ZX SPECTRUM

ZX SPECTRUM

ЭНЦИКЛОПЕДИЯ

НАЧИНАЮЩЕГО ПОЛЬЗОВАТЕЛЯ КОМПЬЮТЕРА

ZX SPECTRUM

ЭНЦИКЛОПЕДИЯ НАЧИНАЮЩЕГО ПОЛЬЗОВАТЕЛЯ КОМПЬЮТЕРА ZX SPECTRUM

НАУЧНО-ПРОИЗВОДСТВЕННЫЙ ЦЕНТР "ТИНОПОС"

НОВОПОЛОЦК 1992г

СОДЕРЖАНИЕ

ВВЕДЕНИЕ - 3
1. ОПИСАНИЕ КОМПЬЮТЕРА - 5
1.1. ЧТО ТАКОЕ ZX-SPECTRUM - 5
1.2. КЛАВИАТУРА - 10
1.3. ОРГАНИЗАЦИЯ ПАМЯТИ КОМПЬЮТЕРА - 13
2. ПОРЯДОК ПОДКЛЮЧЕНИЯ КОМПЬЮТЕРА - 21
2. ПОРЯДОК ПОДКЛЮЧЕНИЯ КОМПЬЮТЕРА - 21
3. ЗАГРУЗКА С МАГНИТОФОНА - 21
4. КОПИРОВАНИЕ ПРОГРАММ - 24
4.1. COPY 86/M - 24
4.2. TF COPY - 25
4.3. COPY-COPY - 25
4.4. ДОПОЛНИТЕЛЬНЫЕ СВЕДЕНИЯ - 26
5. ЯЗЫК ПРОГРАММИРОВАНИЯ БЕЙСИК - 27
6. ОСНОВЫ ПРОГРАММИРОВАНИЯ НА ЯЗЫКЕ БЕЙСИК - 38
6.1. УСЛОВИЯ - 41
6.2. ЦИКЛЫ - 41
6.3. ПОДПРОГРАММЫ - 42
6.4. ОПЕРАТОРЫ READ, DATA И RESTORE - 43
6.5. АРИФМЕТИЧЕСКИЕ ОПЕРАЦИИ - 44
6.6. СТРОКИ СИМВОЛОВ - 46
6.7. ФУНКЦИИ - 47
6.8. МАТЕМАТИЧЕСКИЕ ФУНКЦИИ - 49
6.9. СЛУЧАЙНЫЕ ЧИСЛА - 50
6.10. МАССИВЫ - 51
6.11. ЛОГИЧЕСКИЕ ОПЕРАЦИИ - 52
6.12. НАБОР СИМВОЛОВ - 53
6.13. ГРАФИЧЕСКИЕ СИМВОЛЫ - 55
6.14. ДОПОЛНИТЕЛЬНЫЕ СВЕДЕНИЯ ОБ ОПЕРАТОРАХ PRINT И INPUT - 57
6.15. ЦВЕТА - 59
6.16. ГРАФИКА - 63
6.17. УКАЗАНИЯ - 65
6.18. ПРОГРАММИРОВАНИЕ ЗВУКОВ - 67
6.19. ВНЕШНЯЯ ПАМЯТЬ НА МАГНИТНОЙ ЛЕНТЕ - 69
6.20. УСТРОЙСТВО ПЕЧАТИ - 71
6.21. ВВОД И ВЫВОД - 72
6.22. ПАМЯТЬ - 73
7. ИСПОЛЬЗОВАНИЕ ПРОГРАММ В МАШИННЫХ КОДАХ - 77
8. ЗАЩИТА ПРОГРАММ - 79
9. ОПЕРАЦИИ С ЭКРАНОМ - 81
10. МЕТОДЫ ВВЕДЕНИЯ РУССКОГО ШРИФТА - 84
11. ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ АССЕМБЛЕР - 86
11.1. СУЩНОСТЬ И СТРУКТУРА ЯЗЫКА АССЕМБЛЕРА - 86
11.2. АССЕМБЛЕР GENS3 - 88
12. СИНТЕЗАТОР ЗВУКОВЫХ ЭФФЕКТОВ - 90
13. ВСТРОЕННЫЙ ТАЙМЕР - 92
14. СИСТЕМНЫЕ ПРОГРАММЫ - 93
15. СТРУКТУРА КОМПЬЮТЕРНОЙ ИГРЫ - 94
16. ПОРЯДОК ПОДКЛЮЧЕНИЯ И РЕКОМЕНДАЦИИ ПО РАБОТЕ С ПРИНТЕРОМ - 95
17. ПОРЯДОК ПОДКЛЮЧЕНИЯ И РЕКОМЕНДАЦИИ ПО РАБОТЕ С ДИСКОВОДОМ - 105
Приложение 1. ПОЛНЫЙ НАБОР СИМВОЛОВ - 124
Приложение 2. ШЕСТНАДЦАТИРИЧНАЯ И ДВОИЧНАЯ СИСТЕМЫ СЧИСЛЕНИЯ - 129
Приложение 3. СООБЩЕНИЯ - 129
Приложение 4. ПРИМЕРЫ ПРОГРАММ - 133
Приложение 5. СИСТЕМА КОМАНД МИКРОПРОЦЕССОРА Z80 - 139

ВВЕДЕНИЕ

Персональный компьютер ZX-SPECTRUM - самый популярный домашний компьютер в мире, и, хотя масса фирм уже давно выпускает гораздо более мощные машины, позиций своих SPECTRUM не сдает.

История создания компьютера достаточно интересна. Его создатель Клайв Марлз Синклер (1940 г.р.) свою первую фирму "Синклер радионикс" зарегистрировал в 1961 году, а первое изделие - микроусилитель - выпустил в 1963 году.

Во всех разработках он ставил перед собой две сверхзадачи - минимальные габариты и минимальная цена. Успех Синклера всегда основывался на том, что он со своим товаром был первым, причем часто ориентировался на рынок, который еще не существовал.

В 1979 г. фирма "Коммодор" выпустила свой первый бытовой компьютер "PET" ценой 700 фунтов. Газета "Файнэншнл Таймс" тогда предсказывала, что цены на персональные компьютеры опустятся ниже 100 ф.ст. не ранее, чем через 5 лет, а Синклер уже через полгода выпустил ZX-80 ценой 97 фунтов.

Резкому снижению цены содействовала идея использования телевизора в качестве дисплея, а бытового магнитофона в качестве внешней памяти.

В первые 8 месяцев было продано 20 тыс. компьютеров, и в марте 1981 г. была выпущена новая модель ZX-81 ценой 69 фунтов. В эти дни американская фирма "Таймекс" купила право на производство всех разработок Синклера, как сделанных, так и тех, которые будут выпущены впредь. Фирма "Митцуи" купила исключительное право на распространение ZX-81 в Японии. Решительным рывком вперед стал договор с британской книготорговой сетью о реализации компьютеров по их торговым каналам. За один год товарооборот фирмы вырос с 4,6 до 30 млн.ф.ст., а Синклер уже готовил новую модель- SPECTRUM (март 1982 г.). Были разработаны две версии - 16K и 48K. Эта машина сильно отличалась от своих предшественников, и ее популярность превзошла все ожидания. ZX-SPECTRUM продавались по 15 тыс. штук в неделю, за 2,5 года фирма SINCLAIR RESEARCH OF CAMBRIDGE изготовила и продала около 500 тыс. штук компьютеров.

Задумывался этот компьютер как учебный для изучения программирования, но фирмы, выпускающие программное обеспечение, быстро поняли, что программирование на уровне команд процессора позволяет получить неплохую динамичную графику, и для этого компьютера стали выпускаться увлекательные видеоигры. Получилась своего рода положительная обратная связь. Чем больше СПЕКТРУМов покупалось населением, тем активнее выпускались для него программы, а чем больше на рынке высококачественных программ для компьютера, тем активнее он покупается. Такой же процесс охватил и фирмы "третьего рынка", выпускающие периферийные устройства и аксессуары для компьютеров. К 1984 году, когда фирмы "Атари", "Коммодор" и "Амстрад" выпустили компьютеры, превосходящие СПЕКТРУМ 48, рынок уже был смещен в пользу СПЕКТРУМа, что продолжает чувствоваться и по сей день, а сам СПЕКТРУМ уже выпускался более чем в 30 странах мира.

В 1984 году Синклер выпустил модель СПЕКТРУМа, отличающуюся усовершенствованной клавиатурой, а в конце 1985 года СПЕКТРУМ-128 ("ДЕРБИ"), имеющий 128K оперативной памяти и 32K ПЗУ. Кроме того, новая модель имела звуковой процессор.

В дальнейшем, в 1986 году Синклер под давлением финансовых трудностей, вызванных с одной стороны просчетами при разработке новых моделей машин, а с другой - отсутствием у него серьезной деловой хватки, продал все права на производство СПЕКТРУМ - совместимых моделей французкой фирме "Амстрад".

Продав все права на производство и реализацию своих изделий, Синклер оставил за собой исследовательскую лабораторию в Кембридже.

Последующие модели СПЕКТРУМ+2 (1986 г.) со встроенным магнитофоном и СПЕКТРУМ+3 (1987 г.) со встроенным дисководом выпускались уже фирмой "Амстрад". Основным их преимуществом является полноценная клавиатура, в то время как встроенные магнитофон и дисковод воспринимаются скорее как "нагрузка", непропорционально увеличивающая цену. Особенно если принять во внимание нестандартный диаметр дискет 3,0 дюйма, малую их емкость (180K) и практическую сложность переноса имеющихся кассетных версий программ на диск, граничащую с нецелесообразностью.

В заключение вступления несколько слов о перспективной расширенной модели, планировавшейся к выпуску в 1987 г. В основу СУПЕР СПЕКТРУМа ("ЛОКИ") был положен процессор Z-80H, который может работать с частотой 7 МГц. При такой скорости удается организовать и обслужить два банка памяти по 64К и экран емкостью более 51К. Он имел разрешающую способность 192*256 с возможностью одновременного воспроизведения 64-х цветов (для каждой точки). Эта машина была программно совместима со СПЕКТРУМом, стоила менее 200 фунтов и была бы серьезным конкурентом для "АМИГИ". Но фирма "Амстрад", пользуясь своими правами, опасаясь конкуренции для своих машин, отказала в разрешении на его производство.

Компьютер ZX-SPECTRUM - конструктивно достаточно несложное устройство, и при наличии необходимых комплектующих изделий и деталей для тех, кто немного знаком с электроникой не представляет трудностей его монтаж и отладка в домашних условиях. Простота его воспроизведения и эксплуатации, большие вычислительные возможности СПЕКТРУМа, большой объем наработанного в мире программного обеспечения, особенно игровых программ, по количеству которых ему нет конкурентов, и привели к появлению на внутреннем рынке ряда модификаций компьютера СПЕКТРУМ-48, отличающихся от фирменных машин только конструктивным исполнением и непринципиальными изменениями схемотехнических решений.

Настоящая книга имеет свой целью помочь пользователю изучить этот компьютер, научиться программировать на БЕЙСИКе, использовать все его вычислительные и интеллектуальные возможности. Отдельно рассмотрен интерфейс накопителя на гибком магнитном диске.

При чтении настоящего руководства старайтесь использовать компьютер. Если у вас возникнет вопрос, что будет, если выполнить те или иные действия с компьютером, то ответ вы получите на экране телевизора, введя эти команды в компьютер.

Всякий раз, когда в этой книге вы встретите предложение что-нибудь ввести в компьютер и выполнить на нем те или иные действия, подумайте, чем можно заменить эти действия и попробуйте их проделать. Чем больше собственных программ вы напишите, тем лучше будете понимать, как работает компьютер.

Не бойтесь дать компьютеру не ту команду, нажать не ту клавишу - этим его не испугаешь.

Если все же в результате ваших манипуляций компьютер "зависнет" и перестанет реагировать на нажатие клавиш - нажмите кнопку "СБРОС" - и все придет в нормальное состояние.

1. ОПИСАНИЕ КОМПЬЮТЕРА.

1.1. ЧТО ТАКОЕ ZX-SPECTRUM

Существует масса различных вариантов компьютеров типа ZX-SPECTRUM, изготавляемых на территории бывшего СССР. Они различны как по схемным решениям, так и по конструктивному исполнению.

Кроме "фирменных", т.е. заводского изготовления компьютеров, существует масса самоделок. Однако все они удовлетворяют требованиям программной совместимости с фирменным ZX-SPECTRUM, иначе бы не имело смысла их делать. Из наиболее известных вариантов схемы авторам чаще всего приходилось встречаться с компьютерами типа "Балтика", Ленинградский вариант т.Зонова, так называемый "Пентагон", "Дерби" и др. Самый простой и дешевый компьютер из названных - это Ленинградский вариант. Однако он хорош, если его использовать только как игровой компьютер и возникают существенные трудности при подключении принтера, дисковода, программатора и т.п. Компьютер, выполненный по схеме "Балтика", обладает на первый взгляд некоторой избыточностью, однако при желании к нему просто подключить принтер, контроллер дисковода, программатор, а также возможно при небольшой доработке перейти на другую операционную систему, отличную от системы ZX-SPECTRUM, но очень популярную для 8-ми разрядных машин - CP/M. Вариант "Дерби" обладает увеличенным, по сравнению со стандартным, объемом памяти (128 Кбайт, 1 Кбайт=1024 Байта, в дальнейшем будем говорить просто 128 К), программируемым джойстиком и интерфейсом для принтера типа "CENTRONICS". Однако вариант "Дерби" хорош только для разработки своих больших программ и довольно сложен (по сравнению с Ленинградским вариантом). Что касается "Пентагона", то эта машина имеет в своем составе как параллельный интерфейс так и встроенный контроллер накопителя на гибких магнитных дисках (НГМД, или проще - контроллер дисковода).

Обычно сам по себе ZX-SPECTRUM (будем под этим названием подразумевать и все самоделки, совместимые программно с ним) представляет собой небольшую "коробочку". Он обязательно имеет несколько разъемов сбоку или сзади для подключения магнитофона, джойстика, телевизора, принтера, дисковода и другой периферии. Самые дешевые варианты имеют два разъема (это минимум) - для подключения телевизора (TV) и магнитофона. Может быть два разъема для подключения TV - один это RGB-выход, а второй для подключения на антенный вход TV. Некоторые самоделки включают в свой состав и магнитофон и даже дисковод, однако они довольно дороги, хоть и очень удобны для пользователя.

Рассмотрим теперь характеристики Вашего компьютера:

ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ КОМПЬЮТЕРА

Обьем ОЗУ .....48 кБайт (49152 Байт)
Обьем ПЗУ .....16 кБайт (16384 Байт)
Скорость записи на магнитофон .....1500 Бод (около 180 Байт/сек)
Емкость кассеты магнитофона .....1 мБайт (МК-90)
Скорость вычисления .....950 000 Оп/сек.
Формат экрана .....256*192 точки
Количество цветов 8
Количество градаций яркости 2
Режим мерцания .....есть (8*8 точек)
Цветов бордера (рамки) 8
Таймер .....кварцованный,
дискретность 1/50 С

Компьютер построен на высокоскоростном 8-разрядном процессоре Z80 с тактовой частотой обычно 3,5-4 мГц,что позволило получить производительность не менее 950 000 операций в секунду. Это приблизительно в 4 раза лучше, чем у компьютеров "Радио-86" и "Микроша", близко к производительности компьютера ДВК-3 и даже немного выше, чем у ДВК-2.

Система команд процессора Z80 содержит все команды процессора, применяемого в машинах "Радио-86РК" и "Микроша", и плюс к ним еще в два раза

больше своих.

ZX-SPECTRUM позволяет получать на экране обычного цветного телевизора сложные графические изображения, состоящие из 256*192 точек при 8 цветах, а также текстовые изображения в формате 32*24 символа.

При использовании черно-белого телевизора 8 цветов представляются на нем как 8 градаций яркости.

При включении компьютера немедленно активируется тестовая система, проверяющая исправность компьютера и пытающаяся при обнаружении неисправности обеспечить нормальное его функционирование. Она настолько мощна, что позволяет нормально работать при 3/4 не работающем ОЗУ компьютера и других серьезных неисправностях, что значительно повышает надежность системы.

Через 1,5 секунды после включения тестовая система завершит свою работу и компьютер будет полностью готов к действию - операционная система будет активирована и готова к исполнению команд (в том числе команд встроенного БЕЙСИКа). При этом не требуется загрузка каких-либо программ откуда-либо извне. Пуск машины в работу не сложнее, чем включение обычного калькулятора.

Упрощенная блок-схема компьютера представлена на рис. 1

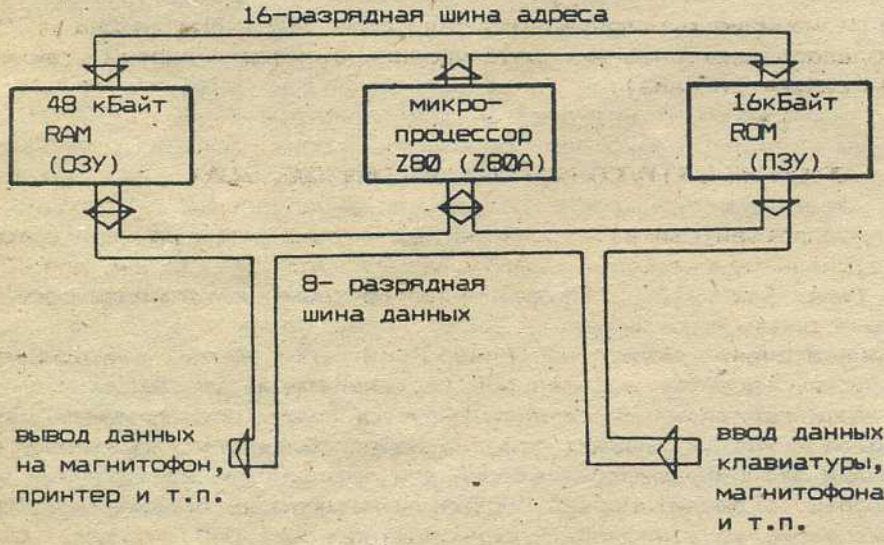


Рис.1

Микропроцессор Z80 (Z80A) - это кремниевый кристалл (ЧИП). Он является самым важным из всех элементов компьютера и предназначен для выполнения программы. Программа для Z80 представляет собой набор кодовых инструкций и согласованных с ними данных.

Назначение выводов микропроцессора Z80 (Z80A) приведено на рис. 2.

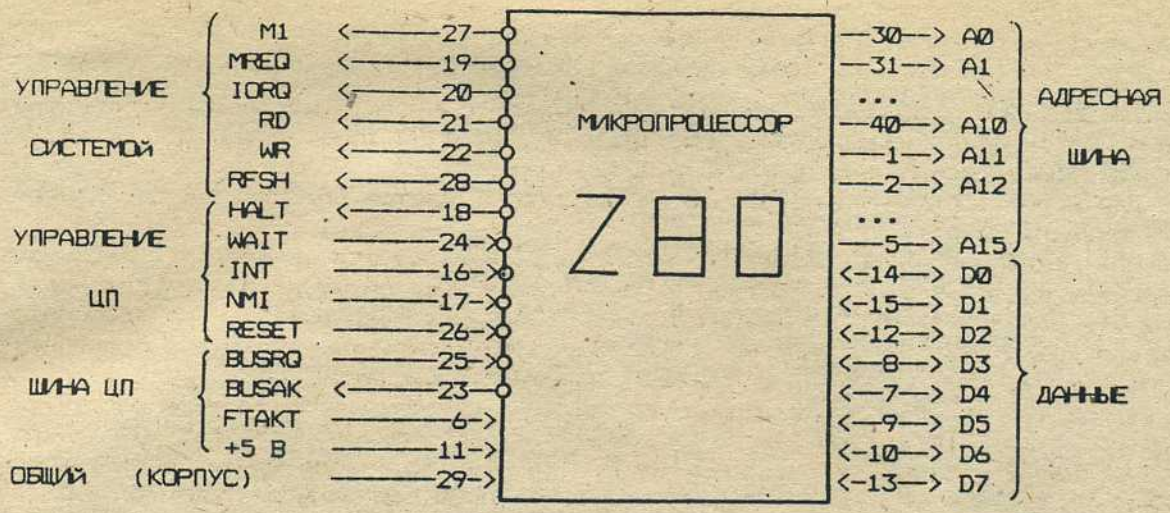


Рис. 2. Назначение выводов микропроцессора Z80 (Z80A). Значок "○" – в обозначении вывода означает, что активный уровень сигнала – низкий (так же как и черта над наименованием сигнала).

ОПИСАНИЕ СИГНАЛОВ МИКРОПРОЦЕССОРА Z80, Z80A

A0–A15 – (адресная шина). Выходы с тремя устойчивыми состояниями. Активный уровень сигналов – высокий. Адресует ОЗУ или УВВ (до 64К для ОЗУ).

D0–D7 – (шина данных). Входы-выходы с тремя устойчивыми состояниями. Активный уровень – высокий.

M1 – (машинный цикл). Выходной активный сигнал – низкий – указывает, что в текущем цикле осуществляется выборка КОП (кода операции).

MREQ – (запрос памяти). Выход с тремя устойчивыми состояниями. Активный уровень – низкий. Сигнал указывает, что на адресной шине установлен адрес для операции чтения или записи в память.

IORQ – (запрос ввода-вывода). Выход с тремя устойчивыми состояниями. Активный уровень сигнала – низкий. Сигнал указывает, что младший байт шины адреса содержит адрес УВВ. Кроме того, этот сигнал выдается для подтверждения прерывания, тем самым указывая, что вектор прерывания может быть помещен на шину данных.

RD – (чтение из памяти). Выход с тремя устойчивыми состояниями. Активный уровень сигнала – низкий. Сигнал указывает, что процессор (ЦП) готов к чтению данных из памяти, из устройства ВВ. Адресованное УВВ или память использует этот сигнал для стробирования при подаче данных на шины данных ЦП.

WR – (запись в память). Выход с тремя устойчивыми состояниями. Активный уровень сигнала – низкий. Сигнал указывает, что на шине данных содержатся данные, предназначенные для записи в память или вывода на устройство вывода.

RFSH – (регенерация). Выход, активный уровень – низкий. Сигнал указывает, что младшие 7 разрядов шины адреса содержат адрес регенерации для ОЗУ и текущий сигнал MREQ должен использоваться для регенерации динамической памяти.

HALT – (останов). Выход, активный уровень – низкий. Сигнал указывает, что ЦП выполнил команду HALT и ожидает появления либо немаскируемого, либо маскируемого прерывания, после которого он продолжит работу. Перед выполнением HALT ЦП заносит в ОЗУ информацию, которая нужна для восстановления.

WAIT – (ожидание). Вход, активный уровень – низкий. Сигнал указывает микропроцессору, что адресуемые память или устройство не готовы к передаче данных. ЦП ждет пока активен этот сигнал.

INT – (запрос на маскируемое прерывание). Вход, активный уровень – низкий. Запрос будет воспринят ЦП в конце выполнения текущей команды, если триггер разрешения прерывания IFF, управляемый внутренними программными

средствами, установлен в определенное состояние.

NMI - (запрос на немаскируемое прерывание). Вход, активный уровень - низкий. Это прерывание имеет более высокий приоритет, чем INT, распознается в конце текущей команды. Сигнал автоматически переводит ЦП к выполнению программы с адреса 0066(HEX).

RESET - (сброс). Вход, активный уровень - низкий. При поступлении сигнала выполняются следующие действия:

а. сброс триггера разрешения прерывания IFF
б. очистка счетчика команд регистров I и R
в. шины адресная и данных - в состоянии с высоким сопротивлением (3-е состояние)
г. для всех управляющих выходных сигналов устанавливаются неактивные уровни.

BUSRQ - (запрос шины). Вход, активный уровень - низкий. Сигнал имеет более высокий приоритет, чем NMI, и всегда распознается в конце текущего машинного цикла. Он используется для организации прямого доступа к памяти (ПДП). Переводит в состояние высокого сопротивления все шины и трехстабильные выходы сигналов управления, после чего этими шинами могут управлять другие внешние устройства.

BUSAK - (подтверждение перевода шин в состояние высокого сопротивления). Выход, активный уровень - низкий. Сигнал подается на запрашивающее устройство.

РЕГИСТРЫ ПРОЦЕССОРА Z80 (Z80A)

Набор основн.	альтерн.	название и назначение регистра
A	A'	аккумулятор (8 Бит)
B	B'	РОН B (8 Бит) - пара BC (16 Бит)
C	C'	РОН C (8 Бит)
D	D'	РОН D (8 Бит) - пара DE (16 Бит)
E	E'	РОН E (8 Бит)
H	H'	РОН H (8 Бит) - пара HL (16 Бит)
L	L'	РОН L (8 Бит)
F	F'	регистр флагов (8 Бит)
IX		регистр сегмента (16 Бит)
IY		регистр сегмента Y (16 Бит)
IR		регистр прерывания и регенерации (16 Бит)
SP		указатель стека (16 Бит)
PC		указатель команды (16 Бит)

Особенностью процессора Z80 является наличие у него "альтернативного" набора регистров, обозначающихся так же, как и регистры основного набора, но со штрихом.

Работа с альтернативными регистрами не отличается от работы с использованием основных регистров, и для программиста их подмена может быть незаметна.

Переход на альтернативный набор регистров удобен, например, при переходе с основной на выполнение фоновой программы.

Всего имеется 21 программно-доступный регистр и еще несколько недоступных, которые процессор использует для хранения промежуточных результатов. Большинство регистров процессора или имеет размер 16 Бит, или обьединены в пары по 16 Бит. Какова же разрядность процессора Z80, 8 или 16 Бит? Ответить на этот вопрос очень сложно. Во всяком случае, у процессора имеется и набор команд 8-Битовой арифметики, и набор команд 16-Битовой арифметики. Наличие в составе процессора двойного АЛУ (арифметико-логического устройства) и 16-разрядной схемы сумматора-вычислителя, а также 16-разрядного инкрементора-декрементора позволяет ему оперировать 16-Битовыми данными столь же легко, как и 8-Битовыми. Таким образом, можно сказать, что "внутри" процессор 16-разрядный, но "снаружи" выглядит как 8-разрядный (ведь внешняя шина данных имеет 8 разрядов).

Именно эта особенность процессора Z80 и обусловила наличие у него

огромного количества команд (около 680 разновидностей) и регистров (21). Но этого создателям процессора Z80 показалось мало. В дополнение к 16 и 8-Битовым операциям они ввели еще и поразрядные (1-Битовые) операции, что на порядок упростило и, приблизительно, в 2 раза ускорило работу с внешними устройствами (например, с дисководом). Это позволяет в целом ряде случаев обойтись без соответствующих схем контроллеров внешних устройств.

Все эти возможности имеет только процессор Z80, и реализует их только соответствующий ассемблер. Одним из наиболее мощных ассемблеров является программа макроассемблер GENS -4/5 из пакета DEVPAC -4 (кроме макроассемблера GENS -4/5 в пакет входит отладчик-дизассемблер MONS -4). Приставка макро перед названием ассемблера означает, что кроме нормальных команд микропроцессора Z80 вы можете сами создавать и использовать макрокоманды. Например, если вам захочется иметь команду умножения MUL (ее нет в системе команд Z80), ее можно заменить программным эквивалентом и использовать так же свободно, как если бы она всегда была в системе команд.

Загружают пакет командой LOAD ""CODE ..., где ... - адрес, начиная с которого будет находиться программа. Затем программы запускаются командой RANDOMIZE USR ..., где ... - тот же адрес, что и в команде LOAD .

16K ROM - ПОСТОЯННАЯ ПАМЯТЬ (ПЗУ)

Представляет собой один ЧИП (от английского chip - "обломок" - на самом деле по русски - микросхема) обьемом 128 кБит (или два ЧИПа по 64 кБит) - или это 16 кБайт памяти, содержащие программу монитора в машинных кодах для микропроцессора Z80.

Программа "монитор 16K" разбита на 3 части:

7 кБайт - операционная система
8 кБайт - интерпретатор БЕЙСИКа
1 кБайт - генератор знаков

48 кБайт RAM - ОПЕРАТИВНАЯ ПАМЯТЬ (ОЗУ)

Вся память компьютера разбита на байты, каждый из которых может содержать число от 0 до 255 (в двоичном коде от 00.000.000 до 11.111.111). Каждый байт может быть записан в память по определенному адресу от 16384 до 65535. Сам адрес может быть записан в память как два байта. Для организации работы операционной системы в оперативной памяти выделена системная область. Остальная память доступна для программ. Распределение памяти приведено на рис. 3

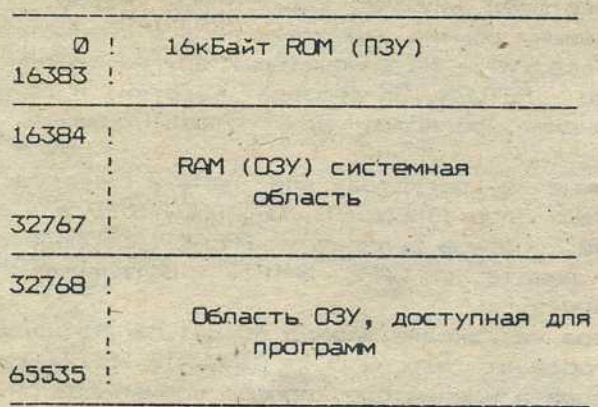


Рис. 3

В большинстве моделей СПЕКТРУМов ОЗУ построено на 8 микросхемах типа 565 РУ5Б(В,Г), имеющих организацию 64К*1 или на 16 микросхемах типа 565РУ5Д1(Д2) - с организацией 32К*1. Память динамического типа.

КЛАВИАТУРА

Клавиатура СПЕКТРУМа имеет обычно 40 клавиш. Каждая клавиша клавиатуры имеет многофункциональное назначение (до 6-7 функций в зависимости от того, в сочетании с какими клавишами она нажимается) и позволяет вводить как отдельные символы, так и целые слова. Более подробно работа с клавиатурой будет рассмотрена в дальнейшем.

ЭКРАН ТЕЛЕВИЗОРА

Экран телевизора содержит 24 строки по 32 знакоместа в каждой и делится на две части. Верхняя часть в 22 строки отображает листинг или вывод из программы.

Нижняя часть используется для ввода команд, строк программы и входных данных, а также для отображения сообщений.

Доступны цвета: черный, голубой, красный, синий, пурпурный (фиолетовый), зеленый, желтый и белый. Края экрана (бордер) могут быть установлены пользователем в необходимый ему цвет. Более подробно описание работы с экраном приведено в разделе 9.

МАГНИТОФОН

В качестве внешнего запоминающего устройства чаще всего применяют магнитофон.

Вопросы работы компьютера СПЕКТРУМ с магнитофоном (подключение магнитофона, загрузка компьютера, копирование записей и т. д.) подробно рассмотрены в разделах 2,3,4.

1.2. КЛАВИАТУРА

Клавиатура СПЕКТРУМа имеет всего 40 клавиш. В принципе это очень мало, и здесь наглядно проявился результат неуклонной борьбы К.Синклера за миниатюризацию и дешевизну своих изделий. Тем не менее, те, кто к ней привык, относятся к ней с уважением и даже любовью.

В СПЕКТРУМе клавиши содержат не только одиночные символы (буквы, цифры и т. д.), но также составные символы (ключевые слова, названия функций и т.д.).

Для того, чтобы реализовать все функции и команды, некоторые клавиши клавиатуры имеют по 5 и более значений, получаемых либо путем выбора соответствующего регистра (т.е. получаемых путем нажатия клавиши CAPS SHIFT или SYMBOL SHIFT одновременно с какой-либо необходимой клавишей), либо путем перевода компьютера в один из возможных режимов работы.

Состояние индицируется курсором - мерцающей буквой, которая показывает, где будет появляться на экране следующий набираемый символ.

Рассмотрим назначение некоторых наиболее характерных клавиш.

ENTER - нажатие этой клавиши обычно завершает ввод в компьютер чего-либо (команды, данных, программной строки). На компьютерах других систем эта клавиша может иметь другое наименование - RETURN, ВК (возврат каретки) и др.

CAPS SHIFT - переключение регистра на печать прописными буквами (аналогично пишущей машинке).

SYMBOL SHIFT - переключение регистра для печати символов (+, -, /, и т.д.), а также некоторых ключевых слов (THEN, TO, OR, AND, STEP и д.р.).

SPACE - пробел. Эта же клавиша выполняет BREAK (прерывание исполнения программы), если ее нажать вместе с CAPS SHIFT. Остальные клавиши имеют буквенное или цифровое обозначение.

При подключении компьютера на экране должно появиться исходное сообщение "1982 (1988,1989 - в зависимости от версии ОС) SINCLAIR RESEARCH LTD". Это означает, что первичные проверки в компьютере прошли нормально, он исправен и готов к работе.

Нажмите клавишу ENTER, на экране в нижнем левом углу появится черный квадрат с буквенным мерцающим обозначением на нем. Это курсор - место на экране, в котором будет помещаться очередной набираемый символ, а также указывает, в каком режиме находится клавиатура. Этих режимов - четыре и им соответствуют пять разных курсоров.

1.2.1. КОМАНДНЫЙ РЕЖИМ.

Курсор - "К". Он означает, что сейчас при нажатии клавиши будет введена

команда, которая закреплена за этой клавишей, или цифра. Компьютер сам понимает, что строка может начинаться только либо с номера строки, либо с команды, поэтому дает курсор "K".

Режим [K] автоматически заменяет режим [L], когда компьютер ожидает команду или программную строку (отличающуюся от вводимых данных) и с этой позиции в строке курсором указывается, что ожидается ввод ключевого слова или строки. Это относится к началу строки или знакоместу сразу же после ":" (за исключением двоеточия в тексте). Если не изменен режим, то нажатие следующей клавиши будет интерпретироваться как ключевое слово, написанное на клавише, либо как цифра.

После того, как команда набрана, и дальше должны пойти ее параметры, режим автоматически переключится на литерный (курсор "L"). Если вы введете ":" (двоеточие), то курсор опять переключится на "K", т. к. двоеточие является разделителем между несколькими командами, если они записываются в одной строке.

1.2.2. ЛИТЕРНЫЙ РЕЖИМ.

Курсоры - L, C. Курсор "L" соответствует печати строчными буквами, а курсор "C"- прописными. Переключиться на курсор "C" можно командой CAPS LOCK. Это выполняется одновременно нажатием CAPS SHIFT и цифры "2". В этом же режиме набираются символы и служебные слова, связанные с клавишей "SYMBOL SHIFT".

И в [K] и в [L] режимах одновременное нажатие клавиши SYMBOL SHIFT и какой-либо клавиши воспринимается как вспомогательный символ, изображенный на клавише, а в случае CAPS SHIFT с цифровой клавишей - как управляющая функция, написанная на цифровой клавише.

Нажатие клавиши CAPS SHIFT с другими клавишами в режиме курсора "K" не влияет на ключевые слова, а в режиме курсора "L" вызывает появление заглавных букв.

1.2.3. РАСШИРЕННЫЙ КОМАНДНЫЙ РЕЖИМ.

Курсор "E". В этом режиме набираются команды, записанные над клавишами или под клавишами. Переход в режим "E" выполняется одновременным нажатием CAPS SHIFT и SYMBOL SHIFT. Он действует только на одно нажатие. Если в режиме "E" нажать какую-либо клавишу, то появится слово, записанное над клавишей, а если нажать эту клавишу совместно с CAPS SHIFT - то слово, записанное под клавишей. Здесь имеется исключение для цифровых клавиш (верхний ряд). Чтобы набрать слово, записанное под клавишей, надо в режиме "E" нажать не CAPS SHIFT, а SYMBOL SHIFT совместно с клавишей. Такая уникальная способность СПЕКТРУМа вводить операторы и функции не по буквам, а одним нажатием клавиши, называется токенизированной формой записи ключевых слов. Сначала это выглядит несколько сложно,зато, но опыт приходит быстро, а с ним и удобство работы.

1.2.4. ГРАФИЧЕСКИЙ РЕЖИМ.

Курсор - "G". В этом режиме набираются символы блочной графики, расположенные на цифровых клавишах, а также символы графики пользователя.

Переход в графический режим выполняется одновременным нажатием клавиш CAPS SHIFT и цифры "9". Если вы в этом режиме нажмете, например, клавишу "5", то на экране появится квадрат, левая половина которого черная, а правая - белая. Нажатие клавиши "5" совместно с CAPS SHIFT изобразит тот же квадрат в инвертированном виде, т. е. левая половина будет белая, а правая - черная.

Графический режим сохраняется до тех пор, пока не будет нажата клавиша CAPS SHIFT совместно с "9".

Цифровые клавиши дают также графические символы, за исключением GRAFICS или DELETE; каждая из буквенных клавиш, кроме V, W, X, Y и Z, могут вызывать появление определенных пользователем графических символов.

П Р И М Е Р.

Если набрать что-то произвольно на экране, то скорее всего это будет неправильная строка и при нажатии ENTER на экране появится знак "?". Чтобы начать работу, надо эту строку стереть. Это выполняется командой DELETE (одновременное нажатие CAPS SHIFT и цифры 0). Когда все символы будут уничтожены, курсор встанет на исходную позицию и примет вид "K", т. е. компьютер ждет от вас команду.

Нажмите клавишу "R". На экране появится команда RUN. Курсор изменится на "L". Еще раз нажмите ту же клавишу - появится буква r.

Нажатие этой клавиши совместно с CAPS SHIFT даст прописную букву R, а совместно с SYMBOL SHIFT - знак "<".

Перейдите в расширенный командный режим - CAPS SHIFT + SYMBOL SHIFT одновременно. Появится курсор "E". Теперь нажатие той же клавиши даст оператор INT, снова войдите в режим "E" и нажмите клавишу совместно с CAPS SHIFT. Получите команду VERIFY. Если какую-либо клавишу задержать в нажатом положении больше 2-3 секунд, то буква будет повторена многократно со скоростью около 6 символов в секунду. Эта удобная функция называется "автоповтор" (кстати, он действует и при стирании).

Ввод с клавиатуры осуществляется в нижнюю половину экрана, каждый символ (или группа символов для ключевых слов) появляется перед курсором. Сам курсор может перемещаться по экрану клавишами: влево - CAPS SHIFT и "5" вправо - CAPS SHIFT и "8" и т.д.

Символ перед курсором может быть удален командой DELETE (CAPS SHIFT и 0) Примечание: целая строка может быть удалена вводом EDIT (CAPS SHIFT и"1") и последующим нажатием клавиши ENTER.

При нажатии ENTER строка, набранная в нижней части экрана, либо выполняется как команда, либо вводится как очередная строка в программу, либо используется как список данных для INPUT - ввода. Если же она содержит синтаксические ошибки, то ошибочное место указывается мерцающим знаком "?".

Когда вводятся строки программы, то листинг отображается в верхней половине экрана. Последняя введенная строка называется текущей и указывается символом ">", его можно перемещать ниже или выше, используя клавиши CAPS SHIFT и "6" или CAPS SHIFT и "7" соответственно. Если введено EDIT (CAPS SHIFT и "1"), то текущая строка переносится в нижнюю часть экрана,где она может редактироваться.

При выполнении команды и программы ввод осуществляется в верхнюю часть экрана и сохраняется до ввода строки программы, либо нажатия клавиши ENTER при наличии пустой строки, либо нажатия клавиши перемещения курсора.

Те, кто работает с компьютерами "СПЕКТРУМ+", "+128", "+2", "+3" могут упростить свою работу благодаря наличию дополнительных клавиш, которые выполняют ряд функций одним нажатием.

DELETE - стирание символа
GRAPH - переход в графический режим
EXTENDED MODE - переход в расширенный командный режим
EDIT - выполнение редактирования строки
CAPS LOCK - переключение регистра на печать прописными буквами
INV. VIDEO - включение инверсного режима (печать белым по черному)
TRUE VIDEO - возвращение из инверсного режима

Кроме того, там имеются дополнительные клавиши для набора знаков препинания: точка, запятая, точка с запятой, кавычки, а также четыре клавиши для управления перемещением курсора при редактировании.

В таблице 1 приведено расписание клавиатуры компьютера СПЕКТРУМ, т.е. значение клавиш в зависимости от режима работы компьютера.

ТАБЛИЦА 1

КЛАВИША	КУРСОР						
	K	L			E		
			CAPS SHIFT	SYMBOL SHIFT		CAPS SHIFT	
A	NEW	a	A	STOP	READ	~	
B	BORDER	b	B	*	BIN	BRIGHT	
C	CONTINUE	c	C	?	LPRINT	PAPER	
D	DIM	d	D	STEP	DATA	\	
E	REM	e	E	>=	TAN	ATN	
F	FOR	f	F	TO	SGN	{	
G	GO TO	g	G	THEN	ABS	}	
H	GO SUB	h	H	^	SQR	CIRCLE	

КЛАВИША	КУРСОР						
	K	L			E		
		CAPS SHIFT		SYMBOL SHIFT	CAPS SHIFT		
I	INPUT	i	I	AT	CODE	IN	
J	LOAD	j	J	-	VAL	VAL$	
K	LIST	k	K	+	LEN	SCREEN$	
L	LET	l	L	=	USR	ATTR	
M	PAUSE	m	M	.	PI	INVERSE	
N	NEXT	n	N	,	INKEY$	OVER	
O	POKE	o	O	;	PEEK	OUT	
P	PRINT	p	P	"	TAB	©	
Q	PLOT	q	Q	<=	SIN	ASN	
R	RUN	r	R	<	INT	VERIFY	
S	SAVE	s	S	NOT	RESTORE	;	
T	RANDOMIZE	t	T	>	RND	MERGE	
U	IF	u	U	OR	CHR$	[	
V	CLS	v	V	/	LLIST	FLASH	
W	DRAW	w	W	<>	COS	ACS	
X	CLEAR	x	X		EXP	INK	
Y	RETURN	y	Y	AND	STR$	]	
Z	COPY	z	Z	:	IN	BEEP	
0	0	0	DELETE	-		FORMAT	*
1	1	1	EDIT	!		DEF FN	*
2	2	2	CAPS LOSK	@		FN	*
3	3	3	TRUE VIDEO	#		LINE	*
4	4	4	INV. VIDEO	$		OPEN $	*
5	5	5	курсор влево	%		CLOSE $	*
6	6	6	курсор вниз	&		MOVE	*
7	7	7	курсор вверх	'		ERASE	*
8	8	8	курсор вправо	(		POINT	*
9	9	9	GRAPHICS	)		CAT	*

Примечание: * Символы верхнего ряда в расширенном командном режиме [E] нажимаются не с [CAPS SHIFT], а с [SYMBOL SHIFT].

1.3. ОРГАНИЗАЦИЯ ПАМЯТИ КОМПЬЮТЕРА

Для обеспечения работы операционной системы компьютера в его памяти выделено несколько областей. Эти области отделены друг от друга некоторыми границами, которые могут быть фиксированы (постоянно), а могут изменяться в зависимости от конкретных требований.

Фиксированные границы обозначены числом, а переменные - именем системной переменной, в которой они хранятся.

КАРТА ПАМЯТИ КОМПЬЮТЕРА

Адреса с 0 до 16383 (16К) - область ПЗУ

В ПЗУ хранятся:

7кБайт - операционная система (ОС)

8кБайт - интерпретатор БЕЙСИКа

1кБайт - генератор знаков

Операционная система (ОС) - это программная надстройка над компьютером. ОС СПЕКТРУМа относится к типу "твердотельных" ОС, т. к. находится в ПЗУ.

Основная функция ОС - операции над файлами, например, загрузка их в память, запуск, выгрузка, сравнение, компоновка.

Всего имеется 5 типов файлов:

BYTE - программа в машинных кодах
PROGRAMM - программа на БЕЙСИКе
SCREEN$ - копия экрана
DATA - значения числовых массивов
DATA$ - значение символьных переменных или массивов

Кроме того, некоторые программы создают свои типы файлов. Однако эти типы файлов не являются стандартными и не обрабатываются ОС.

АДРЕСА С 16384 ДО 22527 (6К), (4000-57FF- в шестнадцатиричном виде) - экранная область - обеспечивает высокое разрешение экрана телевизионного приемника (ТП).

Экранная область доступна для операторов PEEK и POKE.

Каждая позиция экрана представима матрицей 8*8 точек (1 байт на каждый ряд из 8 точек). Однако эти 8 байтов хранятся в памяти не вместе. Полный экран представляет собой 24 строки по 32 символа в каждой. Каждая строка символов прописывается 8-ю строками развертки экрана TV. Итого для воспроизведения (записи) изображения одного экрана выполняется 24*8=192 сканирования и в памяти рядом хранятся байты одноименных рядов матриц соседних позиций экрана.

Высокое разрешение экрана TV достигается за счет того, что в отведенных под экранную область 6К памяти можно закодировать изображение любой из 49152 точек экрана TV (матрица 192*256 точек), что подтверждается расчетом:

1 символ кодируется 8*8=64 точками
количество символов на 1 экран 32*24=768
количество адресуемых точек 768*64=49152

Экран TV разбит на три части по 2К. Строкам с номерами 0-7 соответствуют адреса 16384-18431 (4000-47FF - шестнадцатиричные). Строкам с 8 по 15 соответствуют адреса 18432-20479 (4800-48FF - шестнадцатиричные). Строкам с 16 по 23 соответствуют адреса 20480-22527 (4900-57FF - шестнадцатиричные).

Каждые из этих третей по 2К состоят из 8 массивов по 1/4 кБайта. Каждый массив из 1/4 кБайта (256 байтов) хранит коды символов одной строки экрана. Счет линий начинается сверху экрана.

АДРЕСА С 22528 ПО 23295 (5800-5AFF - шестнадцатиричные) - это область атрибутов - по 1 байту на каждую позицию символа, где хранятся признаки цветов, яркости, мигания. Экран имеет 768 символов, каждый из которых имеет один из восьми цветов "чернил", восьми цветов "бумаги" (фон), признак мигания и признак яркости - повышенной и нормальной.

Зависимость между площадями символа и байтами атрибута несложная, так как байты просматриваются подряд для каждой строки экрана сверху вниз слева направо.

Кодировка байтов атрибутов следующая:

биты 0-2 - код цвета "чернил"
биты 3-5 - код цвета "бумаги" (фон)
бит 6 - признак яркости (0- обычная, 1- повышенная)
бит 7 - признак мигания (0- постоянное свечение, 1 - мигание).

Коды цветов следующие:

0 - черный
1 - синий
2 - красный
3 - фиолетовый
4 - зеленый
5 - голубой
6 - желтый
7 - белый

АДРЕСА С 23296 ПО 23551 - область буфера ZX-принтера - содержит 256 байтов - по 32 байта на каждую печатаемую строку. При отсутствии такого принтера эта область часто используется для хранения небольших процедур в машинных кодах. Здесь они не повреждаются БЕЙСИКом и, кроме того, размещение их в нижней половине памяти компьютера повышает их быстродействие.

