NEXT
1440 RETURN
3000 DATA 145,149,57,27,25,150,25,50,
     26,29,54,26,25,54, 27,30,50,27,25,
     26,150,52,26,150,50,26,53,8,51,27,
     150,149
3010 DATA 149,145,57,27,150,25,50,29,
     51,26,50,31,50, 29,25,51,27,50,30,
     25,28,150,53,26,150,27,50,31,150,149
3020 DATA 33,149,55,27,147,150,50,27,
     50,25,50,26,31,50,26,145,53,26,29,
     52,25,27,50,8,27,146,25,145,25,146
     25, 150,149,145
3030 DATA 42,149,56,27,146,150,27,25,
     145,50,26,29,146,51,27,25,145,52,
     26,29,146,53,27,25,145,53,26,25,
     146,27,150,50,27,150,50,27,25,145,
     50,26,150,149,145
3040 DATA 39,149,150,147,55,25,50,27,
     51,31,52,27,50,25,28,51,8,31,50,
     25,52,26,51,28,26,25,145,53,25,27,
     53,8,30,51, 25,31,50,8,150,149
3050 DATA 49,149,50,25,150,145,51,25,
     30,27,31,51,8,28,50,26,150,8,57,
     27,150,147,55,25,28,26,50,31,8,50,
     28,50,27,50,29,28,26,29,25,30,27,
     31,50,26,8,50,27,150,149,145
3060 DATA 42,149,150,145,57,27,25,150,
     25,150,57,26,25,150,25,150,57,27,
     25,150,25,150,57,26,25,150,51,27,
     146,150,55,8,27,150,50,27,150,55,
     25,27,150,149,145
3070 DATA 28,149,150,147,55,25,50,
     27,51,31,52,27,50,25,28,51,8,31,50,
     25,52,26,51,28,50,26,150,149,145
3080 DATA 1,50,7,4,0,100,7,4,5,7,5,10,6
3090 DATA 40,1,0,40,50,2
3100 DATA 35,1,15,0,0,20,35,1,15,0,0,
     20,35,1,15,0
```

Продолжение статьи «Практикум на ассемблере»

Вторая «критическая точка» находится между 177777 и 0, когда при 177777+1=−1+1=0, а единица, перенесенная из знакового 15-го бита в несуществующий 16-й, просто теряется. Впрочем, процессор все же ПЕРЕНОСИТ этот «сверхнормативный» бит на временное хранение в флаг C. Поэтому данное событие (по сути, также являющееся переполнением, но уже беззнаковой разрядной сетки) получило название «перенос знакового бита».

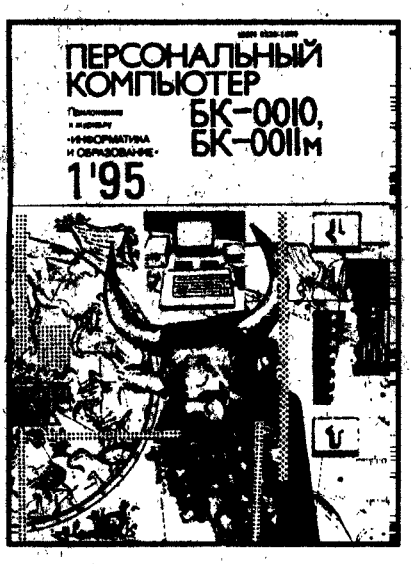
Программисту остается лишь предусмотреть возможность проявления этих событий и подготовить на этот случай соответствующий вариант продолжения своей программы (с помощью команд условного перехода, проверяющих состояние флагов C и V).