В компьютерах 128К в этой области хранятся дополнительные системные переменные для поддержания повышенных возможностей этих машин, и потому туда ничего нельзя записывать.

АДРЕСА С 23552 ПО 23733 - 182 байта (5000-5CB5 - шестнадцатиричные) - область системных переменных.

Байты памяти с 23552 до 23733 предназначены для специального использования. В них размещаются, так называемые, системные переменные. Не надо путать их имена с именами переменных в программе. Компьютер не распознает ссылки к этим переменным из Бейсик-программы по их именам. Имена используются только для мнемонического обозначения этих переменных в этом описании.

Информация, записанная в первом столбце таблицы, имеет следующее значение:

X - переменная не должна изменяться, так как это может нарушить работу системы.

N - изменение переменной не приводит к длительному эффекту. Число - число байтов в переменной (для двухбайтовых переменных младший байт первый.)

Например необходимо изменить значение на V в двухбайтовой переменной по адресу N:

```
10 POKE N,V-256*INT(V/256)
20 POKE N+1,INT(V/256)
```

Для просмотра нового значения можно использовать оператор:

```
PEEK N+256*PEEK(N+1)
```

зн.	адрес	имя	содержание
N8	23552	KSTATE	Используется при сканировании клавиатуры
N1	23560	LAST K	Запоминается последняя нажатая клавиша
1	23561	REPDEL	Время,в 50-ых долях секунды,в течение которого клавиша должна быть зафикси-рована в нажатом состоянии. Чтобы произошел автоповтор. Начальное значение 35, но может быть изменено
1	23562	REPPER	Задержка, в 50-ых долях секунды,между последовательными опросами клавиш. Начальное значение 5
N2	23563	DEFADD	Адрес аргументов функций пользовате-ля если они используются, иначе 0
N1	23565	K DATA	Второй байт управления цветом с кла-виатуры
N2	23566	TVDATA	Байты цвета, AT, TAB управления TV
X38	23568	STRMS	Адреса каналов,подключенных к потокам
2	23606	CHARS	Адрес символьного набора -256. Обычно этот набор находится в ПЗУ, но может быть размещаем и в ОЗУ с указанием в CHARS адреса размещения
1	23608	RASP	Продолжительность звукового сигнала
1	23609	PIP	Длительность задержки, устраняющей дребезг клавиатуры
1	23610	ERR NR	Код сообщения -1. Начальное значение 255 (для "-1"), т.е.PEEK 23610 = 255
X1	23611	FLAGS	Управляющие флаги Бейсика
X1	23612	TV FLAG	Флаг телевизора

зн.	адрес	имя	содержание
X2	23613	ERR SP	Адрес в аппаратном стеке, используемый как адрес возврата при ошибке
N2	23615	LIST SP	Адрес возврата из автоматического листинга
N1	23617	MODE -	Режим курсора: [K], [L], [C], [E] или [G]
2	23618	NEW PPC	Номер строки, на которую должен быть сделан переход
1	23620	NSPPS	Номер оператора в строке, к которому следует переход
2	23621	PPC	Номер строки, оператор в которой выполняется
1	23623	SUB PPC	Порядковый номер выполняющегося оператора в строке
1	23624	BORDCR	Цвет бордюра, умноженный на 8
2	23625	E PPC	Количество текущих строк (с курсором)
X2	23627	VARS	Адреса переменных
N2	23629	DEST	Адрес переменной в задании
X2	23631	CHANS	Адрес канала данных
X2	23633	CURCHL	Адрес данных для ввода-вывода
X2	23635	PROG	Адрес бейсик-программы
X2	23637	NXTLIN	Адрес следующей строки в программе
X2	23639	DATADD	Адрес терминатора последнего символа в DATA
X2	23641	E LINE	Адрес вводимой команды
2	23643	K CUR	Адрес курсора
X2	23645	CH ADD	Адрес следующего интерпретируемого символа: символ аргумента в PEEK , NEWLINE или POKE операторах
2	23647	X PRT	Адрес символа следующего за маркером [?]
X2	23649	WORK SP	Адрес временной рабочей области
X2	23651	STK BOT	Адрес "дна" программируемого стека
X2	23653	STK END	Адрес вершины стека
N1	23655	BREG	B-регистр калькулятора

зн.	адрес	имя	содержание
N2	23656	MEM	Адрес области, используемой как память калькулятора (обычно MEMBOT, но не всегда)
1	23658	FLAG52	Старшие флаги
X1	23659	DF SZ	Число строк (включая и одну чистую) в нижней части экрана
2	23660	S TOP	Количество верхних строк программы в автоматическом листинге
2	23662	OLDPPC	Номер строки, на которую указывает CONTINUE
1	23664	OSPPC	Номер оператора в строке, на которую указывает CONTINUE
N1	23665	FLAGX	Переменные флаги
N2	23666	STR LEN	Длина строковой переменной в операторе присваивания
N2	23668	T ADDR	Адрес следующего символа в синтаксической таблице
2	23670	SEED	Начальное значение для RND, изменяется функцией RANDOMIZE
3	23672	FRAMES	Счетчик кадров - приращение через каждые 20 MS
2	23675	UDG	Адрес первого определяемого пользователем символа
1 1	23677 23678	COORDS	X-координата точки графопостроителя Y-координата точки графопостроителя
1	23679	P POSN	33-позиционное число для позиционирования принтера
1	23680	PR CC	Младший байт адреса позиции для LPRINT для печати
1	23681		Не используется
2	23682	ECHO E	33-позиционное и 24-строковое числа (в нижней половине) конца входного буфера
2	23684	DF CC	Адрес PRINT-позиции в области экрана
2	23686	DF CCL	Подобно DF CC для нижней части экрана
X1 X1	23688 23689	S POSN	33-позиционное число для PRINT позиции 23-строковое число для PRINT позиции

зн.	адрес	имя	содержание
X2	23690	S POSNL	Подобно S POSN для нижней части
1	23692	SCR CT	Счетчик сверток: всегда на 1 больше числа сверток, которые должны быть проведены перед ОСТАНОВОМ со сверткой! Если Вы установите это число больше, чем 1 (скажем 255), то экран будет сворачиваться без запроса к Вам
1	23693	ATTR P	Текущие цветовые атрибуты
1	23694	MASK P	Используется для высвечивания цветов. Бит, установленный в 1,показывает,что! биты атрибутов берутся не из ATTR P, а из того, что указано на экране
N1	23695	ATTR T	Временные атрибуты
N1	23696	MASK T	Временный MASK P
1	23697	P FLAG	Старшие флаги
N30	23698	MEMBOT	Область памяти для калькулятора. Используется для записи чисел, которые не могут быть размещены в программируемом стеке калькулятора
2	23728		Не используется
2	23730	RAMTOP	Адрес последнего байта области, доступной для размещения программы на BASICe.
2	23732	P-RAMT	Адрес последнего байта ОЗУ (вершина)

Следующая программа выдаст вам первые 22 байта области системных переменных:

```
10 FOR N=0 TO 21
20 PRINT PEEK(PEEK 23627+256 * PEEK 23628+N)
30 NEXT N
```

теперь замените строку 20 на

```
20 PRINT PEEK(23755+N)
```

и вы дополнительно получите дамп самой программы.

АДРЕСА С 23734 ДО CHANS-1 - карты памяти микродрайва. При работе с микродрайвом здесь хранится информация, например, об испорченных секторах на ленте и т. п. Если микродрайва нет, то эта область не организуется.

ОБЛАСТЬ ИНФОРМАЦИОННЫХ КАНАЛОВ. Специальная область памяти расположена с адреса, указанного в переменной CHANS. CHANS находится в адресах 23631 и 23632 (шестнадцатиричные 5C4F и 5C50). Область переменной длины. Концом области служит маркер со значением 128 (80H).

В стандартном СПЕКТРУМе без подсоединенных микроприводов имеются 4 основных канала:

канал "K" - обеспечивает ввод от клавиатуры и вывод в нижнюю часть экрана;

канал "S" - обеспечивает вывод в верхнюю часть экрана;

канал "R" - обеспечивает вывод в рабочую область, которая при необходимости может быть увеличена;

канал "P" - обеспечивает вывод на принтер.

Информационные каналы состоят, для каждого канала, из 5 байтов данных. Эти байты содержат:

- адрес программы ввода - 2 байта
- адрес программы вывода - 2 байта
- один символ имени файла - 1 байт

В стандартном СПЕКТРУМе, с учетом четырех каналов и маркера конца, область занимает память с 23734 по 23754 - 21 байт.

ОБЛАСТЬ БЕЙСИК-ПРОГРАММЫ (БП).

Область содержит последовательные строки текста программы. Размер области определяется количеством строк.

Начало программы задается в переменной PROG (адреса 23635-23636). Отметим, что в стандартном СПЕКТРУМе переменная PROG указывает на адрес 23755 и так до тех пор, пока не будет подключен микродрайв или использован дополнительный канал.

ОБЛАСТЬ ПЕРЕМЕННЫХ - это область, где хранятся все переменные, с которыми оперирует БП.

Начало области задается переменной VARS (адреса 23627 и 23628, или шестнадцатиричные 5C4B и 5C4C). Начало этой области остается постоянным во время работы БП. Однако с появлением новых переменных ее длина может меняться. Граница области задается маркером конца (код 128 или шестнадцатиричный 80).

Следующая программа содержит только переменные оператора FOR NEXT

```
10 FOR A=23804 TO 23823; PRINT A; TAB 9; PEEK A; NEXT A RUN
```

ОБЛАСТЬ БУФЕРА РЕДАКТОРА - сюда помещается строка программы, которую вводят или редактируют.

Начало области задается переменной E-LINE (адрес 23641 и 23642, или, 5C59 и 5C5A шестнадцатиричных).

Нижняя часть экрана высвечивает редактируемую строку программы.

По мере поступления входных символов с клавиатуры область редактирования расширяется.

Аналогичная процедура имеет место, когда используется клавиша EDIT для вызова строки БЕЙСИК на нижнюю часть дисплея. Прежде всего область редактирования расширяется до требуемого размера с тем, чтобы разместилась эта строка БЕЙСИКа. Затем эта строка копируется из области программы в область редактирования и , наконец, строка из области редактирования копируется в нижнюю часть экранной области RAM.

Так как область редактирования представляет собой динамическую область, т.е. она меняется при использовании, то нецелесообразно давать пример на БЕЙСИКе.

РАБОЧАЯ ОБЛАСТЬ (буфер команды INPUT)

Эта область памяти используется для выполнения большого числа различных задач, например, ввод данных, соединение в цепочку строк и т. д. Начальный адрес области задается величиной, содержащейся в системной переменной WORKS, которая сама находится по адресу 23649 и 23650 (шестнадцатиричных 5061 и 5062). Когда же требуется дополнительное место в рабочей области памяти, то эта область памяти расширяется. После использования рабочая область освобождается. Это значит, что она сводится к нулю, чтобы избежать использования большего количества адресов, чем это необходимо.

СТЕК ВЫЧИСЛИТЕЛЯ (калькулятора)

Эта очень важная область памяти начинается с адреса, определенного системной переменной STK BOT, которая сама находится в адресах 23651 и 23652 (шест. 5063-5064). Область распространяется до адреса, заданного в системной переменной STK END (адреса 23653 и 23654, шест. 5065 и 5066). Стек вычислителя может содержать числа с плавающей точкой, целые числа, а при использовании строк - он содержит пять байтов, определяющих параметры строки.

Стек обслуживается по правилу: "первым пришел - последним ушел". Можно рассматривать величину, находящуюся вверху стека, если она существует, как последнюю "пришедшую величину".

РЕЗЕРВНАЯ ПАМЯТЬ. Область памяти между стеком вычислителя и стеком машины представляет собой количество памяти, которым располагает пользователь. Например, в 16К-СПЕКТРУМе номинальный размер области равен 8939, когда система

включается. Однако интересно отметить, что приемлемая самая низкая величина для CLEAR равна 23821, которая опускает RAMTOP и расширяет резервную область до 8878 байт.

МАШИННЫЙ СТЕК. Микропроцессор Z80 должен иметь рабочую область для своего использования и это называется машинный стек. Указатель стека Z80 всегда показывает на последний адрес, который должен заполняться.

GOSUB - СТЕК (стек возвратов). Здесь хранится номер строки, к которой надо вернуться после выполнения подпрограммы по RETURN.

Стек расширяется в памяти вниз и каждая команда GOSUB требует три адреса. Самый старший адрес содержит номер операнда в пределах строки БЕЙСИКа, к которой должен быть осуществлен возврат. Второй адрес содержит младшую часть номера строки, определяющей цикл, а третий адрес содержит ее старшую часть.

Демонстрация следующей программы показывает стек GOSUB, который используется для сохранения чисел строки при организации трех вложенных программ.

Демонстрационная программа GOSUB - стека

```
10 GOSUB 20: STOP
20 GOSUB 30: RETURN
30 GOSUB 40: RETURN
40 FOR A=32584 TO 32547 STEP -1: PRINT A, PEEK A: NEXT A: RETURN
RUN
```

Два адреса выше GOSUB - стека всегда содержат величины 0 и 62 (шестнадцатиричный 00 и 3E) и они представляют собой неправильный номер строк.

Программа БЕЙСИК при выполнении лишней команды RETURN перейдет к неправильному номеру строки и выдаст сообщение RETURN без GOSUB.

Системная переменная RAMTOP, которая занимает адрес 23730 и 23731 (шестн. 5CB2 - 5CB3) содержит адрес ячейки, в которой хранится код 62. Эта ячейка рассматривается как последняя ячейка в системной области БЕЙСИКа.

ГРАФИЧЕСКАЯ ОБЛАСТЬ, ОПРЕДЕЛЕННАЯ ПОЛЬЗОВАТЕЛЕМ. До тех пор, пока системная область БЕЙСИКа расширяется вниз вследствие использования команды CLEAR, старшие 168 ячеек памяти содержат представления 21 графического символа, определенного пользователем.

Самый старший адрес в памяти всегда адресуется системной переменной P-RAMT, расположенной в адресах 23732 и 23733 (шестн. 5CB4-5CB5).

Для получения содержимого области любой из памятей используется функция PEEK с адресом в качестве аргумента. Функция возвращает значение байта по этому адресу.

Рассматриваемая ниже программа выводит содержание первых 21 байтов с их адресами:

```
10 PRINT"ADDRESS"; TAB 8; "BYTE"
20 FOR A=0 TO 20
30 PRINT A; TAB 8; PEEK A;
40 NEXT A
```

Для изменения содержимого памяти (только для RAM (ОЗУ)) используется оператор POKE в форме

```
POKE ADDRESS,NEW CONTENTS
```

где "ADDRESS" "NEW CONTENS" - числовые выражения

Пример:

```
POKE 31000, 57
```

"NEW CONTENS" может принимать значения от -255 до 255.

2. ПОРЯДОК ПОДКЛЮЧЕНИЯ КОМПЬЮТЕРА

Итак, в Ваших руках компьютер ZX-SPECTRUM (либо его аналог, выпущенный той или иной фирмой, а может быть даже изготовленный своими руками).

Подключите его соответствующими кабелями к телевизору и магнитофону согласно инструкции по эксплуатации. Если Ваш компьютер укомплектован джойстиком и контроллером дисковода с дисководом, принтером, световым пером или другой периферией, то обязательно их подключите до включения компьютера.

Здесь Вы можете столкнуться с трудностью подключения к Вашему телевизору. Существуют модели компьютеров, подключаемых прямо к антенному входу телевизора, однако их мало и качество изображения оставляет желать лучшего. Чаще всего приходится забираться внутрь Вашего телевизора и устанавливать специальные согласующие платы. Рекомендуем за советом по подключению к конкретной модели телевизора обратиться к специалистам.

Если Вы все правильно подключили, то после включения компьютера на экране телевизора должно появиться исходное сообщение:

1982 Sinclair Research LTD

но в зависимости от модификации компьютера оно может быть и другим (например отличаться годом, либо быть вообще на русском языке).

3. ЗАГРУЗКА С МАГНИТОФОНА

Если Вы недавно приобрели компьютер, то вполне понятно желание начать работу с готовыми фирменными программами, а самостоятельное программирование отложить на более поздний срок.

Загрузка и запуск фирменных программ выполняются весьма просто:

LOAD ""[ENTER].

Помните, что между кавычками не должно быть пробела.

Основную проблему на этом этапе представляет плохая загружаемость программ, о чем свидетельствует сообщение об ошибке

TAPE LOADING ERROR,

и зависание программы после загрузки или самопроизвольный сброс. Причина не обязательно связана с плохой загрузкой, но если лента получена Вами из стороннего источника, то это скорее всего так. Здесь мы приведем некоторые принципы, которыми надо руководствоваться, чтобы улучшить загружаемость программ.

1. По-настоящему надежную загрузку может обеспечить только высококачественная лента, если она записана на том же магнитофоне, на котором и воспроизводится. Понравившиеся Вам программы надо откопировать на своем магнитофоне (см.ниже).

2. Для работы с компьютером годится любой бытовой кассетный магнитофон, причем чем он проще, тем легче добиться надежной работы. В то же время желательно, чтобы он имел регуляторы тембров (по возмжности раздельные). Компьютер весьма чувствителен к отклонениям скорости. Монофонические магнитофоны во всех случаях предпочтительнее, чем стерео. Обязательным элементом является счетчик ленты, позволяющий быстро находить нужную программу.

3. Приступая к загрузке программы, установите уровень громкости на магнитофоне примерно на 70-80% от максимального и, если загрузка не пошла, поэкспериментируйте с ним. Помните, что программы чаще страдают от избыточного уровня, чем от недостаточного. Поиск ведите в сторону уменьшения уровня.

4. Установка регуляторов тембра также имеет большое значение. Нормально "высокие" надо ставить на 80% от максимума, а низкие примерно на 20%. Если программа решительно не загружается, а регулировка громкости ничего не дает, то попробуйте установку 50%-50% и 20%-80%.

5. Загрузка программ сопровождается перемещением цветных полос по бордюрному полю телевизора. По нему можно также многое сказать о настройке магнитофона. Сначала идут в течение 3-х - 5-ти секунд широкие красные и голубые полосы. Это так называемый пилоттон. Он нужен для того, чтобы процессор подготовился к загрузке. Ширина красных и голубых полос должна быть одинаковой. Если голубые полосы шире, то значит уровень слишком большой. Если шире красные

полосы – то слишком низкий. Эти полосы должны медленно перемещаться по экрану. Слишком быстрое их перемещение говорит о несоответствии скоростей Вашего магнитофона и того, на котором была сделана запись. Неравномерное перемещение (с ускорением и рывками) свидетельствует о некачественности лентопротяжной системы магнитофона либо Вашего, либо того, на котором делалась запись.

Бывают случаи когда широкие красные и голубые полосы "превращаются" в широкие желтые и синие. Это может указывать на то, что следует поменять местами установку регуляторов тембра.

6. Если у Вас стереомагнитофон и программа не загружается, попробуйте загрузить ее отдельно по левому и по правому каналам.

7. Если все пути регулировки исчерпаны, но программа не загружается и при этом точно известно,что копия работоспособна, по всей видимости причина в настройке головок у писавшего и воспроизводящего магнитофона. Как правило недорогие монофонические магнитофоны имеют в крышке отверстие для узкой отвертки, чтобы можно было при воспроизведении записи на слух, поворотом подпружиненного винта, регулировать азимутальное положение головки. При этом следует добиваться наиболее четкого и резкого звучания сигналов. Обычно бывает достаточно поворота в ту или иную сторону на угол 120 градусов. Если у Вас только один магнитофон, и Вы используете его также и для записи программ, то регулировать головку нежелательно, либо придется периодически сталкиваться с необходимостью юстировки. Самый лучший вариант – использовать для воспроизведения программ, полученных со стороны, отдельный магнитофон, выбрав его из самых простых и дешевых, а для записи программ – наиболее качественный, по возможности специализированный аппарат.

8. Обычно программы состоят из нескольких блоков, которые загружаются последовательно один за другим. При плохой загрузке бывает очень трудно загрузить программу, разные блоки которой требуют разной установки органов управления магнитофона. Для того, чтобы обойти эту проблему, надо проводить загрузку в копирующую программу. Если, предположим, при загрузке второго блока произойдет сбой, то первый блок уже загружен и остается, так что его повторять не надо. К тому же, по окончании длительной трудоемкой загрузки можно сразу же выполнить качественную копию и больше не иметь с этой программой проблем.

9. Сделать процесс настройки магнитофона наглядным и значительно более простым можно с помощью специальных программ, например с помощью программы TAPER.

3.1. ФОРМАТ КОДИРОВАНИЯ ИНФОРМАЦИИ НА МАГНИТНОЙ ЛЕНТЕ

Каждый блок информации (файл) на магнитной ленте начинается с синхронизирующего сигнала (ЛДР) длительностью 2 или 5 секунд. Частота этого сигнала – около 810 Гц (период 1,25 мс). После этого сигнала идет один период специального синхросигнала – длительность "нуля" – около 0,19 мс, "единицы" – 0,21 мс. Затем следуют байты данных, передаваемые последовательно, начиная со старшего бита. "Нулевой" бит передается одним периодом сигнала частотой около 2047 Гц (период – 0,489 мс). "Единичный" – одним периодом сигнала частотой около 1023 Гц (период – 0,978 мс).

Каждый файл состоит из байта типа (значение от 0 до 255), собственно данных и байта контрольной суммы. Байт типа и байт контрольной суммы не входят в длину файла. В контрольную сумму входит значение байта типа.

Байт типа обычно принимает значение 0 (для заголовка) и 255 (для данных). Если есть необходимость, пользователь может использовать и любой другой тип. От значения байта типа зависит длительность сигнала ЛДР. При значении байта типа от 0 до 7, длительность сигнала равна 5 секундам. Если байт типа больше 7, то длительность сигнала 2 секунды.

Стандартный файл, формируемый компьютером по команде разгрузки, состоит из двух файлов: файла заголовка, длиной 17 байт, и файла данных.

Формат файла заголовка:

Номера байтов	Назначение
1	Тип информации, описываемый заголовком: 0 – программа на Бейсике; 1 – числовой массив; 2 – символьный массив; 3 – "байтовый" файл: программа в машинных кодах или образ экрана.
2 – 11	Имя файла в кодах КОИ-7. Если имя отсутствует, то первый байт равен 255.
12 – 13	Длина файла данных.
14 – 15	Для программы – номер строки для автозапуска программы. Если первый байт равен 80Н, то автозапуск не был задан. Для "байтового" файла – адрес загрузки, для массивов – второй байт содержит имя массива (один символ КОИ-7).
16 – 17	Только для программы – длина программы на Бейсике.

Временные диаграммы записи-считывания:

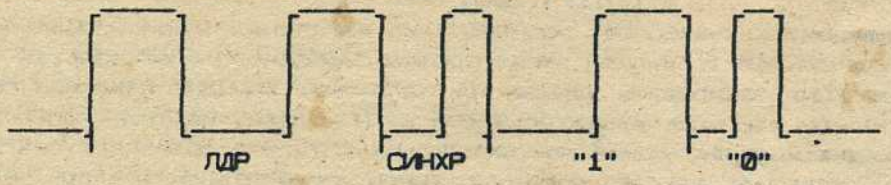


Рис.4.

3.2. ПРОГРАММА TAPER

Программа предназначена для настройки кассетного магнитофона. Она анализирует (раскладывает по частотам) сигнал, приходящий на вход компьютера, и изображает его на экране в виде спектра. Вертикальная шкала имеет три менти. Против этих меток должны находиться пики сигналов во время загрузки. Самая верхняя метка указывает на пик во время считывания "пилоттона", а две другие – показывают уровень "нулей" и уровень "единиц". При прослушивании цифровой записи через программу TAPER пики в участках, отмеченных метками, должны быть ярко выражены и никаких других побочных всплесков не должно быть.

4. КОПИРОВАНИЕ ПРОГРАММ

Если Вы сами написали какую-либо программу на Бейсике, то выполнить ее копию несложно. Для этого достаточно дать команду

SAVE "имя"

и нажать [ENTER]. Если Вы хотите, чтобы программа автостартовала после загрузки, начиная со строки M, то команда имеет вид:

SAVE "имя" LINE M.

Если в программе есть блоки, записанные в машинном коде, то дается команда

SAVE "имя" CODE M,N

где M - адрес, с которого начинается блок кодов; N - длина этого блока в байтах. При загрузке программ блоки, записанные на Бейсике, индицируются на экране как

PROGRAM ... ,

а блоки машинных кодов как

BYTES ...

Если Вы выгружаете свою разработку, то Вы , конечно, знаете те адреса, в которых располагаются Ваши блоки. Однако, если Вы имеете дело с чужой (фирменной) программой, то загрузив ее, Вам не так-то просто ее выгрузить. Во-первых, она автостартует и остановить ее сложно. Во-вторых, Вам неизвестны адреса блоков, из которых она состоит.

Упростить процесс копирования и автоматизировать его можно с помощью специальных копирующих программ. Эти программы имеют свою нестандартную загрузочную систему. Таких программ очень много. Мы здесь рассмотрим несколько наиболее широко распространенных.

4.1. COPY 86/M

Это, по-видимому, наиболее удобный и наглядный копировщик общего применения. В исходном состоянии он имеет объем свободного пространства 45000 байтов, и способен компрессировать данные при загрузке. Компрессирование состоит в том, что например вместо последовательности из N нулевых байтов записывается один "нуль", а следующий байт указывает сколько их (N). Это позволяет загружать в копировщик значительно более 45K. Особенно много места экономится при загрузке экранов, т.к. в графических изображениях часто встречаются длинные последовательности нулей.

Результат компрессии данных отправляется на хранение в область экрана, что изображается появлением на экране полос и точек, напоминающих телеграфный код.

Сводка команд программы COPY 86/M

BREAK	- переход в исходное положение;
L	- загрузить файлы;
C	- выгрузить (скопировать) файлы; Выгрузка производится с паузой 1,5 сек. между записями.
M	- то же, но с паузой 3 сек.;
V (VERIFY)	- проверка выгруженных записей;
D (DELETE)	- удаление ненужных записей;
H	- перевод в шестнадцатиричную систему;
X	- просмотр записей с отбоем ранее поданных команд;
S	- то же, но без отбоя команд;
B	- просмотр Бейсик-программ. [Y] - построчно, [P] - на 4 строки вперед;
A (ALL)	- после этой команды очередная команда будет одновременно распространяться на все записи.

Все команды подтверждаются нажатием [ENTER].

На экране файлы записываются в "окне", имеющем 4 строки. Если записей более чем 4, то одновременно их вывести на экран нельзя, поэтому и необходимы клавиши [S] и [X] для "протягивания" записей через "окно".

При использовании программы следует помнить, что она может не работать с некоторой периферией. Так, она не будет работать, если к компьютеру подключен кемпстон-джойстик.

К сожалению, эта программа не работает и со многими самодельными компьютерами. В таком случае как правило пользуются другой компрессирующей

программой TF COPY.

4.2. TF COPY

После загрузки программы на экране появляется исходное меню. Нажмите "0", программа стартует и перед Вами появится основное меню программы, которое выглядит так:

LOAD SAVE DELETE VERIFY MODE

Нажав первую букву нужной команды, Вы входите в нужный Вам режим.

LOAD - загрузка файлов;
SAVE - выгрузка файлов;
DELETE - удаление файлов;
VERIFY - проверка выгруженных файлов;
MODE - переключение режима.

В программе имеются 3 режима работы:

Режим 1 - 41984 байта свободной памяти;
Режим 2 - 44032 байта свободной памяти;
Режим 3 - 44288 байта свободной памяти.

Все команды должны завершаться нажатим [ENTER].

Обратите внимание, что при переходе на другой режим происходит очистка памяти, т.е. загруженная информация будет уничтожена.

После выбора одного из вышеуказанных режимов программа выдает запрос

DELETE FROM ... TO ...
(Удалить записи с ... по ...)

В ответ надо дать номера тех записей, которые Вы хотите удалить. Например, если Вы хотите уничтожить записи с третьей по шестую, то соответственно надо ввести 3 и 6.

4.3. COPY-COPY

Эта программа находится в эксплуатации довольно давно и представляет классический образец копировщика. Она не выполняет компрессирование и имеет довольно ограниченный объем свободной памяти (42k), но в ней есть несколько оригинальных вспомагательных режимов, которые в ряде случаев делают ее незаменимой.

Сводка команд

Все команды являются ключевыми словами "СПЕКТРУМа" и потому не набираются по буквам. Они требуют завершения нажатием клавиши [ENTER].

LOAD - выполнить загрузку очередной записи (очередного блока);

LOAD N TO M - выполнить загрузку записей, начиная с номера N до номера M. При этом ранее существовавшие записи с номером N и выше будут уничтожены, т.е.

LOAD 1

уничтожит все ранее существовавшие записи и подготовит компьютер к загрузке новых.

Возможные вариации этой команды:

LOAD N TO
LOAD TO M

SAVE - выполнение выгрузки записи (записей) на ленту. Может иметь форму:

SAVE N TO M
SAVE TO M
SAVE N TO
SAVE N TO M STEP K

Параметр K после оператора STEP показывает величину паузы в секундах, которую компьютер выдержит между отдельными блоками.

VERIFY - проверка выгруженных записей. Имеет все те же формы, что и команда LOAD.

LOAD AT ADDR - загрузка блока, начиная с адресса ADDR. Например: LOAD AT 16384 - загрузка экрана.

LIST ADDR - выдача на экран содержимого памяти, начиная с указанного адреса.

LIST = LIST 0.

POKE ADDR,N - поместить по указанному адресу байт N.

Этой командой часто пользуются совместно с командой LIST. Они позволяют вносить изменения в машинный код программы. В принципе с подобными задачами лучше справляются специализированные программы для отладки машинного кода (их часто называют МОНИТОРАМИ),но наличие такой функции в копировщике делает его весьма удобным.

COPY 16384 - команда, которая позволяет копировать блоки длиной 49K.

Этот режим делает данную программу незаменимой во многих случаях. Дело в том, что если программа имеет один значительный блок длиной порядка 49K и не поддается компрессии, то откопировать его копировщиками типа COPY 86/M и TF COPY не удастся, т.к. он не помещается в памяти. (Например программы FIRELORD и URIDIUM фирмы HUSON CONSULTANTS и другие). В режиме COPY 16384 программа COPY-COPY удаляет саму себя из памяти и оставляет только небольшой блок длиной несколько байтов, предназначенный для выгрузки загруженного блока. Выгрузка выполняется нажатием клавиши [CAPS SHIFT]. Возможна только однократная выгрузка, после этого копировщик необходимо снова загрузить. Разумеется магнитофон должен быть включен на запись до нажатия [CAPS SHIFT].

4.4. ДОПОЛНИТЕЛЬНЫЕ СВЕДЕНИЯ

Мы рассмотрели три наиболее распространенные копирующие программы. На самом деле их очень много, но принципы их работы отличаются не очень значительно. В то же время, надо помнить, что с помощью копировщиков можно только копировать программы, в которых не принято специальных мер для защиты от копирования. Многие фирмы такие меры применяют. Это например "спидлок" (ускоренный загрузчик), джеркитон (пилоттон, сопровождающий рывками), блоки избыточной длины (более 50K), фальшхэдеры (блоки длиной 17 байтов, которые воспринимаются копировщиком как заголовок очередного блоко, в то время как они таковыми не являются), укороченный пилоттон, удлиненный пилоттон, слишком широкий или слишком узкий пилоттон, блоки с замеряемой паузой между ними и многие другие методы. Принцип их действия состоит в том, что первый блок программы является БЕЙСИК-загрузчиком, который подготавливает загрузку второго блока. Второй блок в машинных кодах подготавливает нестандартную загрузку прочих блоков, которые уже не могут быть загружены никуда, где предварительно не отработал второй блок, в том числе и в копировщик. Копирование таких программ довольно трудоемкий процесс, но со многими защитами справляется например специальная программа LERM-7 (TC-7).

Другой подход к таким программам состоит в том, что используются специальные периферийные устройства. Их назначение - остановить ("заморозить") программу в каком-либо месте, а затем сделать полный дамп памяти (полную выгрузку всей памяти). Широко известны устройства типа MULTIFACE, позволяющие кроме этого производить выгрузку экрана в любом месте программы, вносить в программу изменения и продолжать исполнение программы с места прерывания. Аналогичными устройствами снабжаются иногда и дисковые системы, например, БЕТА-ДИСК ИНТЕРФЕЙС (TP ДОС) фирмы TECHNDOLOGY RESEARCH. Он имеет специальную кнопку, называемую MAGIC BUTTON (волшебная кнопка) для прерывания работы и сброса ее на диск. Правда запустить откопированную таким образом программу через LOAD не удается. Для этого интерфейс реализует нестандартный подход и выполняет это по команде GO TO "имя".

5. ЯЗЫК ПРОГРАММИРОВАНИЯ БЕЙСИК

Язык Бейсик - диалоговый алгоритмический язык высокого уровня. Характерной особенностью его является непосредственное взаимодействие пользователя с ЭВМ, когда он сразу же получает ответы на свои вопросы, посланные в ЭВМ.

Язык Бейсик близок к популярному алгоритмическому языку ФОРТРАН.

В языке Бейсик предусмотрены режимы работы программный и непосредственный.

Для работы в программном режиме необходимо, чтобы каждый оператор имел номер (метку). Роль метки выполняют десятичные целые числа от 1 до 9999. Последовательность пронумерованных операторов составляет программу.

При работе в непосредственном режиме оператор набирается без номера и немедленно выполняется при нажатии клавиши ENTER. Этот режим применяют при простых вычислениях и отладке программ.

Следует различать понятия "язык Бейсик" и "интерпретирующая Бейсик-система". Интерпретирующая Бейсик-система преобразует программу, написанную на алгоритмическом языке Бейсик, во внутреннее представление конкретной ЭВМ и обеспечивает ее выполнение в режиме интерпретации. Как правило, Бейсик-системы ориентированы на определенный тип ЭВМ и могут отличаться одна от другой по своим возможностям в разных ЭВМ.

Ниже мы рассмотрим один из вариантов языка и системы Бейсик, ориентированный на семейство компьютеров, построенных на базе микропроцессоров Z-80 (семейство ZX-СПЕКТРУМ и СПЕКТРУМ-совместимых моделей).

КОНСТАНТЫ

В языке Бейсик используются целые и вещественные константы. Константы выражаются цельми и десятичными числами. Запись числа осуществляется в форматах с фиксированной и плавающей запятой. Все числа могут иметь точность 9 или 10 знаков. Наибольшее число 10**38, а наименьшее положительное 4*10**(-39). Числа имеют внутреннее представление как числа с плавающей (двоичной) запятой, с выделением одного байта на показатель степени E в интервале от 1 до 255, и четырех байтов на мантиссу M в интервале от 0.5 до 1. Это представляется числом M*2**E(-128).

Поскольку 1/2<=M<1, старший значащий бит мантиссы всегда 1. Следовательно, мы можем заменить его на бит, обозначающий знак: 0 - для положительного числа и 1 - для отрицательного.

Наименьшее целое имеет специальное представление, в котором первый байт 0, второй - байт знака (0 и FFH), а третий и четвертый само число в дополнительном коде (младшие значащие цифры в первом байте).

ПЕРЕМЕННЫЕ

Переменная - это величина, значение которой может меняться в процессе выполнения программы. Числовые переменные имеют имя произвольной длины, начинающееся с буквы и продолжающееся буквами или цифрами. Пробелы и символы управления цветом игнорируются и все буквы преобразуются к минимально упакованному виду.

Управляющие переменные для FOR-NEXT циклов имеют имена длиной в одну букву.

МАССИВЫ

Числовые массивы имеют имена длиной в одну букву, которая может быть такой же, как имя скалярной переменной. Эти массивы могут иметь произвольную размерность и произвольный размер. Начальный индекс всегда 1.

Строки символов более гибкие в своей длине. Имя строковой переменной в отличие от простой переменной заканчивается символом доллара ($).

Строковые массивы также могут иметь произвольное количество измерений и размер. Их имена представляют одну букву и следующий за ней символ $, но не могут совпадать с именем простой строки символов.

Все строки в массивах имеют фиксированную длину, которая определяется числом, задающим последнюю размерность в операторе DIM. Начальный индекс 1.

Подстрока от строки может быть получена, как сечение. Сечение может быть:

а) пустым;

б) числовым выражением;

в) некоторым "строковым выражением", другим "строковым выражением" и использоваться в:

*) строковых выражениях (сечениях);

**) строковых массивах переменных

(индекс 1, индекс 2,..., индекс N, сечение)

или, что тоже самое

(индекс 1, индекс 2,..., сечение).

В случае *) строка выражения имеет значение S$. Если сечение массива пусто, то S$ считается подстрокой от самой себя.

Если сечение представляется числовым выражением с значением 'M', то результатом будет M-ый символ от S$ (подстрока длиной 1).

Если сечение представлено в форме в) и первое числовое выражение имеет значение 'M' (умалчиваемое значение 1), а второе 'N' (умалчиваемое значение S$), и если 1<=M<=N<= чем длина S$, то результатом будет подстрока от S$ с M-ым начальным символом и N-ым конечным.

Если 0<=N<M, то результатом будет пустая строка. В любом другом случае выдается сообщение об ошибке '3'.

Сечение выполняется перед функцией или операцией, которая осуществляется, если скобками не предписано сделать иначе.

Подстрока может назначаться (смотри оператор LET). Если часть строки записывается в строковый литерал, она должна удваиваться.

ФУНКЦИИ.

ИМЯ ФУНКЦИИ	ТИП АРГУМЕНТА	ДЕЙСТВИЕ (ВОЗВРАЩАЕМОЕ ЗНАЧЕНИЕ)
ABS	число	Абсолютное значение
ACS	число	Арккосинус в радианах. Выдает сообщение об ошибке A , если X не лежит в интервале от -1 до 1.
AND	логическая операция. Правый операнд всегда число. Слева может быть: - число, тогда ---->	A AND B = { A, если B<>0; B, если B=0
	- строка, тогда ---->	A$ AND B = { A$, если B<>0; "", если B=0
ASN	число	Арксинус в радианах. Выдает сообщение A , если X не лежит в интервале от -1 до 1.
ATN	число	Арктангенс в радианах.
ATTR	два числовых аргумента X и Y, закючаемые в скобки	Число, двоичный код которого представляет собой атрибуты Y-ой позиции X-ой строки экрана. Бит 7 (старший) равен 1 для мерцающего поля и 0 для немерцающего. Биты с 5 по 3 - цвет фона. Биты с 2 по 1 - цвет закрашивания. Выдает сообщение B, если 0<=X<=23 и 0<=Y<=31. Бит 6 - яркость.

ИМЯ ФУНКЦИИ	ТИП АРГУМЕНТА	ДЕЙСТВИЕ (ВОЗВРАЩАЕМОЕ ЗНАЧЕНИЕ)
BIN		Это не обычная функция. За BIN записывается последовательность нулей и единиц, представляющая собой двоичное представление числа, которое записывается в память.
CHR$	число	Символ, чей код представим числом X, округленным к ближайшему целому.
CODE	строка символов	Код первого символа в строке X (или 0, если X - пустая строка).
COS	число в радианах	Косинус X.
EXP	число	e в степени X.
FN		FN с последующим именем, определенной пользователем функции (см. DEF). Аргументы должны заключаться в скобки. Даже если нет аргументов, скобки все равно должны записываться.
IN	число	Осуществляется ввод на уровне микропроцессора из порта X (0<=X<=FFFFH). Загружается пара регистров BC и выполняется команда ассемблера IN A(C).
INKEY$	нет	Чтение с клавиатуры. Возвращает символ введенный с клавиатуры (в режиме [L] или [C]), если было действительное нажатие клавиши, или пустую строку в противном случае.
INT	число	Округление к ближайшему меньшему целому.
LEN	строка символов	Длина строки в символах.
LN	число	Натуральный логарифм. Выдает сообщение A, если X<=0.
NOT	число	0, если X<>0 и 1, если X=0. Операция имеет четвертый приоритет.
OR	Логическая операция. Оба операнда - числа.	1, если B<>0 A OR B = { A, если B=0 (операция имеет второй приоритет)
PEEK	число	Значение байта в памяти по адресу X, округленному к ближайшему целому.
PI	нет	Число 'пи' (3.14159265....).
POINT	Два числовых аргумента X	1, если точка экрана с координатами (X,Y) закрашена. 0, если точка имеет цвет фона. Выдает сообщение B , если не выполняются условия

ИМЯ ФУНКЦИИ	ТИП АРГУМЕНТА	ДЕЙСТВИЕ (ВОЗВРАЩАЕМОЕ ЗНАЧЕНИЕ)
	и Y, заключенные в скобки.	0<=X<=255 и 0<=Y<=175.
RND	нет	Очередное псевдослучайное число из последовательности, получаемой возведением в 75 степень модуля числа 65537, вычитанием 1 и делением на 65536. Число лежит в интервале 0<=Y<1.
SCREEN$	Два числовых аргумента X и Y, заключенные в скобки.	Символ (обычный или инверсный), который появляется на экране в строке X, позиции Y. Дает пустую строку, если символ не опознан.
SGN	число	-1, если X<0 определяет знак числа 0, если X=0 1, если X>0
SIN	число в радианах	Синус x
SQR	число	Корень квадратный. Выдает сообщение A , если X<0
STR$	число	Строка символов, которая должна быть отображена, если X выводится.
TAN	число	Тангенс угла, заданного в радианах.
USR	число	Вызывает программу в машинных кодах, начальный адрес которой X. При возврате результатом будет содержимое регистровой пары BC.
USR	строка символов	Адрес группы байтов задающих определенный пользователем символ для закрепления его за X.
VAL	строка символов	Вычисление X как числового выражения. Выдает сообщение C, если X содержит синтаксические ошибки или дает строковое (нечисловое) значение. Возможны и другие ошибки.
VAL$	строка символов	Вычисляет X как строковое выражение. Выдает сообщение C , если X содержит синтаксическую ошибку или дает не строковое (числовое) значение.

ОПЕРАЦИИ.

Префиксные:

- число отрицательное значение

Инфиксные (двухоперандовые):

+ сложение для чисел, конкатенция для строк
- вычитание
* умножение
/ деление

^ возведение в степень (стрелка вверх). Сообщение B, если левый операнд отрицательный.

= равенство :
> больше : Оба операнда должны быть одного типа.
< меньше : Результат равен 1, если сравнение
>= больше или равно : истинно и равен 0, если нет.
<= меньше или равно :
<> не равно :

Функции и операции имеют следующие приоритеты:

индексация и сечения – 12
все функции за исключением
NOT и префиксного минуса – 11
возведение в степень – 10
префиксный минус – 9
*, / – 8
+, – (вычитание) – 6
=, >, <, <=, >=, <> – 5
NOT – 4
AND – 3
OR – 2

ОПЕРАТОРЫ.

Принятые обозначения:

A – одна буква;
V – переменная;
X,Y,Z – числовые выражения;
M,N – числовые выражения, которые округляются к ближайшему целому;
E – некоторое выражение;
F – выражение имеющее строковое значение;
S – последовательность операторов, разделенных двоеточием;
C – последовательность символов управления цветом.
Каждый заканчивается запятой или точкой с запятой.
Цветовой символ имеет форму операндов:
PAPER, INK, FLASH, BRIGHT, INVERSE или OVER.

Текст произвольного выражения может располагаться в любом месте строки (за исключением номера строки, который должен размещаться в начале строки).

Все операторы, кроме INPUT, DEF и DATA могут использоваться и как команды, и в программах. Команда или строка программы может содержать несколько операторов, разделенных двоеточием (':').

Нет ограничений на положение оператора в строке, хотя есть и некоторые ограничения в IF и REM.

Все операторы языка сведены в следующую таблицу:

ОПЕРАТОР	ДЕЙСТВИЕ ОПЕРАТОРА
BEEP X,Y	Воспроизводит звук длительностью X сек. и высотой Y полутонов вверх от основного тона "ДО" (или вниз, если Y отрицательное).
BORDER M	Устанавливает цвет рамки (бордера) экрана. Выдает сообщение об ошибке K, если M>9.
BRIGHT M	Устанавливает яркость выводимого символа: 0 – для обычной яркости; 1 – для повышенной яркости; 8 – сохраняет существующую яркость.
CAT ***	Без MICRODRIVE не работает.

ОПЕРАТОР	ДЕЙСТВИЕ ОПЕРАТОРА
CIRCLE X,Y,Z	Изображает дугу или окружность с центром в точке с координатами (X,Y) и радиусом Z.
CLEAR	Уничтожает все переменные и очищает занимаемую ими память. Выполняет RESTORE и CLS, устанавливает PLOT-позицию в нижнюю левую точку экрана и очищает GO SUB стек.
CLEAR N	Подобно CLEAR, но дополнительно изменяет системную переменную RAMTOP на 'N' и задает новый GO SUB стек.
CLOSE# ***	Без MICRODRIVE не работает.
CLS	(CLEAR SCREEN) Очищает экран.
CONTINUE	Продолжает выполнение программы, начатой ранее и остановленой с сообщением, отличным от 0 . Если было сообщение 9 или L , то выполнение продолжается со следующего оператора, в других случаях - с того оператора, где случилась ошибка. Если сообщение возникло в командной строке, то CONTINUE вызовет попытку повторить командную строку и перейдет в цикл, если было сообщение 0:1; дает сообщение 0 , если было 0:2 или дает сообщение N , если было 0:3 или более. В качестве CONTINUE используется ключевое слово CONT на клавиатуре.
COPY	Пересылает копию 22 строк экрана на принтер, если он подключен. Помните, что по COPY нельзя распечатать находящийся на экране автоматический листинг. Выдает сообщение D , если нажать клавишу BREAK. (При работе с НГМД копирует файлы).
DATA E1,E2, E3,...	Часть списка данных. Должна располагаться в программе.
DEF FN A(A1, A2,...,AK)= E	Определяемая пользователем функция. Должна располагаться в программе. A,A1,A2 и т. д. единственные буквы или буквы и $ для строковых аргументов, значений. Используется форма DEF FN A(), если нет аргументов.
DELETE F ***	Без MICRODRIVE не работает.
DIM A(N1,N2, ...,NK)	Уничтожает массив с именем 'A' и устанавливает числовой массив 'A' с 'K' измерениями и присваивает всем его элементам значение 0.
DIM A$(N1,N2 ,...,NK)	Уничтожает массив или строку с именем 'A$' и устанавливает символьный массив 'A$' с 'K' измерениями и присваивает всем его элементам значение ' '.Массив может быть представлен,как массив строк фиксированной длины 'NK' с 'K-1' размерностью. Сообщение '4' выдается, если недостаточно места для размещения массива. Массив не определен до его описания в операторе DIM.
DRAW X,Y	То же самое, что и DRAW X,Y,0. Чертит прямую линию.

ОПЕРАТОР	ДЕЙСТВИЕ ОПЕРАТОРА
DRAW X,Y,Z	Изображает линию от текущей графической позиции в точку с приращениями X,Y по дуге в Z радиан. Выдает сообщение 'B' при выходе за пределы экрана.
ERASE ***	Без MICRODRIVE не работает. (Удаление файлов с ГМД).
FLASH N	Определяет, будет ли символ мерцающим или с постоянным свечением. N=0 для постоянного свечения, N=1 для мерцания, N=8 для сохранения предыдущего состояния.
FOR A=X TO Y	то же самое, что и FOR A=X TO Y STEP 1
FOR A=X TO Y STEP Z	Уничтожает скалярную переменную 'A' и устанавливает управляющую переменную 'X', предел 'Y', шаг приращения 'Z', зацикливает адрес, указанный в утверждении после оператора FOR. Проверяет, если начальное значение больше (если STEP>=0) или меньше (если STEP<0), чем предел, то происходит переход к утверждению NEXT A или выдача сообщения 1 , если нет (см. NEXT): Сообщение 4 выдается, если недостаточно места для размещения управляющей переменной.
FORMAT F ***	Без MICRODRIVE не работает. (Форматирование ГМД).
GO SUB N	Выполнение подпрограммы с адреса N. Записывает строку с оператором GO SUB в стек, затем, как GO TO N. Выдает сообщение 4, если не все подпрограммы завершились RETURN.
GO TO N	Продолжает выполнение программы со строки 'N'. Если 'N' пропущено, то с первой строки после этой.
IF X THEN S	Если 'X' истинно (не равно 0), то выполняется 'S'.'S' включает все операторы до конца строки. Форма 'IF X THEN НОМЕР СТРОКИ' недопустима.
INK N	Устанавливает цвет закрашивания (т.е. цвет, которым будут изображаться символы на цвете фона). 'N' в интервале от 0 до 7 указывает цвет. N=8 - оставить цвет без изменения, N=9 - увеличение контраста. Выдает сообщение K , если 'N' не лежит в интервале от 0 до 9.
INPUT...	Где '...' есть последовательность вводимых символов, разделяемых как в операторах PRINT запятыми, точками с запятой или апострофами. Вводимыми символами могут быть: а) некоторый PRINT-символ, начинающийся не с буквы; б) имя переменной; в) строка имен переменных строкового типа; PRINT-символы в случае а) представляются так же, как и в операторе PRINT, за исключением того, что они все выводятся в нижнюю часть экрана. В случае б) компьютер останавливается и ждет ввода некоторого выражения с клавиатуры, значение которого будет присвоено переменной. Ввод осуществляется обычным образом, а синтаксические ошибки выдаются мерцающим [?]. Для строкового выражения вводной буфер устанавливается

ОПЕРАТОР	ДЕЙСТВИЕ ОПЕРАТОРА
	для размещения двух таких строк (который при необхо- димости может быть увеличен). Если первый вводимый символ STOP, то программа останавливается с сообщени- ем 'H'. Случай в) подобен случаю б) с той лишь раз- ницей, что вводимая информация представляет собой строковый литерал неограниченной длины, и STOP в этом случае не сработает. Для останова Вы должны на- жать клавишу 'КУРСОР ВНИЗ'.
INVERSE N	Символ управления инверсией выводимого символа. Если N=0, символ выводится в обычном виде с прорисовкой цветом закрашивания (INK) на цвете фона (PAPER). Если N=1, то цветовое решение изображения символа меняется на обратное. Выдает сообщение K , если 'N' не 0 или 1
LET V=E	Присваивает значение 'E' переменной 'V'. Ключевое сло- во LET не может быть опущено. Скалярная переменная не определена, пока не встретится в операторах LET,READ или INPUT. Если 'V' индексируемая строковая переменная или сечение строкового массива (подстрока), то присваивание осуществляется с усечением справа или до- полнением пробелами до фиксированной длины.
LIST	То же, что и LIST 0.
LIST N	Выводит часть программы в верхнюю часть экрана, на- чиная с первой строки, меньшей, чем 'N', и делает 'N' текущей строкой.
LLIST	То же, что и LLIST 0.
LLIST N	Подобно LIST, но вывод осуществляется на принтер.
LOAD F	Загружает программу и переменные с магнитофона.
LOAD F DATA ()	Загружает числовой массив с магнитофона.
LOAD F DATA A$()	Загружает строковый массив (A - любая буква) с магнитофона.
LOAD F CODE M,N	Загружает старшие 'N' байтов, начиная с адреса 'M' с магнитофона.
LOAD F CODE M	Загружает байты, начиная с адреса 'M'.
LOAD F CODE	Загружает байты по тому же адресу, с которого они были разгружены .
LOAD F SCREEN$	Аналогично LOAD F CODE 16384,6912. Очищает файл экра- на и загружает его с магнитофона.
LPRINT	Подобно PRINT, но использует принтер.
MERGE F	Подобно LOAD F, но не затирает всю старую программу в памяти, а заменяет только те строки и переменные. У

ОПЕРАТОР	ДЕЙСТВИЕ ОПЕРАТОРА
	которых совпадают номера или имена с такими же на ленте.
MOVE F1,F2**	Без MICRODRIVE не работает.
NEW	Запускает заново систему программирования БЕЙСИК, уничтожая старую программу и переменные и используемую память, включая и байт адреса в системной переменной RAMTOP, но сохраняет системные переменные UDG, P RAMT, RASP и PIP.
NEXT A	а) находит управляющую переменную 'A'; б) прибавляет к ней значение STEP; в) если STEP>=0, а значение 'A' стало не больше значения "ПРЕДЕЛ" или STEP<0, а значение 'A' не меньше, чем значение "ПРЕДЕЛ", то происходит переход к оператору цикла. Сообщение 2 выдается, если не найдена переменная 'A'. Сообщение 1 выдается, если 'A' не является управляющей переменной цикла.
OPEN# ***	Без MICRODRIVE не работает.
OUT M,N	Выводит байт 'N' в порт 'M'. Операции выполняются на уровне микропроцессора (загружает в регистровую пару BC адрес 'M', а в регистр A - 'N' и выполняет команду ассемблера OUT (C),A). 16384<=M<=65535, -255<=N<=255, иначе выдается сообщение B.
OVER N	Управляющий символ надпечатывается по выведенной строке.Если N=0, то выводимый символ затирает существующий в этой позиции. Если N=1, то новый символ соединяется со старым, образуя цвет закрашивания (INK) - если один из символов (но не оба сразу) имеет цвет закрашивания, или цвет фона, если оба символа имеют один и тот же цвет.
PAPER N	Подобен INK, но управляет цветом фона.
PAUSE N	Останавливает выполнение программы и задерживает изображение на экране на 'N' кадров (50 кадров в секунду- частота кадровой развертки) или до нажатия любой клавиши. 0<=N<=65535, иначе выдается сообщение 'B'. При N=0 время задержки не учитывается и продолжается до нажатия любой клавиши.
PLOT C;M,N	Выводит точку закрашивающего цвета (обработанную OVER и INVERSE) с координатами (ABS (M), ABS (N)) и смещает графическую позицию (PLOT POSITION). Если цветовой символ 'C' не специфицирован иначе, то цвет закрашивания в позиции, где расположена эта точка, изменяется на текущий сплошной закрашивающий цвет, а другие указания (цвет фона, мерцание, яркость) остаются без изменений. 0<=ABS(M)<=255, 0<=ABS(N)<=191, иначе выдается сообщение B.
POKE M,N	Записывает значение 'N' в байт памяти по адресу 'M'. 16384<=M<=65535, 0<=N<=255,иначе выдается сообщение B.

ОПЕРАТОР	ДЕЙСТВИЕ ОПЕРАТОРА
PRINT...	Где '...' - последовательность PRINT-символов, разделенных запятыми, точками с запятой или апострофами, которые выводятся в экранный файл для отображения на экране телевизора. Точка с запятой сама действия не вызывает, а используется для разграничения символов. Запятая порождает управляющий символ 'ЗАПЯТАЯ', а апостроф символ ENTER. В конце оператора PRINT, если он не заканчивается точкой с запятой или апострофом, автоматическм выводится символ ENTER . PRINT-символом может быть: а) пустая строка (т.е. ничего); б) числовое выражение. Если значение выражения отрицательное, то выводится знак минус. Если X<=10**(-5) или X>=10**13, то вывод осуществляется в показательной форме. Мантисса представляется 8-ю цифрами (с нормализацией) и десятичной точкой (отсутствует только тогда, когда в мантиссе одна цифра) после первой цифры. Показатель степени записывается после буквы 'E' с последующим знаком и двумя цифрами порядка. Иначе X выводится как обычное десятичное число с 8-ю значащими цифрами. в) строковое выражение. Заключается в кавычки. В строке возможны пробелы до и после символов. Управляющие символы вызывают определяемое ими действие! Не отражаемые на экране символы выводятся как '?'. г) AT M,N Вывод в строку 'M', позицию 'N'. д) TAB N Вывод управляющего символа TAB с последующими двумя байтами 'N' (первый байт старший). Вызывает TAB-останов. е) Оператор цвета в форме PAPER, INK, FLASH, BRIGHT, INVERSE или OVER оператора.
RANDOMIZE	То же, что и RANDOMIZE 0.
RANDOMIZE N	Устанавливает системную переменную SEED, используемую для вычисления очередного значения функции RND. Если N<>0, то SEED принимает значение 'N'. Если N=0, то SEED принимает значение другой системной переменной FRAMES, подсчитывающей кадры, отображаемые на экране, что обеспечивает вполне случайное число. Оператор допускает сокращение RAND (см.клавишу). Выдает сообщение B , если 'N' не лежит в интервале от 0 до 65535.
READ V1,V2,..,V	Присваивает переменным одной за другой значения, последовательно представленные в списке DATA.
REM...	Не выполняется. '...' может быть последовательностью символов (исключая ENTER). Может включать (':') для указания отсутствия операторов в строке REM.
RESTORE	То же, что и RESTORE 0.
RESTORE N	Перезаписывает указатель данных в первый оператор DATA в строке с номером меньшим, чем N. Следующий оператор READ начнет считывание отсюда.

ОПЕРАТОР	ДЕЙСТВИЕ ОПЕРАТОРА
RETURN	Возврат из подпрограммы. Ссылается на оператор GO SUB в стеке и передает управление на строку после него. Выдает сообщение 7 , если нет указываемого оператора в стеке. Характерная ошибка - операторы GO SUB не сбалансированы с операторами RETURN.
RUN	То же самое, что и RUN 0.
RUN N	CLEAR, и затем GO TO N.
SAVE F	Записывает на ленту программу и переменные.
SAVE F LINE M	Записывает на ленту программу и переменные таким образом, что при загрузке программа автоматически выполняется со строки 'M'.
SAVE F DATA ( )	Запись на ленту числового массива.
SAVE F DATA A$( )	Запись на ленту строкового массива (A - любая буква).
SAVE F CODE M,N	Записывает на ленту 'N' байтов, начиная с адреса 'M'.
SAVE F SCREEN$	Аналогично SAVE F CODE 16384.6912. Выдает сообщение F ,если F - пустая строка или имеет длину более 10
STOP	Останавливает выполнение программы с выдачей сообщения 9. CONTINUE (продолжение) будет осуществляться со следующего оператора.
VERIFY	То же, что и LOAD, за исключением того, что данные не загружаются в ОЗУ, а сравниваются с находящимися там. Выдает сообщение R, если обнаружен хотя бы один не совпадающий байт.

6. ОСНОВЫ ПРОГРАММИРОВАНИЯ НА ЯЗЫКЕ БЕЙСИК

Краткое содержание: программы, номера строк, редактирование программ с использованием клавиш: <вверх>, <вниз>, EDIT, команды RUN, LIST, GO TO, CONTINUE, INPUT, NEW, REM, PRINT, STOP в INPUT-данных, BREAK.

Наберите эти две строки программы вычисления суммы двух чисел:

```
20 PRINT A
10 LET A=10
```

так, чтобы на экране появилось:

```
+----------------------------------------+
!  10 [>] LET A=10                       !
!  20     PRINT A                        !
!                                        !
!                                        !
!                                        !
!                                        !
!                                        !
!                                        !
!                                        !
!                                        !
!  [K]                                   !
+----------------------------------------+
```

Строка программы должна начинаться с номера, который не записывается в память, а служит лишь для указания порядка следования строк в программе, что важно при ее выполнении.

Теперь наберите:

```
15 LET B=15
```

и введите.

Подобного не возможно было бы сделать, если бы нумерация начиналась с 1 и 2, а не с 10 и 20, как в нашем случае (номер может быть в интервале от 1 до 9999).

Допустим, теперь Вам понадобилось изменить строку 20 на следующую:

```
20 PRINT A+B
```

это можно сделать используя команду EDIT.

Символ [>] в строке 15 называется программным курсором, а строка на которую он указывает, называется текущей. Это обычно последняя введенная строка, но Вы имеете возможность переместить программный курсор выше или ниже, используя соответствующие клавиши управления курсором. Установите его в строку 20. Когда Вы наберете команду (CAPS SHIFT +1) EDIT, то в нижней части экрана появится копия текущей строки 20. Нажмите и удерживайте клавишу перемещения курсора вправо до тех пор, пока курсор не переместится на конец оператора и затем введите '+B'(без нажатия ENTER). Строка в нижней части экрана примет вид:

```
20 PRINT A+B
```

теперь нажмите ENTER и это вызовет замену старой строки 20 на новую, записанную в нижней части экрана. На экране это будет выглядеть так:

```
+----------------------------------------+
! 10     LET A=10                       !
! 15     LET B=15                       !
! 20 [>] PRINT A+B                      !
!                                        !
!                                        !
!                                        !
!                                        !
!                                        !
!                                        !
!                                        !
! [K]                                   !
+----------------------------------------+
```

Запустите программу, нажав RUN и ENTER и получите на экране сумму.

Выполните теперь команду PRINT A,B. Переменные сохраняются даже после завершения программы.

Есть еще одно применение команды EDIT. Допустим, Вам надо удалить всю строку, набранную в нижней части экрана. Для этого Вы можете нажать и удерживать до конца строки клавишу DELETE, но можно сделать быстрее: нажать EDIT, что вызовет копирование текущей строки в нижнюю часть экрана, затем нажать ENTER, строка заменит такую же в программе, а нижняя часть экрана очистится.

Введите строку:

```
12 LET B=B
```

Теперь для удаления этой строки наберите:

```
12 (и затем ENTER)
```

программный курсор станет между строками 10 и 15, но клавишами управления курсором Вы можете установить его в любую строку. Еще раз выполните: 12 и ENTER, курсор снова установится между строками 10 и 15, теперь нажмите EDIT и строка 15 будет скопирована в нижнюю часть экрана. Оператор EDIT копирует вниз строку, следующую за строкой с новым номером. Нажмите ENTER для очистки нижней части экрана.

Теперь введите:

```
30 (и затем ENTER)
```

программный курсор установится после конца программы, если Вы теперь нажмете EDIT, вниз будет переслана строка 20.

И, наконец, выполните команду:

```
LIST 15
```

теперь Вы увидите на экране:

```
15 LET B=15
20 PRINT A+B
```

Строка 10 не отображается на экране, но она сохраняется в Вашей программе. Вы можете убедиться в этом, нажав ENTER.

Команда LIST 15 указывает, что надо отобразить листинг со строки с номером 15 и устанавливает в эту строку программный курсор. Это бывает удобно при просмотре очень больших программ.

Другое назначение номера строки - служит меткой оператора при ссылке к нему из другого места программы (в GO TO N).

Команда LIST без операндов выдает листинг с первой строки.

Команда NEW очищает память компьютера от старых программ и переменных.

Теперь выполним программу, переводящую значения температуры в градусах по Фаренгейту в температуру по Цельсию:

```
10 REM TEMPERATURE CONVERSION
20 PRINT "DEG F","DEG C"
30 PRINT
40 INPUT "ENTER DEG F",F
50 PRINT F,(F-32)*5/9
60 GO TO 40
```

Вы видете, что заголовок выводится в строке 20, и у Вас возникнет вопрос, что же делает строка 10? Компьютер игнорирует эту строку, это коментарий (REMARK или REMINDER). Все, что следует после REM, компьютером игнорируется до конца строки.

Вычисления доходят до строки 40 и компьютер переходит в ожидание ввода значения переменной F. Вы должны ввести это значение. Наберите число и нажмите ENTER. Компьютер выведит результат и снова перейдет в ожидание следующего числа. Этот переход обеспечивается в строке 60 в операторе GO TO 40.

Если на запрос очередного числа ответить STOP, то компьютер остановится с выдачей сообщения: 'H STOP IN INPUT, 40:1', которое поясняет причину останова и место останова (первый оператор в строке 40).

Если теперь Вы желаете вновь продолжить выполнение программы, то введите CONTINUE и компьютер запросит очередное число.

При использовании CONTINUE компьютер запоминает (до выдачи им 'OOK') номер последней выполнявшейся строки и продолжает выполнение именно с этой строки.

Просмотрите внимательно оператор PRINT в строке 50, запятая в нем очень важна. Запятые в операторе PRINT используются для указания того, что вывод

следующего после запятой знака должен продолжаться либо с левого края экрана, либо с его середины, в зависимости от того, какая это по порядку запятая в данном операторе. Так, в строке 50, запятая предписывает выводить значения температуры в градусах Цельсия с середины экрана.

Если использовать в операторе PRINT вместо запятой точку с запятой (';'), то очередные данные будут выводиться непосредственно после предыдущих без пробела.

Оператор в строке 30 выводит чистую строку.

Оператор PRINT всегда начинает вывод с начала следующей строки, но это можно изменить, поставив в конце предыдущего оператора PRINT запятую или точку с запятой:

```
50 PRINT F,
60 PRINT F;
```

Не путайте эти знаки с двоеточием (':'), которое используется только для разделения разных операторов в одной строке. Теперь наберем еще несколько программных строк:

```
100 REM THIS POLITE PROGRAM REMEMBER YOUR NAME
110 INPUT N$
120 PRINT "HELLO";N$;"!"
130 GO TO 110
```

Эта программа никак не связана с набранной нами ранее программой, но их обе можно держать в памяти компьютера одновременно.

Для того, чтобы выполнить отдельно только последнюю программу, надо ввести команду:

```
RUN 100
```

Эта программа вводит строку символов, что указывается строковыми кавычками, если их опустить, то компьютер попытается найти переменную с таким именем и использовать ее значение в качестве INPUT-данных. Например, ответьте программе

N$ (удалив кавычки)

Это сделает оператор INPUT в строке 110 подобным оператору

```
LET N$=N$
```

Если Вы решили ввести STOP под строковый ввод, то должны установить курсор в начало строки, используя клавишу управления курсором <влево>.

Действие команды 'RUN 100' подобно действию оператора 'GO TO', но имеются и различия. RUN 100 очищает все переменные и экран, и после этого выполняет GO TO 100. Другое отличие в том, что Вы можете указать RUN без номера строки, и тогда выполнение начинается с первой строки, а оператор GO TO всегда должен содержать номер строки.

Обе приведенные программы останавливались нами вводом команды STOP, но могут быть программы, которые невозможно остановить подобным образом, например:

```
200 GO TO 200
    RUN 200
```

Остановить эту программу можно, если нажать клавиши CAPS SHIFT и SPACE, это вызовет ввод команды BREAK, которая остановит вычисления с выдачей сообщения 'L BREAK INTO PROGRAM'. Команда BREAK может быть использована и во время выполнения операций с магнитофоном или принтером, в этом случае выдается сообщение 'D BREAKCONT REPEATS'. Команда CONTINUE в этом случае (как и в большинстве других), вызовет повторение оператора, в котором произошел останов. Ввод команды CONTINUE после сообщения 'L BREAK INTO PROGRAM' продолжит выполнение со следующего оператора.

Запустите вторую программу снова и, когда она запросит ввод, введите:

N$ (удалив ковычки)

Поскольку значение 'N$' не определено, то будет выдано сообщение: '2 VARIABLE NOT FOUND'. Если теперь Вы выполните:

```
LET N$="SOMETHING DEFINITE"
```

На что компьютер ответит '0 OK, 0:1', а затем введете CONTINUE, то увидете, что программа завершится нормально.

Как уже отмечалось, сообщение 'L BREAK INTO PROGRAM' особое, так как выдача после него CONTINUE не вызывает повторение команды, вызвавшей останов.

Все приведенные утверждения PRINT, LET, INPUT, RUN, LIST, GO TO, CONTINUE,

NEW и REM могут быть использованы либо как операторы в программе, либо как команды. Хотя RUN, LIST, CONTINUE и NEW чаще используются как команды, но могут быть использованы и в программе.

6.1. УСЛОВИЯ

Краткое содержание: IF, STOP, = , > , < , <= , >= , <>

Последовательность выполнения операторов программы не всегда предсказуема, в определенных местах программы компьютер может принимать решение о дальнейшем ходе вычислений. Оператор, реализующий это, имеет форму:

IF - некоторое истинное или ложное выражение, THEN - некоторое действие.

Например, выполните команду NEW, а затем наберите и выполните программу (это игра для двух человек):

```
10 REM GUESS THE NUMBER (угадывание числа)
20 INPUT A:CLS
30 INPUT "GUESS THE NUMBER",B
40 IF A=B THEN PRINT "THIS IS CORRECT":STOP
50 IF B<A THEN PRINT "THIS IS TOO SMALL, TRUE AGAIN"
60 IF B>A THEN PRINT "THIS IS TOO BIG, TRUE AGAIN"
70 GOTO 30
```

Здесь оператор IF имеет форму:

IF условие THEN ...

Где '...' - последовательность операторов, разделенных двоеточием обычным образом. Если 'условие' истинно, то выполняются операторы, следующие после THEN, в противном случае они пропускаются и выполнение программы продолжается со следующего оператора.

Простейшим условием может быть сравнение двух чисел или двух строк. Числа могут быть либо равны, либо одно больше другого, а строки либо равны, либо одна следует после другой в алфавитном порядке. Для задания условия используются отношения:

= , < , > , <= , >= , <> .

Например, выражение 1<2 , -2<1 , -3<1 истинны, а выражения 1<0, 0>2 ложны.

Строка программы 40 сравнивает числа 'A' и 'B', и, если они равны, завершает работу, выполняя команду STOP. При этом будет выдано сообщение '9 STOP, STATEMENT, 40:3', показывающее, что команда STOP была выдана в 3-ем операторе в 40-й строке.

Знаки условия набирают на клавиатуре следующим образом:

>	- SYMBOL SHIFT вместе с T	- больше
<	- SYMBOL SHIFT вместе с R	- меньше
<=	- SYMBOL SHIFT вместе с Q	- меньше или равно (нельзя набирать < и =)
>=	- SYMBOL SHIFT вместе с E	- больше или равно
<>	- SYMBOL SHIFT вместе с W	- не равно

6.2. ЦИКЛЫ

Краткое содержание: FOR, NEXT, TO, STEP

Допустим, нам необходимо составить программу, подсчитывающую сумму вводимых пяти чисел. Это можно было бы сделать так:

```
10 LET TOTAL=0
20 INPUT A
30 LET TOTAL=TOTAL+A
40 INPUT A
50 LET TOTAL=TOTAL+A
60 INPUT A
70 LET TOTAL=TOTAL+A
80 INPUT A
90 LET TOTAL=TOTAL+A
100 INPUT A
110 LET TOTAL=TOTAL+A
120 PRINT TOTAL
```

Получилась большая и не очень оптимальная программа. Можно решить эту

задачу более рационально, если ввести счетчик и оператор GO TO:

```
10 LET TOTAL=0
20 LET COUNT=1
30 INPUT A
40 REM COUNT=NUMBER OF TIME THAT A HAS BEEN INPUT SO FAR
50 LET TOTAL=TOTAL+A
60 LET COUNT=COUNT+1
70 IF COUNT<=5 THEN GO TO 30
80 PRINT TOTAL
```

Теперь, изменив условие в строке 70, можно ввести не только 5, но и любое количество чисел, для организации в программе таких счетчиков существуют специальные операторы FOR и NEXT, которые всегда используются вместе.

Наша программа при использовании этих операторов будет выглядеть так:

```
10 LET TOTAL=0
20 FOR C=1 TO 5
30 INPUT A
40 REM C=NUMBER OF TIMES THAT A HAS BEEN INPUT SO FAR
50 LET TOTAL=TOTAL+A
60 NEXT C
70 PRINT TOTAL
```

Здесь 'C'- управляющая переменная цикла должна иметь имя в одну букву. 'C' последовательно принимает значения 1, 2, 3, 4 и 5 (предел - конечное значение управляющей переменной цикла) и при каждом проходе выполняются строки 30, 40 и 50. Затем после того, как 'C' примет пятое значение, выполнится 70-я строка. Приращение значения управляющей переменной составляет 1, но это значение можно изменить, используя указание STEP как часть оператора FOR. Итак, общая форма оператора FOR выглядит следующим образом:

FOR 'упр.перем.'='нач.знач.' TO 'предел' STEP 'шаг прираш.' Здесь 'начальное значение', 'предел', 'шаг прирощения' - есть выражения, принимающие числовое значение. Если Вы замените строку 20 программы на

```
20 FOR C=1 TO 5 STEP 3/2
```

то 'C' последовательно примет значения 1, 2.5 и 4.

Выполните программу, выводящую числа от 1 до 10 в убывающей последовательности

```
10 FOR N=10 TO 1 STEP -1
20 PRINT N
30 NEXT N
```

Следующая программа выводит числа домино:

```
10 FOR M=0 TO 6
20 FOR N=0 TO 6
30 PRINT M;":";N;" ";
40 NEXT N
50 PRINT
60 NEXT M
```

Значение STEP, равное 0, вызовет бесконечное повторение цикла, этого не рекомендуется делать.

6.3. ПОДПРОГРАММЫ

Краткое содержание: GO SUB, RETURN

Иногда бывает удобно некоторые фрагменты программы представить в виде отдельных частей, по нескольку раз используемых в различных местах программы. Такие части оформляются как подпрограммы, которые могут вызываться в любом месте программы.

Для этого используются операторы GO SUB (GO TO SUBROUTINE) и RETURN в форме:

```
GO SUB N
```

где 'N' номер первой строки в подпрограмме. Этот оператор подобен GO TO N с той разницей, что при использовании GO SUB компьютер запоминает следующий после GO SUB оператор, которому и передается управление после выполнения подпрограммы. Делается это посредством помещения номера оператора (адреса возврата) в специальную область памяти, называемую GO SUB-стек.

RETURN выбирает верхний адрес возврата из GO SUB стека и продолжает выполнение программы с оператора, следующего после оператора с этим номером.

Приведем пример использования подпрограммы:

```
100 LET X=10
110 GO SUB 500
120 PRINT S
130 LET X=X+4
140 GO SUB 500
150 PRINT S
160 LET X=X+2
170 GO SUB 500
180 PRINT S
190 STOP
500 LET S=0
510 FOR y=1 TO X
520 LET S=S+y
530 NEXT y
540 RETURN
```

В общем случае, подпрограмма может вызывать другие подпрограммы и даже саму себя (такая подпрограмма называется рекурсивной).

6.4. ОПЕРАТОРЫ READ, DATA и RESTORE

Краткое содержание: READ, DATA, RESTORE

В некоторых предыдущих программах мы видели, что информация или данные могут быть введены в компьютер при помощи оператора INPUT. Иногда это может быть очень утомительно, особенно если многие данные повторяются каждый раз при выполнении программы. Вы можете сэкономить много времени, используя команды READ, DATA и RESTORE.

Например:

```
10 READ A,B,C
20 PRINT A,B,C
30 DATA 10,20,30
40 STOP
```

Оператор READ состоит из слова READ, за которым следует список имен переменных, разделенных запятыми. Он выполняется значительно эффективнее оператора INPUT, особенно когда вводимые значения присваиваются переменным. В этом случае компьютер ищет величины переменных в утверждении DATA.

Каждый оператор DATA - это список значений (числовых или строчных), разделенных запятыми. Вы можете вводить их в программе, где Вам угодно, так как компьютер игнорирует их, за исключением тех случаев, когда их использует оператор READ.

Сначала компьютер выбирает первое значение из списка DATA для величины из оператора READ. В следующий раз он берет второе значение из списка DATA, и, таким образом, выбираемые последовательно операторы READ обрабатываются с использованием списка DATA. (Если оказалось, что список просмотрен до конца, то возникает ошибка). Заметьте для себя, что является бесполезным введение оператора DATA в виде управляющего оператора, так как оператор READ не обнаружит его. Оператор DATA выполняется только в программе.

Можно посмотреть, как эти требования выполнены в вышеприведенной программе:

строка 10 дает указание компьютеру читать три значения данных и присвоить их переменным A, B и C, строка 20 говорит о том, что надо их вывести (PRINT)

в строке 30 оператор DATA задает значения для A, B и C, строка 40 заканчивает программу.

Информация в операторе DATA может быть частью FOR...NEXT цикла.

Например:

```
10 FOR N=1 TO 6
20 READ D
30 DATA 2,4,6,8,10,12
40 PRINT D
```

```
50 NEXT N
60 STOP
```

Когда выполняется эта программа, то можно увидеть, как READ оператор перебирает весь список DATA.

Оператор DATA может содержать также и строчные переменные.

Например:

```
10 READ D$
20 PRINT "THE DATE IS",D$
30 DATA "JUNE 1ST,1982"
40 STOP
```

Это простой способ получения выражений из DATA списка: старт и выполнение до тех пор, пока не будет достигнут конец, однако Вы можете использовать и программный переход для DATA списков. В этом случае используется оператор RESTORE с указанием после него номера строки с оператором DATA, и все последовательно встречающиеся в программе операторы READ вводят данные подряд, начиная с первого оператора DATA под указанным номером. Вообще-то Вы можете не указывать номера строки в операторе RESTORE, и, в этом случае, указатель данных становится на первый оператор в программе.

Попробуйте выполнить такую программу:

```
10 READ A,B
20 PRINT A,B
30 RESTORE 10
40 READ X,Y,Z
50 PRINT X,Y,Z
60 DATA 1,2,3
70 STOP
```

В этой программе переменным, вводимым в строке 10, будут присвоены значения A=1 и B=2. Оператор RESTORE 10 сбрасывает указатель данных в начальное положение и строка 40 присвоит значения переменным X,Y и Z начиная с первого значения в DATA.

Выполните программу без строки 30 и Вы увидите сами, что из этого получится.

6.5. АРИФМЕТИЧЕСКИЕ ОПЕРАЦИИ

Краткое содержание: операции +, -, *, /
выражения, усл. обозначения, имена переменных

Вы уже видели несколько примеров, в которых ZX SPECTRUM может оперировать числами. Можно выполнять четыре арифметических операции: +, -, * и / (помните, что * используется для умножения, а / используется для деления), и при этом определяется значение переменной, задаваемой именем.

Пример:

```
LET TAX = SUM*15/100
```

Отсюда видно, что вычисления могут быть комбинированными. Комбинации такого типа, как

```
SUM*15/100
```

называются выражениями. Выражение - это самый короткий путь для указания компьютеру на то, что вычисления надо делать одно за другим. В нашем примере выражение

```
SUM*15/100
```

указывает: возьми значение переменной с именем "SUM", умножь его на 15 и затем раздели на 100. Если Вы не можете еще этого сделать, мы рекомендуем посмотреть вводную часть этой книги, чтобы ознакомиться с тем, как ZX SPECTRUM работает с числами и каков порядок, в котором выполняются математические выражения.

Краткое повторение:

Умножение и деление выполняются первыми. Они имеют более высокий приоритет, чем сложение и вычитание.

Относительно друг друга умножение и деление имеют равные приоритеты. Существует правило, по которому умножение и деление выполняются последовательно слева направо. Когда все они выполнятся, то затем будут выполняться сложение

и вычитание по порядку (также слева направо).

Для задания приоритета в компьютере ZX SPECTRUM используются числа в интервале от 1 до 16. Например, операции '*' и '/' имеют приоритет 8, а '+' и '-' - 6. Этот порядок вычислений является жестким, но его можно изменить при помощи скобок. Выражение в скобках вычисляется первым, а затем подставляется в общее выражение, как одно число.

Вы можете использовать операцию сложение ('+') для сцепления строк (конкатенции) в выражениях.

Имя строковой переменной состоит из буквы с последующим знаком '$', имя управляющей переменной в FOR-NEXT цикле должно состоять из одной буквы, а имена обычных числовых переменных могут выбираться произвольно. Они могут содержать несколько букв и цифр, но первой всегда должна быть буква.

Вы можете вставлять в имена пробелы для удобства чтения, поскольку компьютер не считает их частью имени. Запись имени прописными или заглавными буквами не делает их различными.

Примеры допустимых имен переменных:

```
X
T42
THIS NAME IS SO LONG THAT I SHALL NEVER BE ABLE TO TYPE
     IT OUT AGAIN WITHOUT
MAKING A MISTAKE
```

Примеры недопустимых имен переменных:

```
2001                  (начинается с цифры)
3 BEARS               (начинается с цифры)
M*A*S*H               (знак '*' - не буква и не цифра)
FOTHERINGTON-THOMAS (содержит знак '-')
```

Числа в выражениях могут задаваться в экспоненциальной форме. Попробуйте выполнить:

```
PRINT 2.34E0
PRINT 2.34E1
PRINT 2.34E2      и т.д до
PRINT 2.34E15
```

Помните, что оператор PRINT дает лишь числа в значащих цифрах. Попробуйте выполнить еще:

```
PRINT 4294967295, 4294967295-429E7
```

и Вы увидете, что компьютер может воспринять только цифры 4294967295.

Компьютер ZX SPECTRUM использует арифметику с плавающей точкой, при этом различные части числа (мантисса и порядок) хранятся в отдельных байтах, что приводит к не всегда точным результатам даже для целых чисел. Выполните:

```
PRINT 1E10 + 1 - 1E10, 1E10 - 1E10 + 1
```

1E10 и 1E10 + 1 не различаются компьютером как разные числа (1E10 усекается справа).

Еще один, более наглядный пример:

```
PRINT 5E9 + 1 - 5E9
```

Погрешность в 5E9 составляет около 1, а с прибавлением единицы фактически округлится до 2.

Числа 5E9 + 1 и 5E9 + 2 для компьютера равны. Наибольшее целое, которое может воспринять компьютер, равно 2**32 - 1 или (4 294 967 295).

Строка "" без единого символа называется пустой или нулевой строкой. Не путайте ее с пробелом. Наберите:

```
PRINT "HAVE YOU FINISHED "FINNEGANS WAKE" YET?"
```

Когда Вы нажмете клавишу ENTER, Вы получите мерцающий знак вопроса, указывающий ошибочное место в строке. Когда, при интерпретации этой строки, компьютер найдет двойную кавычку, открывающую "FINNEGANS WAKE", то сочтет ее закрывающей кавычкой для строки "HAVE YOU FINISHED" и затем не сможет вывести "FINNEGANS WAKE". Здесь надо помнить специальное правило: если Вы хотите вывести кавычки внутри строки, они должны удваиваться. Например:

```
PRINT "HAVE YOU FINISHED""FINNEGANS WAKE"" YET?"
```

В данном случае будет выведено только фраза, обрамленная двойными кавычками "FINNEGANS WAKE" .

6.6. СТРОКИ СИМВОЛОВ

Краткое содержание: сечения, использование TO

Примечание: эти операции отсутствуют в стандартном Бейсике.

Пусть имеется строка символов, тогда ее подстрокой будет некоторая последовательность символов из этой строки, так "STRING" является подстрокой от "BIGGER STRING", а "W STRING" и "BIG STRING" не являются.

Существует действие, называемое сечением для определения подстрок и которое может применяться к строковым выражениям. Общая его форма:

'СТРОКОВОЕ ВЫРАЖЕНИЕ' ('начало' TO 'конец')

Следующее выражение истинно:

"ABCDEF" (2 TO 5) = "BCDE".

Если опущено 'начало', то по умолчанию подразумевается 1, если - 'конец', то подразумевается длина всей строки. так:

```
"ABCDEF"(TO 5) = "ABCDEF"(1 TO 5) = "ABCDE"
"ABCDEF"(2 TO) = "ABCDEF"(2 TO 6) = "BCDEF"
"ABCDEF"( TO ) = "ABCDEF"(1 TO 6) = "ABCDEF"
```

Последнее выражение можно было бы записать и так:

```
"ABCDEF"(),
```

что тоже верно. Можно опускать и слово TO:

```
"ABCDEF"(3) = "ABCDEF"(3 TO 3) = "C"
```

'Начало' и 'конец' должны находиться в пределах строки, иначе будет выдано сообщение об ошибке. Так выражение

```
"ABCDEF"(5 TO 7)
```

вызывает сообщение '3 SUBSCRIPT WRONG' , так как 'конец' превышает длину строки (6).

Если 'начало' больше, чем 'конец', либо обе границы лежат за пределами строки, то результатом будет пустая строка:

```
"ABCDEF"(8 TO 7) = ""
"ABCDEF"(1 TO 0) = ""
```

'Начало' и 'конец' не могут быть отрицательными, иначе выдается сообщение 'B INTEGER OUT OF RANGE'.

Следующая программа иллюстрирует эти правила:

```
10 LET A$ = "ABCDEF"
20 FOR N=1 TO 6
30 PRINT A$(N TO 6)
40 NEXT N
50 STOP
```

Можно также присваивать значения подстроке. Попробуйте:

```
10 LET A$ = "I AM THE ZX SPECTRUM"
20 PRINT A$
30 LET A$(5 TO 8) = "******"
40 PRINT A$
```

Подстрока A$(5 TO 8) имеет длину только в 4 символа, поэтому будут использованы только первые четыре звездочки. Это особенность присвоения значения подстроке: длинные данные усекаются справа, а короткие дополняются пробелами до длины подстроки. Это действие называют "прокрустианом" в честь мифического разбойника Прокруста, который своим гостям либо отрубал ноги, либо вытягивал их, если они не подходили по длине к его кровати. Если Вы теперь выполните

```
LET A$() = "HELLO THERE" и
PRINT A$","
```

Вы увидете, что будут выведены дополнительные пробелы, так как 'A$()' считается подстрокой. Для правильного выполнения следует писать:

```
LET A$ = "HELLO THERE".
```

Можно использовать скобки, что позволяет вычислять значение строкового выражения перед тем, как брать сечение. Например:

```
"ABC" + "DEF"(1 TO 2) = "ABCDE"
("ABC" + "DEF")(1 TO 2) = "AB" .
```

6.7. ФУНКЦИИ

Краткое содержание: DEF, LEN, STR$, VAL, SGN,
ABS, INT, SQR, FN

Функции - это зашитые в BASIC-систему подпрограммы, которые, получая на входе одни значения, называемые аргументами, возвращают другие значения - результаты. Функции используются в выражении простым включением в него имени функции с последующими аргументами.

При вычислении выражения, вычисляется и значение функции. Например, функция LEN возвращает длину заданного в ней строкового аргумента. Вы можете записать:

```
PRINT LEN "SINCLAIR" ,
```

а компьютер выведет ответ '8', т.е. количество букв в слове 'SINCLAIR' (для ввода с клавиатуры имени функции LEN, Вы должны войти в необходимый режим, нажав клавиши CAPS SHIFT и SYMBOL SHIFT, курсор изменится с [L] на [E], и нажать клавишу K).

Если в одном выражении используются и функции и операции, то функции будут вычислены перед выполнением любых операций. Однако, Вы можете изменить этот порядок, применяя скобки.

Функция STR$ преобразует число в символьный вид, подобный формату вывода чисел оператором PRINT:

```
LET A$ = STR$ 1E2
```

аналогично по действию команде

```
LET A$ = "100"
```

или выполните

```
PRINT LEN STR$ 100.0000
```

и получите ответ 3, так как STR$ 100.0000 ="100" .

Функция VAL обратится К функции STR$ и преобразует строку в число. Так,

```
VAL "3.5" = 3.5
```

или

```
VAL "2*3" = 6
```

или даже так

```
VAL ("2" + "*3") = 6
```

В последнем случае происходит вычисление двух выражений, сначала строкового с получением строки "2*3", затем числового с получением строки "6".

Можно попасть в затруднительное положение, например:

```
PRINT VAL "VAL" "VAL" " " "2" " " " " " " "
```

Помня, что внутри строки кавычки удваиваются, мы видим, что в нашем случае может понадобиться учетверение или даже увосьмирение.

Имеется еще одна функция подобная VAL - это VAL$. И аргументом, и результатом этой функции является строка символов. Она работает как VAL, примененная дважды, раскрывая все кавычки в строках:

```
VAL$" " "FRUIT PUNCH" " " = "FRUIT PUNCH"
```

Сделайте

```
LET A$ = "99"
```

и затем выведете все следующие значения:

```
VAL A$
VAL "A$"
VAL " " "A$" " "
VAL$ A$
VAL$ "A$"
VAL$ " " " A$" " "
```

Некоторые из них сработают, а некоторые нет, проанализируйте все ответы.

Функция SGN - это так называемая математическая функция сигнум (знак). и аргумент и результат ее числовые. Результат равен:

1 , если аргумент положителен;
0 , если аргумент равен 0;
-1 , если аргумент отрицателен.

Функция ABS преобразует аргумент в положительное число:

```
ABS -3.2 = ABS 3.2 = 3.2
```

Функция INT (от 'INTEGER PART' - целая часть) преобразует дробное число к

целому отбрасыванием дробной части:

```
INT 3.9 = 3
```

Сложности возникают при отрицательном аргументе, так как округление происходит к ближайшему целому, не большему, чем аргумент:

```
INT -3.9 = -4
```

Функция SQR вычисляет корень квадратный от числа, например:

```
SQR 4 = 2
SQR 0.25 = 0.5
SQR 2 = 1.4142136 (приближенно).
```

Если аргумент отрицательный, то выдается сообщение:

```
'A INVALID ARGUMENT'.
```

Вы также можете сами определить для себя какую-нибудь функцию, указав FN и имя этой функции (букву, если аргумент числовой или букву и $, если строковый). Аргументы должны быть обязательно заключены в скобки.

Вы можете определить функцию вводом оператора DEF в некотором месте программы. Например, зададим функцию вычисляющую квадрат числа:

```
10 DEF FN S(X) = X*X: REM THE SQUARE OF X
```

DEF вводится в соответствующем режиме (SIMBOL SHIFT и 1). Теперь функция может использоваться в программе:

```
PRINT FN S(2)
PRINT FN S(3+4)
PRINT 1 + INT FN S(LEN "CHICKEN"/2 + 3)
```

Функция INT всегда округляет до целого; для округления с точностью 0.5 надо добавить к результату '.5' . Вы можете задать для себя такую функцию:

```
20 DEF FN R(X) = INT(X + 0.5): REM GIVES X ROUNDED TO
                  THE NEAREST INTEGER.
```

И можете затем попробовать ввести:

```
FN R(2.9) = 3              FN R(2.4) = 2
FN R(-2.9)= -3             FN R(-2.4) = -2
```

Введите и выполните следующее:

```
10 LET X=0: LET Y=0: LET A=10
20 DEF FN P(X,Y)=A + X*Y
30 DEF FN Q()=A + X*Y
40 PRINT FN P(2,3), FN Q()
```

Есть одна тонкость в этой программе: во-первых, функция FN Q не использует аргументов, но скобки при этом должны обязательно использоваться; во-вторых, операторы DEF не выполняемые, компьютер после выполнения строки 10, просто переходит к выполнению строки 40. Помните, что DEF может быть только оператором, но не командой; в третьих, 'X' и 'Y' - имена целых переменных в программе и в то же время имена аргументов в функции FN P. Функция FN P использует в вычислении результата значения аргументов 'X' , 'Y' и переменной 'A', не являющейся аргументом. Так, когда вычисляется FN P(2,3), значение 'A' равно 10, как и определено в программе, а значения 'X' и 'Y' соответственно 2 и 3, так как они - аргументы, и результат будет 10+2*3 = 16. При вычислении FN Q() участвуют только переменные программы, так как аргументов нет, и ответ в этом случае будет 10 + 0*0=10. Теперь изменим строку 20 на

```
20 DEF FN P(X,Y) = FN Q()
```

В этом случае FN P(2,3) будет возвращать значение 10.

Некоторые версии Бейсика имеют функции LEFT$ и RIGHT$, TL$:

```
LEFT$ (A$,N) - возвращает подстроку, содержащую 'N'
               первых символов строки 'A$';
RIGHT$(A$,N) - возвращает подстроку, содержащую 'N'
               последних символов в строке 'A$';
TL$(A$)      - возвращает подстроку, содержащую все
               символы строки 'A$', кроме первого.
```

Вы можете определить такие функции на своем компьютере:

```
10 DEF FN T$(A$) = A$(2 TO): REM TL$
20 DEF FN L$(A$,N) = A$(TO N): REM LEFT$
```

Проверьте их работу со строками длиной 0 и 1.

П Р И М Е Ч А Н И Е. Функция может иметь до 26 числовых аргументов и в

то же время, до 26 строковых.

6.8. МАТЕМАТИЧЕСКИЕ ФУНКЦИИ

Краткое содержание: **,PI,EXP,LN,SIN,COS,TAN,ASN,ACS,ATN

В этой главе описываются математические функции, которые могут быть выполнены на ZX SPECTRUM. Вполне возможно, что Вам никогда не придется воспользоваться ими и, если Вы сочтете их слишком сложными, можете пропустить эту главу. Все сказанное относится к функциям: ** (возведение в степень), EXP, LN, тригонометрическим функциям: SIN, COS, TAN, и обратных к ним: ASN, ACS, ATN.

**** и EXP**

Вы можете возвести число в некоторую степень путем многократного умножения его самого на себя необходимое число раз. Это обычно изображается записью числа, обозначающего степень, справа вверху от числа, обозначающего основание. Но такую форму записи трудно реализовать в компьютере, поэтому там используют специальный символ (направленная вверх стрелка, в данном случае замененная двумя звездочками: '**'). Например, степени двойки можно представить так:

```
2 ** 1 = 2
2 ** 2 = 2 * 2 = 4      (два в квадрате)
2 ** 3 = 2 * 2 * 2 = 8 (два в кубе)
```

Таким образом, запись A**B означает: 'умножь 'A' само на себя 'B' раз'. Но это предполагает, что B положительное целое число.

Для нахождения определения для этого действия при других значениях A и B, мы запишем выражение:

```
A ** (B + C) = A ** B * A ** C
```

Здесь надо помнить, что операция '**' имеет более высокий приоритет, чем '*' и '/'. Вы можете быть уверены в правильности этого выражения если 'B' и 'C' целые положительные числа, но если это не так, а вы все-таки решили выполнить возведение в степень, то вы должны знать, что:

```
A ** 0 = 1
A **(-B) = 1/(A ** B)
A **(1/B) = корню B-ой степени из A
A **(B*C) = (A ** B) ** C
```

Полезно помнить, что:

```
A **(-1) = 1/A
A **(1/2) = SQR A
```

Поэкспериментируйте с этим, попробовав выполнить такую программу:

```
10 INPUT A,B,C
20 PRINT A**(B+C),A**B * A**C
30 GO TO 10
```

Компьютер станет выводить два числа, если Вы, правильно конечно, набрали программу. Число A, кстати, не должно быть отрицательным.