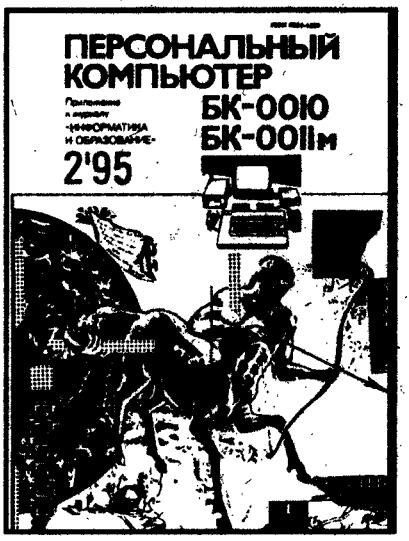
Из последнего числового кольца можно видеть и то, что, например, число 177776 — отрицательное, которому в десятичной системе соответствует −2. Для проверки этого утверждения вспомним правило из школьной алгебры, по которому (−2)+2=0. Проделаем аналогичное действие с восьмеричными числами: 177776+2=200000$_8$=0. Другой пример: 100004+077774=200000$_8$=0 (т. е. 32764$_{10}$+32764$_{10}$=0).

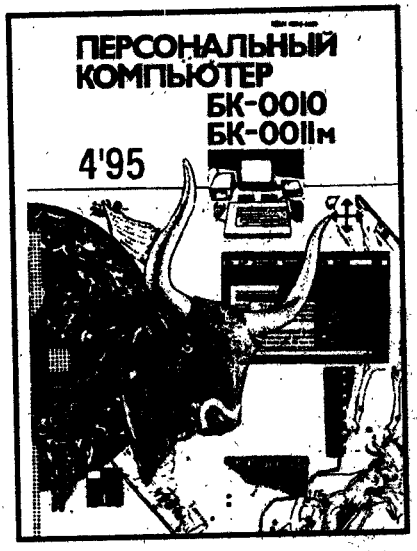
Чтобы вычислить любое восьмеричное отрицательное число по его абсолютному значению, существует команда NEG. Выполните в отладчике две команды: MOV #A,R1 и NEG R1, вставляя вместо A любое положительное восьмеричное число, для которого вы хотите узнать отрицательный эквивалент и наоборот. Из этого правила «выпадает» только число 100000, которое заменяется самим собой, так как для него (наибольшего отрицательного числа) нет соответствующего положительного числа. Взгляните еще раз на числовое кольцо — для 100000 нет пары, так же как и для нуля.

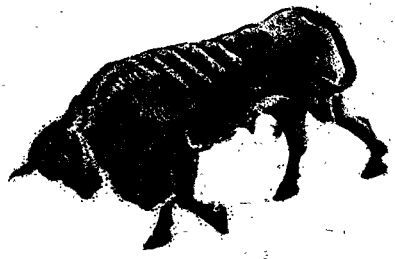
(Окончание следует)

Содержание выпусков журнала за 1995 г.









ПЕРСОНАЛЬНЫЙ КОМПЬЮТЕР БК-0010 — БК-0011М

Внимание!

Новый адрес редакции журнала «Персональный компьютер БК-0010 — БК-0011М»:

метро «Охотный ряд», ул. Тверская, д. 5, Международная Академия Информатизации, комн. 308.

Телефон: (095) 292-53-87.

Адрес для переписки (125315, Москва, а/я 17) и E-Mail (mail@infoobr.msk.su) остаются без изменения.

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
Усенков Д. Ю.

Подписано в печать с оригинал-макета издательства «Компьютика» 30.11.95. Тираж 1500 экз. Формат 70×100 1/16. Бумага офсетная. Усл.печ.л. 6,5. Заказ № 11
Цена 5000 руб. (по подписке). В розничной продаже цена договорная.
Отпечатано в типографии "Принт-Сервис"

С НОВЫМ ГОДОМ!

ПЛАНИРУЕМОЕ СОДЕРЖАНИЕ ВЫПУСКА 1'96

Практикум на ассемблере *(окончание)*

СПРАВОЧНЫЙ ЛИСТОК:
принципиальная схема БК-0011М

Самодельный фотопистолет для БК

Программы для рисования пространственных объектов

Видеоэффект «луча фонарика» на экране

НАЧИНАЮЩЕМУ ПОЛЬЗОВАТЕЛЮ:
драйверы принтеров