Другой типичный пример использования этой операции -это вычисление дохода. Предположим, что Вы вложили часть своих денег в общественное строительство, которое приносит Вам 15% годовых. После года Вы будете иметь уже не только 100% от того, что имели в начале, а плюс 15% дохода, что составит 115%. Для вычисления другим способом, Вы умножаете вашу сумму денег на 1.15 и получаете тот же результат. В конце следующего года Вы снова получите прибыль, что в сумме составит 1.15 * 1.15 = 1.15 ** 2 = 1.3225 от Вашей первоначальной суммы. В итоге после Y лет Вы будете иметь в 1.15 ** Y раз больше денег.

Выполнив операторы:

```
FOR Y=0 TO 100:PRINT Y,10*1.15**Y:NEXT Y
```

Вы увидите, что начиная с 10 фунтов, можно получать все больший и больший доход с капитала.

Такой тип поведения функции, когда после фиксированного числа интервалов времени, значения функции пропорциональны количеству умножений этого числа самого на себя, называется экспоненциальным законом.

Предположим Вы записали:

```
10 DEF FN A(X)=A**X
```

здесь A определено в операторе LET, его значение передается для вычисления

степени.

Имеется определенное значение A,которое делает функцию FN A иллюстрирующей специальную математическую функцию. Это значение называется 'E'.

ZX SPECTUM имеет специальную функцию, называемую EXP и определяемую как:

```
EXP X = E ** X
```

К сожалению, 'E' не может быть представлено точным числом. Вы можете увидеть пять его первых десятичных знаков, выполнив

```
PRINT EXP 1
```

так как EXP 1 = E ** 1 = E. Конечно, это лишь первое приближение, Вы никогда не сможете записать 'E' абсолютно точно.

LN

Обратной к экспоненциальной является логарифмическая функция. Логарифм (по основанию A) числа X есть степень, в которую надо возвести A, чтобы получить X, это записывается так: LOGA X. (Выражение A**LOGA X=X так же верно как и LOGA(A**X)=X)

Вам должно быть уже известно,как используется логарифм по основанию 10 для умножения. Такой логарифм называется общим. ZX SPECTRUM имеет функцию LN, которая вычисляет логарифм по основанию 'E', называемый натуральным. Для вычисления логарифма с другим основанием, надо разделить натуральный логарифм искомого числа на натуральный логарифм основания:

```
LOGA X = LN X/LN A
```

PI

Допустим имеется некоторый круг. Вы можете найти его периметр (длину окружности), умножив его диаметр на число, называемое PI. Подобно 'E', PI представляется бесконечной десятичной дробью. Его начало:

```
3.141592653589...
```

Слово 'PI' в ZX SPECTRUM обозначает это число. Выполните, например:

```
PRINT PI
```

SIN, COS, TAN и ASN, ACS, ATN

Тригонометрические функции измеряют те случаи, когда точка перемещается вокруг окружности единичного радиуса. Точка стартует с позиции "3-х" часов и перемещается против часовой стрелки. Начало координат находится в центре этой окружности. Тогда SIN угла между радиусом, соединяющим движущуюся по окружности точку с началом координат, будет ордината этой точки, а COS - абсциса. Необходимо помнить, что если точка находится слева от оси Y, то косинус отрицацельный, а если точка находится под осью X, то отрицацельный синус. Необходимо помнить, что:

```
SIN (A + 2*PI) = SIN A
COS (A + 2*PI) = COS A
```

имеются и другие тригонометрические функции:

```
TAN - тангенс;
ASN - арксинус;
ACS - арккосинус;
ATN - арктангенс.
```

Помните, в ZX-SPECTRUMe тригонометрические функции вычисляются в радианах. Для перевода из градусов в радианы необходимо число разделить на 180 и умножить его на PI, а для обратного преобразования необходимо разделить на PI и умножить на 180.

6.9. СЛУЧАЙНЫЕ ЧИСЛА

Краткое содержание: RANDOMIZE, RND.

В этой главе описывается функция RND и ключевое слово RANDOMIZE. Их не надо путать, хотя они обе расположены на клавише 'T'. Для RANDOMIZE допустимо сокращение RAND.

При обращении к функции RND, она возвращает случайное число в интервале от 0 до 1 (может принимать значение 0, но никогда 1). Попробуйте выполнить:

```
10 PRINT RND
20 GO TO 10
```

вы увидете как меняется результат.

Фактически RND не абсолютно случайное число, а выбирается из определенной

последовательности длиной в 65536 чисел, поэтому обычно говорят, что RND - псевдослучайное число. Для получения случайного числа в интервале отличном от 0...1 можно использовать выражения, например:

```
1.3 + 0.7 * RND
```

даст интервал от 1.3 до 2.

Для получения случайных целых чисел используйте функцию INT (округляет с отбрасыванием дробной части). Например:

```
1 + INT(RND * 6)
```

будет давать числа 1, 2, 3, 4, 5, 6.

Пусть имеется программа:

```
10 REM THROWING PROGRAM     (выбрасывание кости)
20 CLS
30 FOR N=1 TO 2
40 PRINT 1+INT(RND*6);' ';
50 NEXT N
60 INPUT A$:GO TO 20
```

нажимая ENTER, Вы каждый раз будете получать номер, выпавший на кости.

Утверждение RANDOMIZE используется для установления начала последовательности случайных чисел для функции RND. Как можно увидеть из программы:

```
10 RANDOMIZE 1
20 FOR N=1 TO 5:PRINT RND,:NEXT N
30 PRINT:GO TO 10
```

после каждого выполнения RANDOMIZE 1 случайная последовательность будет начинаться с числа 0.00222735596. В утверждении RANDOMIZE Вы можете использовать любые числа в интервале от 1 до 65535. Нельзя использовать RANDOMIZE без числа, а также RANDOMIZE 0. Например, имеется программа:

```
10 RANDOMIZE
20 PRINT RND:GO TO 10
```

В каждой итерации будет печататься не случайное число. Для улучшения случайности распределения можно заменить GO TO 10 на GO TO 20.

Большинство версий Бейсика используют RND и RANDOMIZE для генерации случайных чисел, но это не единственное их применение.

Ниже приводится текст программы, моделирующей выбрасывание монеты и подсчета числа выпадений 'орла' и 'решки'. (Перевод имен программы: HEADS-орлы, TAILS-решки, COIN-монета)

```
10 LET HEADS=0:LET TAILS=0
20 LET COIN=INT(RND*2)
30 IF COIN=0 THEN LET HEADS=HEADS+1
40 IF COIN=1 THEN LET TAILS=TAILS+1
50 PRINT HEADS;',';TAILS
60 IF TAILS<>0 THEN PRINT HEADS/TAILS;
70 PRINT:GO TO 20
```

если программа выполняется достаточно долго, то отношение 'орлов' к 'решкам' приблизительно равно 1.

6.10. МАССИВЫ

Краткое содержание: DIM.

Допустим, у Вас имеется список из чисел, каким-то образом описывающих 10 человек. Для записи их в память компьютера, Вы должны будете завести переменную на каждого человека. Это не удобно, так как приходится обращаться к данным, называя каждый раз новую переменную, например BLOGGS1, BLOGGS2 и т.д. до BLOGGS10. Как это неудобно, Вы можете убедиться из программы:

```
 5 REM THIS PROGRAMM WILL NOT WORK
10 FOR N=1 TO 10
20 READ BLOGGSN
30 NEXT N
40 DATA 10,2,5,19,16,3,11,1,0,6
```

Имеется специальный аппарат для подобного случая, - это применение массивов. Переменные в массиве являются его элементами, обладают общим именем и

различаются только номером, записываемым после имени (индексом).

В нашем примере имя будет B (подобно управляющим переменным в FOR-NEXT утверждениях, имя массива должно быть уникальным в данной программе), а десятью переменными будут B(1),B(2) и т.д. до B(10).

Элементы массивов называются индексируемыми переменными. Перед использованием массива необходимо зарезервировать под него память, это делается в операторе DIM (от английского DIMENSION). В нашем случае это будет оператор DIM B(10), который определяет массив с именем B и размерностью 10 (т.е. 10 индексируемых переменных B(1), B(2),...,B(10)) и присваивает всем элементам массива значение 0.

Итак, теперь мы можем записать:

```
10 FOR N=1 TO 10
20 READ B(N)
30 NEXT N
40 DATA 10,2,5,19,3,11,1,0,6
```

Можно также обьявлять массивы с более, чем одной размерностью. Например, в двумерном массиве первый индекс можно сравнить с номером строки, а второй - с позицией в строке. Такой массив как бы описывает страницу. Если ввести третье измерение для номера страницы, то массив будет описывать книгу в виде:

(номер страницы, номер строки, номер столбца).

Обьявим двумерный массив 'C' с размерностью 3 и 6:

```
DIM C(3,6)
```

что даст 3*6=18 индексируемых переменных:

```
C(1,1) , C(1,2) , ... . , C(1,6)
C(2,1) , C(2,2) , . . . , C(2,6)
C(3,1) , C(3,2) , . . . , C(3,6)
```

Могут быть также строковые массивы. Строки в таких массивах отличаются от скалярных тем, что имеют фиксированную длину, а присваивание им значения осуществляется с усечением справа или добавлением до полной длины пробелами. Имя строкового массива образуется добавлением справа к имени специального символа, перечеркнутой буквы S (знак доллара '$').

Допустим Вам необходимо обьявить массив A$ на 5 строк по 10 символов в каждой, Вы должны записать:

```
DIM A$(5,10)
```

теперь Вы можете обращаться как целиком к отдельной строке, так и к каждому символу в строке:

```
A$(1)=A$(1,1)A$(1,2)...A$(1,10)
A$(2)=A$(2,1)A$(2,2)...A$(2,10)
.........
A$(5)=A$(5,1)A$(5,2)...A$(5,10)
```

можно также рассматривать элемент строкового массива как массив символов. Пусть обьявлен массив A$(2,7). Можно записать и так A$(2)(7). Следующая программа:

```
10 LET A$(2)="1234567890"
20 PRINT A$(2),A$(2,7)
```

даст '1234567 7'

Можно использовать также сечения массивов:

```
A$(2,4 TO 8)=A$(2)(4 TO 8)='45678'
```

Помните, что в строковых массивах все строки имеют фиксированную длину. Эту длину определяет последнее число размерности массива в операторе DIM. Если обьявлен одномерный массив, то он определяет массив символов: DIM A$(10).

6.11. ЛОГИЧЕСКИЕ ОПЕРАЦИИ

Краткое содержание: AND, OR, NOT.

Если мы взглянем на описанную ранее форму оператора IF:

```
IF условие THEN
```

то увидим, что 'условие' описывается отношениями (=,<,>,>=,<=, <>), связывающими два числа или строки. Здесь можно также использовать логические операции AND(и), OR(или) и NOT(не).

Некоторое выражение 'и' некоторое другое выражение истино, если истины

оба этих выражения. Например:

```
IF A$='YES' AND X>0 THEN PRINT X
```

'X' будет напечатано только тогда, когда
A$='YES' и X>0

Некоторое выражение 'или' некоторое другое выражение истинно, если истинно хотя бы одно из этих выражений. 'Не' выражение истинно, если ложно само выражение и наоборот.

OR имеет низший приоритет, затем идет AND, затем NOT.

Условие '<>' обратно в логическом смысле условию '=', то есть:

A<>B тоже, что и NOT A=B
NOT A<>B тоже, что и A=B

Тем, кто боится сложностей, следующие разделы можно опустить.

1. Условия =,<,>,<=,>=,<> дают числовой результат 1 для истины и 0 если ложь. Например, оператор PRINT 1=2, 1<>2 выведет 0 для '1=2', которое ложно, и 1 для '1<>2', которое истино.

2. В операторе "IF условие THEN ...", само условие может быть числовым выражением. Если значение после вычисления равно 0, то считается, что это ложь, если другое значение (включая и 1), то считается, что это истина. Таким образом IF-оператор можно представить:

```
IF условие <> 0 THEN ...
```

Операции AND, OR, NOT могут также использоваться и в числовых выражениях:

X AND Y имеет значение X, если Y<>0 и 0, если Y=0
X OR Y имеет значение 1, если Y<>0 и X, если Y=0
NOT Y имеет значение 0, если Y<>0 и 1, если Y=0

например:

```
10 INPUT A
20 INPUT B
30 PRINT(A AND A>B)+(B AND A<B)
40 GO TO 10
```

в каждой операции будет выводиться большее из двух чисел A или B.

Пример использования OR:

```
LET TOTAL PRICE=PRICE LESS TAX*(1.15 OR V$='ZERO RATER'
```

В условном выражении можно также использовать символьные строки, но только с операцией AND:

X$ AND Y имеет значение X$, если Y^=0, и " ", если Y=0
где " " - пустая строка.

Выполните следующую программу, которая вводит две строки, а затем выводит их в алфавитном порядке:

```
10 INPUT "TYPE IN TWO STRING" A$,B$
20 IF A$>B$ THEN LET C$=A$:LET A$=W$:LET B$=C$
30 PRINT  A$;"  ";("<"  AND  A$<B$)+("="  AND  A$=B$)
40 PRINT " ";B$
50 GO TO 10
```

6.12. НАБОР СИМВОЛОВ

Краткое содержание: CODE, CHR$, POKE, PEEK, USR, BIN.

Буквы, цифры, знаки пунктуации обозначаются символами и образуют алфавит или набор символов, используемый компьютером. Отдельные символы, называемые знаками, образуют целые слова, например:

PRINT,STOP и т.д.

Компьютер ZX SPECTRUM использует 256 символов с кодами от 0 до 255. Все они приведены в приложении 1. Для преобразования из символьной формы во внутреннюю кодовую и наоборот служат две функции CODE и CHR$.

CODE применяется к строке символов и возвращает код внутреннего представления первого символа в строке или 0, если строка пустая.

CHR$ применяется к числу и возвращает один символ, код которого представлен этим числом.

Следующая программа выводит весь отображаемый символьный набор:

```
10 FOR A=32 TO 255:PRINT CHR$ A;:NEXT A
```

Все эти символы (кроме знака фунта и '©' в кружочке) образуют код ASCII

(AMERICAN STANDART CODES FOR INFORMATION INTERCHANGE).

Следующие символы не входят в ASCII, но используются в ZX SPECTRUM. Первые из них это 15 черно-белых значков, называемых графическими символами и используемых для изображения рисунков. Их можно ввести с клавиатуры, используя так называемый "графический режим". Если Вы нажмете GRAPHICS(CAPS SHIFT 9), то курсор изменится на <G>. Теперь цифровые клавиши с 1 по 8 выдают графические символы, обозначенные на клавишах, а если при этом удерживать SHIFT, то они будут выдавать инверсные символы, т.е.черное становится белым, а белое - черным.

Независимо от SHIFT, клавиша с цифрой '9' обеспечивает Вам возврат к обычному (<L>-курсор) режиму, а клавиша '0' - функцию DELETE.

После графических символов на клавиатуре располагаются символы алфавита от A до Z. Графические значения этих клавиш могут определяться самим пользователем, а затем использоваться в графическом режиме. Определение графики этих клавиш проиллюстрируем на примере определения символа буквы греческого алфавита 'ПИ'.

1. Каждый символ представляется точками в матрице 8 X 8, поэтому вначале начертим диаграмму,приведенную на рисунке. По периметру символа оставим по одной клетке для отделения его от других знаков.

2. Закрепим этот символ за клавишей 'P', так, чтобы при нажатии клавиши в графическом режиме выдавался символ 'ПИ'.

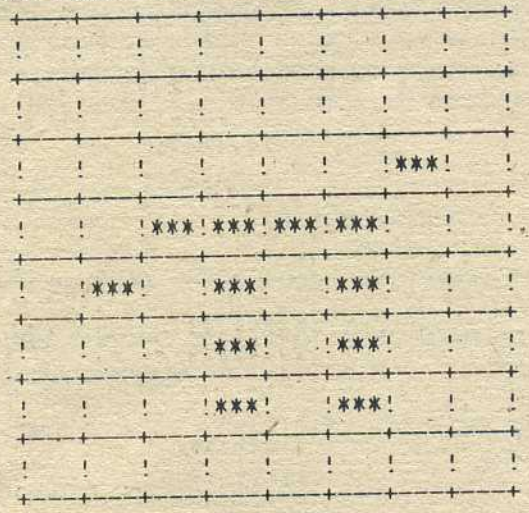


Рис. 5

3. Запрограммируем это изображение. Каждый определяемый пользователем символ запоминается в памяти восмью знаками, по одному на каждый ряд. Можно записать их используя функцию BIN с обозначением цифрой 0 чистой точки и 1 - закрашенной точки.

```
BIN 00000000
BIN 00000000
BIN 00000010
BIN 00111100
BIN 01010100
BIN 00010100
BIN 00010100
BIN 00000000
```

Эти восемь двоичных чисел запоминаются в памяти в восьми ячейках, каждая из которых имеет свой адрес, для нашего символа адрес первого из восьми байтов в группе будет USR "P". Второй байт имеет адрес USR "P"+1 и т.д. до USR "P"+7.

USR - функция преобразования строки символов в адрес первого байта в строке. Строковый аргумент может содержать единственный символ, который будет обозначать символ, определяемый пользователем. Имеются и другие функции применения USR с числовым аргументом, мы рассмотрим их позже.

Поясним все сказанное программой:

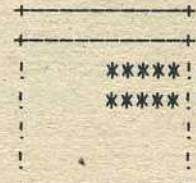
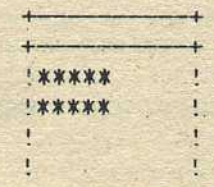
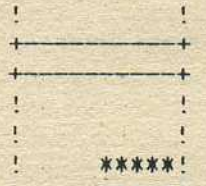
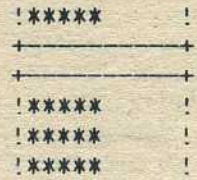
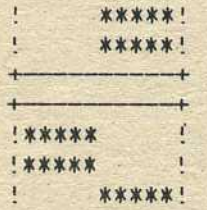
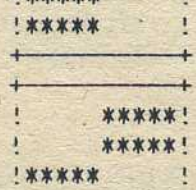
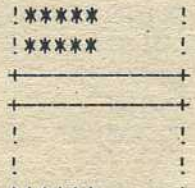
```
10 FOR N=0 TO 7
20 INPUT ROW: POKE USR "P"+N,ROW
```

```
30 NEXT N
```

Данная программа вводит 8 двоичных чисел, определяющих графику символа, закрепляемого за клавишей 'P'.

Оператор 'POKE' записывает данные непосредственно в память, минуя обычный аппарат Бейсика. Обратным оператору 'POKE' является оператор 'PEEK', который служит для отображения содержимого ячейки памяти.

6.13. ГРАФИЧЕСКИЕ СИМВОЛЫ

СИМВОЛ	код	набор	СИМВОЛ	код	набор
	128	<G> 8		143	<G> SH8
	129	<G> 1		142	<G> SH1
	130	<G> 2		141	<G> SH2
	131	<G> 3		140	<G> SH3
	132	<G> 4		139	<G> SH4
	133	<G> 5		138	<G> SH5
	134	<G> 6		137	<G> SH6
	135	<G> 7		136	<G> SH7

Вернемся к знакам. Первые 32 знака с кодами от 0 до 31 - это управляющие символы. Они не отображаются, вместо них на экране телевизора отображается знак

'7'. Назначение этих символов описанно в приложении 1.

Три символа с кодами 6, 8 и 13 имеют специальное назначение при работе с экраном.

CHR$ 6 печатает пробел, используемый как запятая в операторе PRINT.

PRINT 1;CHR$ 6;2 даст тот же результат что и PRINT 1,2.

Но это не совсем корректное ипользование, правильнее будет написать:

```
10 LET A$="1"+CHR$ 6+"2"
20 PRINT A$
```

CHR$ 8 - символ забоя, обеспечивает возврат на одну позицию назад. Оператор PRINT "1234";CHR$ 8;"5" даст строку: '1235'.

CHR$ 13 - перевод строки, продолжает вывод с новой строки. При работе с экраном используются также символы с кодами 16 и 23, которые подробнее будут рассмотрены ниже.

Все символы расположены в кодовой таблице в алфавитном порядке по возрастанию кодов. Причем все прописные буквы расположены после заглавных.

Существует правило по которому сравнивают две строки. Сначала сравниваются первые символы, если они различаются, то строка, содержащая символ с меньшим кодом считается меньшей, а если они равны, то выбирается следующая пара символов, и так до тех пор, пока не встретятся несовпадающие символы, либо пока одна из строк не кончится - она и будет считаться меньшей. В противном случае строки считаются равными.

Отношения =, <, >, <=, >=, <> применяются к строкам символов так же как и к числам. Знак "<" означает "находится впереди в кодовой таблице", а '>' - "находится позади".

Для иллюстрации всего сказанного приведем программу, которая вводит две строки, а затем выводит их в упорядоченном виде.

```
10 INPUT "TYPE IN TWO STRING", A$, B$
20 IF A$>B$ THEN LET C$=A$: LET A$=B$: LET B$=C$
30 PRINT A$;" ";
40 IF A$<B$ THEN PRINT "<";: GO TO 60
50 PRINT "="
60 PRINT " ";B$
70 GO TO 10
```

Следующая программа закрепляет определенные пользователем символы для игры в шахматы за клавишами:

```
P - за пешкой;     (POWN)
R - за ладьей;     (ROOK)
N - за конем;      (KNIGHT)
B - за слоном;     (BISHOP)
K - за королем;    (KING)
Q - за королевой.  (QUEEN)
 5 LET B=BIN 01111100:LET S=BIN 00111000:
   LET D=BIN 00010000
10 FOR N=1 TO 6:READ P$:REM 6 PIECES
20 FOR F=0 TO 7:REM READ PIECE INTO 8 BYTES
30 READ A:POKE USR P$+1,A
40 NEXT F
50 NEXT N
100 REM BISHOP
110 DATA "B",0,D,BIN 00101999,BIN 01000100
120 DATA BIN 01101100,C,B,0
130 REM KING
140 DATA "K",0,D,C,D
150 DATA C,BIN 01000100,C,0
160 REM ROOK
170 DATA "R",0,BIN 01010100,B,C
180 DATA C,B,B,0
190 REM QUEEN
200 DATA "Q",0,BIN 01010100,BIN 00101000,D
210 DATA BIN 01101100,B,B,0
```

```
220 REM PAWN
230 DATA "P",0,0,D,C
240 DATA C,D,B,0
250 REM KNIGHT
260 DATA "N",0,D,C,BIN 01111000
270 DATA BIN 00011000,C,B,0
```

6.14. ДОПОЛНИТЕЛЬНЫЕ СВЕДЕНИЯ ОБ ОПЕРАТОРАХ PRINT и INPUT

Краткое содержание: CLS, PRINT-параметры: их отсутствие вообще, выражение(числовое или строковое): TAB числовое выражение, AT числовое выражение, PRINT разделители:",";";"'", INPUT-параметры, LINE - строчная переменная, свертка, SCREEN$.

Выражения, значение которых используются в операторе PRINT, называются PRINT-параметрами. Они разделяются запятыми или точкой с запятой, называемыми PRINT-разделителями.

В операторе PRINT возможно отсутствие некоторых параметров. В этом случае ставятся две запятые подряд.

AT 'строка','столбец'

этот параметр перемещает позицию вывода в место, определяемое номером строки и столбца. Номер строки меняется от 0 (верхняя) до 21, а номер столбца от 0 (левый) до 31.

```
PRINT AT 11,16;"*"
```

выведет '*' в середине экрана.

Оператор SCREEN$. Его действие противоположно действию оператора PRINT AT, он использует те же параметры, номер строки и столбца, но их значения заключаются в скобки. Оператор SCREEN$ сообщает Вам, какой символ находится на экране в указанной позиции. Выполнив:

```
PRINT SKREEN$ (11,16)
```

мы получим '*' выведенную предыдущим оператором.

В качестве возвращаемого значения могут использоваться: алфавитно-цифровые символы, специальные символы, пробелы. Линии, нарисованные с помощью операторов: PLOT, DRAW, CIRCLE, определяемые пользователем символы и графические символы, возвращаются, как пустая строка. То же, когда функция OVER используется для получения комбинированных знаков.

TAB столбец

этот параметр перемещает позицию вывода в указанный столбец на той же строке, или переходит на новую строку, если столбец был последним.

Помните, что компьютер уменьшает номер позиции по модулю 32 (т.е. делит на 32 и использует остаток). Так 'TAB 33' равнозначно 'TAB 1'.

К примеру:

```
PRINT TAB 30,1;TAB 12;"CONTENTS";AT 3,1;"CHAPTER";
TAB 24;"PAGE"
```

выведет на экран оглавление для первой страницы книги.

Рассмотрим пример, иллюстрирующий уменьшение по MOD 32:

```
10 FOR N=0 TO 20
20 PRINT TAB 8*N;N
30 NEXT N
```

Более наглядный пример получится при замене в 20 строке '8' на '6'.

Несколько замечаний:

1) В рассмотренных примерах в качестве ограничителей использовалась ';'. Можно использовать ',' (или вообще ничего). При этом необходимо следить за установкой текущей позиции вывода.

2) Нельзя использовать для вывода две нижние строки экрана (22,23) т.к. они используются для получения оператором INPUT данных. Последняя используемая строка - 21.

3) Можно использовать параметр AT для установки позиции вывода в то место, где уже имеется выведенная информация, при этом каждый новый символ уничтожает

старый.

Еще одним оператором, используемым совместно с PRINT, является CLS, он производит очистку экрана, подобно операторам CLEAR и RUN.

При заполнении всего экрана происходит его свертка. В этом можно убедиться проделав:

```
CLS:FOR N=1 TO 22:PRINT N:NEXT N
```

и далее выполнить 'PRINT 99' некоторое количество раз.

Или вариант с ОСТАНОВОМ вывода, для просмотра текста. Чтобы убедится в этом выполним:

```
CLS:FOR N=1 TO 100:PRINT N:NEXT N:
```

когда экран заполнится, вывод остановится и в нижней части экрана появится запрос: "SCROLL ?". После просмотра нажмите 'Y' (да) и вывод продолжится. Возможен отрицательный ответ 'N' (нет), STOP (SYMBOL SHIFT + 'A') или SPACE (BREAK). В последнем случае компьютер остановит программу и выдаст сообщение: '0 BREAK-CONT REPEATS'.

Оператор INPUT используется для ввода различных значений. Например:

```
INPUT "HOW OLD ARE YOU?",AGE
```

компьютер выведет на экран (в нижней части) вопрос, в ответ на который Вы должны ввести свой возраст. Фактически INPUT содержит те же параметры, что и PRINT, так "HOW OLD ARE YOU?" и 'AGE' оба являются операторами INPUT. Однако существует и некоторые отличия.

Первое: это дополнительный параметр-переменная, значение которой Вы должны присвоить(в нашем примере 'AGE').

Второе: Вы можете выводить значение переменной, как часть запроса, заключив ее для этого в скобки.

Пример:

```
LET MY AGE=INT(RND*100):INPUT("I AM",MY AGE;".");
"HOW OLD ARE YOU?",YOUR AGE
```

Значение 'MY AGE' выдает компьютер, значение 'YOUR AGE' вводите Вы сами, по мере выдачи операторов INPUT происходит свертка экрана. Рассмотрим пример использования AT в INPUT-операторе:

```
10 INPUT"THIS IS LINE 1",A$;AT 0,0;"THIS LINE IS 0",A$;
   AT 2,0;"THIS IS LINE 2",A$;AT 1,0;
   "THIS IS STILL LINE 1";A$
```

когда "THIS IS LINE 2" будет выведено, нижние строки станут сдвигаться вверх, освобождая место, но нумерация останется прежней.

Выполним:

```
10 FOR N=0 TO 19:PRINT AT N,0;N;NEXT N
20 INPUT AT 0,0;A$;AT 1,0;A$;AT 2,0;A$;AT 3,0;A$;
   AT 4,0;A$;AT 5,0;A$;
```

Когда информация начнет смещаться в область действия операторов PRINT произойдет свертка экрана. Еще одним параметром оператора INPUT является LINE, он предназначен для ввода строчных переменных. Рассмотрим пример:

```
INPUT LINE A$
```

Если ввести какую-либо строчную переменную без строковых кавычек, то ее значение будет присвоено A$. Заметим, что мы не можем использовать параметр LINE для числовых переменных.

Управляющие символы CHR$22 и CHR$23 выполняют функции, подобные параметрам TAB и AT. Их преимущество состоит в том, что можно использовать имена переменных, а для TAB и AT это невозможно. Эти управляющие символы обрабатываются как числа. Аналогом AT является управляющий символ CHR$22. первое значение определяет строку, второе столбец.

```
PRINT CHR$22+CHR$1+CHR$0;
```

то же, что и

```
PRINT AT 1,0;
```

как значения параметров рассматриваются только CHR$1 и CHR$0 (CHR$22 не учитывается).

Аналогом TAB является управляющий символ CHR$23. Значения задаваемых им параметров находятся в пределах от 0 до 65535

```
PRINT CHR$23+CHR$A+CHR$B
```

то же, что и

```
PRINT TAB A+256*B
```

Вы можете использовать POKE для отключения свертки, выполнив :

```
POKE 23692,255
```

компьютер станет сворачивать экран без запроса 255 раз, прежде чем запросит свертку. Так например запустите:

```
10 FOR N=0 TO 10000
20 PRINT N:POKE 23692,255
30 NEXT N
```

и следите сколько сверток сделает компьютер. Запустите следующую программу, проверяющую знание таблицы умножения:

```
10 LET M$=" "
20 LET A=INT(RND*12)+1:LET B=INT(RND*12)+1
30 INPUT(M$)' '"WHAT IS";(A);"*";(B);"?";C
100 IF C=A*B THEN LET M$="RIGHT": GOTO 20
111 LET M$="WRONG.TRY AGAIN.": GOTO 30
```

Можно несколько изменить программу, так чтобы не зная правильного ответа, можно было узнать его. К примеру, компьютер спрашивает, сколько будет 2*3. Не зная ответа, Вы вводите 2*3 и получаете его. Для этого замените в 30 строке 'C' на 'C$', в 100 строке на 'VAL C$' и дополнительно введите строку:

```
40 IF C$< >STR$ VAL C$ THEN LET M$="TYPE IT PROPERLY,
   AS NUMBER.":GOTO
```

для исключения подсказки поменяйте 'C$' в строке 30 на 'LINE C$'.

6.15. ЦВЕТА

Краткое содержание: INK, PAPER, FLASH, BRIGHT, INVERSE, OVER, BORDER

Выполним следующую программу :

```
10 FOR M=0 TO 1:BRIGHT M
20 FOR N=1 TO 10
30 FOR C=0 TO 7
40 PAPER C:PRINT C"   ";:REM 4 COLOURED SPACES
50 NEXT C:NEXT N:NEXT M
60 FOR M=0 TO 1:BRIGHT M:PAPER 7
70 FOR C=0 TO 3
80 INK C:PRINT C;"   ";
90 NEXT C:PAPER 0
100 FOR C=4 TO 7
110 INK C:PRINT C;"   ";
120 NEXT C:NEXT M
130 PAPER 7:INK 0:BRIGHT 0
```

Она продемонстрирует Вам возможности вывода компьютером ZX SPECTRUM на цветной телевизор восьми цветов (включая черный и белый) и двух уровней яркости. Если телевизор черно-белый, Вы увидите различные градации серого цвета.

Ниже дана кодировка цветов:

0-черный
1-синий
2-красный
3-фиолетовый
4-зеленый
5-голубой
6-желтый
7-белый

Для черно-белого телевизора этот ряд представляет собой последовательность перехода серых полутонов от черного до белого. Для использования цветов уясним строение графического экрана. Он состоит из 768 позиций (24 строки по 32 знакоместа), каждая из которых представ-ляет из себя матрицу 8 на 8 пикселей. Вспомним:

0 - белая точка;
1 - черная точка.

Позиция символа (знакоместо) также рассматривается с этих позиций:

INK-цвет тона, PAPER-цвет фона, т.о. знакоместо состоит из INK и PAPER обычной и повышенной яркости, а так же мерцающих и немерцающих. Все это имеет следующую кодировку:

1) для знакоместа (8 на 8 пикселей) форму символа определяют чистые и закрашенные точки (0 и 1), цвета фона и тона определяются PAPER и INK.

2) цвета фона и тона кодируются от 0 до 7 каждый.

3) яркость: 0-обычная, 1-повышенная.

4) мерцание: 0-постоянно, 1-мерцание.

Заметим, что для одного знакоместа в 64 пикселя мы не можем установить более одного цвета для фона и одного цвета для тона. Это же относится и к яркости , и к мерцанию. Цвет, яркость и мерцание задаются для знакоместа (а не для отдельного пикселя) и являются его атрибутами. Для изменения этих атрибутов предназначены операторы: INK, PAPER, BRIGHT, FLASH. Выполним:

```
PAPER 5
```

теперь вывод будет осуществляться на голубой фон (т.к. 5-код голубого цвета).

Формат операторов:

```
PAPER число от 0 до 7
INK число от 0 до 7
BRIGHT 0 или 1   / 0-выкл. 1-вкл.
FLASH  0 или 1   / 0-выкл. 1-вкл.
```

Отметим, что использование чисел, больших, чем указывалось выше, допустимо, но дает другой эффект. К примеру "8" может использоваться во всех четырех операторах как средство, позволяющее определить значение ранее установленных атрибутов. Так,

```
PAPER 8
```

не изменит цвета фона (т.к. такого цвета нет), а поможет выяснить значение предыдущего PAPER. Операторы: INK 8, BRIGHT 8, FLASH 8 выдадут значения этих атрибутов. "9" может использоваться только для INK и PAPER, как средство "контраста". Цвета INK и PAPER, которые Вы используете, должны быть контрастны друг другу. Так к белому цвету подходят темные тона: черный, синий, красный, фиолетовый; к черному цвету подходят светлые тона: зеленый, голубой, желтый, белый.

Выполним:

```
INK 9:FOR C=0 TO 7:PAPER C:PRINT C:NEXT C
```

Можно запустить программу, выдающую на экран дисплея цветные полосы:

```
INK 9:PAPER 8:PRINT AT 0,0;:FOR N=1 TO 1000:
PRINT N;:NEXT N
```

Цвет фона будет контрастен цвету тона в каждой выводимой позиции. Цветной телевизор построен на способности человеческого глаза воспринимать только три первичных цвета - синий, красный и зеленый. Другие цвета образуются из их сочетаний. К примеру, фиолетовый цвет образуетя, как комбинация синего с красным (код фиолетового цвета '3', он является суммой кодов синего '1' и красного '2'). Видеть все восемь цветов на одном участке экрана невозможно, т.к. это будет темное пятно. Но там, где цвета частично накладываются друг на друга мы увидим цветовую гамму. В качестве примера выполним программу (отметим, что INK получена с использованием SHIFT и 8 в [G]-режиме).

```
 10 BORDER 0:PAPER 0:INK 7:CLS
 20 FOR A=1 TO 6
 30 PRINT TAB 6;INK 1;"[][]...[][]":REM 18 INK SQUARES
 40 NEXT A
 50 LET DATALINE=200
 60 GO SUB 1000
 70 LET DATALINE=210
 80 GO SUB 1000
 90 STOP
200 DATA 2,3,7,5,4
210 DATA 2,2,6,4,4
1000 FOR A=1 TO 6
1010 RESTORE DATALINE
1020 FOR B=1 TO 5
```

```
1030 READ C:PRINT INK C;"[]...[]";:REM 6 INK SQUARES
1040 NEXT B:PRINT :NEXT A
1050 RETURN
```

Существует функция ATTR, позволяющая определить,какие атрибуты были заданы для позиции экрана. Это сложная фынкция, и она будет рассмотрена в конце главы.

Операторы: INVERSE и OVER не управляют атрибутами, но тем не менее определяют способ вывода на экран. В этих операторах используются значения параметров '0' и '1'. Если Вы дадите: INVERSE 1 , то выводимый символ изменит свою обычную форму (вывод будет осуществляться в негативном изображении).

В обычном виде мы пишем черным по белому, в инверсном белым по черному.

Оператор: OVER 1 устанавливает режим расширенного вывода. В обычном режиме при выводе символа на знакоместо, там стирается все выведенное ранее, при расширенном выводе можно накладывать символы друг на друга. Это позволяет выводить составные символы, например стилизованные шрифты.

Программа для вывода готического шрифта:

```
10 OVER 1
20 FOR N=1 TO 32
30 PRINT "0";CHR$8;"""";
40 NEXT N
```

Отметим, что управляющий символ CHR$8 возвращает на одну позицию.

Возможен еще один способ использования INK и PAPER. Их можно вводить как параметры PRINT. Точно так же можно использовать и другие операторы, рассмотренные в этой главе, отметив при этом, что их действие распространяется только до конца PRINT.

В примере:

```
PRINT PAPER 6;"X";:PRINT "Y"
```

только 'X' будет выведен на желтый фон.

Расписание клавиатуры верхнего ряда в различных режимах.

	[K],[L],[C]	[G]	[E]	
				PEEK
	SYMBOL CAPS	любой	CAPS SYMBOL	PEEK
1	EDIT	фон голубой	тон голубой	DEF FN
2	CAPS LOCK	фон красный	тон красный	FN
3	TRUE VIDEO	фон фиалетовый	тон фиалетовый	LINE
4	INVERS VIDEO	фон зеленый	тон зеленый	OPEN

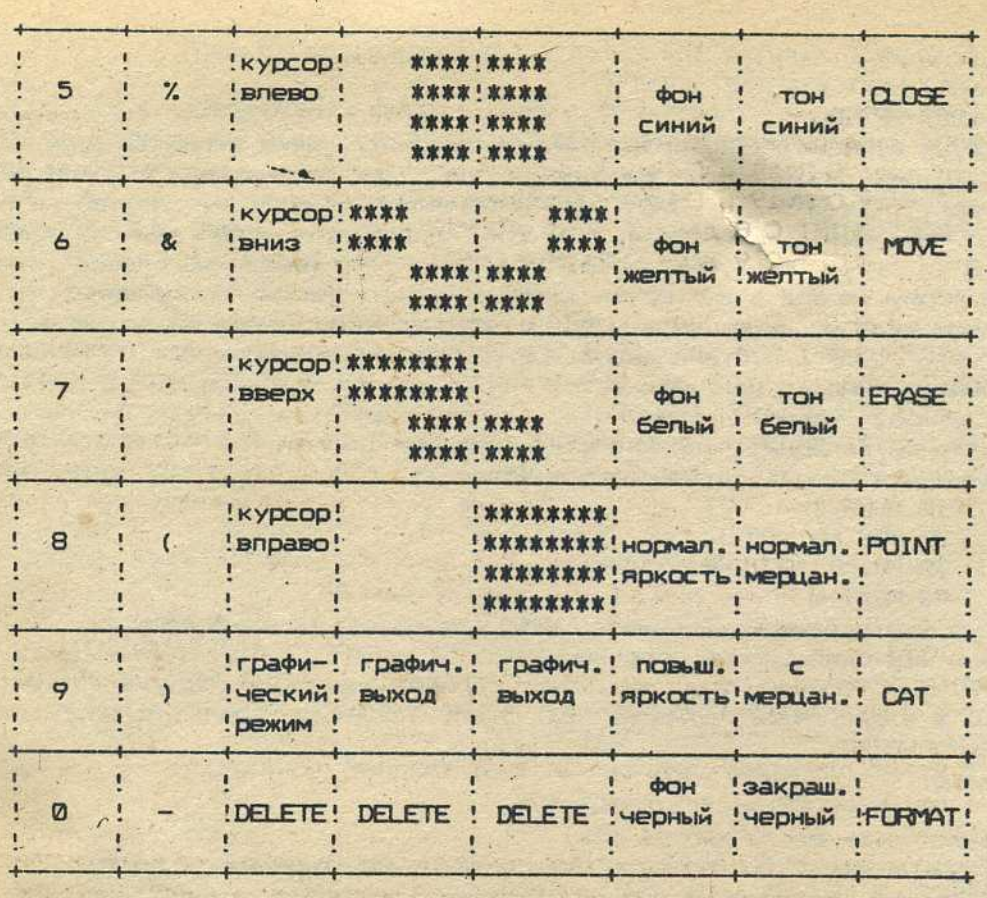


Рис. 6

INK и другие операторы не действуют в нижней части экрана, предназначенной для ввода команд и INPUT-данных. Для изменения цветов в этой части экрана служит оператор:

BORDER 'цвет'

Кодировка цветов прежняя. Возможны мерцание и повышенная яркость, для этого используйте соответствующие параметры в INPUT (наподобие PRINT). Эти параметры действуют до конца оператора или до тех пор, пока запрашиваемые данные не будут введены. Выполним:

```
INPUT FLASH 1;INK 1;"WHAT IS yOUR NUMBER ?";N
```

Возможно изменение цветов и с помощью управляющих символов, подобных управляющим символам для AT и TAB.

```
CHR$16 соответствует INK
CHR$17    —//—     PAPER
CHR$18    —//—     FLASH
CHR$19    —//—     BRIGHT
CHR$20    —//—     INVERSE
CHR$21    —//—     OVER
```

так

```
PRINT CHR$16+CHR$9;
```

то же, что и

```
PRINT INK 9;
```

Можно пользоваться либо управляющими символами, либо операторами. Их можно ставить как после номера строки, так и в конце строки. Для удобства можно пользоваться в расширенном режиме [E] цифрами (см.рис). Цифры от 0 до 7 устанавливают цвет INK, если CAPS SHIFT нажата, и цвет PAPER, если не нажата. Если нажать цифру в [E]-режиме,то будут выведены CHR$17 и CHR$ (код цвета). Если в это время была нажата CAPS SHIFT, то будут выведены:CHR$16 и CHR$ (код цвета). Если Вы захотите уничтожить вводимое, нажмите DELETE два раза, после первого

тех пор. пока курсор не выйдет к предыдущему управляющему символу.

Действия в расширенном режиме [E]:

8 дает CHR$19 и CHR$0 - нормальная яркость

9 дает CHR$19 и CHR$1 - повышенная яркость

CAPS SHIFT C 8 дает CHR$18 и CHR$0 - не мерцающее

CAPS SHIFT C 9 дает CHR$18 и CHR$1 - мерцающее

В [L]-режиме:

CAPS SHIFT C 3 дает CHR$20 и CHR$0 - обычный вывод

CAPS SHIFT C 4 дает CHR$20 и CHR$1 - инверсный (негативный) вывод.

Функция ATTR имеет следующий формат:

ATTR ('строка','столбец')

Значения двух параметров функции подобны значению параметров в AT. В результате выполнения будут выведены значения атрибутов для соответствующей позиции экрана. Выводимый результат - это число, представляющее сумму четырех чисел:

1) 128 - если знакоместо мерцающее, 0 - если обычное
2) 64 - если повышенная яркость, 0 - если обычная
3) 8 - код цвета фона
4) код цвета тона

Пример: знакоместо мерцающее, обычной яркости, желтый фон, синий тон

128+0+(8*6)+1=177

проверим это выполнив:

```
PRINT AT 0,0; FLASH 1; PAPER 6; INK 1;" "; ATTR(0,0)
```

Упражнения:

1) PRINT "B"; CHR$8; OVER 1;"/";

здесь '/' перечеркнет 'B'. Этим способом можно выводить слова из комбинированных знаков на ZX SPECTRUM: два фона или два тона дают фон, один из них дает тон. Это интересное свойство. Если Вы повторите вывод одного символа дважды, то он не будет отображен. Так, если дать:

```
PRINT CHR$8;OVER 1;"/"
```

дополнительно к описанному выше утверждению, то мы в результате увидим 'B' не перечеркнутое '/'. Так ли это?

2) выполним :

```
PAPER 0:INK 0
```

Действуют ли эти операторы в нижней части экрана? что мы увидим если добавить

```
BORDER 0 ?
```

3) выполним программу:

```
10 POKE 22527+RND*704,RND*127
20 GO TO 10
```

Результатом программы явится смена цветов знакомест, распределенных по экрану случайным образом. Возможно Вы увидите смену цветов на диагональных ступенях. Это является следствием того, что мы пользуемся квазислучайным распределением, которое лишь приближенно воспроизводит случайное распределение.

6.16. ГРАФИКА

Краткое содержание: PLOT, DRAW, CIRCLE, POINT

Эта глава описывает возможности компьютера ZX SPECTRUM по отображению графической информации. Экран компьютера содержит 22 строки по 32 символа в каждой, что составляет 22x32=704 символьные позиции. Каждая символьная позиция представляется квадратом 8X8 точек, называемых пикселями.

Пиксель задается двумя числами - его координатами. Первое задает координату X, то есть удаление (в пикселях) до левой границы экрана, второе - задает координату Y (удаление от нижней границы экрана). Координаты записываются в скобках, например: (0,0), (255,0), (0,175) и (255,175). Они задают соответственно нижний левый, нижний правый, верхний правый и верхний левый углы экрана.

Оператор PLOT X,Y вызывает высвечивание закрашивающим цветом (INK) пикселя с указанными координатами.

Например программа:

```
10 PLOT INT(RND*256),INT(RND*176)
```

```
20 INPUT A$
30 GOTO 10
```

будет высвечивать некоторый случайный пиксель при каждом нажатии ENTER.

Есть и более интересные программы. Например, следующая программа вычерчивает график функции SIN X для X в интервале от 0 до 2*PI:

```
10 FOR N=0 TO 255
20 PLOT N,88+80*SIN(N/128*PI)
30 NEXT N
```

Или программа:

```
10 FOR N=0 TO 255
20 PLOT N,80*SQR(N/64)
30 NEXT N
```

которая чертит график SQR X (часть параболы) в интервале от 0 до 4.

Помните, что координаты пикселей отличаются от адресации строк и позиций в подкоманде AT.

Помощь при построении изображений Вам могут оказать операторы DRAW и CIRCLE.

Оператор DRAW чертит линию, заданную в форме

```
DRAW X,Y
```

Началом линии является пиксель, на котором завершился один из предыдущих операторов PLOT, DRAW или CIRCLE (этот пиксель называется текущей PLOT-позицией. Операторы RUN, CLEAR, CLS и NEW устанавливают ее в левый нижний угол экрана). Таким образом оператор DRAW задает длину и направление вычерчивания линии, но не начальную точку.

Поэксперементируйте с такими командами:

```
PLOT 0,100:DRAW 80,-35
PLOT 90,150:DRAW 80,-35
```

Чертить можно также в цвете, но при этом надо иметь в виду, что цвет устанавливается для целой символьной позиции и не может быть задан для отдельного пикселя. Следующая программа демонстрирует это:

```
10 BORDER 0: PAPER 0: INK 7: CLS: REM BLACK OUT SCREEN
20 LET X1=0: LET Y1=0: REM START OF LINE
30 LET C=1: REM FOR INK COLOUR, STARTING BLUE
40 LET X2=INT(RND*256): LET Y2=INT(RND*176): REM RANDOM
   FINISH OF LINE
50 DRAW INK C;X2-X1,Y2-Y1
60 LET X1=X2: LET Y1=Y2: REM NEXT LINE STARTS WHERE LAST
   ONE FINISHED
70 LET C=C+1: IF C=8 THEN LET C=1: REM NEW COLOUR
80 GO TO 40
```

Вы можете использовать в операторах PLOT и DRAW управляющие символы PAPER, INK, FLASH, BRIGHT, INVERSE и OVER так же, как и в операторах PRINT и INPUT. Управляющие символы записываются между ключевым словом и координатами и оканчиваются запятой или точкой с запятой (смотри строку 50).

При помощи DRAW можно также вычертить отрезок дуги, используя для этого дополнительное число, задающее угол (в радианах) этой дуги:

```
DRAW X,Y,A
```

Если 'A' положительно, то дуга вычерчивается влево, а если отрицательно, то вправо. При 'A' равном 2*PI вычерчивается полная окружность. Например:

```
PLOT 100,100:DRAW 50,50,PI
```

вычертит полуокружность с начальной точкой (100, 100) и конечной точкой (150,150). Вычерчивание начнется в направлении юго-восток, а закончится в направлении на северо-запад.

Оператор CIRCLE вычерчивает полный круг задаваемый координатами его центра и радиусом:

```
CIRCLE X,Y,радиус
```

Как и в операторах PLOT и DRAW Вы можете указать в этом операторе различные цвета.

Функция POINT возвращает характеристики цвета заданного пикселя. Например строка программы:

```
CLS:PLOT(0,0):PRINT POINT(0,0)
```

выведет:

```
PAPER 7:INK 0
```

Допускается также задавать управляющие символы INVERSE и OVER в операторе PLOT, по умолчанию они предполагаются равными 0 (отключено), но Вы можете задать и 1, при этом:

PLOT INVERSE 1 - устанавливает для заданного пикселя цвет фона,

PLOT OVER 1 - изменяет цвет пикселя на противоположный, если был цвет тона, то становится цвет фона и наоборот.

PLOT INVERSE 1; OVER 1; - сохраняет цвет пикселя без изменения, но меняет текущую PLOT-позицию.

Другой пример использования OVER с записью черным по белому:

```
PLOT 0,0:DRAW OVER 1;255,175
```

вычерчивает линию по диагонали.

Теперь попробуйте:

```
PLOT 0,0:DRAW INVERSE 1;255,175
```

и перечертите ее командой

```
DRAW OVER 1;-250,-175
```

это не изменит картинку, так как при черчении как вперед, так и назад используются одни и те же пиксели.

Имеется способ получения необычных цветов в одном квадрате, с использованием определяемых пользователем символов. Выполните эту программу:

```
1000 FOR N=0 TO 6 STEP 2
1010 POKE USR"A"+N, BIN 01010101:
     POKE USR"A"+N+1,BIN 10101010
1020 NEXT N
```

Она задает определяемый пользователем символ для шахматной доски, который закрепляется за клавишей 'A'. Для символа используется красный закрашивающий цвет и желтый цвет фона, но на экране этот символ будет казаться оранжевым.

Еще один пример - программа, которая строит график некоторой функции. На первый ее запрос Вы отвечаете числом 'N', задающим область значений аргумента (т.е. график будет строиться для значений аргумента в диапазоне от -N до +N). Второй ответ - это выражение в виде символьной строки, задающей функцию, использующую 'X' в качестве аргумента:

```
10 PLOT 0,87:DRAW 255,0
20 PLOT 127,0:DRAW 0,175
30 INPUT S,E$
40 FOR F=0 TO 255
50 LET X=(F-128)*S/128:LET Y=VAL E$
60 IF ABS Y > 87 THEN LET T=0:GO TO 100
70 IF NOT T THEN PLOT F,Y+88:LET T=1:GO TO 100
80 LET OLDY=INT(Y+5)
100 DRAW 1,Y-OLDY
110 NEXT F
```

Выполните ее, введя 10 для числа 'N' и '10*TAN X' для функции. Будет вычерчен график функции TG X при X, изменяющемся от -10 до +10.

6.17. УКАЗАНИЯ

Краткое содержание: PAUSE, INKEY$, PEEK

Если Вы решили задержать выполнение программы на некоторое время, то Вам следует использовать оператор

```
PAUSE N,
```

который останавливает выполнение программы и отображает картину в течение 'N' телевизионных кадров (50 кадров в сек в Европе или 60 - в Америке). 'N' может быть вплоть до 65535, что составляет 22 минуты. Если N=0, то это означает, что оператор PAUSE не имеет ограничений по времени.

Выполнение программы всегда может быть возобновлено до окончания времени, определенного в операторе PAUSE нажатием любой клавиши.

Пример программы моделирования секундной стрелки часов:

```
10 REM FIST WE DRAW THE CLOCK FACE
```

```
 20 FOR N=1 TO 12
 30 PRINT AT 10-10*COS(N/6*PI),16+10*SIN(N/6*PI);N
 40 NEXT N
 50 REM NOW WE START THE CLOCK
 60 FOR T=0 TO 200000: REM THIS THE TIME IN SECONDS
 70 LET A=T/30*PI: REM A IS THE ANGLE OF THE SECOND
                      HAND IN RADIANS
 80 LET SX=80*SIN A: LET SY=80*COS A
200 PLOT 128,88: DRAW OVER 1;SX,SY: REM DRAW SECOND HAND
210 PAUSE 42
220 PLOT 128,88: DRAW OVER 1;SX,SY: REM ERASE SECOND HAND
400 NEXT T
```

Эти часы остановятся, проработав приблизительно 55,5 часов, что задается в операторе с номером 60. Оператор 210 производит отсчет времени. Казалось бы здесь должен быть оператор PAUSE 50 (Европа), для точного отсчета одной секунды, но тогда бы мы не учли время, затрачиваемое на выполнение остальных операторов программы. Рассматриваемый вариант часов обеспечивает двухпроцентную точность или, иными словами, уход на полчаса в день.

Возможны и более точные способы измерения времени. Для этого можно использовать содержимое специальных областей памяти. В этом случае данные из памяти могут быть вызваны с помощью функции PEEK. В качестве примера рассмотрим выражение:

```
(65536*PEEK 23674 + 256*PEEK 23673 + PEEK 26672)/50
```

оно дает количество секунд, прошедших с тех пор, как компьютер был включен (вплоть до 3-х суток и 21-го часа). Ниже приводится модифицированная программа моделирования часов:

```
 10 REM FIRST WE DRAW THE CLOCK FACE
 20 FOR N=1 TO 12
 30 PRINT AT 10-10*COS(N/6*PI),16+10*SIN(N/6*PI);N
 40 NEXT N
 50 DEF FN T()=INT((65536*PEEK 23674+256*PEEK 23673+
        PEEK 23672)/50): REM NUMBER OF SECOND SINCE START
100 REM NOW WE START THE CLOCK
110 LET T1=FN T()
120 LET A=T1/30*PI: REM A IS THE ANGLE OF THE SECOND
                      HAND IN RADIANS
130 LET SX=72*SIN A: LET SY=72*COS A
140 PLOT 131,91: DRAW OVER 1; SX,SY: REM DRAW HAND
200 LET T=FN T
210 IF T<=T1 THEN GO TO 200: REM WAIT UNTIL TIME FOR
                             NEXT HAND
220 PLOT 131,91: DRAW OVER 1; SX,SY: REM RUB OUT OLD HAND
230 LET T1=T: GO TO 120
```

Эти часы обеспечивают точность 0,01% или уход на 10 секунд в день. Однако, это возможно при условии, что Вы не использовали оператор BEEP, ввод/вывод на магнитофон и принтер. Все эти операции увеличивают погрешность.

Числа PEEK 23674, PEEK 23673 и PEEK 23672 выделяют адреса ячеек памяти компьютера и используемых для подсчета 1/50 долей секунды. В каждой из ячеек подсчитывается сумма от 0 до 255, после достижения величины 255 в любой из ячеек она сбрасывается в 0. Первой начинает отсчитывать ячейка PEEK 23672. Каждую 1/50 секунды ее содержимое увеличивается на 1. Когда в ячейке накопится величина, равная 255, то она сбрасывается в 0, а значение ячейки PEEK 23673 увеличивается на 1. Через каждые 256/50 сек. содержимое этой ячейки переходит из состояния 255 в 0, а содержимое ячейки PEEK 23674 увеличивается на 1.

При значениях 0 для ячейки PEEK 23674 и 255 для ячеек PEEK 23673 и PEEK 23672 (этот момент наступит через 21 минуту) наше выражение примет значение:

```
(65536*0 + 256*255 + 255)/50=1310.7
```

но здесь имеется скрытая опасность. Через следующую 1/50 сек. ячейки будут содержать соответственно следующие значения: 1, 0, 0.

Пока производится вычисление выражения, компьютер может оценить значение

ячейки PEEK 23674 как 0 до завершения циклического переноса. В результате получим:

(65536*0 + 256*0 + 0)/50=0,

что безнадежно неверно.

Простое правило позволяет решить эту проблему: "Следует вычислять выражение дважды в некоторой последовательности и использовать сохраненный ответ".

Пример:

```
10 DEF FN M(X,Y)=(X+Y+ABS(X-Y))/2: REM THE LARGER OF X AND Y
20 DEF FN U()=(65536*PEEK 23674 + 256*PEEK 23673 +
   PEEK 23672)/50: REM TIME, MAY BY WRONG
30 DEF FN T()=FN M(FN U(),FN U()): REM TIME RIGHT
```

Вы можете изменять значения числовых счетчиков так, чтобы получать реальное время того момента, когда компьютер был включен. Например, надо установить 10 часов вечера. Вы посчитали, что это

10*60*60*50 = 18000000 50-х долей секунды и значит

18000000 = 65536*27 + 256*119 + 64.

Для присвоения трем ячейкам значений 27, 119 и 64 необходимо выполнить

```
POKE 23674,27:POKE 23673,119:POKE 23672,64
```

Функция INKEY$ (без аргументов) считывает с клавиатуры. Если Вы нажали некоторую клавишу (или SHIFT и какую-нибудь клавишу), результатом будет символ, который дает эта клавиша в режиме курсора <L>, или пустая строка.

Выполните программу, которая использует эту функцию:

```
10 IF INKEY$ <>" " THEN GO TO 10
20 IF INKEY$=" " THEN GO TO 20
30 PRINT INKEY$;
40 GO TO 10
```

Помните, что функция INKEY$ не будет подобно INPUT ждать Вас. Если Вы не выполните ввод, то считайте, что Ваш шанс упущен.

6.18. ПРОГРАММИРОВАНИЕ ЗВУКОВ

Краткое содержание: BEEP.

ZX SPECTRUM может воспроизводить звуки при помощи оператора BEEP:

BEEP продолжительность, высота звука.

Где 'продолжительность' и 'высота звука' некоторые числовые выражения. Продолжительность задается в секундах, а высота в полутонах от основного тона 'ДО': при положительных числах - выше ноты 'ДО', а при отрицательных - ниже ноты 'ДО'. На диаграмме приведены все значения нот одной октавы:

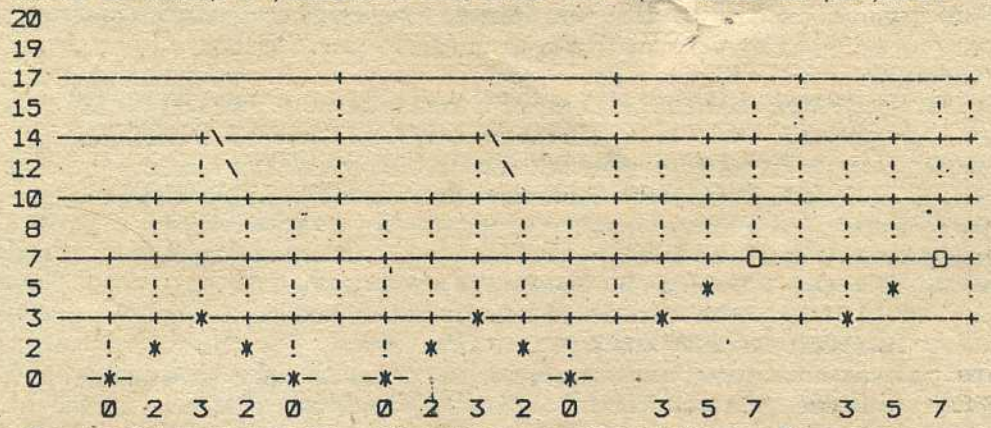
Для получения более высоких или более низких нот, Вы должны прибавить или отнять 12 для каждой октавы вверх или вниз.

Например:

```
10 PRINT "FRERE GUSTAV"
20 BEEP 1,0: BEEP 1,2: BEEP .5,3: BEEP .5,2: BEEP 1,0
30 BEEP 1,0: BEEP 1,2: BEEP .5,3: BEEP .5,2: BEEP 1,0
40 BEEP 1,3: BEEP 1,5: BEEP 2,7
50 BEEP 1,3: BEEP 1,5: BEEP 2,7
60 BEEP .75,7: BEEP .25,8: BEEP .5,7: BEEP .5,5: BEEP .5,3:
   BEEP .5,2: BEEP 1,0
70 BEEP .75,7: BEEP .25,8: BEEP .5,7: BEEP .5,5: BEEP .5,3:
```

```
      BEEP .5,2: BEEP 1,0
   80 BEEP 1,0: BEEP 1,-5: BEEP 2,0
   90 BEEP 1,0: BEEP 1,-5: BEEP 2,0
```

Когда Вы запустите эту программу, Вы услышите похоронный марш из первой симфонии Мольера, ту часть, когда гобблины хоронят рыцаря. Запись начала этой мелодии в ключе до-минор с указанием значений нот приведена на рисунке:



Если Вы желаете исполнить мелодию в другом ключе, Вы должны вставить в выражение некоторую переменную 'KEY'. Например для второй строки программы:

```
   20 BEEP 1,KEY+0: BEEP 1,KEY+2: BEEP .5,KEY+3:
      BEEP .5,KEY+2: BEEP 1,KEY+0
```

Теперь при выполнении программы Вы можете присвоить переменной 'KEY' значения: 0 - для до-минор, 2 - для ре-минор, 12 - для до-минор верхней октавы и т.д. Переменная 'KEY' может также принимать значения кратные 1/2, 1/4 и т.д.

Таким же образом можно изменять и длительность звучания нот. Но помните, что компьютер может одновременно исполнять только одну ноту, что не позволяет воспроизводить сложные мелодии.

Попробуйте запрограммировать собственную мелодию. Начните с самой простой. Если Вы не знаете нотной грамоты, можете изучить ее прямо на компьютере. Например фрагмент программы:

```
FOR N=0 TO 1000: BEEP .5,N: NEXT N
```

будет исполнять последовательно ноты до предельно высокой и завершится с сообщением об ошибке 'B'. Вы можете также параллельно выводить значения 'N', чтобы знать значение исполняемой ноты.

Фрагмент программы:

```
   10 BEEP .5,0: BEEP .5,2: BEEP .5,4: BEEP .5,5: BEEP .5,7:
      BEEP .5,9: BEEP .5,11: BEEP .5,12: STOP
```

исполняет гамму до-минор, в которой используются чистые ноты от среднего 'ДО' до верхнего 'ДО'. Однако в этой гамме неестественные интервалы, скрипач бы исполнил ее так:

```
   20 BEEP .5,0: BEEP .5,2.039: BEEP .5,3.86: BEEP .5,4.98:
      BEEP .5,7.02: BEEP .5,8.84: BEEP .5,10.88: BEEP .5,10.88
      BEEP .5,12: STOP
```

Эти же интервалы будут естественными для гаммы, исполняемой в любом ключе, отличном от 'ДО'.

Некоторая музыка, например индийская, использует интервалы меньшие чем полутон. Вы можете без особого труда запрограммировать это в операторе BEEP. Например, для звука на четверть тона выше среднего 'ДО' надо указать значение высоты звука равное 0.5.

Вы можете сделать клавиатуру компьютера клавишами музыкального инструмента выполнив переключение:

```
POKE 23609,255
```

Второе число здесь определяет продолжительность нахождения в этом состоянии (попробуйте изменять его от 0 до 255).

Можно также вывести музыку на внешние устройства, подключаемые к выходным разъемам 'MIC' и 'EAR'.

6.19. ВНЕШНЯЯ ПАМЯТЬ НА МАГНИТНОЙ ЛЕНТЕ

Краткое содержание: LOAD, SAVE, VERIFY, MERGE.

Основные команды работы с магнитофоном LOAD, SAVE и VERIFY уже рассматривались во вводном описании. Вы могли видеть,что LOAD затирает старую программу в памяти компьютера при загрузке новой программы с ленты. Есть другая команда MERGE, не делающая этого. Эта команда стирает лишь те строки старой программы или переменные,которые совпадают с номерами строк новой программы или именами новых переменных.

Теперь Вы знаете 4 оператора для работы с магнитофоном:

SAVE – записывает программу и переменные на магнитофон;

VERIFY – проверяет копию программы на ленте, сравнивая ее с памятью компьютера;

LOAD – очищает память компьютера от всех программ и загружает в нее новые программы,считанные с магнитофона;

MERGE – подобна LOAD, только не очищает всю память, а лишь заменяет те строки программы или переменные,у которых совпадают номера или имена с такими же на магнитной ленте.

За каждой из этих команд следует ключевое слово – имя программы, определенное первоначально в команде SAVE. Пока компьютер ищет указанную программу, он выводит имена всех программ, уже прочитанных с ленты.

В операторах LOAD, VERIFY и MERGE вместо имени можно указать пустую строку. Тогда будет взят первый встретившийся файл.

Если использовать оператор SAVE типа:

```
SAVE STRING LINE NUMBER
```

программа запишется на ленту так, что когда она будет вновь считана по команде LOAD (но не MERGE), она автоматически установится на строку с указанным номером и сама инициирует свое выполнение. (Здесь и далее "STRING" – имя записываемого или уже записанного на магнитную ленту файла. Оно должно состоять не более чем из 10 символов и заключаться в кавычки).

Кроме текстов программ, на ленту можно записывать также массивы или данные.

Записать на ленту массив Вы можете, используя команду SAVE с DATA таким образом:

```
SAVE STRING DATA ARRAY NAME ()
```

здесь "STRING" – имя, присваиваемое файлу данных, которое может состоять из букв или букв и символа "$" (перечеркнутая буква S). Для строковых данных это требование здесь не важно. Загружаются такие данные по команде:

```
LOAD STRING DATA ARRAY NAME ()
```

Нельзя использовать оператор MERGE.

Если загружается строковый массив, то после обнаружения его на ленте, компьютер выдает: "CHARACTER ARRAY:" и далее имя этого массива.

Существует возможность записи на магнитную ленту и отдельных байтов информации. Так, например, это может быть телевизионная картинка или определяемые пользователем графические символы и т.д. Для этого используется ключевое слово "CODE", например:

```
SAVE "PICTURE" CODE 16384,6912 ,
```

здесь первое число – адрес первого байта в области памяти, где расположены данные, а второе число – количество байтов, которое нужно записать на ленту (6912 – обьем в байтах одного экрана, а 16384 – адрес экрана в памяти). Загружаются эти данные по команде:

```
LOAD "PICTURE" CODE
```

после CODE можно указать числа:

```
LOAD "PICTURE" CODE START, LENGTH
```

LENGTH (длина) – определяет сколько байтов данных надо загрузить с ленты. Если длина больше, чем записано на ленту, то выдается сообщение "R Tape loading error" (ошибка загрузки с ленты). Этот параметр можно опустить и тогда компьютер считает все данные, которые записаны на ленте.

START (начало) – указывает адрес, с которого должны загружаться данные и может быть отличным от адреса, указанного в SAVE. Вы можете опускать этот параметр в команде LOAD.

Выражение CODE 16384,6912 можно заменить на SCREEN$:

SAVE "PICTURE" SCREEN$

и затем

LOAD "PICTURE" SCREEN$

Это тот случай, когда VERIFY не работает. В остальных случаях VERIFY можно использовать везде, где используется SAVE.

В заключение. Везде, где указывается имя файла на ленте, используются только первые 10 символов. Существуют 4 типа информации, которые могут быть записаны на ленту:

- программы и переменные (совместно);
- числовые массивы;
- строковые массивы;
- непосредсвенно байты.

Когда команды VERIFY,LOAD и MERGE осуществляют поиск данных на ленте, они выводят на экран все считанные ими с ленты имена с указанием типа данных в виде: "PROGRAM:", "NUMBER ARRAY:", "CHARACTER ARRAY:" или "BYTES:".

Если имя - пустая строка, эти команды берут первый встретившийся файл с указанным типом.

Команда SAVE служит для записи информации на ленту под заданным именем. Сообщение об ошибке 'F' Invalid file name выдается, если вместо имени указана пустая строка или число символов в имени 11 и более. SAVE всегда выдает сообщение

"Start tape, then press any key" (запусти магнитофон и нажми любую клавишу),

и ждет нажатия, после чего записывает данные на ленту.

1. Программа и переменные

SAVE "NAME" LINE NUMBER

записывает программу на ленту таким образом, что последующая команда LOAD автоматически вставляет в программу

GO TO LINE NUMBER

и начинает ее выполнять.

2. Байты

SAVE "NAME" CODE START, LENGTH

записывает на ленту "LENGTH" байт, начиная с адреса START.

SAVE "NAME" SCREEN$

эквивалентно

SAVE "NAME "CODE 16384, 6912

и записывает один телевизионный экран.

3. Массивы

SAVE "NAME" DATA LETTER ()

или

SAVE "NAME" DATA LETTER$ ()

записывает числовой или строковый массив (требование "$" не относится к "NAME").

Команда VERIFY проверяет (сравнивает) информацию в памяти и на ленте.

Может выдавать сообщение "R Tape loading error":

1. Программа и переменные

VERIFY "NAME"

2. Байты

VERIFY "NAME" CODE START, LENGTH

Если данных в файле "NAME" более, чем указано в "LENGTH",то выдается сообщение об ошибке "R".

VERIFY "NAME" CODE START

здесь осуществляется сравнение байтов в файле "NAME" с данными в памяти, начиная с адреса "START".

VERIFY "NAME" CODE

Этот оператор осуществляет сравнение данных на ленте с данными в памяти, начиная с адреса, с которого записывался на ленту первый байт данных.

VERIFY "NAME" SCREEN$

или эквивалентно

```
VERIFY "NAME" CODE 16384, 6912
```

однако это будет проверка уже проверенного файла.

3. Массивы

```
VERIFY "NAME" DATA LETTER ()
VERIFY "NAME" DATA LETTER$ ()
```

Команда LOAD загружает новые данные с ленты, стирая старые данные в памяти.

1. Программа и переменные

```
LOAD "NAME"
```

Может выдавать сообщение " 4 Out of memory", если нет места для новой программы. В этом случае старая программа не уничтожается.

2. Байты

```
LOAD "NAME" CODE START, LENGTH
```

Если данных в файле "NAME" больше, чем указано в "LENGTH", то выдается сообщение "R".

```
LOAD "NAME" CODE START
```

производит загрузку данных из файла "NAME" в память, начиная с адреса "START".

```
LOAD "NAME" CODE
```

загружает данные по адресу, с которого записывались данные на ленту в файл "NAME".

3. Массивы

```
LOAD "NAME" DATA LETTER ()
```

или

```
LOAD "NAME" DATA LETTER$ ()
```

Уничтожает в памяти массив с именем "LETTER" или "LETTER$", формирует новый массив и переписывает туда данные из файла "NAME". Может выдать сообщение "4 Out of memory" при нехватке памяти под массив. В этом случае старый массив не уничтожается.

Команда MERGE загружает новые данные с ленты, не уничтожая старые.

1. Программа и переменные

```
MERGE "NAME"
```

дописывает программу "NAME" к некоторой программе, находящейся в памяти. Может выдать сообщение "4 Out of memory".

2. Байты

Не поддерживается.

3. Массивы

Не поддерживается.

Пример. Записать на ленту информацию о 21-м определенном пользователем символе.

```
SAVE "CHESS" CODE USR "A", 21*8
```

обратная загрузка:

```
LOAD "CHESS" CODE
LOAD "CHESS" CODE USR "A"
```

6.20. УСТРОЙСТВО ПЕЧАТИ

Краткое содержание: LPRINT, LLIST, COPY.

Эта глава описывает операторы БЕЙСИКА, необходимые для работы с принтером ZX.

Два оператора LPRINT и LLIST подобны операторам PRINT и LIST, но с той лишь разницей, что они работают не с телевизором, а с принтером.

Попробуйте для примера выполнить следующую программу:

```
10 LPRINT '"THIS PROGRAM"'
20 LLIST
30 LPRINT '"PRINTS OUT THE CHARACTER SET." '
40 FOR N=32 TO 255
50 LPRINT CHR$ N
60 NEXT N
```

Оператор COPY позволяет распечатать экран телевизора. Например, по LIST текст программы будет выведен на экран, а затем по COPY его можно распечатать на

принтере.

Вы всегда можете прекратить вывод на печать, выдав BREAK (CAPS SHIFT и SPACE).

Если Вы задали операторы управления принтером без подключенного реального устройства, то вывода просто не будет и выполнение программы продолжится со следующего оператора.

Теперь попробуйте выполнить такую программу:

```
10 FOR N=31 TO 0 STEP -1
20 PRINT AT 31-N,N; CHR$ (CODE"0"+N)
30 NEXT N
```

Вы получите последовательность символов, расположенных по диагонали экрана, начиная с правого верхнего угла. Теперь заменим в строке 20 'AT 31-N,N' на 'TAB N', программа будет работать также, как и прежде. Теперь заменим в строке 20 PRINT на LPRINT и заметим, что развертки по диагонали не получается. А заменив теперь 'TAB N' на 'AT 31-N,N' и сохранив LPRINT, получим по одному символу на строку, что и требовалось получить.

Вообще, при печати, перевод строки осуществляется в следующих случаях:

а). При заполнении буфера строки;

б). После LPRINT, если это не конец оператора и в нем встретились запятая или точка с запятой;

в). Если запятая, апостроф или TAB требуют новой строки;

г). При окончании программы, если остались не выведенные данные.

6.21. ВВОД И ВЫВОД

Краткое содержание: OUT, IN.

Компьютер может считывать некоторую информацию и записывать ее в свою оперативную память по командам PEEK и POKE. Вся память компьютера, и ПЗУ,и ОЗУ, представляется совокупностью адресов от 0 до 65535, каждый из которых адресует один байт.

Таким же образом можно адресовать и еще 65535 адресов, называемых портами ввода-вывода. Они используются процессором для связи с клавиатурой и принтером, и могут управляться операторами БЕЙСИКА IN и OUT.

IN аналогичен оператору PEEK:

```
IN ADDRESS
```

он использует один аргумент - адрес порта, и позволяет считать один байт из указанного порта.

UUT подобен оператору POKE:

```
OUT ADDRESS, VALUE
```

и записывает данные VALUE в порт вывода с адресом ADDRESS.

ZX-SPECTRUM оперирует с шестнадцатиразрядным адресом, который мы будем обозначать буквой A:

```
A15,A14,....,A2,A1,A0
```

Биты адреса A0, A1, A2, A3 и A4 очень важны. Как правило, они находятся в состоянии логической 1, но если хотя бы один из них находится в состоянии логического 0, то это предписывает компьютеру некоторые действия. Не более чем один из этих пяти битов может быть в 0.

Биты A6 и A7 игнорируются, так что, если Вы знакомы с электроникой, то можете использовать их по своему усмотрению.

Биты A8 и A9 используются иногда для получения дополнительной информации.

Информационный байт будем обозначать буквой D:

```
D7, D6, D5, D4, D3, D2, D1, D0
```

Теперь представим список адресов портов. Имеется целый ряд входных адресов для чтения с клавиатуры, а также входного разъема 'EAR'. Сама клавиатура разбита на 8 полурядов по 5 клавиш в ряду.

```
IN 65278 считывает ряд от CAPS SHIFT до V;
IN 65022 считывает ряд от A до G;
IN 64510 считывает ряд от Q до T;
IN 63486 считывает ряд от 1 до 5;
IN 61438 считывает ряд от 0 до 6;
IN 57342 считывает ряд от P до Y;
```

IN 49150 считывает ряд от ENTER до H;
IN 32766 считывает ряд от SPACE до B.

Эти адреса могут быть вычислены из выражения:

254+256*(255-2**N) при N, пробегающем от 0 до 7.

В байте, считанном с клавиатуры, биты от D0 до D4 служат для обозначения пяти клавиш в данном полуряду. D0 для крайней клавиши, а D4 - для той, что ближе к центру. Состояние 0 одного из этих битов указывает, что соответствующая ему клавиша нажата. D6 принимает свое значение при чтении с разъема 'EAR'.

Выходной порт 254 обслуживает громкоговоритель (D4) и разъем 'MIC' (D3), а также установку цвета (D2, D1, D0).

Порт 251 обеспечивает связь с принтером, как чтение, так и запись. Чтение- для проверки готовности принтера к работе.

Порты 254, 247 и 239 используются для связи с дополнительными устройствами.

Запустите следующую программу:

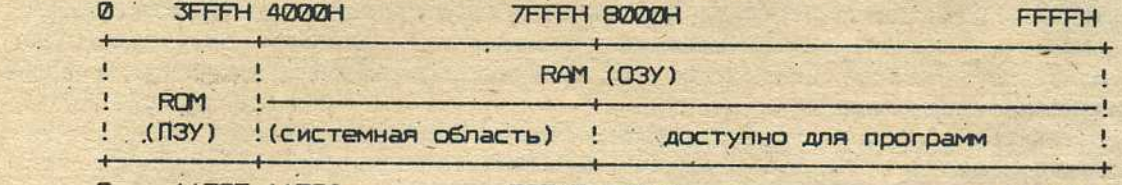
```
10 FOR N=0 TO 7: REM HELF-ROW NUMBER  (номер полуряда)
20 LET A=254+256*(255-2**N)
30 PRINT AT 0,0: IN A: GO TO 30
```

понажимайте по одной клавише в каждом полуряду. После нажатия очередной клавиши введите BREAK, а затем NEXT N.

6.22. ПАМЯТЬ

Краткое содержание: CLEAR.

Вся память компьютера разбита на байты, каждый из которых представим числом от 0H до 255. Каждый байт может быть записан в память по определенному адресу от 0 до FFFFH (H - здесь и далее означает шестнадцатиричное представление числа). Сам адрес может быть записан в память, как два байта. На диаграмме показано распределение памяти компьютера ZX-SPEKTRUM:



Для получения содержимого области любой из памятей используется функция PEEK с адресом в качестве аргумента. Функция возвращает значение байта по этому адресу.

Рассматриваемая ниже программа выводит содержимое первых 21 байтов из ROM с адресами:

```
10 PRINT "ADDRESS"; TAB 8; "BYTE"
20 FOR A=0 TO 20
30 PRINT A; TAB 8; PEEK A
40 NEXT A
```

Для изменения содержимого памяти (только для RAM (ОЗУ)), используется оператор POKE в форме:

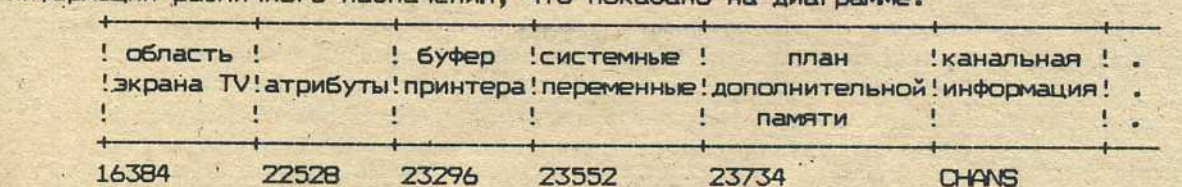
POKE ADDRESS,NEW CONTENTS ,

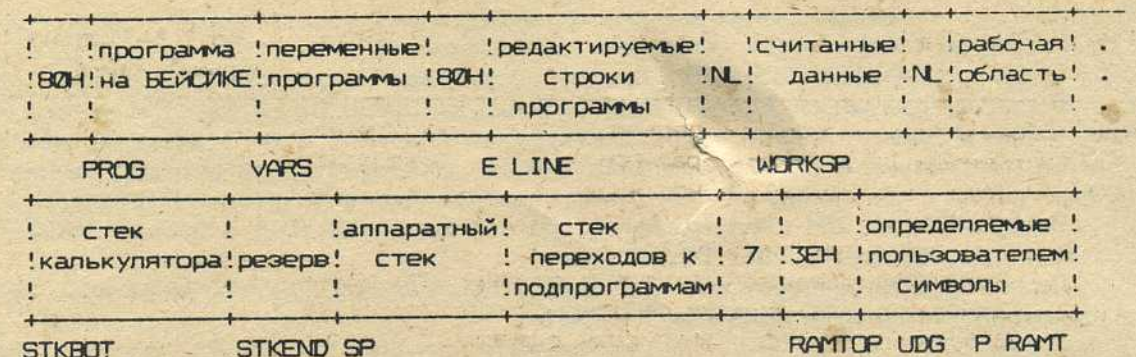
где "ADDRESS" и "NEW CONTENTS" - числовые выражения. Например:

POKE 3100,57

"NEW CONTENTS" может принимать значения от 0 до 255.

Вся память распределяется на области, предназначенные для хранения информации различного назначения, что показано на диаграмме:





Область телевизионного экрана содержит образ текущего кадра. Она доступна для операторов PEEK и POKE. Каждая позиция экрана представима матрицей 8X8 точек (один байт на каждый ряд из 8-ми точек). Однако эти восемь байт хранятся в памяти не вместе.

Полный экран представляет собой 24 строки по 32 символа. Каждая строка экрана прописывается 8-мью строками развертки телевизионного экрана. Итого для записи одного экрана выполняется 172 сканирования, и в памяти рядом хранятся байты одноименных рядов, матриц соседних позиций экрана.

Область атрибутов содержит данные о цвете и других параметрах каждой позиции экрана. Используется в формате ATTR.

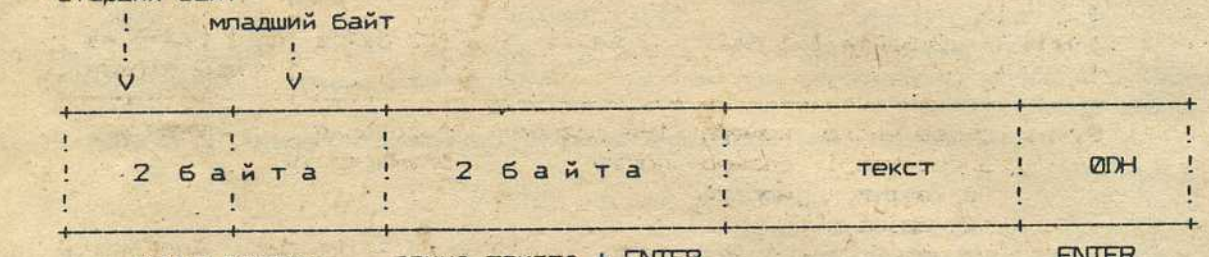
Буфер принтера содержит символы, передаваемые на печать.

Область системных переменных содержит данные управления вычислениями. Они полностью описаны в следующей главе. Некоторые из них (CHANS, PROG, VARS, E LINE и т. д.) содержат адреса границ между системными областями памяти. Но это не переменные БЕЙСИКА и их имена не распознаются компьютером.

Область планов дополнительной памяти используется только с MICRODRIVE.

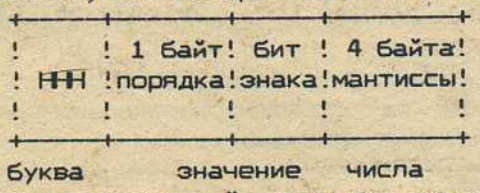
В области канальной информации содержатся данные об устройствах ввода-вывода, а именно: клавиатуре (с нижней половиной экрана), верхней половине экрана и принтере.

Каждая строка области БЕЙСИК-программ имеет формат:



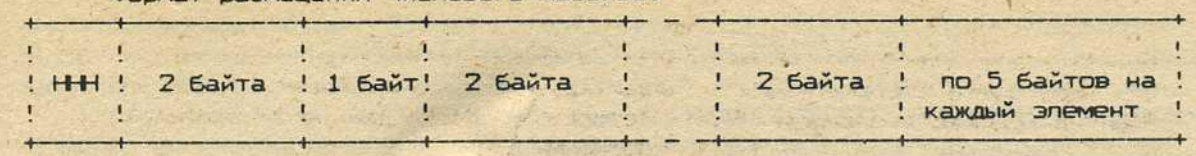
Числовые константы в программе представлены в двоичной форме, используя CHR$ 14 и следующие за ним 5 байт самого числа.

Переменные имеют различные форматы представления в памяти. Ниже представлен формат записи переменной, имя которой состоит из одной буквы:



Формат размещения переменной, если имя имеет более, чем одну букву:

Формат размещения числового массива:



буква — общая длина элементов+1 на каждую размерность — номера размерностей — первая размерность — последняя размерность

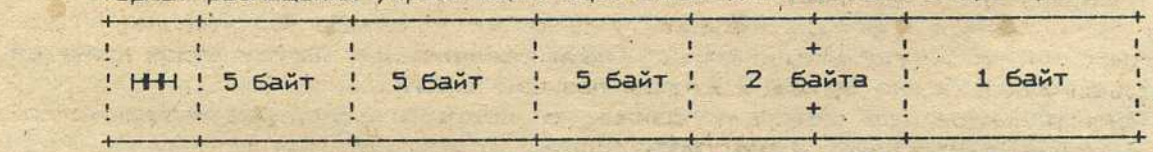
Порядок элементов следующий:

1. Элементы, для которых первая размерность равна 1.
2. Элементы, для которых первая размерность равна 2.
3. Элементы, для которых первая размерность равна 3 и т. д.

Затем в том же порядке по следующей размерности и т. д. Например, элементы массива с размерностью (3,6) расположатся в памяти в следующем порядке:

B(1,1), B(1,2), B(1,3), B(1,4), B(1,5), B(1,6), B(2,1), B(2,2),...,B(3,6)

Формат размещения управляющих переменных для FOR-NEXT операторов:

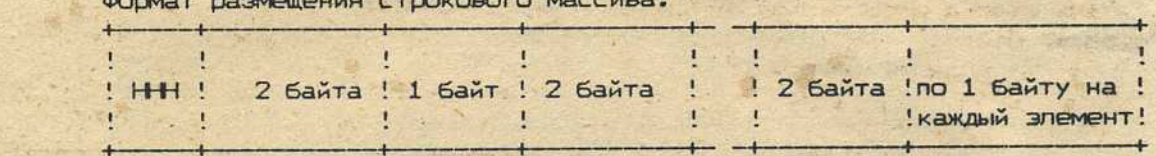


буква — начальное значение — ограничение — приращение — строка цикла — номер оператора

Формат размещения строки символов:

буква — количество символов

Формат размещения строкового массива:



буква — общее число элементов+1 на каждую размерность — номера размерностей — 1-я размерность — последняя размерность — элементы

Программируемый стек - есть часть интерпретатора БЕЙСИКа. Аппаратный стек используется микропроцессором Z80 для запоминания адресов возврата.

Резерв в данной версии не используется.

Назначение стека переходов к подпрограммам описано в главе 5.

Байт, адресуемый по RAMTOP, содержит верхний адрес, используемый БЕЙСИКОМ. Даже оператор NEW, который очищает ОЗУ, не изменяет содержимого области определяемых пользователем символов.

Вы можете изменить адрес RAMTOP в операторе CLEAR:

```
CLEAR NEW RAMTOP,
```

по которому:

1. очищаются все области переменных;
2. очищается область экрана (подобно CLS);
3. переустанавливается позиция PLOT в левый нижний угол экрана;
4. выполняется функция RESTORE;
5. очищается стек переходов и устанавливается новое значение RAMTOP.

Оператор RUN также выполняет действия CLEAR, хотя и не изменяет значение RAMTOP.

Используя оператор CLEAR, Вы можете смещать RAMTOP, увеличивая область для БЕЙСИКА, уменьшая тем самым область определяемых пользователем символов. Можно несколько увеличить доступную часть RAM (ОЗУ), используя директиву NEW. Например, выполнение NEW, затем CLEAR 23800 помогает компьютеру при переполнении ОЗУ.

Все указанные действия могут приводить к двум сообщениям об ошибке и выдаче звукового сигнала:

"4 OUT OF MEMORY" (переполнение памяти);

"G NO ROOM FOR LINE" (нет места для строки программы).

Можно изменить длительность подачи звукового сигнала, изменив число по адресу 23608. По умолчанию предполагается 64.

Числа (за исключением 0) могут записываться в показательной форме как:

M*2**E,

где M - мантисса в интервале 0.5 ... 1 (не может быть 1);

E - экспонента, положительное или отрицательное число.

Допустим Вы записали "M" в двоичной системе счисления, "M"- дробное и имеет двоичную точку (подобно десятичной точке), то тогда будет:

1/2 ----> .1

1/4 ----> .01

3/4 ----> .11 и т. д.

Наше число "M" меньше, чем 1, значит у него нет битов перед двоичной точкой, а поскольку оно больше 0.5, то левый бит следующий за точкой - это 1.

Для записи числа в память мы используем 5 байтов в следующем порядке:

а) записываем первые 8 бит мантиссы во второй байт (мы помним,что первый бит - это 1), вторые 8 бит в третий байт и т. д. до пятого байта;

б) заменяем первый бит второго байта, в котором записана 1 на знаковый бит (0 для +, 1 для -);

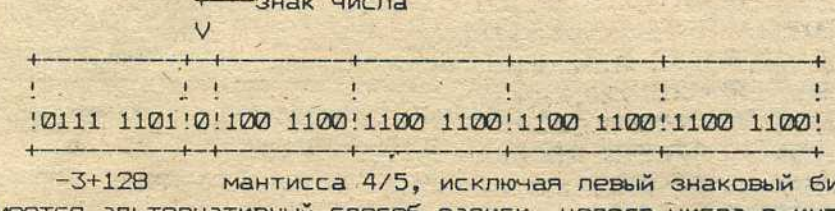
в) записываем в первый байт сумму экспоненты и числа 128.

Например, мы хотим записать число 1/10:

$$1/10 = 4/5 * 2^{-3}$$

Мантисса будет .11001100110011001100110011001100B (поскольку 33-й бит равен 1, мы должны округлить 32-й бит, записав 1 вместо 0), а экспонента равна -3.

Теперь применив наших три правила, запишем 5 байт:



Имеется альтернативный способ записи целого числа в интервале -65535 ... +65535:

а) первый байт равен 0;

б) второй байт равен 0 для положительного числа и FFH для отрицательного числа;

в) 3 и 4 байты содержат младшие и старшие значащие биты числа (или число +131072, если оно отрицательное);

г) 5 байт равен 0.

7. ИСПОЛЬЗОВАНИЕ ПРОГРАММ В МАШИННЫХ КОДАХ

ОС СПЕКТРУМА (как впрочем и Бейсик) позволяет загружать, выгружать и запускать на выполнение программы в машинных кодах процессора Z80. При загрузке и выгрузке такие программы представлены файлами типа BYTE, являющимися полной аналогией файлов типа .COM компьютера IBM PC. Эти программы полностью резидентны (т.е. при работе целиком находятся в ОЗУ компьютера), что повышает скорость их работы практически до скорости работы процессора (950 000 операций в секунду). Это самые быстрые программы из всех возможных. Такие программы - результат компиляции программ на языках высокого уровня (Паскаль, СИ, Фортран, Бейсик, специальные языки); результат ассемблирования программ на языке Ассемблера Z80; непосредственное написание программ в кодах с помощью специальных отладчиков или просто вручную. (Перечень мнемокодов команд микропроцессора Z80 приведен в приложении 5).

Имеются три момента, подлежащие обсуждению.

а) выбор области RAM.

Программист должен определить требуемый обьем памяти. В SP имеются несколько областей памяти, которые могут использоваться, но демонстрация программ в этой главе будет представлена в области RAM с адреса 32000 и дальше. Программа в кодах в этой области, если требуется, может сохраняться на ленту (SAVE) или загружаться с ленты (LOAD) как блок данных. Вообще пользователь может использовать область ОЗУ от RAMTOP до UDG (см. системные переменные)

Командой BASIC CLEAR N Вы можете поменять стандартное значение переменной RAMTOP и тем самым увеличить объем памяти под программы в машинных кодах (при этом уменьшив BASIC область).

б) ввод кодов программы.

В SP можно вводить коды только по команде POKE. Однако операнд этой команды может быть задан как десятичное или двоичное число, или как выражение. Поэтому следующие строки все равносильны:

```
10 POKE 32000,201
10 POKE 32000,BIN 1100 1001
10 LET A=201; POKE 32000,A
```

и каждая может оказаться полезной в определенном случае.

Рекомендуемый метод однако заключается в описании команд машинного кода в их соответствующих шестнадцатиричных кодах.

```
10 LET D=32000 : REM Hex loader
20 DEF FN A(A$,B)=CODE A$(B)-48-7*(CODE A$(B)>57)
30 DEF FN C(A$)=16*FN A(A$,1)*FN A(A$,2)
40 READ A$: IF A$<>"**" THEN POKE D,FN C (A$): LET D=D+1: GO TO 40
```

Загрузчик будет считывать данные из DATA, в котором каждая пара шестнадцатиричных символов заключена в кавычки и отделена от других пар запятыми. Пара символов "**" используется как признак конца.

Пример:

```
50 DATA "00","01","02","**"
   RUN
```

приведет к тому, что адрес 32000 содержит 0, адрес 32001 - 1, 32002 - 2.

в) выполнение программы.

Программы в машинных кодах запускаются на исполнение командой RANDOMIZE USR ..., где ... - адрес (десятичный), с которого программа запускается.

Кроме того, для запуска таких программ в Бейсике имеется функция USR, возвращающая содержимое регистровой пары BC процессора Z80 в виде целого положительного числа от 0 до 65535. Все программы такого типа должны заканчиваться командой процессора RET (возврат из подпрограммы) 201 (шестнадцатиричный C9).

Примеры:

```
RANDOMIZE USR 0 ..... запуск программы в кодах с адреса 0000
                      т.е. в ПЗУ. Такая команда полностью
                      аналогична нажатию кнопки "сброс".
RANDOMIZE USR 55000 ..... запуск программы в кодах с адреса 55000
LET A=USR 55000 ..... запуск программы в кодах с адреса 55000
```

и присвоение переменной "A" значения, оказавшегося в регистровой паре BC процессора в момент возврата.

PRINT USR 55000 ..... запуск программы в кодах и печать значения регистровой пары BC процессора.

RANDOMIZE USR USR 55000 ..... очень интересная комбинация, запускается программа в кодах с адреса 55000, а затем запускается программа в кодах с адреса, возвращенного первой программой в регистровой паре BC процессора.

Приведем примеры программ.

Следующие программы представлены в формате Ассемблера и оператор DATA используется совместно с шестнадцатиричным загрузчиком.

Пример 1. Команда "нет операции" (NOP).

Эта команда очень простая и программа машинного кода, содержащая одну или более строк "нет операции" заканчивается "RET".

Следующая программа Ассемблера показывает простое использование команды NOP.

адрес	маш.код	обозн.	коммент.
7D00	00	NOP	NOP
7D01	C9	RET	возврат

Следующая программа BASIC показывает как изменить выше приведенную программу на ассемблере для SPECTRUM.

Программа 1 'NOP'

```
50 DATA "00","C9","**"
60 LET A=USR 32000
70 PRINT "Нормальное завершение NOP"
```

Пример 2. Команды для непосредственной загрузки регистров.

В следующем примере B и C регистры загружаются постоянными.

7D00	06 00	LD B,+00	Рег. B=0
7D02	0E 00	LD C,+xx	Пользователь вводит различные значения
7D04	C9	RET	

В программе 2 команды 'LD B,+dd' и 'LD C,+dd' обе используются. Пользователь может задавать различные значения в качестве входных операндов.

Программа 2.

```
50 DATA "06","00","0E","00"
51 DATA "C9","**"
60 INPUT "введите значения для регистра C (только 0-255)";N
70 CLS
80 POKE 32003,N
90 PRINT AT 10,0;"Регистр C теперь содержит"; CHR$ 32,USR 32000
100 GO TO 60
```

Вы можете записать Вашу программу на ленту:

SAVE "NAME" CODE адрес, количество байт

Программы в кодах не имеют атрибута самозапуска, из-за чего они не могут сами запуститься после загрузки, если не применять для этого особых программных ухищрений. Поэтому для их загрузки и запуска применяют маленькие программки-загрузчики на Бейсике.

При использовании программ в кодах надо учитывать, что при их выполнении не производится контроль ошибок и может произойти зависание компьютера, устранимое только кнопкой <сброс>.

Написание программ в кодах - занятие весьма кропотливое, так как приходится помнить не только коды команд, но и их длину (1, 2, 3, 4 байта), внимательно следить за адресами, в которых размещается программа и данные.

Использование ассемблера позволяет получить столь же эффективные и быстрые программы, как и при написании их в кодах, но с гораздо меньшими затратами сил и времени. Одновременно обеспечивается возможность работы программы в любых адресах (при перетрансляции) и легкость ее модификации.

Принцип программирования на ассемблере состоит в том, что вместо кодов

машинных команд Вы пишите их условные обозначения - мнемоники, а вместо адресов - именные метки. Программа Ассемблер превращает мнемоники в коды команд, а метки - в конкретные адреса этих команд. Такая операция называется ассемблированием, и в результате ее получается программа в кодах, используемая также, как и обычные программы в кодах.

8. ЗАЩИТА ПРОГРАММ

Защита программ состоит в исключении возможности просмотра текста программы и внесения в него изменений лицом, не уполномоченным на это автором программы.

Методов защиты программ существует очень много, и здесь мы рассмотрим только основные принципы и обозначим подходы к этой проблеме.

Можно выделить три основных направления:

1. Исключить возможность остановки (прерывания работы программы).
2. Исключить возможность подачи команды LIST для распечатывания текста программы.
3. Сделать листинг программы нечитаемым.

ИСКЛЮЧЕНИЕ ВОЗМОЖНОСТИ ОСТАНОВКИ ПРОГРАММЫ

Самое первое, что Вы должны сделать - это выполнить программу автостартующей со строки номер N. Это выполняется при загрузке программы командой SAVE "имя" LINE N. Теперь программа после загрузки будет сама стартовать с указанной строки. Тем не менее она может быть остановлена командой BREAK или по сообщению об ошибке.

Отключить клавишу BREAK можно, если в той строке, с которой происходит автостарт, поместить команду POKE 23613, PEEK 23730-5.

Если программа Вами хорошо отлажена, то остановку по ошибке можно исключить. Надо помнить, что чаще всего при взломе программы сознательно вносят ошибку во время исполнения оператора INPUT. Так, если программа просит от пользователя ввести какое-либо число, он сознательно нажимает на клавишу с буквой. Программа останавливается с сообщением VARIABLE NOT FOUND. Рекомендуем не использовать оператор INPUT, а организовывать опрос клавиатуры на основе функции INKEY$.

Если в строке автостарта поместить нижеприведенные команды, то при попытке сделать BREAK или при появлении ошибки программа будет сбрасываться:

```
LET ERR = 256*PEEK 23614 + PEEK 23613: POKE ERR,0: POKE ERR+1,0
```

Примерно тот же эффект дает помещение в стартовой строке команды POKE 23659,0.

Одним из способов остановки автостартующей программы является использование команды MERGE "" вместо LOAD "" при загрузке.

Если Ваша программа имеет размер больше 7K, то защитить ее от этого приема можно, если перед выгрузкой подать следующие прямые команды:

```
LET X = 256*PEEK 23636 + PEEK 23635: POKE X,60: POKE X+1,0
```

Другой способ защиты от MERGE состоит в том, чтобы перед выгрузкой Вашей программы на ленту дать ложную информацию о том, что якобы первая строка Вашей программы имеет чрезмерно большую длину. Это делается прямой командой:

```
POKE (PEEK 23635 + 256*PEEK 23636 +3),255.
```

ОТКЛЮЧЕНИЕ КОМАНДЫ LIST

Введите первой строкой программы 1 REM. Перед выгрузкой программы на ленту дайте прямую команду:

```
POKE (PEEK 23635 + 256*PEEK 23636),100
```

Теперь программу можно запускать (RUN), но читать нельзя.

КАК СДЕЛАТЬ ТЕКСТ ПРОГРАММЫ НЕЧИТАЕМЫМ

Простейший прием, применяемый во многих программах состоит в том, что устанавливают одинаковым цвета символов (INK) и фона (PAPER).

Например: 10 INK 7: PAPER 7

В тех местах, где программа должна сделать вывод на экран в операторе

PRINT вставляют в качестве временных правильные цвета:

```
20 PRINT INK 0; "ZX SPECTRUM"
```

Другой способ состоит в искажении набора символов путем задания "фальшивого" значения системной переменной CHARS. Попробуйте, например:

```
POKE 23606,8: PRINT "ZX SPECTRUM":POKE 23606,0
```

Если адресовать системную переменную CHARS в те области памяти, где вообще ничего нет, например POKE 23607,200, то все символы будут выглядеть как пробелы и на экране вообще ничего не будет. Вам же нужно перед всяким оператором PRINT включать его после PRINT.

Еще один регулярно используемый прием состоит в организации "нулевой" строки, которая реально исполняется программой, но не может быть удалена при редактировании. Чтобы первая строка стала нулевой, дайте команду POKE 23755,0: POKE 23756,0.

Однако, кроме защиты собственной программы, очень часто стоит задача "вскрыть" защиту чужой программы. Если Вы имеете дело с чужой (фирменной) программой, то загрузив ее, Вам не так просто ее выгрузить. Во-первых, она автостартует и остановить ее очень сложно, во-вторых, Вам неизвестны адреса блоков, из которых она состоит. Однако все эти проблемы можно решить с помощью специальных программ копировщиков, которых разработано очень много. Но и копировщик может оказаться бессилен, если в копируемой программе предусмотрены специальные, и порой весьма изощренные, меры защиты от копирования. Для преодоления таких препятствий разработаны специальные программы-копировщики, которые позволяют копировать защищенные программы. Но даже если удается скопировать программу, далеко не всякую программу можно прочитать, внести в нее изменения, так как помимо защиты от копирования во многих фирменных программах предусмотрены меры защиты программ от их остановки и модификации. Ваша попытка вмешаться в программу может привести к самым разным последствиям – от зависания компьютера до полного обнуления его памяти – в зависимости от того, какое "наказание" придумал автор защиты.

9. ОПЕРАЦИИ С ЭКРАНОМ

Типичной задачей для тех, кто начинает создавать собственные программы на "СПЕКТРУМе", является создание красочных изображений в графике высокого разрешения, хранение их в памяти компьютера и вывод на экран.

Мы не будем рассматривать графические операции с операторами DRAW, CIRCLE, PLOT, а также печать символов графики пользователя, а рассмотрим операции с экраном, созданном в графическом редакторе. Наиболее удобным в работе является графический редактор ARTSTUDIO, но можно использовать и какой-либо другой доступный (например THE ARTIST).

ВОЗМОЖНОСТИ РЕДАКТОРА ARTSTUDIO:

1. Система работы с перекрывающимся меню.
2. Использование для "рисования":
   - карандаша с произвольно назначаемым сечением грифеля;
   - кисти с произвольными размерами и формой;
   - краскораспылителя с регулируемым факелом краски;
   - валика для заливки с различной текстурой (кирпичи, рельеф и т.п.);
   - лупы для рассматривания мелких деталей 2-х, 4-х, 8-кратной.
3. Работа с окнами: размножение, переносы, повороты, удаление, увеличение и т.д.
4. Работа с изображением: растяжение, сплющивание, наложение и т.д.
5. Работа со стандартными геометрическими фигурами: точки, линии, окружности, треугольники, лучи и т.д.
6. Режим "резиновых" геометрических фигур.
7. Печать надписей любым шрифтом в любом направлении. Создание своих шрифтов.
8. Операции с файлами на кассетах (дискетах).
9. Работа с цветом и атрибутами.
10. Отмена любого неправильного действия.

Работа с ARTSTUDIO очень удобна благодаря перекрывающимся меню, где перечислены все функции графического редактора. Достаточно перевести курсор (он выполнен в виде стрелки) на требуемое поле и нажать „Ø" для исполнения данного действия или перехода в тот или иной режим.

Созданное в редакторе изображение Вы можете выгрузить на ленту в виде блока кодов, длина которого 6912 байтов. Если теперь загрузить его командой

```
LOAD "" CODE 16384, 6912
```

то изображение сразу будет загружено на экран, но хранить его там, естественно, нельзя. Можно загрузить изображение в произвольное место и вызывать его оттуда на экран по мере необходимости. Примем адрес для хранения картинки 30000.

```
LOAD "" CODE 30000, 6912
```

Переброску на экран можно организовать в БЕЙСИКе.

```
500 FOR I = 1 TO 6912
510 POKE 16383+I, PEEK 29999+I
520 NEXT I
530 RETURN
```

Всякий раз, когда Вам надо вызвать изображение на экран, Вы можете делать это командой GO SUB 500.

К сожалению этот метод очень медленно работает, что связано с ограниченными возможностями БЕЙСИКа. Построение экрана происходит более 20 секунд.

Возможность быстрой переброски экрана (менее 1 сек.) предоставляет программирование в машинном коде. Процессор Z80 для этой цели имеет мощную команду перемещения блоков LDIR (ее код 237,176). Для ее работы необходимо, чтобы в регистре HL процессора находился адрес начала блока, подлежащего переброске (в нашем случае - 30000), в регистре BC - длина блока (6912), а в регистре DE - адрес места назначения (16384). Тогда программа в машинных кодах будет выглядеть так:

МАШИННЫЙ КОД	АССЕМБЛЕР	КОММЕНТАРИЙ
33 48 117	LD HL, 48 117	Загрузить в регистр HL адрес 30000. Обратите внимание на то, что 48 + 117*256 = 30000.
1 0 27	LD BC, 0 27	Загрузить в регистр BC 6912.
17 0 64	LD DE, 0 64	Загрузить в регистр DE 16384.
237 176	LDIR	Выполнить переброску блока в новое место.
201	RET	Возврат в вызывающую программу.

Таким образом, программа в машинных кодах выглядит так: 33, 48, 117, 1, 0, 27, 17, 0, 64, 237, 176, 201.

Поместим ее в произвольное место оперативной памяти, например начиная с адреса 65000 следующей программой, написанной на БЕЙСИКе.

```
10 FOR I=1 TO 12
20 READ A
30 POKE 59999+I,A
40 NEXT I
50 DATA 33,48,117,1,0,27,17,0,64,237,176,201
```

Теперь всякий раз, когда Вам нужно поместить картинку на экран, Вам надо стартовать эту процедуру, что выполняется командой RANDOMIZE USR 65000.

Если у Вас в памяти компьютера хранится не одно, а несколько графических изображений, то их можно также перебрасывать на экран этой процедурой, надо только изменить адрес, поступающий в регистр процессора HL, что можно сделать командой POKE по адресу 65001 и по адресу 65002.

Предположим, что у Вас есть вторая картинка, помещенная начиная с адреса 37000. Тогда INT (37000/256) = 144, а 37000-256*144 = 136. Выполним POKE 65001,136 и POKE 65002,144. Команда RANDOMIZE USR 65000 выполнит переброску второй картинки и т.д.

Очевидно, что поскольку каждый экран (включая атрибуты) занимает 6912 байтов , то одновременно в памяти нельзя хранить более 6 изображений, что во многих случаях бывает недостаточно. Если учесть, что каждое изображение имеет длительные последовательности нулей, то имеется возможность компрессировать каждое изображение. Это позволяет во многих случаях вдвое сократить обьем памяти, занимаемой графикой, а иногда даже больше.

Мы предлагаем Вашему вниманию две процедуры в машинном коде. Первая процедура вызывается FNs и служит для компрессирования изображения с экрана и размещения его в заданном пользователем адресе. Она имеет два параметра h и l, которые задают адрес для хранения экрана. h - старший байт адреса, а l - младший байт.

Сама процедура FNs расположена начиная с адреса 56600. Длина - 60 байтов.

Вторая процедура FNt служит для декомпрессирования изображения и переброски его на экран. Она имеет два тех же параметра h,l. Адрес размещения этой процедуры - 56500. Длина - 40 байтов.

Чтобы создать процедуру FNs, Вам надо набрать и запустить следующую БЕЙСИК-программу. После того, как она отработает, выгрузите созданную процедуру на ленту командой SAVE "FNs" CODE 56600,60 и в дальнейшем можете использовать в своих программах. Точно так же исполняется копия процедуры FNt:

```
SAVE "FNt" CODE 56500,40.
```

ПРОЦЕДУРА FNs

```
8300 LET b=56600: LET l=60: LET x=0: RESTORE 8310
8301 FOR i=0 TO l-1: READ a
8302 POKE (b+i),a: LET z=z+a
8303 NEXT i
8304 LET x=INT(((x/l)-INT(x/l))*l)
8305 READ a: IF a<>z THEN PRINT "???", STOP
8310 DATA 42, 11, 92, 1, 4
8311 DATA 0, 9, 86, 14, 8
8312 DATA 9, 94, 237, 83, 82
```

```
8313 DATA 221, 33, 0, 64, 6
8314 DATA 1, 126, 44, 32, 8
8315 DATA 36, 245, 124, 254, 91
8316 DATA 40, 16, 241, 78, 185
8317 DATA 32, 4, 4, 32, 238
8318 DATA 5, 18, 19, 120, 18
8319 DATA 19, 24, 227, 241, 18
8320 DATA 19, 120, 18, 237, 83
8321 DATA 22, 221, 201, 0, 0
8322 DATA 56, 0, 0, 0, 0
```

После того, как процедура выполнит компрессирование экрана и перебросит его в место, отведенное для длительного хранения, Вы можете узнать адрес, в который можно помещать следующий экран. PRINT PEEK 23296 + 256*PEEK 232297. Но помните, что делать это можно только немедленно после окончания компрессирования и переброски блока.

ПРОЦЕДУРА FNt

```
8350 LET b = 56500: LET l=35: LET z=0: RESTORE 8360
8351 FOR i=0 TO l-1 : READ a
8352 POKE (b+i),a: LET z=z+a
8353 NEXT i
8354 LET z=INT(((z/l)-INT(z/l))*l)
8355 READ a: IF a<>z THEN PRINT "???": STOP
8360 DATA 42, 11, 92, 1, 4
8361 DATA 0, 9, 86, 14, 8
8362 DATA 9, 94, 33, 0, 64
8363 DATA 26, 245, 19, 26, 19
8364 DATA 71, 241, 119, 35, 16
8365 DATA 252, 124, 254, 91, 32
8366 DATA 240, 201, 0, 0, 0
8367 DATA 28, 0, 0, 0, 0
```

ПРИМЕР ПРИМЕНЕНИЯ ПРОЦЕДУР КОМПРЕССИИ И ДЕКОМПРЕССИИ

Предположим, что Вам надо компрессировать и отправить на хранение три экрана scr1, scr2, scr3, причем первый из них идет в адрес 30000 (48, 117).

Тогда основная программа может выглядеть так:

```
10  DEF FN s(h,l) = USR 56600
20  DEF FN t(h,l) = USR 56500
100 LET a=23296: LET b=23297
110 LET h1=117: LET l1=4897
120 LET ad=l1+(h1*256)
130 PRINT ab: LOAD "scr1"CODE 16384,6912
140 RANDOMIZE FNs(h1,l1)
150 LET L2 = PEEK a: LET h2 = PEEK b
155 LET ab = l2+h2*256
160 PRINT ad: LOAD "scr2"CODE 16384,6912
170 RANDOMIZE FNs(h2,l2)
180 LET l3=PEEK a: LET h3=PEEK b
185 LET ad=l3+256*h3
190 PRINT ad: LOAD "scr3"CODE 16384,6912
200 RANDOMIZE FNs(h3,l3)
210 PAUSE 50
300 RANDOMIZE FNt(h1,l1)
310 PAUSE 50
320 RANDOMIZE FNt(h2,l2)
330 PAUSE 50
340 RANDOMIZE FNt(h3,l3)
350 PAUSE 50: GO TO 300
```

10. МЕТОДЫ ВВЕДЕНИЯ РУССКОГО ШРИФТА

Обеспечение возможности работы компьютера с символами русского алфавита является, наверное, одной из первейших задач, которые возникают перед тем, кто хочет сам писать программы на СПЕКТРУМе.

Существуют два принципиально различных подхода для решения этой задачи. Первый подход состоит в использовании символов графики пользователя, а второй – в создании нового набора символов, размещении его в оперативной памяти и назначении его в качестве действующего с помощью системной переменной CHARS.

ИСПОЛЬЗОВАНИЕ ГРАФИКИ ПОЛЬЗОВАТЕЛЯ

В графическом режиме Вы можете, во-первых, набирать графические символы блочной графики, расположенные на цифровых клавишах, а также можете воспользоваться символами графики пользователя. Их может быть до 21; они берутся нажатием на клавиши от "A" до "U" в графическом режиме (курсор "G"). Изображение символов графики пользователя должны быть сформированы в специальной области памяти, на которую указывает системная переменная UDG. Адрес, в котором расположено изображение символа (8 байтов), может быть получен с помощью функции USR. Таким образом, заслав, начиная с адреса USR "A", 8 байтов, формирующих изображение скажем буквы Я, мы будем иметь возможность нажатием клавиши "A" в графическом режиме печатать на экране букву "Я".

Учитывая, что многие буквы латинского и русского алфавита имеют сходное начертание (например O, A, X, H, P, и др.), можно задать прописные буквы, не имеющие аналогов и при этом уложиться в разрешенный 21 символ.

Изображение буквы "Я" формируется из точек экрана (пикселей) например так:

```
00000000    0
00xxxxx0    0*128+0*64+1*32+1*16+1*8+1*4+1*2+0*1 = 62
0x0000x0    0*128+1*64+0*32+0*16+0*8+0*4+1*2+0*1 = 66
0x0000x0    0*128+1*64+0*32+0*16+0*8+0*4+1*2+0*1 = 66
00xxxxx0    0*128+0*64+1*32+1*16+1*8+1*4+1*2+0*1 = 62
00x000x0    0*128+0*64+1*32+0*16+0*8+0*4+1*2+0*1 = 34
0x0000x0    0*128+1*64+0*32+0*16+0*8+0*4+1*2+0*1 = 66
00000000    0
```

Таким образом, в результате такого синтеза шаблона буквы "Я", мы можем представить ее последовательностью из 8 байтов, разместив в соответствующем месте – 0, 62, 66, 66, 62, 34, 66, 0.

Точно так же можно изобразить и остальные желаемые символы. Для тех, кто не хочет глубоко вдаваться в суть процесса, мы предлагаем небольшую готовую программу, которую Вы можете набрать и выгрузить на ленту. Всякий раз, когда Вы хотите составить какую-либо программу, в которой возможна необходимость в русских буквах, загрузите эту программу и стартуйте ее (RUN). После остановки программы дайте LIST и продолжайте со строки 100 набирать свою программу.

```
10 FOR z=1 TO 20
20 READ A$
30 FOR X=0 TO 7
40 READ Y
50 POKE USR A$ + X,Y
60 NEXT X
70 NEXT Z
80 DATA "B",0,126,64,124,66,66,124,0
81 DATA "G",0,126,64,64,64,64,64,0
82 DATA "D",0,28,36,36,36,36,126,66
83 DATA "J",0,65,73,62,73,73,65,0
84 DATA "I",0,66,70,74,82,98,66,0
85 DATA "L",0,30,34,34,34,34,98,0
86 DATA "P",0,126,66,66,66,66,66,0
87 DATA "O",0,66,66,36,24,16,96,0
88 DATA "C",0,68,68,68,68,68,126,2
89 DATA "H",0,66,66,66,126,2,2,0
90 DATA "N",0,65,73,73,73,73,127,0
```

```
91 DATA "M",0,65,73,73,73,73,127,1
92 DATA "E",0,60,66,30,2,66,60,0
93 DATA "U",0,76,82,114,82,82,76,0
94 DATA "A",0,62,66,66,62,34,66,0
95 DATA "S",24,66,70,74,82,98,66,0
96 DATA "R",0,64,64,124,66,66,124,0
97 DATA "T",0,192,64,124,66,66,124,0
98 DATA "F",0,127,73,73,73,127,8,0
99 DATA "Q",0,66,66,114,74,74,114,0
```

В результате работы этой программы устанавливается следующее соответствие между клавишами в графическом режиме и символами русского алфавита:

"Q" – "Ы"	"R" – "Ь"	"U" – "Ю"	"O" – "У"	"A" – "Я"
"D" – "Д"	"G" – "Г"	"J" – "Д"	"C" – "Ц"	"N" – "Ш"
"E" – "Э"	"T" – тв.знак	"I" – "И"	"P" – "О"	"B" – "И"
"F" – "Ф"	"H" – "Ч"	"L" – "Л"	"D" – "Б"	"M" – "Щ"

ИЗМЕНЕНИЕ НАБОРА ЗНАКОВ

Набор знаков компьютеров "СПЕКТРУМ" состоит из 96 символов и занимает 96*8=768 байтов. Стандартный набор находится в ПЗУ по адресу 15616. Этот адрес определен в системной переменной CHARS.

PRINT PEEK 23606 +256*PEEK23607+256 <ENTER> 15616 O.K.

При этом по адресу 23606 находится 0, а по адресу 23607 – 60. Изменение набора знаков состоит в том, чтобы сформировать с помощью какой-либо специализированной программы или вручную желаемый набор знаков, поместить его в любое свободное место оперативной памяти и изменить содержимое ячеек 23606 и 23607, чтобы оно указывало на адрес места расположения нового набора. Рассмотрим эти процедуры на конкретном примере.

1. Для формирования набора знаков очень удобно использовать графический редактор "ARTSTUDIO". Эта программа не только является идеальным графическим редактором, позволяющим легко выполнять графические изображения, но и позволяет конструировать шрифты, а также выгружать сформированный шрифт на ленту.

Для этого, загрузив ARTSTUDIO, необходимо выйти в меню "TEXT". В этом меню существует режим "FOND EDITOR" – это как раз и есть редактор символов, которым можно символы поворачивать, двигать, сделать инверсными, либо вообще изменить до неузнаваемости, создать свои оригинальные символы.

После создания своего символьного набора не забудьте сбросить его на ленту (или диск).

2. Допустим, мы хотим разместить новый шрифт, начиная с адреса 30000. Загрузим его с ленты LOAD ""CODE 30000,768.

3. Системная переменная CHARS должна указывать на адрес, находящийся на 256 байтов ниже, т.е. на 30000-256=29744.

4. Определим старший байт двухбайтной переменной CHARS. INT (29744/256)=116.

5. Определим младший байт: 29744 – 1169*256 = 48.

6. Переключение на печать сформированным Вами шрифтом выполняется командой POKE 23606,48: POKE 23607,116. Обратное переключение на шрифт, записанный в ПЗУ Вашего компьютера – командой POKE 23606,0: POKE 23607,60.

11. ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ АССЕМБЛЕРА

11.1 СУЩНОСТЬ И СТРУКТУРА ЯЗЫКА АССЕМБЛЕРА

Языки программирования для компьютера можно разделить на три основных уровня: машинные, алгоритмические высокого уровня и ассемблера.

Машинным языкам, находящимся на самом нижнем уровне, свойственен тот существенный недостаток, что, будучи языками цифр, они неудобны для описания вычислительных процессов и поэтому требуют от программистов больших усилий при написании и отладке программ. Достоинство машинного языка состоит в том, что для программирования на нем требуется лишь знание системы команд процессора , а для выполнения составленных таким образом программ вычислительная машина не нуждается ни в каком трансляторе. Кроме того, при использовании машинного языка может быть достигнута максимальная гибкость в реализации технических возможностей Вашего компьютера.

Алгоритмические языки высокого уровня (например, ФОРТРАН, БЕЙСИК, ПЛ/М, ПАСКАЛЬ, АДА и др.) занимают верхнее положение в иерархии языков программирования. Будучи приближены к привычной математической нотации и в ряде случаев обеспечивая естественную форму описания вычислительных процессов, они достаточно просты и удобны в программировании, но не всегда позволяют в полной мере реализовать технические возможности компьютера , а результирующие машинные программы, получаемые после трансляции программ с алгоритмических языков, обычно неэффективны с точки зрения объема и быстродействия. Недостатком этих языков является также и то, что их применение предполагает наличие транслятора, представляющего собой сложный программный комплекс и требующего для своей разработки большого труда.

Языки ассемблера, занимая промежуточное положение между машинными языками и языками высокого уровня (будучи при этом гораздо ближе к машинным языкам), объединяют в себе некоторые достоинства самого нижнего и самого верхнего уровней языков программирования. Свое название языки ассемблера получили от имени программы, преобразующей программу, написанную на таком языке, в машинные коды.

Язык ассемблера, обеспечивая возможность символических имен в программе и избавляя программиста от утомительной работы по распределению памяти компьютера для переменных и констант, существенно облегчает труд программиста и повышает его производительность по сравнению с программированием на машинном языке. Язык ассемблера позволяет так же гибко и полно реализовать технические возможности компьютера, как и машинный язык. Транслятор программ с этого языка, т.е. ассемблер, гораздо проще и компактней транслятора программ, требующегося для алгоритмического языка высокого уровня, а результирующая машинная программа на выходе ассемблера может быть столь же эффективной, как и программа, которую сразу написали на машинном языке. Поэтому неудивительно, что первым транслятором, который создается для нового компьютера, является обычно ассемблер. Это особенно справедливо для компьютеров, в которых ограниченный объем основной памяти может значительно усложнить задачу создания транслятора с языка высокого уровня и заставить разработчиков программного обеспечения в течение долгого времени довольствоваться языком ассемблера.

На языке ассемблера, как правило, программируется операционная система компьютера или по крайней мере наиболее ответственные и "узкие" места в такой системе.

Таким образом, язык ассемблера и сам ассемблер обычно составляют основу для создания программного обеспечения компьютера и, в частности, операционных систем и систем реального времени, к которым предъявляются высокие требования с точек зрения объема занимаемой машинной памяти и скорости выполнения программ.

Каждый язык ассемблера является машинно-зависимым языком и отражает аппаратурные особенности (в частности, состав программно-доступных регистров) того компьютера, для которого он создан.

Программа, написанная на языке ассемблера, состоит из последовательности предложений, или операторов, и называется исходной программой или исходным модулем.

Как правило, оператор в языке ассемблера содержит следующие четыре

поля:

1) метки;
2) операции;
3) операндов;
4) комментария.

Из этих полей необходимо лишь поле операции, а остальные поля (или некоторые из них) могут отсутствовать.

Поле метки предназначено для записи символического имени данного оператора. Символическое имя требуется, например, в случае, когда на оператор есть ссылка в программе хотя бы в каком-нибудь другом операторе. Обычно в качестве символического имени допускается применять начинающуюся с буквы алфавитно-цифровую последовательность знаков, максимальная длина которой различна в разных языках ассемблера и, как правило, не превышает восьми знаков.

Поле операции содержит мнемоническую буквенную запись операции, выполняемой оператором. Число знаков в мнемонической записи может варьировать, в зависимости от типа операции, в пределах от одной буквы до некоторого максимального числа букв, определяемого конкретным языком. Для операторов машинных команд мнемокодом операции является мнемокод соответствующей машинной команды.

Поле операндов отведено для одного или нескольких операндов. Если операндов несколько, то они обычно отделяются друг от друга запятыми. В качестве операндов могут быть числа, символические имена и выражения, причем в выражениях используются знаки арифметических операций. Иногда отдельные части операндов, отдельные операнды или группы операндов заключаются в круглые скобки. В операндах могут применяться специальные знаки, специфицирующие режим выполнения оператора.

В **поле комментария** программист может помещать произвольный текст, не влияющий на выполнение оператора и служащий для пояснения этого оператора или фрагмента исходной программы.

Все перечисленные поля отделяются друг от друга по крайней мере одним пробелом. Нередко комментарий от операндов отделяется не пробелами, а знаком ";".

В общем случае в языках ассемблера выделяют следующие группы операторов:

- операторы машинных команд;
- операторы псевдокоманд;
- макрокоманды;
- комментарии;
- команды управления ассемблером.

Операторы машинных команд соответствуют символической записи машинных команд компьютера . Каждый такой оператор в результате трансляции, или ассемблирования, преобразуется в соответствующую машинную команду.

Операторы псевдокоманд (команды определения) предназначены для описания переменных и констант, резервирования памяти , управления счетчиком команд, указания эквивалентных величин, задания начала и конца программного модуля и т.д. Мнемоника операций в псевдокомандах различна в разных языках ассемблера. Приставка "псевдо" в названии этой группы операторов напоминает о том, что операторы псевдокоманд не всегда порождают элементы в объектной программе, являющейся результатом трансляции исходной программы.

Макрокомандой называется такой оператор в языке ассемблера, который при трансляции заменяется последовательностью других операторов языка. Эту последовательность, в свою очередь, называют макрорасширением макрокоманды. Каждой макрокоманде соответствует, кроме того, макроопределение, которое может находиться в том же программном модуле, что и макрокоманда, или чаще всего в библиотеке макроопределений. **Макроопределение** представляет собой совокупность операторов, используемых ассемблером для формирования макрорасширения всякий раз, когда в исходном модуле встречается макрокоманда. В целом макроопределение можно понимать как программную "заготовку" для получения макрорасширения, причем в общем случае размер и характер макрорасширения зависят от параметров, записанных в макрокоманде. Совокупность макрокоманд и средств их обработки в ассемблере называют макросредствами языка ассемблера. Макросредства имеются не

во всех языках ассемблера .

Комментарии предназначены для записи пояснений в тексте исходной программы. Не следует смешивать оператор комментария с комментарием, который может записываться, например, в операторе машинной команды. Как правило, оператор комментария начинается с некоторого выделенного знака (например, знака "*" или ";"), за которым может следовать произвольный текст.

Команды управления ассемблером могут управлять режимом и формой выдачи листинга, обеспечивать выбор внешних устройств для ввода исходной программы и вывода результатов трансляции, осуществлять приостановку и продолжение трансляции и выполнять другие операции. Строго говоря, команды управления ассемблером не являются частью языка ассемблера.

11.2. АССЕМБЛЕР GENS3

Для компьютера ZX SPECTRUM одним из лучших ассемблеров считается профессиональный двухпроходный ассемблер GENS3, MONS3 - отладчик (сейчас появилась более поздняя версия GENS4).

GENS3 загружается в компьютер при помощи команды LOAD :

```
LOAD "GENS3" CODE xxxxx
```

(xxxxx - необходимый адрес загрузки , например 24064)

для инициализации программы ввести :

```
RANDOMIZE USR xxxxx
```

Если в процессе работы необходимо переинициализировать программу, введите для холодного старта (разрушаются результаты предыдущей работы):

```
RANDOMIZE USR xxxxx + 56
```

для теплого старта (сохранение ранее созданных файлов):

```
RANDOMIZE USR xxxxx + 61
```

Необходимо помниить, что GENS3 один раз инициализируется по адресу xxxxx. Например, если Вы ввели и инициализировали GENS3 по адресу 24064, то для холодного старта используется адрес 24120 и для теплого-старта 24125.

После инициализации по адресу xxxxx появится вопрос :

```
" BUFFER SIZE ? "
```

Введите число от 0 до 9 (по умолчанию принимается 4. 1 единица равна 256 байт). Это число устанавливает величину включающего ("INCLUDE") буфера. Величина 0 устанавливает размер буфера равный 64 байтам.

11.2.1. ФОРМАТ УТВЕРЖДЕНИЙ АССЕМБЛЕРА

Каждая строка текста, обрабатываемого GENS3, должна иметь следующий формат, где некоторые поля необязательны :

```
LABEL      MNEMONIC    OPERAND   COMMENT
-------------------------------------------
START          LD        HL,LABEL   "PICK UP" LABEL
```

Пробелы и знаки табуляции (вставляемые редактором) обычно игнорируются. Строка рассматривается так: определяется первый символ и последующие действия зависят от его значения:

" ; " - вся строка рассматривается как комментарий, т.е. игнорируется;

" * " - ожидается следующий символ, с которым образуется команда ассемблера. Все символы после команды воспринимаются как комментарии;

" " - знак конца строки, просто игнорируется;

" " - (пробел или табуляция). Если первый знак - пробел или табуляция, то GENS3 ожидает следующий, отличный от пробела или табуляции (значений), символ и считает его началом мнемоники процессора Z80.

11.2.2. ДИРЕКТИВЫ АССЕМБЛЕРА

GENS3 распознает некоторые мнемоники, называемые директивами ассемблера. Они не оказывают воздействия на процессор Z80 во время выполнения оттранслированной программы т.е. не транслируются в выполняемые команды, они только управляют процессом трансляции и влияют на результирующий обьектный код.

ORG < выражение >

присваивает значение выражения указателю размещения (LOCATION COUNTER).

EQU < выражение >

директива должна предворяться меткой. Устанавливает значение метки равным значению выражения. Выражение не должно содержать символов, значения которых еще не определены.

DEFB < выражение > , < выражение > .....

каждое выражение должно определять 8 битов. Байту с текущим адресом присваивается значение выражения.

DFW < выражение > , < выражение > .....

слову (2 байта) с текущим адресом присваивается значение выражения.

DEFS < выражение >

резервирует блок памяти размером, равным значению выражения.

DEFV < строка >

записывает N байт ASCII - представления строки.

ENT < выражение >

устанавливает исполнительный адрес ассемблированного кода равным значению выражения.

11.2.3. ДИРЕКТИВЫ УСЛОВНОЙ ТРАНСЛЯЦИИ

Директивы условной трансляции позволяют программисту включать в программу секции текста, в зависимости от выполнения заданных условий. Это достигается использованием следующих директив:

" IF " , " ELSE " , " END "

IF < выражение >

если значение выражения равно нулю, ассемблирование прекращается до тех пор, пока не встретится директива ELSE или END.

ELSE

эта директива "включает" или "выключает" ассемблирование, если ассемблирование было "выключенно" и наоборот.

END

завершает блок условной трансляции, т.е. "включает" ассемблирование, если оно было "выключено" директивами IF или ELSE.

11.2.4. КОМАНДЫ АССЕМБЛЕРА

Команды ассемблера, как и его директивы, не переводятся в коды и не влияют на результирующий обьектный код. Они только управляют форматом листинга.

* e вызывает включение в листинг трех пустых строк, что полезно для разделения модулей.

* н < строка >
строка , указанная в команде , считается заголовком и после команды "* e" ассемблер помещает этот заголовок в листинг. Команда "* н" автоматически осуществляет "* e".

* S вызывает приостановку выдачи листинга на экран (но не на принтер). Вывод листинга возобновляется после нажатия любой клавиши.

* L - прекращает выдачу листинга, начиная с текущей строки.

* L + возобновляет выдачу листинга, начиная с текущей строки.

* D + вызывает выдачу значения LOCATION COUNTER в десятичной системе счисления вместо шестнадцатиричной. Значения выводятся без отметки системы счисления.

* D - вызывает выдачу значения LOCATION COUNTER в шестнадцатиричной системе счисления.

* c - прекращает включение в листинг полученных кодов команд, сокращая длину строки листинга на 9 символов. Позволяет размещать большинство ассемблированных строк на одной 32 - х символьной экранной строке.

* c + возвращает к полному листингу ассемблера.

* F < имя файла >
это очень мощная команда, позволяющая ассемблировать программы, записанные на магнитной ленте. Производит считывание ленты в буфер (1 блок за 1 раз) и затем ассемблирует из буфера, что позволяет транслировать большие программы, т.к. текстовый файл не хранится в памяти машины.

12. СИНТЕЗАТОР ЗВУКОВЫХ ЭФФЕКТОВ

Ваш компьютер обладает системой синтезатора звуковых эффектов, позволяющей получать различные звуки. В языке Бейсик этим синтезатором управляет всего одна команда BEEP, позволяющая подавать звуковые сигналы и исполнять несложные мелодии.

Если Вы увлекаетесь музыкой, есть множество различных музыкальных программ. Например, программы WHAM!. MUSIC BOX позволяют исполнять различные произведения в разложении на два канала тонального сигнала и ритм-бокс, с изображением партитуры на нотном стане. Каждый из каналов имеет диапазон 4,5 октавы и свою нотную партитуру. Ритм-бокс имеет три перестраиваемых эффекта и автоподстраивающийся барабан. Партитура запоминается, редактируется, записывается на магнитофон и загружается с него. После отладки, партитуру встроенным компилятором можно откомпилировать, превратив в программу в кодах, которую можно уже использовать как составную часть любой Вашей программы.

Рассмотрим в качестве примера музыкальный редактор WHAM!:

Музыкальный редактор WHAM! представляет собой двухголосый синтезатор, имеющий четыре октавы и имитатор ударников. Все это с возможностью редакции и сохранения на магнитной ленте.

МЕНЮ:

1 - загрузить ноты;
2 - запомнить ноты;
3 - прослушать текущую мелодию;
4 - скомпилировать ноты;
5 - изменить темп;
6 - войти в редактор;
7 - HELP.

ЗАГРУЗКА НОТ

При нажатии на клавишу "1" в нижней части экрана появится надпись:

TAPE MEMORY DRIVE

При нажатии на клавишу "T" ноты будут загружаться с магнитофона; в этом случае на последующий запрос необходимо ввести имя файла на ленте и включить "воспроизведение" на магнитофоне (для отмены этого режима нажмите клавишу "SPACE", а затем "RUN" для перезапуска WHAM!).

"M" - загрузка из памяти. В памяти WHAM! хранится пять мелодий и есть место для хранения еще одной мелодии. При нажатии на клавишу "M" на экране появятся названия мелодий; нажав на клавишу, соответствующую номеру мелодии, Вы поместите ее текст в редактор, где ее можно прослушать или изменить.

"D" - загрузка с устройства типа микродрайва.

Для возврата в основное меню используйте клавишу "SPACE".

СОХРАНЕНИЕ НОТ

При нажатии в основном меню на клавишу "2" в нижней части экрана также появится надпись:

TAPE MEMORY DRIVE

Нажатие на клавишу "T" приведет к запросу имени файла с последующим сохранением текста мелодии из буфера редактора на магнитной ленте в файле с указанным именем.

"M" - сохранение текста мелодии в памяти WHAM!. При нажатии на эту клавишу появится запрос названия мелодии, а затем номер, под которым она будет храниться в памяти (естественно, пока включено питание Вашего компьютера). Если Вы укажете номер от 1 до 5, то перекроете текст мелодии, который уже там находился. Для того, чтобы это избежать и существует свободное место под номером 6.

"D" - работа с микродрайвом.

"SPACE" - возврат в основное меню.

ИСПОЛНЕНИЕ ТЕКУЩЕЙ МЕЛОДИИ

Нажатие на клавишу "3" приводит к исполнению мелодии, текст которой

находится в данный момент в буфере редактора. Здесь нажатие на любую другую клавишу приведет к возврату в режим редактирования.

КОМПИЛИРОВАНИЕ

Эта функция очень важна. Она позволяет скомпилировать текст мелодии, находящийся в редакторе в коды, которые могут загружаться отдельно от WHAM' и исполнять созданную Вами мелодию.

После нажатия на клавишу "4" в основном меню необходимо ответить на ряд вопросов:

- адрес загрузки (например, 50000);
- имя файла;
- режим работы кодов после загрузки:
  1. исполнение прекращается только при нажатии на клавишу;
  2. исполнение идет только при нажатой клавише;
  3. исполнение прекращается как при нажатии на клавишу, так и при достижении конца музыки.

Выбрав нужный режим работы, нажмите соответствующий номер, после чего запишите кодовый файл на магнитофон.

ИЗМЕНИТЬ ТЕМП

Клавиша "5" позволит Вам подобрать темп мелодии. Изменить темп можно с помощью курсора клавишами "5" и "8". Нажатие любой другой клавиши приведет к возврату в редактор.

РЕДАКТОР

Нажатие на клавишу "6" вызывает переход к редактору. При этом в нижней части экрана появляется информационное табло, которое определяет:

- выбранную октаву;
- канал, который в данный момент редактируется (редактирование выполняется отдельно по каждому каналу);
- номер ноты, начиная с единицы (для каждого канала в отдельности).

Октава выбирается клавишами "1" ... "4".

Для нотных клавиш используются клавиши 3-го и 4-го ряда клавиш компьютера, учитывая их относительное смещение:

```
 ! ДО# ! РЕ# !     ! ФА# ! СОЛЬ# ! СИ-Ь !     ! ДО# ! РЕ# !
 !     !     !     !     !       !       !     !     !     !
 !  A  !  S  !     !  F  !   G   !   H   !     !  K  !  L  !
 ---------------------------------------------------------------
 ! ДО  ! РЕ  ! МИ  ! ФА  ! СОЛЬ ! ЛЯ  ! СИ  ! ДО  ! РЕ  ! МИ  !
 !     !     !     !     !       !     !     !     !     !     !
 ! CS  !  Z  !  X  !  C  !   V   !  B  !  N  !  M  ! SS  !SPACE!
 ---------------------------------------------------------------
 Т Е К У Щ А Я   О К Т А В А          /\ СЛЕД. ОКТАВА
 ---------------------------------    ---------------
```

Редактор имеет ряд управляющих клавиш:

P – воспроизводит следующую ноту (движение на одну ноту вперед);

Ø – перемещение на одну ноту назад;

O – ускоренное движение вперед с одновременным воспроизведением мелодии;

9 – ускоренное движение назад (бесшумное);

R – возврат в начало записи;

Q – воспроизведение мелодии, начиная с помеченной ноты; последующее нажатие на любую клавишу прерывает эту функцию;

ENTER – стирание следующей ноты текущего канала; для перехода на другой канал используйте клавишу "T".

УДАРНИКИ

Для работы с ударниками используются клавиши "Y", "U" и "I". Редактирование звучания этих клавиш выполняется после нажатия на клавишу "8". При этом на экране появляются два ряда прямоугольников, по три в каждом, под

которыми находятся три буквы - "Y", "U" и "I".

В нижнем ряду находится мигающий курсор. Его можно передвигать с помощью клавиш управления движением курсора ("5"..."8"). В начальный момент курсор находится над буквой "Y" в нижнем ряду. В этом прямоугольнике (как и в соседних по горизонтали) показана частота ударников в виде частотной кривой. Она может изменяться (нажатием на клавишу "0") до максимума, а затем вновь становится минимальной. Установите требуемую частоту для данного ударника, а затем переместите курсор на следующий ударник.

Верхний ряд прямоугольников определяет прерывистость звучания ударников. Для этого используется клавиша "0".

Для возврата в редактор нажмите любую другую клавишу.

Кроме вышеуказанных WHAM! имеет еще один шумовой эффект (на 2-ом канале) - это клавиша "E".

ПРОЧИЕ КЛАВИШИ

В редакторе используются еще некоторые функциональные клавиши:

W - петля; когда WHAM! доходит до нее, то он автоматически возвращается к началу. Петлю надо делать на каждом канале отдельно. При нажатии на "W" возникает вопрос, на который необходимо ответить "Y" - если петля нужна и "N" - если петлю надо удалить;

7 - очистка буфера редактора;

5 - установка цвета бордера.

Если возможности ритм-бокса WHAM! Вам недостаточны, имеется более мощный специализированный ритм-бокс фирмы EINSTEIN SOFTWARE. Эта программа имеет уже 10 ударных инструментов, используемых в различных комбинациях, и запоминает до 10 ритмов произвольного размера и длины. Имеется 10 готовых ритмов. Ритмы можно переключать прямо в процессе исполнения.

Есть синтезаторы речи - LMOWA или TUKER, позволяющие компьютеру произносить слова и фразы.

13. ВСТРОЕННЫЙ ТАЙМЕР

Ваш компьютер обладает широкими возможностями по управлению различными процессами и объектами в реальном времени.

Скорость реакции Вашего компьютера весьма велика и позволяет, например. управлять небольшим промышленным роботом с электроприводом или же программно имитировать работу контроллеров таких устройств, как графический принтер или графопостроитель. В случае графического принтера компьютер одновременно управляет двумя электодвигателями, одним шаговым двигателем и игольчатой головкой, и имеет значительный запас по скорости, что позволяет ему одновременно с печатью еще и оптимизировать процесс печати, программно корректировать люфты печатающего устройства и регулировать контрастность печати.

Для работы в реальном масштабе времени часто необходим датчик времени - таймер, позволяющий отсчитывать временные интервалы.

ТЕХНИЧЕСКАЯ ХАРАКТЕРИСТИКА ТАЙМЕРА:

Дискретность отсчета времени ..... 0,02 сек.(50 Гц)
Режим работы ..... маскируемое прерывание
Стабилизация частоты ..... кварцевым генератором
Максимальный фиксируемый интервал ..... 328 000 секунд
Программная установка ..... есть
Программная блокировка ..... есть

Этот таймер используется ОС для обслуживания клавиатуры, что не мешает использовать его Вам - только помните, что блокировка таймера приводит к отключению клавиатуры от ОС. Однако сама клавиатура остается работоспособной и может использоваться Вашими программами (при этом к ней надо обращаться как к портам ввода). При блокировке таймера его показания не сбрасываются, но фиксируются - фактически таймер продолжает работать, но подсчет временного интервала прекращается. При снятии блокировки это позволит Вам продолжать подсчет временного интервала с того места, на котором Вы его прервали.

Для блокировки таймера достаточно всего лишь запретить процессору прерывания командой DI (шестнадцатиричный код команды F3) - и таймер остановится. Разрешив прерывания командой EI (шестнадцатиричный - FB), Вы разблокируете таймер.

Пример использования встроенного таймера приведен в разделе "Основы программирования на языке Бейсик" при описании оператора PEEK на стр. 65.

14. СИСТЕМНЫЕ ПРОГРАММЫ

С некоторыми из таких программ Вы уже познакомились в разделах синтезатор звуковых эффектов. На самом деле таких программ очень много, как и возможностей для их применения.

ЯЗЫКИ ПРОГРАММИРОВАНИЯ:

Бейсик-интерпретатор - BETA-B.48, BETA 3.1, LASER-BASIC, MEGA-BASIC
Бейсик-компиляторы - F-COMPILER, FP 48 K V1.7, TOBOS-FP
Паскаль - PASCAL HP-80, HP-4 TM 8, HP-4 TM 16
Си - HISOFT C
Форт - FIG-FORTH, ED 50, EP 50
Лого - LOGO
Пролог - PROLOG
Ассемблер - GENS-4, ZEUS, ASSEMBLER, EDI 1 AS
Отладчик кодов - MONS-4, MONITOR 48, MONITOR 16, IR 48K

БАЗЫ ДАННЫХ:

Картотеки - MASTERFILE V 09, TOOLKIT
Электронные таблицы - OMNYCALK

ТЕКСТОВЫЕ ЭКРАННЫЕ РЕДАКТОРЫ:

TASWORD TWO, TASWORD 3, TASWORD CS, TAS.RUS, FILES

ГРАФИЧЕСКИЕ РЕДАКТОРЫ:

ART STUDIO, ARTIST, DINAMIC-3, M.DRAW, LP MK 2.3

Все графические редакторы, кроме последнего, работают с клавиатурой, джойстиком или манипулятором "мышь". Последний графический редактор работает со световым пером.

ИМИТАТОРЫ ДИСКОВОЙ ОПЕРАЦИОННОЙ СИСТЕМЫ:

RAM DOS 2

ТЕСТЫ СИСТЕМЫ:

TEST-PROG, TEST-BAS, MEMORY TEST, TST 80.V3

ТРАССИРОВЩИКИ ПЕЧАТНЫХ ПЛАТ:

PLATA V.3, ТРАССА-3

СПЕКТРОАНАЛИЗАТОРЫ И ЦИФРОВЫЕ ОСЦИЛЛОГРАФЫ:

OSCILLATOR, TARE'R, DIAG, TAPE HEAD

МАТЕМАТИЧЕСКИЕ ПРОГРАММЫ:

FUN F(X,Y), STATIST, FOURIER

РЕДАКТОРЫ ЗНАКОГЕНЕРАТОРА ПОЛЬЗОВАТЕЛЯ:

UDG DEFINER, PAINTBOX, UDG

СИНТЕЗАТОР РЕЧИ:

FONGEN, LMOWA, SPEAKER, SPEAKEASY

ПАКЕТЫ ДЛЯ БЕЙСИКА:

ZXED, COMPRESSOR, 64*32, RUS, RENUM, 64 COLOMNY

КОПИРОВЩИКИ:

COPY 86/M, TF COPY, COPY-COPY

15. СТРУКТУРА КОМПЬЮТЕРНОЙ ИГРЫ

С точки зрения пользователя, компьютерная игра представляет собой несколько файлов, записанных на кассете для магнитофона. Типы, размеры и количество файлов варьируется, однако можно выделить три основных типа структур:

загрузчик на Бейсике - блок (блоки) кодов

загрузчик на Бейсике - загрузчик в кодах - блок (блоки) кодов

программа на Бейсике со встроенными кодами

Примером первой структуры может служить игра LUNA CRABS, второй - SILENT SERVICE, третьей - игра WHEELIE 2.

Наиболее общим является второй тип структуры - его и рассмотрим подробнее. Начнем с первого файла типа PROGRAMM. Если его загрузить не командой LOAD "", а командой MERGE "", то можно увидеть примерно следующую программу - загрузчик на Бейсике:

```
0 ... : LOAD "..." SCREEN$ : LOAD "..." CODE : RANDOMIZE USR ...
```

Здесь команда LOAD "..." SCREEN$ загружает на экран картинку-заставку, рассматривая которую Вам будет легче ждать конца загрузки всего остального. Следующая команда LOAD "..." CODE загружает загрузчик в кодах, и последняя команда RANDOMIZE USR ... его запускает.

Как видите, достаточно загрузить такую программу (она, как правило, имеет атрибут самозапуска и сама начинает работать), и все остальные файлы будут загружены и запущены без всякого вмешательства, аналогично тому, как на компьютере IBM PC при запуске пакетного файла (.BAT-файлы) он сам начинает загружать и запускать нужные Вам программы в нужном порядке.

Обратите внимание на номер строки. Набрать такую строку с клавиатуры невозможно, как, впрочем, и отредактировать. "Нулевая" строка формируется специальным образом и служит для того, чтобы не дать Вам забраться в программу. Обычно кроме зануления строки применяются еще и специальные меры, чтобы Вы не увидели на экране текст программы.

16. ПОРЯДОК ПОДКЛЮЧЕНИЯ И РЕКОМЕНДАЦИИ ПО РАБОТЕ С ПРИНТЕРОМ

По-видимому, принтер - это наиболее желаемое периферийное устройство, перспективу возможности приобретения и подключения которого к СПЕКТРУМу имеет в виду большинство пользователей. К СПЕКТРУМу можно подключить практически любой принтер, но при этом могут возникать те или иные трудности.

Прежде всего надо определиться для чего он Вам нужен. Если основное направление Вашей работы - программирование, то Вам может подойти небольшой узкопечатный ZX-принтер, который подключается к разъему компьютера без каких-либо интерфейсов и не требует никакого программного обеспечения для работы (все уже содержится в ПЗУ компьютера). Такой принтер имеет ширину бумажной ленты 10...13 см и выводит 32 символа в строке. Он также может выполнять графическую копию экрана. Качество печати далеко не идеальное, но такие устройства сравнительно дешевы, просты и удобны в работе и, если их применять в основном для распечатки текстов программ или результатов расчетов, то вполне соответствуют поставленным задачам.

Если же Вам необходимо получение качественных документов на листах стандартного формата, необходимо применять полноразмерные принтеры. Полноразмерные принтеры выпускаются двух типов - это точечно-матричные принтеры и принтеры типа "ромашка". Разница в конструкции печатающего элемента. В первом случае это набор иголок, оставляющий через красящую ленту мозаичный отпечаток на бумаге. Из точек складывается изображение символа. В принтерах второго типа печатающие литеры закреплены на вращающемся прорезном диске ("ромашке"). Этот диск может быть сменным для замены набора литер.

Разница в конструкции вызывает и различия в эксплуатации. Матричные принтеры значительно быстрее работают, могут гибко (программно) перестраиваться с одного шрифта на другой, но по качеству печати только самые лучшие экземпляры могут приближаться к принтерам типа "ромашка".

В основном принтеры "ромашка" применяются в тех случаях, когда имеется необходимость в большом количестве деловой переписки, и качество исполненного документа свидетельствует об уровне представительства. Надо также отметить, что выдача графических изображений на принтерах этого типа исключена.

Наиболее гибким и многофункциональным устройством является точечно-матричный принтер. Далее мы остановимся на этих устройствах и отметим, какие вопросы Вы должны для себя решить, прежде чем приобретать и подключать такой принтер.

ТИП ИСПОЛЬЗУЕМОГО ИНТЕРФЕЙСА

Этот пункт является весьма важным, т.к. от него зависит способ подключения к компьютеру. Существуют два основных метода обмена информацией между компьютерами и периферийными устройствами - параллельный и последовательный. При параллельном подключении все биты, составляющие байт посылки, передаются одновременно по параллельным шинам данных. При последовательном байт передается бит за битом по очереди. Исторически за интерфейсами, реализующими параллельный метод передачи, закрепилось название "Цетроникс", а за последовательными - RS232.

Большинство современных компьютеров уже имеют встроенный интерфейсный порт того или иного типа, но СПЕКТРУМ-48 его не имеет. Поэтому, приобретая принтер, надо подумать и об интерфейсе. Наибольшее распространение получили следующие модели интерфейсов:

1. **ИНТЕРФЕЙС-1.** Мы о нем уже упоминали. Он имеет последовательный порт RS232, к которому может подключаться принтер, имеющий такой же интерфейс. Никакой программной поддержки этот интерфейс не требует, она находится в его ПЗУ.

2. **ZX-LPRINT III.** Это интерфейс может настраиваться и поддерживает аппаратуру как с интерфейсом "Цетроникс", так и с RS232.

3. **TASPRINT** - интерфейс типа "Цетроникс". Требует специальной программы (драйвера) для своей работы.

4. **KEMPSTON-E** - интерфейс типа "Цетроникс", также требует драйвера.

Компьютеры "СПЕКТРУМ+128" и "СПЕКТРУМ+2" имеют уже встроенный порт RS232.

Зато они не могут работать с ZX-принтерами, т.к. область буфера принтера там занята дополнительными системными переменными, хотя существуют программные пути устранения этого недостатка.

Обратим Ваше внимание однако на то, что использовать этот порт непросто. Дело в том, что он поддерживается только в режиме 128К, а наибольшая часть прикладных программ (текстовые редакторы, графические редакторы и пр.), с которыми хотелось бы использовать принтер, работают в режиме 48К.

НАЛИЧИЕ ДОПОЛНИТЕЛЬНОГО НАБОРА ЗНАКОВ

Далеко не все импортные принтеры имеют встроенный русский набор знаков, что необходимо учитывать. Наилучшие модели имеют возможность загрузки шрифта от компьютера.

ПРИМЕНЯЕМАЯ БУМАГА

Некоторые принтеры используют термобумагу, что создает весьма ощутимые неудобства в связи с отсутствием ее в широкой продаже. Кроме того, качество исполнения документа в этом случае ниже.

КРАСЯЩАЯ ЛЕНТА

Многие принтеры работают с лентой, находящейся в специальных неразборных кассетах. Это дает удобство при смене ленты, но может поставить Вас в тупик, если доступа к таким кассетам Вы не имеете и применение обычной ленты на катушках конструктивно невозможно.

СКОРОСТЬ ПЕЧАТИ

Если Вы предполагаете исполнение многочисленных копий документов, этот параметр может быть важным. Обычно точечно-матричный принтер имеет скорость печати 80...100 зн./сек. Даже если этот параметр Вам безразличен, имейте в виду, что чем он выше, тем совершеннее принтер. Наилучшие модели имеют скорость выше 180 зн./сек.

КОЛИЧЕСТВО ИГОЛОК В ПЕЧАТАЮЩЕЙ ГОЛОВКЕ

Как правило, принтер среднего качества имеет 9 иголок в печатающей головке. Если у него только 7 иголок, то некоторые буквы, например y, j, p могут лишиться своих нижних элементов. Наиболее совершенные модели имеют до 24 иголок.

НАЛИЧИЕ РЕЖИМА NLQ

Режим NLQ - NORMAL LETTER QUALITY обеспечивает печать символами, приближающимися по качеству к стандартной пишущей машинке. Такой режим имеется далеко не у всех принтеров, и печать в этом режиме выполняется значительно медленнее, зато он позволяет получать представительные документы.

КОДЫ УПРАВЛЕНИЯ ПРИНТЕРОМ

Принтер может не только распечатывать текст, который поступает в него из компьютера, но выполнять дополнительно и целый ряд специфических действий (подчеркивать текст, переключаться с одного набора знаков на другой, изменять размеры шрифта, выполнять печать двойным ударом и еще многие другие действия). Эти переключения делаются с помощью управляющих кодов, которые в нужных местах вставлены в текст. Как правило, управляющие коды состоят из последовательности кодов, начинающейся с символа 27 (ESCAPE), поэтому часто коды управления принтером называют ESCAPE-кодами. В зависимости от того, какому коду какое действие принтера соответствует, различают несколько стандартов этих кодов. Наиболее широко распространена система кодов фирмы EPSON, которая применяется на принтерах этой фирмы. В зависимости от того, поддерживает ли принтер этот стандарт кодов, он может быть или не быть EPSON-совместимым. Желательно, чтобы такая совместимость была, тогда Вы сможете распечатывать подготовленные документы не только на своем принтере, а также упростить задачу настройки фирменных программ на работу с принтером, т.к. они, как правило, готовятся в расчете на EPSON-совместимые принтеры.

DIP- ПЕРЕКЛЮЧАТЕЛИ

Эти переключатели служат для задания определенных режимов работы принтера. Многие их функции дублируются и могут быть изменены кодами управления принтером, но те условия, которые заданы этими переключателями и являются исходными в момент включения принтера. Чем их больше, тем больше возможностей имеет принтер. Нормально их должно быть порядка 24 шт.

БУФЕР ПРИНТЕРА

Принтер должен иметь определенный обьем свободной оперативной памяти, где он размещает пакет, поступающий от компьютера, и выполняет с ним преобразования (перед выводом на печать). Именно наличие буфера позволяет принтеру, например, печатать и при обратном проходе головки (справо налево), что вдвое повышает его быстродействие. При наличии достаточно емкого буфера компьютер очень быстро может перебросить в него выводимый текст, и Вы сможете, например, готовить к печати очередной блок, пока принтер ведет печать. Если же емкость буфера мала, компьютеру приходится все время подгружать буфер,и Вы не можете использовать его для других целей. В принципе, для большинства практических задач емкость порядка 0.5К достаточна, но лучше, если она больше 2К, а совершенные модели имеют буфер больше 4К.

ВЫБОР ТЕКСТОВОГО РЕДАКТОРА

Текстовые редакторы представляют наиболее удобные условия для работы с принтером, поэтому, предполагая подключение принтера, надо иметь в виду и необходимость наличия редактора. Наиболее широко распространены различные версии текстового редактора TASWORD. Эта программа отличается простотой в работе, предоставляет возможность изображения на экране как 32 символов в строке, так и 64-х. Имеет несколько кодов управления принтером, которые вставляются в текст в виде символов блочной графики. Достаточно просто переделывается на работу с русским или другим национальным шрифтом.

Наиболее совершенным и гибким в работе является редактор THE LAST WORD 2. Среди программ для бытовых компьютеров любых моделей трудно найти сравнимый по возможностям. Он может воспроизводить текст на экране в режимах 40, 48, 60 и 80 знаков в строке. Имеет очень удобное меню настройки условий печати, обширный набор операторов управления принтером, который к тому же может гибко перестраиваться без выхода из программы. Имеет встроенный таймер и калькулятор, средства поиска информации и мощные средства для оформления текста и составления комплексных документов.

Строго говоря, это не просто программа, а пакет программ, т.к. к нему прилагаются дополнительные файлы, позволяющие перенастроить редактор на работу в дисковых системах "TR DOS", "OPUS", "GORDON" и др., а также дополнительные программные средства для обеспечения возможности работы с текстами, выполненными в других редакторах и базах данных.

ПАРАЛЛЕЛЬНЫЙ ИНТЕРФЕЙС ПРИНТЕРА

Рассмотрим два наиболее распространенных параллельных интерфейса - ИРПР-М (CENTRONICS) и ИРПР. Чаще всего в компьютерах используется упрощенный вариант CENTRONICS - с минимально достаточным числом линий связи. Эти линии, названия сигналов, а также два основных типа разъемов принтера отображены в таблице 1.

AMPHENOL 36 контактов	CANON 25 контактов	Обозначение сигналов
1	1	STROBE - СТРОБ - при логическом нуле данные на линиях Д0-Д7 действительны, при логической единице - недействительны. Вход принтера.
2 - 9	2 - 9	Д0-Д7, данные, вход принтера
11	11	BUSY - ЗАНЯТ - лог."0"/"1" означает, что принтер может/не может принимать данные. Выход принтера.
16, 19 - 30	18 - 25	Земля, 0В.

Остальные линии интерфейса в упрощенном варианте не используются. Передача данных от компьютера к принтеру осуществляется по линиям Д0-Д7 с помощью сигналов СТРОБ и ЗАНЯТ. Принтер выставляет сигнал ЗАНЯТ в лог. "1" в случаях ввода данных, состояния "ошибки", переполнение буфера и т.п. Компьютер по этому сигналу приостанавливает передачу данных и не передает импульс с лог. "0" по линии СТРОБ до тех пор, пока принтер не перейдет в состояние готовности, т.е. лог. "0" на линии ЗАНЯТ. Принтер считывает данные с линий Д0-Д7 только после того, как сигнал СТРОБ принял состояние лог. "0". Длительность импульса СТРОБ с лог. "0" не менее 0,5 мкс.

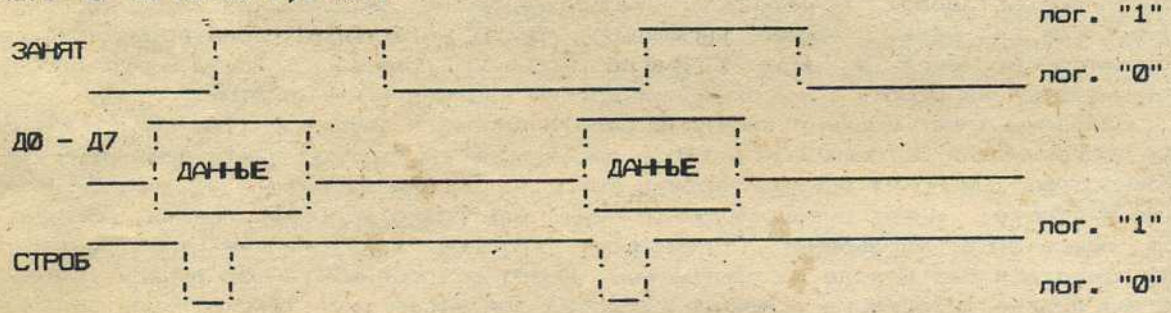
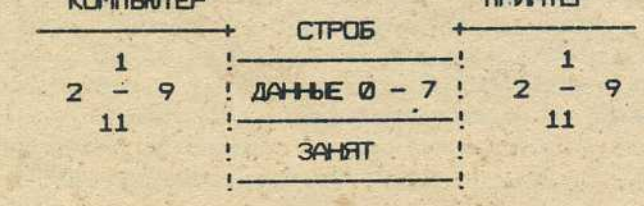


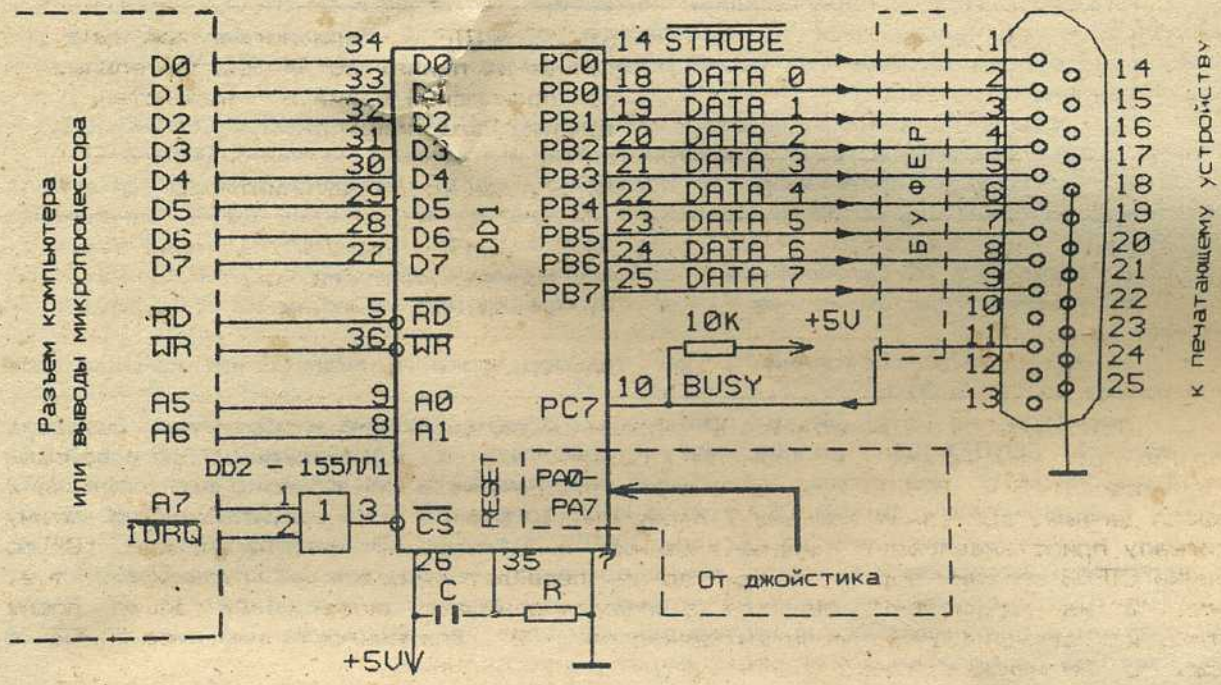
Диаграмма обмена данными в ИРПР-М (CENTRONICS)

Для реализации интерфейса типа CENTRONICS удобно использовать микросхему КР580ВВ55А, которая представляет собой программируемый интерфейсный адаптер (ПИА). В состав ПИА входят три 8-ми разрядных параллельных порта ввода/вывода А, В, С. В нашей схеме порт В используется для ввода информации DATA 0-7 на вход принтера, по младшему разряду порта С - РС0 передается сигнал СТРОБ, а старший разряд - РС7 служит для ввода сигнала ЗАНЯТ принтера. Порт А зарезервирован под джойстик. Программа поддержки интерфейса заносит в управляющий регистр ПИА число 136, что соответствует настройке ПИА в режим вывода по порту В и 4-м младшим разрядам порта С, и ввода по 4-м старшим разрядам порта С.

Заметим, что после включения питания, в результате RESETа, все порты ПИА настраиваются в режим ввода информации. Кроме того, адресация порта А выбрана таковой, что позволяет использовать его в качестве порта джойстика, который может быть подключен, как показано на рис. 8. Джойстик будет выбираться при нуле на шинах А5, RD, IORQ, что соответствует адресации KEMPSTON интерфейса.

Принтер подсоединяется к интерфейсу с помощью кабеля длиной 1,5-2 метра.





C - 10 мкФ/6В DD1 - 580ВВ55А
R - 120 кОм

Рис.7

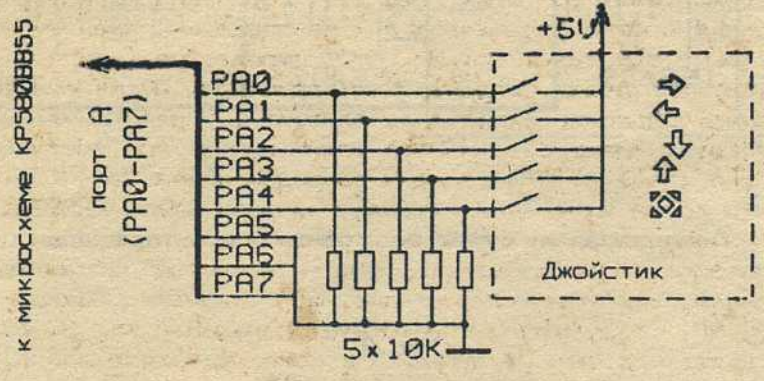


Рис.8

Для увеличения нагрузочной способности на выходе интерфейса можно включить буферный элемент, в качестве которого можно использовать любые ТТЛ-микросхемы без инверсии, например, 155ЛП4, 155ЛП10, 155ЛП11, 555АП5, 555АП6, 589АП16 и т.п.

Интерфейс ИРПР - параллельный интерфейс, отличающийся от CENTRONICS, как полярностью сигналов, так и протоколов обмена. В упрощенном варианте для работоспособности интерфейса, кроме 8-ми информационных шин DATA 0-7, необходимы два управляющих сигнала:

CS - строб источника, аналогично сигналу СТРОБ интерфейса CENTRONICS, в низком уровне свидетельствует о достоверности информации на шинах DATA 0-7;

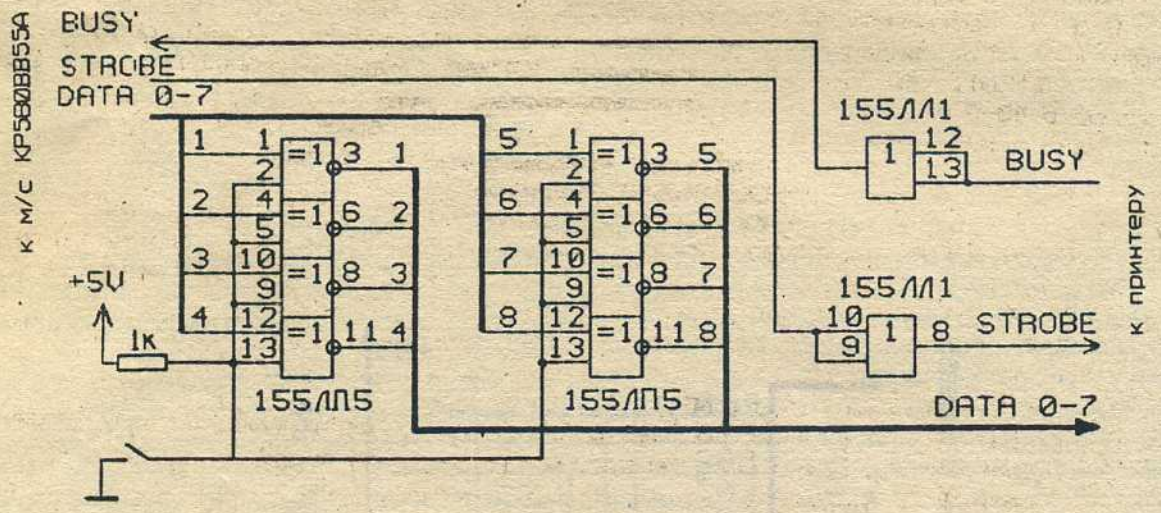
AC - запрос приемника, в состояниях 0/1 сигнализирует о готовности неготовности принтера к приему данных, т.е. аналогичен сигналу ЗАНЯТ интерфейса CENTRONICS.

Сигналы CS и AC связаны между собой следующим образом:

SC изменяет состояние из "1" в "0" только при AC=0, а из "0" в "1" при AC=1;

AC остается в состоянии "1" до тех пор, пока сигнал CS не изменит свое состояние из "0" в "1".

Несмотря на отличия в протоколах обмена, схема и программа поддержки интерфейса CENTRONICS вполне могут работать с принтерами, оснащенными интерфейсом ИРПР. Для этого необходимо инвертировать информационные шины DATA 0-7 и буферизировать сигнал CS, т.к. в ИРПР требуются токи нагрузки около 40 мА. Сигнал AC используется вместо сигнала ЗАНЯТ, а сигнал CS вместо сигнала СТРОБ. В качестве инверторов и буферов можно применять микросхемы "Исключающее ИЛИ" - 155ЛП5. В этом случае, подавая на вторые входы элементов микросхем "1" или "0", Вы получите на выходах инвертированный или неинвертированный сигнал, таким образом Ваш параллельный интерфейс будет универсальным.



Универсальный буфер параллельного интерфейса

Рис.9

Программа поддержки интерфейса CENTRONICS, приведенная ниже, обеспечивает выполнение команд LLIST, LPRINT, а также, по команде "RANDOMIZE USR 23370", позволяет получать "твердую" копию текста с экрана.

C E N T R O N I C S

```
1 OUT 127,136: OUT 95, 1: LET a=PEEK 23631: LET b=PEEK 23632: LET
c=a+256*b+15: POKE c,0: POKE c+1,91
2 LOAD "CENTRONICS"CODE
3 CLS : PRINT  FLASH 1:"          +++ CENTRONICS +++          "
4 PRINT AT 5,2: "RANDOMIZE USR 23370 for COPY"
5 PRINT FLASH 1:AT 10,7:"H T K  + P L U S + ": STOP
6 SAVE "CENTRONICS LINE 1: SAVE "CENTRONICS CODE 23296,256
```

```
5B00  FE 06 CA 19 5B FE 16 28
5B08  04 FE 17 20 2C E1 E1 D9
5B10  E1 F5 1F 30 01 41 78 18
5B18  06 F5 3A FE 5B CB 3F 21
5B20  FF 5B 96 38 0B E1 CB 47
5B28  3E 20 CD B6 5B 10 F9 C9
5B30  CD F8 1F F1 FE 06 CB 18
5B38  EF FE A5 38 05 D6 A5 C3
5B40  10 0C FE 80 38 70 06 01
5B48  18 DE 0E 00 06 00 C5 C5
5B50  D5 78 CD B0 22 EB CD 7C
5B58  5B C1 D1 CD B6 5B 79 C6
5B60  0B 4F FE 00 28 04 C1 4F
5B68  18 E4 F1 78 C6 08 47 3E
5B70  0D CD B6 5B 3E A8 B8 38
5B78  02 18 D3 C9 21 00 3D 01

5B80  20 60 C5 D5 E5 06 0B 1A
5B88  FE FF 20 0C 1A EE FF BE
5B90  20 10 23 14 10 F6 18 19
5B98  1A BE 20 06 23 14 10 F8
5BA0  18 0F E1 D1 0C 79 01 08
5BA8  00 09 C1 4F 10 D4 3E 20
5BB0  C9 79 E1 D1 C1 C9 FE 0D
5BB8  20 07 E5 21 FF 5B 36 00
5BC0  E1 FE 20 38 11 F5 E5 21
5BC8  FF 5B 7E 2B BE 3E 0D 00
5BD0  00 00 23 34 E1 F1 C5 F5
5BD8  01 BF E2 CD 54 1F D2 00
5BE0  0D DB 5F 17 38 F5 F1 05
5BE8  05 D3 3F 06 E3 F5 3E 0E
5BF0  D3 5F 3C D3 5F F1 C1 FE
5BF8  0D C0 3E 0A 18 D8 50 00
```

Программа состоит из 2-х частей: на Бейсике и в машинных кодах Z80. После старта, Бейсик-программа загружает коды и настраивает интерфейс. Коды приведены в распечатке содержимого памяти в шестнадцатиричном (HEX) виде. Каждая строка распечатки начинается с четырехзначного HEX адреса первого из восьми HEX байтов, напечатанных в этой строке.

Если Ваш принтер поддерживает графический режим работы, то его можно использовать для распечатки графических копий экрана Вашего СПЕКТРУМа. Для этого служит программа "COPY", приведенная ниже. Программа рассчитана на работу с наиболее распространенными, EPSON-совместимыми принтерами.

C O P Y

```
5 CLS: PRINT FLASH 1:AT 10,7:"H T K  + P L U S + "
6 PRINT AT 20,2: "RANDOMIZE USR 23296 for COPY"
10 LOAD "COPY"CODE
20 PAUSE 0: STOP
30 SAVE "COPY" LINE 1: SAVE "COPY" CODE 23296,256
```

```
5B00  3E 88 D3 7F 3E FF D3 5F
5B08  00 00 00 00 00 21 F4 5B
5B10  CD A5 5B 06 18 97 CD 28
5B18  5B 3E 14 CD F7 5B CC 28
5B20  5B 05 78 FE 02 20 EE C9
5B28  21 37 5B 77 CD B4 5B C5
5B30  78 CD EE 0D EB 06 04 14
5B38  10 FD CD 4B 5B C1 0D 79
5B40  FE 01 20 EB 21 CF 5B CD
5B48  A5 5B C9 21 6F 5B 23 36
5B50  7F D5 E5 21 73 5B 23 36
5B58  F9 23 23 36 F1 CD F7 5B
5B60  28 02 36 F9 2B 2B 0E 00
5B68  06 04 28 02 06 08 1A CB
5B70  3F 28 04 CB B9 CB B1 CD
5B78  F7 5B C5 06 10 28 02 06

5B80  08 7E 90 77 23 23 7E 90
5B88  77 2B 2B C1 14 10 DF 79
5B90  CD D1 5B 79 CD F7 5B CC
5B98  D1 5B E1 7E D6 08 77 FE
5BA0  47 D1 30 AD C9 7E FE 80
5BA8  38 04 D6 80 18 23 CD D1
5BB0  5B 23 18 F1 CD E9 5B DE
5BB8  21 21 C6 5B CD F7 5B 2B
5BC0  E4 21 CB 5B 88 DF 18 2A
5BC8  05 00 82 1B 4B 00 51 0D
5BD0  8A F5 DB 5F 17 38 FB F1
5BD8  D3 3F 3E FE D3 5F 3E FF
5BE0  D3 5F 00 00 00 00 00 00
5BE8  C9 0E 05 0D C8 3E 20 CD
5BF0  D1 5B 18 F7 18 33 97 E5
5BF8  21 FF 5B CB 46 E1 C9 00
```

Программа COPY вводится в память компьютера аналогично программе "CENTRONICS". Для получения графической копии экрана необходимо выполнить команду "RANDOMIZE USR 23296". Программу можно модифицировать:

- POKE 23551,1 - уменьшенная копия экрана;
- POKE 23551,0 - увеличенная копия экрана;
- POKE 23530,x - где x - величина отступа от левого края;
- POKE 23542,y - где y от 151 до 156 - размер по вертикали.

Программы "CENTRONICS" и "COPY" располагаются в области буфера, поэтому при выполнении команды "NEW", они уничтожаются.

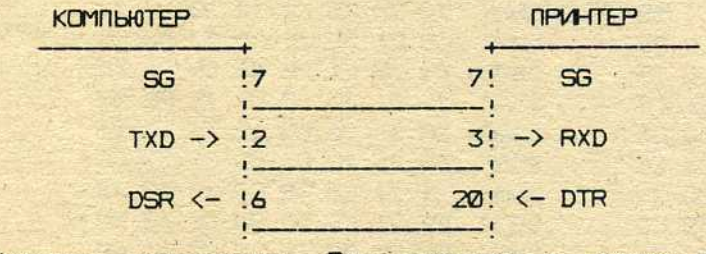
Следует отметить, что коды управления графическим режимом работы принтеров разных фирм часто не совпадают, поэтому, для работы с другими моделями принтеров, программа должна быть изменена.

ПОСЛЕДОВАТЕЛЬНЫЙ ИНТЕРФЕЙС - RS-232C

C2 (RS-232C) - самый распространенный последовательный интерфейс как синхронной и асинхронной связи с периферийными устройствами, в дуплексном и полудуплексном режимах обмена данными. Для передачи информации и сигналов управления в этом интерфейсе используется двуполярное напряжение от - (3-12)V до + (3-12)V. Полный интерфейс содержит более десятка сигналов, тогда как для подключения принтера достаточно использовать лишь некоторые из них. В таблице приведено распределение основных сигналов интерфейса RS-232C по контактам стандартного 25-ти контактного разъема типа CANNON DB-25P.

NN контакта	Сигнал	Назначение сигнала	Направление
1	PG	Корпус	——
2	TXD	Передаваемые данные	Выход
3	RXD	Принимаемые данные	Вход
4	RTS	Запрос на передачу	Выход
5	CTS	Готовность к передаче	Вход
6	DSR	Готовность компьютера	Вход
7	SG	Сигнальное заземление	——
8	DCD	Контроль приема	Вход
20	DTR	Готовность терминала	Выход

Связь компьютера и принтера в стандарте RS-232C может быть осуществлена либо по протоколу программного управления обменом - XON-XOFF (дуплексном или полудуплексном), либо с помощью аппаратно управляемого протокола DTR. Предлагаемый Вашему вниманию интерфейс RS-232C поддерживает протокол DTR. Сопряжение по этому протоколу приведено ниже.



Необходимо переключить Ваш принтер в режим работы по протоколу DTR, скорость передачи данных 1200 бод, без контроля четности, с одним стоп-битом и автоматическим переводом строки при возврате каретки. Для настройки принтера обычно применяются микропереключатели, положения которых описано в инструкции по эксплуатации принтера.

Передача данных происходит под управлением сигнала DTR. После включения и инициализации, принтер переводит линию DTR в состояние "Включено", т.е. + (3-12)V. Компьютер через линию DSR интерфейса RS-232C, опрашивает линию DTR и,

если она включена, начинает последовательную передачу данных по линии TXD на линию RXD-принтера. Когда буфер данных принтера заполняется, принтер запрещает компьютеру передачу данных , переводя линию DTR в состояние "Включено", т.е. - (3-12)V. После освобождения буфера данных, принтер снова включает линию DTR. Линия DTR не включается, если принтер находится в состоянии "OFFPLINE" (при вмешательстве оператора, конце бумаги или аварии).

Схема интерфейса приведена на рис.10. Интерфейс представляет собой порт с адресом 63DEC (3P HEX), для ввода/вывода информаций по младшему разряду шины данных - D0. Для ввода данных необходимо использовать какую-либо микросхему с третьим состоянием и инверсией, например 555АП3 и т.п. На выходе можно использовать любые маломощные транзисторы. Особенностью схемы является необходимость двуполярного питания выходного каскада. При отсутствии у Вас источника отрицательного напряжения, можно собрать несложный преобразователь напряжения, как показано на рис 11.

Программа поддержки интерфейса RS-232C позволяет выполнять команды СПЕКТРУМа LLIST и LPRINT. Коды программы располагаются в верхних адресах памяти и инициализируются командой RANDOMIZE USR 65100.

Для получения графических копий экрана можно воспользоваться программой "COPY". Для этого, после загрузки и инициализации программы "RS-232C" нужно загрузить программу "COPY" и выполнить в ней следующие изменения:

POKE 23505,205: POKE 23506,114: POKE 23507,254: POKE 23508,201

Кроме того, в отличие от работы с текстами, в принтере должен быть выключен режим автоматического перевода строки (LF) при возврате каретки (CR). Команда печати графической копии экрана та же, что и для параллельного интерфейса - RANDOMIZE USR 23296. Остаются в силе и модификации программы "COPY".

R S - 2 3 2 C

```
10  CLEAR 65099
20 LOAD "RS-232C"CODE 65100
30 RANDOMIZE USR 65100
40 STOP
50 SAVE "RS-232C" LINE 10: SAVE "RS-232C"CODE 65100,105
```

```
FE48                  21 9E A1 22     FE88  A9 FE D1 1E 08 CB 1A 38
FE50  1C 5C C9 58 FE C4 15 52         FE90  06 3E 0F D3 BF 18 04 00
FE58  FE 17 20 04 3E 09 18 12         FE98  AF D3 BF CD A9 FE 1D 20
FE60  FE 80 38 0E FE A5 30 04         FEA0  EC AF D3 BF CD A9 FE FB
FE68  3E 3F 18 06 D6 A5 CD 10         FEA8  C9 C5 01 01 D9 10 FE 0D
FE70  0C C9 F5 CD 54 1F 38 05         FEB0  20 FB C1 C9 00 5A CD 44
FE78  F1 CF 0C 18 2A DB BF 1F         FEB8  F8 CD A0 FA
FE80  38 F1 F3 3E 0F D3 BF CD
```

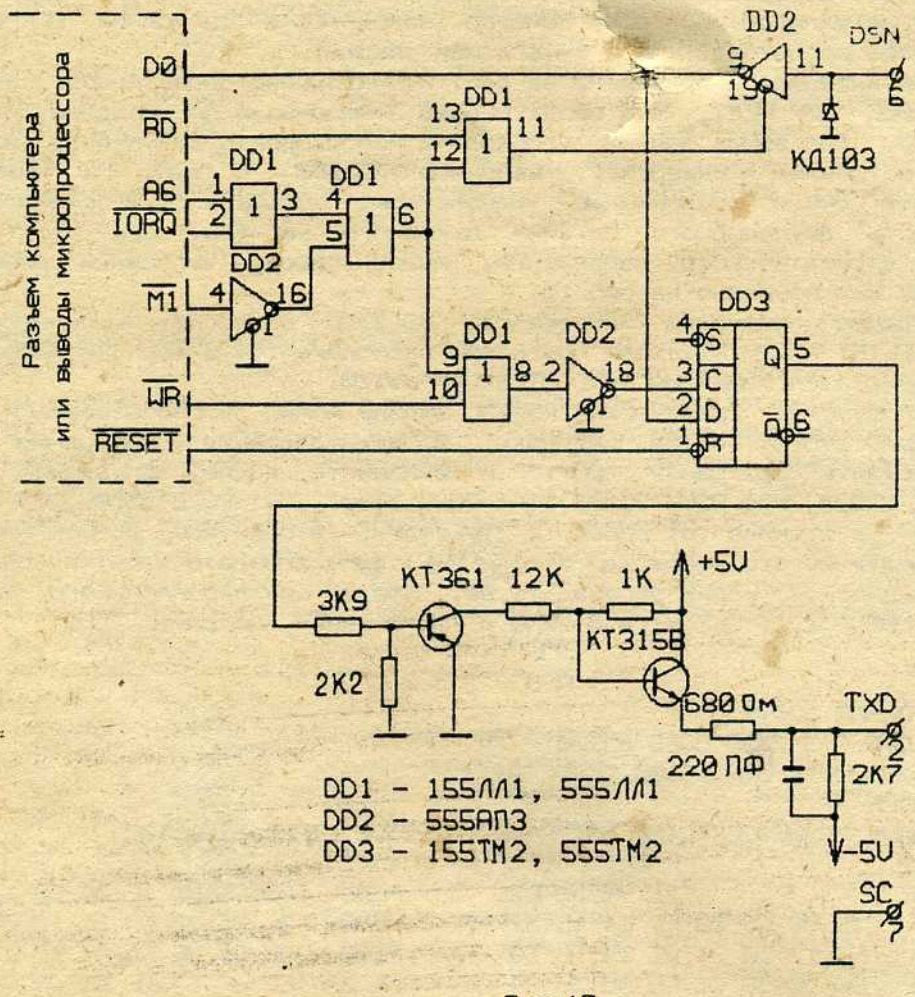
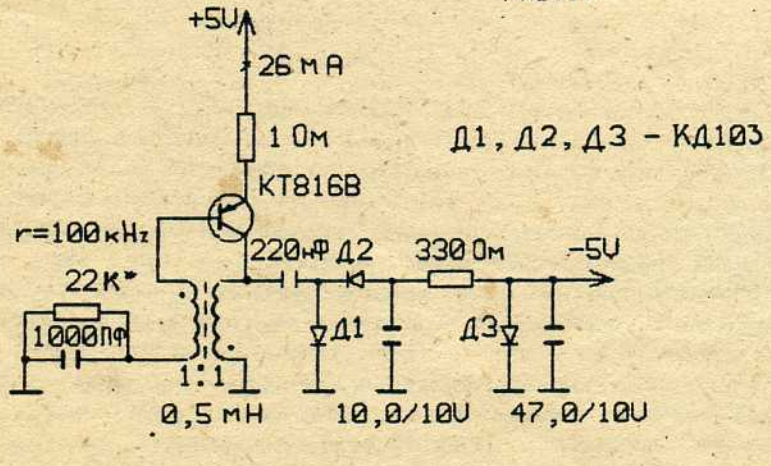

Разъем компьютера
или выводы микропроцессора
D0
RD
A6
IORQ
M1
WR
RESET
DD1
DD2
DD3
DSN
КД103
КТ361
12К
1К
+5U
3К9
2К2
КТ315Б
680 Ом
TXD
220 ПФ
2К7
-5U
SC
DD1 - 155ЛЛ1, 555ЛЛ1
DD2 - 555АП3
DD3 - 155ТМ2, 555ТМ2

Рис.10

Рис.11


+5U
26 мА
1 Ом
КТ816В
Д1, Д2, Д3 - КД103
r=100кHz
220мФ
Д2
330 Ом
-5U
22К*
1000пФ
Д1
Д3
1:1
0,5 мН
10,0/10U
47,0/10U

17. ПОРЯДОК ПОДКЛЮЧЕНИЯ И РЕКОМЕНДАЦИИ ПО РАБОТЕ С ДИСКОВОДОМ

Многие пользователи ZX-SPECTRUMa уже столкнулись с неудобством ввода и хранения информации на магнитной ленте. Это прежде всего низкая надежность, большое время записи-считывания программ, а также их поиск на ленте. Чтобы избежать всего этого в более дорогих компьютерах используют накопители на гибких магнитных дисках (НГМД). Такая возможность есть и у Вашего ZX-SPECTRUMa. Для подключения к нему НГМД (дисковода) существует специальная плата дискового интерфейса (иначе контроллер). Наибольшее распространение получил дисковый интерфейс типа "BETA-DISK". Контроллер, как и сам дисковод, требует два напряжения питания: 5В и 12В. Поэтому при подключении его к Вашему компьютеру необходимо запастись более мощным источником питания, чем требуется для компьютера (это обычно 5В - 2А, 12В - 1А).

Длина кабеля от компьютера к контроллеру должна быть минимально возможной и составлять не более 20 см. Кабель лучше всего выполнить витыми парами проводов. Кабель от контроллера к дисководу может быть до 3-х метров. Распайку кабелей необходимо вести строго по схеме компьютера и контроллера.

Чаще всего приходится делать небольшие доработки в самом компьютере. Их необходимо выполнить согласно инструкции по подключению контроллера.

17.1. ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ ДИСКОВОГО ИНТЕРФЕЙСА "BETA-DISK"

Дисковый интерфейс "BETA-DISK" (далее - дисковый интерфейс) рассчитан на дисководы 5,25; 3,5 дюйма с количеством дорожек 40 или 80. На линию может быть подключено до 4 дисководов, хранящих данные в формате двойной плотности, что дает максимальную емкость 2,5 млн. символов. Интерфейс может использоваться со "SPECTRUM"-совместимыми компьютерами (далее SPECTRUM). Программа Бейсика запускается автоматически или после перезагрузки. Дисковая операционная система TRDOS размещается в ПЗУ и использует только 112 байт ОЗУ, при этом простой синтаксис, использующий ключевые слова "SPECTRUM", позволяет получить доступ к файлам на диске из TRDOS или программ в машинных кодах. TRDOS управляет числовыми и строковыми массивами, серийными (последовательными) файлами и файлами прямого доступа.

17.2. ВВЕДЕНИЕ В ДИСКОВУЮ СИСТЕМУ

Дисковая система-это профессиональный способ хранения программ и файлов с данными в домашних и персональных компьютерных системах. Этот способ имеет много преимуществ по сравнению с системами, базирующимися на ленте, такими, как кассетная лента и "MICRODRIVERS". Дисковая система более надежна, легка в использовании и намного быстрее. Терминология, связанная с гибкими дисками, часто приводит к недоразумению, краткое пояснение даст Вам возможность сделать некоторые выводы и лучшим образом использовать Ваш дисковый интерфейс с TECHNOLOGY RESEARCH.

ДИСКИ И ДИСКОВОДЫ

Дисковый интерфейс "BETA DISK" TECHNOLOGY RESEARCH позволяет использовать дисководы на 5,25 и 3,5 дюйма. Вы могли слышать, что их называют гибкими дисками, дискетами, мини или микродисками. Мы их будем называть просто дисками. В настоящее время 5,25-дюймовые диски наиболее популярны, однако существуют тенденции к применению более маленьких дисков на 3,5 дюйма. Гибкий, одетый в пластик, 5,25-дюймовый диск находится в маленьком картонном или пластиковом чехле, однако "гибкий" диск нельзя гнуть. Меньшие диски на 3,5 дюйма находятся в более прочном защищенном пластиковом чехле, что означает более высокую цену, чем у 5,25-дюймовых. После того, как диски вставлены в дисковод, они вращаются внутри своих чехлов со скоростью 300 об/мин. Для защиты диска лучший эффект дает пластиковая накладка по краю центральной прорези, благодаря этой накладке слабый диск фиксируется механизмом дисковода. В кассетном магнитофоне лента

движется по неподвижной головке, в дисководе не только диск вращается по отношению к головке, но и головка движется по диску от края к центру. Индексное отверстие в чехле и подобное отверстие в диске дают возможность "вести" дорожку на диске, данные записываются и читаются через щелевую прорезь или "окно данных". Для защиты ценных данных существует защита от записи в виде маленькой липкой ленточки, она предохраняет диск от записи или от несанкционированного стирания. Чехол 3,5-дюймового диска имеет устройство, подобное магнитофонной кассете. На рынке имеются 40 и 80-дорожечные дисководы. Многие дисководы позволяют использовать и 40, и 80-дорожечные диски. Некоторые дисководы имеют только одну головку для записи/чтения, т.е. односторонние. Такие дисководы могут читать и записывать только на одной стороне диска, другие дисководы имеют две головки для записи/чтения (двухсторонние) и позволяют использовать обе стороны диска. Золотое правило заключается в том, чтобы убедиться, что до переписи дисков, предназначенных для работы с различными дисководами, с них сделаны копии, в идеале, если один пользователь диска использует 40 или 80-дорожечный дисковод, то так же должен действовать и другой.

Однако диски, которые отформатированы и записаны на дисководе одного типа, могут быть прочитаны на дисководе другого типа. Обязательно: диск должен быть такого же размера. Приведенная ниже схема показывает совместимость между дисководами различных типов.

ФОРМАТ ДИСКА	40T SS	40T DS	80T SS	80T DS
ДИСКОВОД 40T SS	C	?	X	X
40T DS	C	C	X	X
80T SS	R	?	C	?
80T DS	R	R	C	C

где T - дорожка
SS - односторонний
DS - двухсторонний
C - совместимость
R - совместимость (только чтение)
H - несовместимость
? - несовместимость (в зависимости от типа дисковода)

Например, 40-дорожечный двухсторонний дисковод совместим (может читать и записывать) с дисками, отформатированными на 40-дорожечном одностороннем дисководе, но 80-дорожечный двухсторонний дисковод может только читать с дисков, отформатированных на 40-дорожечном дисководе.

ДОРОЖКИ, СЕКТОРЫ И ПЛОТНОСТЬ

Процесс форматирования позволяет TRDOS и дисководу делить диск на 40 или 80 дорожек (в соответствии с дисководом) и каждую дорожку - на секторы. Количество секторов на дорожке и количество байт в секторе зависит от дисковой операционной системы. TRDOS размещает 16 секторов на дорожке и 256 байт в секторе. Такое большое количество секторов малого размера имеет несколько преимуществ: во-первых, если необходимо сохранить небольшое количество данных, то при этом будет использоваться не очень большая часть диска. Это способствует тому, что может быть записано большее количество файлов; во-вторых, при использовании файлов прямого доступа это обеспечивает большую гибкость программы и увеличивает скорость работы. Необходимо помнить, что TRDOS использует нулевую дорожку для директория. Исходя из изложенной выше информации можно сосчитать количество секторов и объем отформатированного диска. Эта емкость является одной из важных числовых характеристик любой системы и ее можно рассматривать в связи с количеством и размером секторов. Следующая таблица показывает емкость TRDOS в килобайтах для различных дисководов:

40 дорожек, одна сторона=39*16=624 сектора*256=156 K
40 дорожек, две стороны= 79*16=1264 сектора*256=316 K
80 дорожек, одна сторона=79*16=1264 сектора*256=316 K
80 дорожек, две стороны=159*16=2544 сектора*256=636 K

Для математического запоминания это равно 4 килобайтам на дорожку или 4 секторам на 1 килобайт. Последняя оценка наиболее важна, при изучении

содержимого диска при помощи CAT Вы всегда будете знать количество свободных секторов, разделив это количество на 4, Вы узнаете, сколько килобайт свободного пространства осталось. Заметим, что мы не упоминали о плотности. Одинарная плотность и двойная плотность относятся к методу упаковки записываемых данных на диске. TRDOS использует двойную плотность, благодаря чему мы имеем 16 секторов из 256 секторов на дорожке.

ПОДГОТОВКА К ЗАПУСКУ

Подключите контроллер дисковода к "SPECTRUM" согласно руководству по эксплуатации. Включите питание "SPECTRUM", дисковода и телевизора. При включении (даже при наличии нескольких дисководов) активным является только дисковод "A" (DS0). После этого на экране появится следующее изображение:

```
TR-DOS VER.5.XX
(C) 1986 TECHNOLOGY RESEARCH LTD.
(U.K.)
A>
```

где 5.XX - номер версии и A - подсказка TRDOS.

Возможно, что после включения питания "SPECTRUM" не сможет распознать дисковый интерфейс и вышеупомянутая надпись не появится, тогда "сбросьте" "SPECTRUM", используя кнопку "Сброс" (описано в следующей главе). Чтобы перейти к TRDOS, введите следующее:

```
RANDOMIZE USR 15616 <ENTER>
```

где <ENTER> - клавиша "ENTER". После этого появится вышеуказанная надпись.

ПРЕДУПРЕЖДЕНИЕ !!!

Никогда не разъединяйте дисковый интерфейс и "SPECTRUM", когда включено питание.

Никогда не оставляйте диск в закрытом дисководе при включении и выключении питания, это может привести к порче диска. Особенно это касается дисководов с постоянно нагруженными головками.

АВТОЗАГРУЗКА

При включении источника питания или после сброса Вы автоматически оказываетесь в TRDOS и системой всегда выбирается дисковод "A". TRDOS загрузит Бейсик-программу "BOOT", высветится обычная подсказка (см. предыдущую главу). Рекомендуется использовать следующие процедуры для автозагрузки:

(1) включение питания "SPECTRUM".
(2) загрузочный диск с Бейсик-программой "BOOT" на дисководе "A".
(3) сброс компьютера, как это описано в следующей главе.

АВТОПРОВЕРКА СПЕЦИФИКАЦИИ ДИСКОВОДА

Когда запущена TRDOS, она автоматически проверяет спецификацию дисковода. Предполагается, что Вы уже имеете практику и поэтому дисковод включили до или одновременно с компьютером. TRDOS может устанавливать шаговую частоту дисковода. Это позволяет полностью использовать большую шаговую частоту модемных дисководов. TRDOS также проверяет, является ли спецификация дисковода 40-, либо 80-дорожечной, односторонней или двухсторонней. Эти тесты будут проводиться, даже если диск не вставлен в дисковод. Если есть более одного дисковода, то они тоже будут проверяться, когда к каждому отдельному дисководу происходит первое обращение.

КОМАНДЫ "40" И "80"

Некоторые из дисководов раннего выпуска не имеют остановки по концу и, как следствие, автопроверка может не работать. Это означает, что TRDOS предполагает дисковод на 80 дорожек. Команда "40" информирует систему, что текущий дисковод имеет 40 дорожек: 40 <ENTER> Для дисководов, которые могут переключаться с 40 на 80 дорожек, возможно переключение из одного положения в другое после того, как TRDOS проверит спецификацию. Поэтому Вы должны сообщить TRDOS, какое из этих положений Вы используете, введя команду "40" или "80": 80 <ENTER>.

17.3. КНОПКА "СБРОС"

Кнопка "СБРОС" предназначена для сброса TRDOS и SOS. SOS означает "STANDART OPERATING SYSTEM" (стандартная операционная система). Нажмите кнопку "СБРОС". Экран на секунду очистится, в то же время TRDOS начнет работу. После "сброса" восстанавливается нормальная емкость RAMTOP, все переменные памяти очищаются, поэтому это очень быстрый и простой способ убедиться в том, что все результаты и данные из прошлой операции переместились и "SPECTRUM" очистился. Сброс также обеспечивает быстрый переход от одной деятельности к другой, если имеющиеся данные далее не потребуются.

17.4. СПРАВОЧНИК

КОМАНДА	ФУНКЦИЯ
*"A:"	устанавливает по умолчанию дисковод "A"
*"B:"	устанавливает по умолчанию дисковод "B"
*"C:"	устанавливает по умолчанию дисковод "C"
*"D:"	устанавливает по умолчанию дисковод "D"
40	информирует TRDOS, что по умолчанию дисковод с 40 дорожками
80	информирует TRDOS, что по умолчанию дисковод с 80 дорожками
CAT	отображает на экране каталог диска
CAT#	распечатывает каталог диска
CLOSE	закрывает последовательный файл или файл прямого доступа
COPY	копирует файлы с одного диска на другой
COPY S	копирует файлы в системе с одним дисководом
COPY B	дублирует диск в системе с одним дисководом
ERASE	удаляет файлы с диска
LIST	отображает детали содержимого диска
LIST#	распечатывает детали содержимого диска
LOAD	загружает программу с диска
INPUT	читает из последовательного файла или файла прямого доступа
MERGE	объединяет Бейсик-программу с диска с такой же программой в оперативной памяти
MOVE	реорганизовывает и упаковывает вместе файлы на диске
NEW	изменяет существующее имя файла
OPEN	открывает канал для последовательного файла или файла прямого доступа
PEEK	считывает сектор файла с диска в оперативную память
POKE	сохраняет данные из оперативной памяти на диске
PRINT	записывает последовательные файлы и файлы прямого доступа
RANDOMIZE USR 15616	осуществляет переход в TRDOS из SOS
RANDOMIZE USR 15619	осуществляет вызов команды TRDOS из SOS
RETURN	осуществляет возврат в SOS из TRDOS
RUN	загружает и запускает программу с диска
SAVE	записывает программу на диск
VERIFY	верифицирует программу, записанную на диске

где # - номер канала, через который выводится информация.

Приведенная выше таблица - это перечень всех команд TRDOS. Перед исполнением этих команд мы советуем Вам прочитать соответствующие разделы данного руководства. При использовании, показанные слова являются ключевыми словами "SPECTRUM", полученными в обычном режиме SOS. Некоторые из команд требуют дополнительных пояснений, которые будут даны в соответствующих разделах.

17.5. СИНТАКСИС КОМАНД

TRDOS является гибкой системой и Вы можете получить доступ к дисковой системе:

(A) непосредственно из TRDOS
(B) прямой ввод из "SPECTRUM" SOS
(C) из Бейсика (D) из программы в машинных кодах

Всегда в TRDOS Вы увидите подсказку в виде дисковода и стрелки:

A>
B>

В соответствии с операцией, после подсказки может следовать команда TRDOS: A>RUN "BOOT" Команды TRDOS могут быть введены непосредственно после подсказки TRDOS.

ПЕРЕХОД ИЗ TRDOS В SOS

Для перехода в SOS используется команда RETURN. В нижеследующем примере слова, помещенные в скобки, являются пояснением, а не частью синтаксиса: A> (подсказка TRDOS на экране) A>RETURN (нажали "N"-сейчас на экране) (C) SINCLAIR COPYRISHT...(нажали "ENTER"-сейчас на экране)

Запомните, пожалуйста, что клавиша "ENTER" нажимается для завершения команды RETURN. Если обратное не оговорено, то это предполагается во всем руководстве, даже если "ENTER" не пишется.

ПЕРЕХОД ИЗ SOS В TRDOS

Для перехода из SOS в TRDOS, когда мигает курсор, требуется ввести следующее: RANDOMIZE USR 15616. Это достигается так: за ключевым словом на букву T (RANDOMIZE) следует режим E буквы L (USR) и, затем, цифры. Как в примере предыдущего раздела может быть нажата клавиша "ENTER" для завершения команды. Если у Вас подключен только один дисковод, то Вы возвратитесь на дисковод "A". Если подключено более одного дисковода, то эта команда возвращает Вас к тому дисководу, который был выбран последним. Хотя находящаяся в памяти программа уже не отображается на экране после выполнения определенных команд (таких, как каталог диска), Вы можете вернуться в SOS, командой LIST вывести программу на дисплей.

ВЫЗОВ TRDOS ИЗ SOS И БЕЙСИК-ПРОГРАММ

Синтаксис команд всегда одинаков, но может быть использован соответствующий префикс. Если команда вызывается из SOS или программы, когда команда вводится непосредственно из SOS, она выполняется и происходит возврат в SOS, в этом случае префикс будет следующим:

RANDOMIZE USR 15619 : REM:

Например:

RANDOMIZE USR 15619 : REM: CAT "B:"

Использование 15619 вместо 15616 сохраняет управление в SOS. Действие команды CAT осуществляется на дисководе "B". Если команда не выполняет функцию прогона программы, такую как RUN, она будет определять действия с подсказкой дисковода, если она введена из TRDOS или "OK" в "SINCLAIR", если это сделано из SOS:

A>CAT <ENTER>

приводит по окончании к подсказке "A>"

RANDOMIZE USR 15619: REM: CAT <ENTER>

приводит к "OK"

Хотя возможно выполнение команды из SOS, это не рекомендуется:

во-первых, намного меньше вероятность появления неприятностей, если все действия осуществлять из TRDOS,

во-вторых, надо намного меньше нажимать клавиш;

в-третьих, отображаются буквы дисковода по умолчанию, что позволяет избежать нелепых ошибок, которые могут привести к потере ценных данных.

Когда команды TRDOS включены в программу Бейсика, префикс следующий:

RANDOMIZE USR 15619 : REM :

В добавление можно сказать, что команда должна быть последней в строке. При написании программы или при преобразовании ее для запуска с диска вместо кассеты, Вы должны подчиняться следующему правилу:

"КОМАНДА TRDOS ЗАКАНЧИВАЕТ СТРОКУ".

Например: 10 INK 7: PAPER 1: LOAD ""CODE:
GOSUB 500: CLS: PRINT"PRESS ANY KEY"

преобразованная для диска:

10 INX 7: PAPER 1: RANDOMIZE USR 15619: REM:
LOAD "TITLE" CODE
20 GOSUB 500: CLS: PRINT "PRESS ANY KEY"

Заметьте, что "" неприменимо для диска, в добавление к команде TRDOS, завершающей строку 10, Вы должны указать имя, под которым сохранен код.

ВЫБОР АКТИВНОГО ДИСКОВОДА

Дисковод по умолчанию - это тот, к которому можно получить доступ любой командой, в которой не определена буква дисковода. При включении питания или сбросе системы активным дисководом является "A". TRDOS поддерживает 4 дисковода: A, B, C, D. Чтобы сменить активный дисковод по умолчанию, вводится следующая команда:

*"DRIVE:"

где DRIVE-A, B, C или D. Например:

*"B:" сменить на "B"
*"A:" сменить на "A"

ЗАМЕЧАНИЕ: при вводе этой команды можно использовать верхний или нижний регистр для определения дисковода, хотя на экране всегда перед стрелкой отображаются заглавные буквы. После смены дисковода, например с "A" на "B", все следующие команды будут иметь доступ к дисководу "B", пока не будет оговорено что-либо другое. Если подключен только один дисковод, на экране останется "A>" и гибкий режим с несколькими дисководами, описанный в этом руководстве, не реализуется.

ВРЕМЕННЫЙ ВЫБОР ДИСКОВОДА

Иногда нужно оставаться на одном дисководе по умолчанию, но получить доступ к другому дисководу. Синтаксис для этого - это суффикс к команде, указывающий требуемый дисковод. Суффикс напоминает смену дисковода, но без "*", то есть: "A" или "B", или "C", или "D".

Пример полного высказывания из SOS:

RANDOMIZE USR 15619:REN:LOAD "B:PROGRAMM".

Это позволяет загрузить "PROGRAMM" с дисковода "B", безотносительно того, какой дисковод установлен по умолчанию:

LOAD "B:PROGRAMM".

Это позволяет загрузить "PROGRAMM" с дисковода "B", но оставить по умолчанию дисковод "A" для дальнейшей работы TRDOS.

17.6. ФОРМАТИРОВАНИЕ ДИСКА

Диск должен быть отформатирован перед его использованием в компьютере. Это означает, что секторы на каждой дорожке должны быть проверены, идентифицированы и помечены TRDOS, благодаря этому TRDOS будет удерживать дорожку, на которой что-то сохраняется. TRDOS содержит программу форматирования диска и никакого дополнительного программного обеспечения загружать не надо. Форматирование может быть произведено в любое время, даже с программой в памяти. Чтобы отформатировать диск, вставьте его в дисковод по умолчанию , закройте его, введите ключевое слово FORMAT (E SHIFT, SYMBOL SHIFT O) за которым следует название диска в кавычках. Название может иметь не более 8 символов в любом регистре, включая пробелы. Например:

FORMAT "DISCONS".

Нажмите "ENTER" и ждите. Время, которое отводится TRDOS для разметки секторов, изменяется в зависимости от того, является ли диск односторонним или двухсторонним, 40- или 80-дорожечным. Если дисководы двухсторонние, то обе стороны размечаются автоматически. По окончании на экране высветится:

DISCONS
624/624 или 1264/1264 или 2544/2544
A>

Эта картинка показывает имя диска, за которым следует количество отформатированных секторов и, затем, максимально возможное значение для определенного формата диска. Если первое число больше, чем второе - у Вас сбойный диск. Максимальное количество секторов изменяется в зависимости от спецификации дисковода. Нулевая дорожка всегда используется системой, которая оставляет 39 дорожек на 40-дорожечном диске, 79 дорожек на 80-дорожечном или 156 дорожек на 80 дорожечном двухстороннем диске. Имея 16 секторов на дорожке, мы получим 624, 1264 или 2544 как максимально возможное их количество.

ОДНОСТОРОННИЙ ФОРМАТ

Существуют случаи, когда необходимо отформатировать диск, как односторонний. Если дисковод односторонний, то никакого специального форматирования не требуется, используйте стандартную команду TRDOS FORMAT. Если дисковод двухсторонний, то первый символ в имени должен быть "$", т.е.

```
FORMAT "$COUPER"
```

После того, как Вы ввели "ENTER" и форматирование завершилось, на экране появится:

```
$COUPER
624/624 или 1264/1264 в зависимости от того, 40 или 80
A>
```

17.7. КАТАЛОГ ДИСКА

Имеются две команды для отображения каталога диска на экране. Первая и наиболее часто используемая, это команда CAT. Вторая - команда LIST. К командам CAT и LIST можно получить доступ пока программа сохраняется в памяти. Команда CAT отображает заголовок, тип и размер сектора файлов и используется для многих целей. Команда LIST отображает каталог с расширенной информацией и идеальна для анализа программы. Синтаксис для получения каталога диска такой:

```
CAT или
LIST
```

Вы можете отобразить каталог диска не только на дисководе по умолчанию: CAT "B:" или LIST "B:"

Эти команды могут быть также вызваны из SOS:

```
RANDOMIZE USR 15616:REN:CAT "A:" или
RANDOMIZE USR 15616:REN:LIST "A:"
```

Информация, отображаемая на экране командой CAT:

```
название диска
количество файлов
количество стертых файлов
дисковод: имя файла: тип файла: размер файла: (в два столбца)
количество свободных секторов
подсказка TRDOS
```

Например:

```
TITLE ACCOUNTS          значение
4 FILE (S)              диск в дисководе "A"
1 DEL. FILE (S)         4 файла + 1 стертый
A: BONE <B> 12 !B=БЕЙСИК-ПРОГРАММА    12 СЕКТОРОВ   (3к)
A: BONE1<C>  6 !C=МАШИННЫЙ КОД         6 СЕКТОРОВ   (1,5к)
A: BONE2<I> 13 !I=ПОСЛЕД./ПРЯМ.ДОСТУП 13 СЕКТОРОВ   (3,25к)
A: BONE3<D>  7 !D=МАССИВ ДАННЫХ        7 СЕКТОРОВ   (1,75к)
FREE 2503          2544-38=2506, СТЕРТЫЙ ФАЙЛ ЗАНИМАЕТ 3
                   СЕКТОРА, ОСТАВЛЯЯ СВОБОДНЫМИ 2503 СЕКТОРА.
A>                 АКТИВНЫЙ ДИСКОВОД "A"
```

Если количество отображаемых файлов больше 30, появится надпись "SCROLL?" (листать?). При нажатии любой клавиши "SPECTRUM" продолжает отображение. Пример результата выполнения команды LIST:

```
TITLE POOLPENS          DISK DRIVE B:
4 FILE (S)              80 TRECK D. SIDED
1 DEL. FILE (S)         FREE SECTOR 2480
FILE NAME               START LENGTH LINE
POOLCALC <B> 5          00298 01200   25
POOLI    <C>32          32768 08000
POOLFACT <I> 7          01780 01780
POOLBONE <D> 8          30000 32000
```

Картинка показывает все обычные подробности диска, который находится в дисководе "B": что он 80-дорожечный, двухсторонний, с четырьмя файлами и 2480 свободными секторами (около 620K свободны), в добавление к подробностям CAT о названии, типе и размере добавляются начальный адрес, длина и, если мы в Бейсике, начальная строка программы.

РАСПЕЧАТКА КАТАЛОГА

Часто хочется иметь информацию о содержимом диска без использования каждый раз компьютера для отображения каталога. Бумага и карандаш - это обычный метод, хотя он неудобен и часто подвержен ошибкам, TRDOS позволяет Вам создать твердую копию каталога диска, предполагая, что с Вашей системой соединен принтер. Следующий пример основан на применении "INTERFASE 1". В SOS обычная процедура открытия канала должна быть инициализирована перед печатанием, это требует возврата в SOS, если мы работаем в TRDOS и канал еще не открыт. В SOS обычный синтаксис для ввода в действие "INTERFASE 1":

```
FORMAT "1": 9600 : OPEN 4: "1"
```

Когда канал открыт, возврат в TRDOS предпочтителен, так, как при этом обеспечивается лучший контроль и меньший набор (префикс "RANDOMIZE"). По возвращении в TRDOS команды CAT и LIST точно такие же, как команды CAT и LIST, рассмотренные выше.

```
CAT 4:          отошлет каталог на принтер через канал 4
LIST 4."B:"     отошлет расширенный каталог дисковода "B" на принтер
```

Руководство по "SINCLAIR" описывает каналы с 0 по 15. Каналы с 0 по 3 зарезервированы для "SPECTRUM". В действительности используются каналы с 4 по 15. Не важно, какой именно, какой бы канал ни был открыт, он может быть использован в командах CAT и LIST.

17.8. КОПИРОВАНИЕ ФАЙЛОВ

Имеется три команды для копирования файлов:

```
COPY - для обычного копирования файлов;
COPY S - для копирования файлов в системе с одним дисководом;
COPY B -для дублирования в системе с одним дисководом.
```

Основной элемент синтаксиса - COPY, это ключевое слово "SPECTRUM" на клавише "Z". Основной синтаксис:

```
COPY "NEWFILE","OLDFILE"TYPE
```

В SOS синтаксис требует набор имен нового и старого файлов в кавычках и определения типа файлов, и их наборов с клавиатуры:

```
БЕЙСИК-ПРОГРАММА
ПРОГРАММА В МАШИННЫХ КОДАХ CODE
ФАЙЛ МАССИВА ДАННЫХ DATA
ФАЙЛ ПОСЛЕД. И ПРЯМ. ДОСТУПА
```

Заметьте, что программа Бейсика имеет пустой тип. Все структуры должны иметь точную форму имени, включая верхний и нижний регистры, так же, как и тип файла. Лучше всего иметь отображение на экране, полученное при помощи CAT. Очень важно помнить, что когда бы ни были два имени заключены в кавычки, т.е. для команд COPY и NEW, новое имя должно быть заключено в первые кавычки. В добавление нужно сказать, что никогда не нужно набирать новый заголовок, т.к. он такой же, как и старый.

КОПИРОВАНИЕ НА ТОТ ЖЕ ДИСК

Вставьте диск, содержащий файлы, которые надо копировать, в дисковод "A" (на самом деле нет разницы, какой дисковод). Первое, что необходимо сделать, это вызвать CAT на диске. Необходимо сообщить TRDOS имя файла, который будет копироваться и имя, под которым он будет записан, т.е.:

```
COPY "VAT69","WHISKY" CODE
```

Заметьте, что новое и старое имена файлов должны быть в кавычках и разделяться запятой. Представив себе, что существует мифическая программа в машинных кодах, относящаяся к спиртным напиткам. Мы должны добавить "CODE". Нажмите "ENTER" и команда исчезнет на пару секунд, снова выдав команду CAT, мы увидим на диске новый файл VAT69 <C>. Нельзя записать два файла на один и тот же диск с одинаковыми именами. Можно иметь одинаковые имена, но одно в верхнем, другое в нижнем регистрах, одно имя может быть у Бейсик-программы, а второе у программы в машинных кодах.

```
STOCKFILE <C>          применимо, потому, что
STOCKFILE <B>          различны типы файлов
```

Использование этого принципа дает возможность повторить предыдущий пример без перехода от общего термина "WHISKY" к частностям:

```
COPY "whisky" CODE     (первое "WHISKY" -большая латынь, второе
                        "whisky"- малая латынь)
```

или

```
COPY "WHISKY2","WHISKY" CODE
```

Мы рассмотрели копирование только на тот же диск, процедура копирования на другой диск зависит от того, есть ли у Вас второй дисковод. Если у Вас только один дисковод, то обычная команда COPY не может быть использована, для системы с одним дисководом предназначены команды COPY S и COPY B. Первая команда (COPY S) используется для копирования одного файла с одного диска на другой, используя тот же дисковод. Вторая команда (COPY B) используется для дублирования диска на другом диске, т.е. копирование всех файлов. Примеры команды COPY S:

```
COPY S "WINES"  или
COPY S "BEER" CODE
```

где "WINES" и "BEER" CODE - существующие файлы. Отметим разницу с предыдущим примером, ранее мы вводили новое имя в кавычках после подсказки, сейчас мы сообщаем TRDOS, какую программу скопировать, т.е., существующее имя идет в кавычках после COPY S. Чтобы убедиться, что Вы не вставили в дисковод не тот диск, Вам подсказывают, что надо вставить диск и нажать ENTER. Прочитав диск, программа подсказывает Вам заменить его другим и ввести новое имя, под которым программа будет скопирована. Практике создания дублирующей копии всех программ не нужно придавать большого значения. При операциях, основанных на использовании кассеты, это требует много времени, когда должна копироваться ценная программа, лента перематывается, а затем программа верифицируется. Это будет долгая процедура. С TRDOS и диском это займет меньше минуты, чтобы скопировать, а затем, при желании, перезагрузить программу, чтобы проверить, что на самом деле все в порядке. В действительности, дублирование программы состоит в создании копии на другом диске, который существует только как дубликат. Команда "COPY B" действует как расширение команды "COPY S". Программа запускается вводом "COPY B", после этого на экране появится подсказка сменить диск и какие клавиши нажать.

КОПИРОВАНИЕ И ДУБЛИРОВАНИЕ ПРИ ДВУХ ДИСКОВОДАХ

Копирование на другой дисковод автоматически означает, что Вы будете копировать на другой диск, хотя возможно копирование на тот же диск. При любой комбинации двух дисководов мы будем использовать дисководы "A" и "B". Из синтаксиса будет видно, как можно будет ввести другие комбинации. Введите ключевое слово COPY, за которым следуют названия в кавычках, но на сей раз установите временный индикатор дисковода. Находясь на различных дисках, мы можем при желании использовать то же имя и на другом диске, т.е.:

```
COPY "A:WHISKY","B:WHISKY"
```

Хотя дисковод по умолчанию - это "B", мы копируем с диска "B" на диск "A" и используем то же имя, которого не было на диске "A", поменяв диски между двумя

дисководами, мы бы имели:

```
COPY "B:WHISKY"."A:WHISKY"
```

т.е. мы копируем файл с дисковода "A" на дисковод "B". Операция дублирования намного проще при использовании системы с двумя дисководами. Диск, который будет копироваться, помещается в один дисковод, а принимающий диск (чистый и отформатированный, или частично использованный)- в другой дисковод. Мы будем использовать дисковод "A", как источник и дисковод "B", как принимающий. Командный синтаксис напоминает копирование единственного файла со значком "*", который заменяет имя файла:

```
COPY "B:*","A:*"
```

Введя команду TRDOS мы сделаем передышку. Если имена файлов не дублируются и принимающий диск имеет достаточно места для операции, все закончится возвратом к подсказке "A>". Если имя файла дублируется, то появится сообщение об ошибке "OVERWRITE EXISTING FILE? Y/N" (переписать существующий файл? да/нет), введите "Y" и файл. Это позволит Вам делать более поздние изменения и копировать один файл, если имя дублируется в двух различных файлах. Преимущества команды COPY со "*", это скорость, не сравнимая со скоростью ввода каждого файла отдельно.

17.9. ПЕРЕИМЕНОВАНИЕ, УНИЧТОЖЕНИЕ ФАЙЛОВ И ВЕДЕНИЕ ДИСКА

ПЕРЕИМЕНОВАНИЕ ФАЙЛОВ - КОМАНДА NEW

Возможность смены имени любого файла на дисках-одна из наиболее гибких команд, из имеющихся в Вашем распоряжении. Смена имени не имеет значения для работы программы, кроме того, Вы можете корректировать ошибки, возникшие при наборе имени программ. В отличие от других команд TRDOS, диск должен быть в дисководе по умолчанию и это должен быть дисковод "A". Это одна из дисковых команд, для которых Вы должны находиться в TRDOS, а не в SOS. Поэтому мы собираемся находиться в TRDOS, использовать дисковод "A" и ключевое слово "SPECTRUM" NEW (клавиша "A"). Сначала введите каталог диска, чтобы получить на экране имя файла, которое Вы будете менять, введите ключевое слово "NEW", за ним новое имя файла (как всегда, в кавычках), затем существующее, разделив их запятой:

```
NEW "BOOT","PROG"
```

Нажмите "ENTER" и подсказка исчезнет на пару секунд. Когда она появится снова Вам покажется, что ничего не произошло, повторите CAT, снова появится каталог, показывающий, что "BOOT" заменил "PROG".

УДАЛЕНИЕ ФАЙЛА - КОМАНДА ERASE

Если файл на диске устарел и больше не потребуется, его можно удалить. Командное ключевое слово - ERASE и оно получается входом в режим E, удержанием клавиши "SHIFT" и нажатием клавиши "7":

```
ERASE "OLDPROG"
ERASE "OLDPROG" DATA
```

УПАКОВКА ПРОСТРАНСТВА ДИСКА - КОМАНДА MOVE

Когда файл удален, сектора, которые он занимал, можно использовать для других файлов. Чтобы восстановить "утерянные" удаленными файлами сектора, мы используем команду MOVE. Это ключевое слово "SPECTRUM" получается входом в режим и нажатием одновременно клавиши "SHIFT" и клавиши "6":

```
MOVE            или
MOVE "B:"
```

Хотя приведенный выше пример правилен, предпочтительнее выполнять команду так, чтобы рабочий дисковод был дисководом по умолчанию. Это команда, которая обязательно должна быть использована внутри TRDOS, а не в SOS. Перейдя в TRDOS, мы выполнили команду CAT для изучения диска, возможно сделали некоторые удаления и заканчиваем командой восстановления пространства для использования MOVE. Всю работу по реорганизации диска и его директория делает сама команда MOVE. По окончании снова появляется подсказка TRDOS. Если сейчас ввести команду CAT, то каталог покажет 0 удаленных файлов и увеличившееся количество свободных секторов. Как и все программы в машинных кодах, команда MOVE быстрая, но время

ее работы зависит от количества и размера хранимых и удаленных файлов и их расположения на диске. Чтобы сократить время ожидания до минимума, нужно запускать MOVE сразу после удаления файлов. Это дает преимущество, так как это позволяет избежать многочисленных повторных реорганизаций и диск всегда точно будет показывать свободное пространство, он готов загрузиться до максимума без остановки и предварительных служебных действий.

17.10.СОХРАНЕНИЕ, ВЕРИФИЦИРОВАНИЕ, ЗАГРУЗКА, ЗАПУСК И СЛИЯНИЕ ПРОГРАММ

Синтаксис для этих программ подобен синтаксису для системы, применяющей кассету. Все команды оперируют с файлами. Файл на диске может быть Бейсик-программой, программой в машинных кодах, файлом с массивом данных, последовательным файлом или файлом прямого доступа. Загрузка и сохранение файлов с массивами данных, последовательных файлов и файлов прямого доступа описаны дальше, данный раздел охватывает Бейсик-программы и программы в машинных кодах. Во время всех пяти операций:СОХРАНЕНИЕ, ВЕРИФИЦИРОВАНИЕ, ЗАГРУЗКА, ЗАПУСК И СЛИЯНИЕ ПРОГРАММ - клавиша "BREAK" используется для прерывания команды. Если в дисководе нет диска, команда будет прервана и на экране появится сообщение об ошибке "NO DISK" (нет диска). Также, если диск не содержит файлов, отобразится сообщение об ошибке "NO FILE(S)" (нет файла).

КОМАНДЫ SAVE И VERIFY

Команда SAVE запишет программу на диск, имя программы должно быть определено и заключено в кавычки. Для Бейсик-программ тип файла не требуется, однако номер строки может быть определен после того, как автоматически запущено первое слово "LINE". Если номер строки неопределен, то программа запустится с самой первой строки. Например:

```
SAVE "HONDACO" LINE 100
SAVE "GRAPH" LINE
SAVE ":DESIGN"
```

Для программы в машинных кодах мы должны определить тип файла CODE , за которым следует начальный адрес и количество байт, которые нужно сохранить, т.е.:

```
SAVE "DISUOIT"CODE  47800,955
SAVE "B:COLLEC"CODE 32768,40000
```

Команда "VERIFY" проверяет, является ли файл, записанный на диск, таким же, как в памяти. Команда "VERIFY" может быть использована для верификации программ Бейсика, программ в машинных кодах и файлов с данными, т.е.:

```
VERIFY "A:DESIGN"
VERIFY "DISCOUIT"CODE 47800, 955
VERIFY "MONEY"DATA M()
```

Если файл на диске и в памяти отличны друг от друга, то сообщение об ошибке верификации отобразится на экране.

КОМАНДЫ LOAD И RUN

Если Бейсик-программа записана с автозапуском с номера строки, например:

```
SAVE "INTEREST"LINE 25
```

то она автоматически запустится при использовании команд LOAD или RUN. Если Бейсик-программа записана без автозапуска, то команда LOAD произведет загрузку программы и будет ее "листать", в то время, как команда RUN - загрузит и будет "прогонять" программу без автозапуска. Например:

```
LOAD "INTEREST"
RUN "DESIGN"
```

Как Вы можете предположить, TRDOS проинформирует Вас о недостаточном объеме памяти. Это произойдет, если верхняя граница памяти (RAMTOP) расположена слишком низко. Если Вы введете LOAD или RUN без имени файла, TRDOS загрузит и запустит Бейсик-программу "BOOT". Как видно из вышеприведенных примеров, синтаксис команд такой же, как и в Бейсике "SPECTRUM". Программы в машинных кодах могут быть загружены с того же адреса, в котором они были записаны, т.е.:

LOAD "SALECALS" CODE

или загружены с другого адреса, путем определения загрузочных инструкций, т.е.:

LOAD "MARGIN" CODE 51000

При запуске программ в машинных кодах адрес автозапуска должен быть таким же, как начальный адрес программы, т.е.:

RUN "DISCOUNT" CODE 47800

Это означает, что код, который получается для автозапуска с 47800, будет правильным. Альтернативой является использование двухстрочного загрузчика, т.е., записав следующую программу на диск, введя "DISTLOAD" LINE 10

10 RANDOMIZE USR 15619: REM: LOAD "DISCOUNT"CODE 47800

20 RANDOMIZE USR 47838

Чтобы запустить программу, введите команду RUN "DISTLOAD", которая загрузит машинный код и запустит его с адреса 47838. Двухстрочный загрузчик предпочтительнее, потому, что команда TRDOS должна быть последней в строке.

КОМАНДА MERGE

Объединение в TRDOS такое же, как и в SOS, оно использует то же ключевое слово (режим E, SYMBOL SHIFT+T) и служит той же цели, т.е. объединению в памяти "SPECTRUM" Бейсик-программ с диска с программой, уже находящейся в памяти, например:

MERGE "SOBCOUL"

MERGE "B:FUNST2"

17.11.ПЕРЕМЕЩЕНИЕ ПРОГРАММ, РАСПОЛАГАЮЩИХСЯ НА КАССЕТЕ

Если программа записана на кассете, то при копировании ее на диск возникнут проблемы. Вы загружаете Вашу программу с кассеты в SOS как обычно, в "SPECTRUM" программа может быть записана на диск непосредственной командой с клавиатуры, т.е.:

RANDOMIZE USR 15619: REM: SAVE "XXXX" или

RANDOMIZE USR 15619: REM: SAVE "XXXX"CODE NNNN,DSS

Синтаксис этих программ может потребовать преобразований, чтобы они могли работать в TRDOS. Для других программ, которые не могут быть преобразованы и запущены с диска, может быть использована "ВОЛШЕБНАЯ КНОПКА" (MAGIC BUTTON).

ПРЕОБРАЗОВАНИЕ ПРОГРАММ

Программа часто является смесью Бейсика и машинного кода для увеличения скорости работы. Элемент Бейсика может изменяться от простого загрузчика до более сложного интерфейса между кодом и пользователем. Для программ, которые используют Бейсик в качестве интерфейса, преобразование для использования в TRDOS относительно простое. Другие программы могут потребовать дополнительной поддержки, чтобы дать возможность запуска в TRDOS. Некоторые сложные программы могут потребовать использования "ВОЛШЕБНОЙ КНОПКИ", как единственного средства разрешения проблемы. Вы должны иметь возможность доступа ко всем командам LOAD или SAVE в первоначальной Бейсик-программе, которая требует изменений, т.е. первым делом надо "пролистать" ее. Применяемые методы изменяются в зависимости от особенностей программы. В некоторых случаях количество команд LOAD и SAVE минимально и быстрого просмотра листинга достаточно для обнаружения команд, другие программы могут иметь несколько операций LOAD и SAVE, простейший путь при этом, это перемещаться по строкам программы в поиске инструкций LOAD и SAVE. Как только такая команда найдена, она модифицируется путем добавления обычного префикса:

RANDOMIZE USR 15619: REM:

Это выражение вставляется перед имеющимися в программе LOAD и SAVE. Мы также должны запомнить еще два пункта, во-первых, команда TRDOS должна быть последней в программной строке, во-вторых, нужно принять во внимание размещение дисковых файлов. Первый пункт часто требует небольших изменений в нумерации и содержании строк. Следующий пример иллюстрирует оба пункта:

Существующая программа:

```
500 IF X=5 THEN INPUT "NAME"; N$; SAVE N$ DATA C$
    ( ): VERIFY N$ DATA C$ ( ):GOSUB 700
505 IF X=5 THEN GOSUB 800:INPUT "TITLE"; T$; SAVE
    T$; SAVE T$ CODE 50000,575
506 ..............
```

новая программа

```
500 IF X=5 THEN INPUT "NAME"; N$; RANDOMIZE USR
    15619; REM; SAVE N$ DATA C$ ( )
502 GOSUB 700
504 IF X=5 THEN GOSUB 800; INPUT "TITLE"; T$;
    RANDOMIZE USR 15619; REM; SAVE T$
505 RANDOMIZE USR 15619; REM; SAVE T$ CODE 50000,575
506 ..............
```

Нумерация строк меняется от программы к программе. Неиспользуемые номера строк 502 и 504 были нужны для того, чтобы позволить вставить существенную первую строку в команды TRDOS. Также могла понадобиться корректировка других строк, включающих непосредственно команды TRDOS. Для создания достаточного места для вставок все время нужно наблюдать за ходом программы.

ИТОГОВЫЕ ПРАВИЛА ПРЕОБРАЗОВАНИЯ:

(1) КОМАНДА TRDOS ДОЛЖНА БЫТЬ ПОСЛЕДНИМ УТВЕРЖДЕНИЕМ В СТРОКЕ.
(2) НЕОБХОДИМО ДОБАВИТЬ КО ВСЕМ СУЩЕСТВУЮЩИМ КОМАНДАМ LOAD И SAVE ТОЛЬКО ПРЕФИКС RANDOMIZE USR 15619; REM;
(3) УБЕДИТЕСЬ, ЧТО ИСПОЛЬЗУЕТСЯ ПРАВИЛЬНЫЙ ДИСКОВОД, ЕСЛИ ЕСТЬ БОЛЕЕ ОДНОГО ДИСКОВОДА.
(4) ВСЕ ФАЙЛЫ ДОЛЖНЫ ИМЕТЬ ИМЯ.
(5) ПРОВЕРЬТЕ ХОД ПРОГРАММЫ ПРИ ДОБАВЛЕНИИ ПРОМЕЖУТОЧНЫХ СТРОК.

"ВОЛШЕБНАЯ КНОПКА" ("MAGIC")

"ВОЛШЕБНАЯ КНОПКА" - ее назначение - сохранять базирующиеся на кассете программы и запускать их с диска без преобразования. Существует много программ, написанных таким образом, что их трудно или невозможно преобразовать, или запустить с диска. Это могут быть программы со сложной защитой или программы, в которых в основной программе используется SAVE и LOAD для файлов с данными, поскольку при этом невозможно использование команд TRDOS, требуется другой подход. Предположим, что программа загружена и файлы с данными созданы. Вместо использования в программе собственных команд SAVE, используется "ВОЛШЕБНАЯ КНОПКА" для того, чтобы выгрузить всю программу в файлы на диск, последующая перезагрузка означает загрузку всего пакета вместо простого файла с данными. Неудобство состоит в том, что мы имеем нестандартную версию всего пакета для каждого типа файла с данными, это снижает скорость работы и гибкость работы TRDOS на диске. Для этой цели должен быть подготовлен отформатированный чистый диск и при использовании должен быть вставлен в дисковод "A". Если все это сделано, следующий текст подробно укажет Вам необходимые операции (предполагается, что Вы в TRDOS):

(1) возвратитесь в Бейсик вводом "RETURN <ENTER>".
(2) очистите "SPECTRUM", введя "RANDOMIZE USR 0 <ENTER>".
(3) загрузите и запустите программу с ленты как обычно.
(4) после того, как программа загружена и запущена, нажмите "ВОЛШЕБНУЮ КНОПКУ" и тут же отпустите ее.

После этого память компьютера выгружается на диск и хранится на диске, как файл(ы).

```
TITLE ;TEST
/ FILE (S)
Ø DEL.FILE
A:@    <C>192:@7        <C>64
A:@5   <C>64 :@4        <C>64
A:@3   <C>64 :@1        <C>64
A:@2   <C>1
```

Общее имя файлов - "@".

Чтобы перезагрузить файл (для всех компьютеров семейства "SPECTRUM") используется синтаксис, отличный от обычного LOAD или RUN. Вместо ключевого слова SOS "GOTO" (буква G), используется:

```
GOTO "@"CODE
```

Программа может быть переименована или скопирована на другой диск со своим именем, запущена командой GOTO:

```
NEW "GAMES1","@"CODE
COPY "B:GAMES1","A:@"CODE
```

При запуске "GAMES1", если диск находится в дисководе "A", синтаксис будет следующим:

```
GOTO "GAMES1"CODE
```

Перед тем, как использовать "ВОЛШЕБНУЮ КНОПКУ" для записи программ, Вы должны убедиться, что диск не содержит ни одного файла с именем "@", "@1", и т.д. Некоторые программы используют нестандартную клавишу сканирования программы. Для таких программ необходимо сделать символ "$" первым символом в имени файла:

```
GOTO "$GAMES1"CODE
```

Если программа проверена перед копированием на другой диск на предмет заголовка, то имя файла потребует префикса "$", если возникнет проблема в клавише для сканирования:

```
NEW "$GAMES2","@"CODE
COPY "$GAMES2"CODE
```

Во время загрузки этих файлов изображение на экране будет содержать несколько случайных элементов. Часто это бывает набор элементов строк, которые выглядят наподобие азбуки Морзе. Когда экран очистится, программа запустится в области действия.

17.12.ФАЙЛЫ МАССИВОВ ДАННЫХ

Файлы SOS для загрузки или сохранения не строчного массива, названного MANEY:

```
LOAD "MANEY"DATA M( )   или
SAVE "MANEY"DATA M( )
```

Для строчного массива требуется добавлять знак $:

```
LOAD "MANEY"DATA M$( )
```

Подобно большинству команд TRDOS загрузка и сохранение массивов данных достигается путем использования синтаксиса SOS с префиксом Бейсика и, возможно, управляющим признаком. Массивами данных редко можно управлять непосредственно из TRDOS. Особенность массивов TRDOS такова, что они могут быть записаны и загружены из запущенной программы. Синтаксис SOS для работы с файлами из Бейсик-программы:

```
RANDOMIZE USR 15619; REM; LOAD "MANEY"DATA M( )
RANDOMIZE USR 15619; REM; SAVE "MANEY"DATA M( )
```

Замечание: этот синтаксис предполагает использование дисковода A. Если у Вас имеется система из двух одинаковых дисководов, то, в качестве альтернативы, предположим, что диск с данными вставляется во второй, укажите "B", если главная программа и основной дисковод "A". В этом случае будет использоваться временный управляющий признак:

```
RANDOMIZE USR 15619; REM; LOAD "B:MANEY"DATA
```

При копировании, стирании или создании нового файла данных, команда всегда должна заканчиваться указанием типа файла, т.е. DATA, иначе TRDOS предполагает, что это файл Бейсика:

```
COPY "A:MANEY", "B:MANEY"DATA.
```

17.13.ФАЙЛЫ ПОСЛЕДОВАТЕЛЬНОГО И ПРЯМОГО ДОСТУПА

ОБЩЕЕ ОПИСАНИЕ

Файлы с обычным форматом данных рассматривались в предыдущем разделе. В TRDOS возможны два дополнительных типа файлов данных, это файлы последовательного и прямого доступа. Чтобы использовать эти два типа файлов, Вы

должны сначала открыть канал. "SPECTRUM" имеет 15 каналов. SOS резервирует каналы с 0 по 3 для собственных нужд, а каналы с 4 по 15 доступны TRDOS. Когда открыт канал для файла последовательного или прямого доступа, то используется 336 байтов оперативной памяти. Числовые и строковые переменные могут быть сохранены в файле с данными, числовые переменные преобразуются в строковые компьютером, строка заканчивается "ENTER" (в ASCII код 13). Файл последовательного доступа сохраняет данные частями, наподобие магнитофонной кассеты, чтобы прочитать строчку в конце файла, нужно начать с начала файла. Файлы прямого доступа, с другой стороны, сохраняют данные как набор записей. Любая запись в этих файлах может быть записана или прочитана путем определения номера этой записи. Поэтому получить доступ к последней записи так же просто, как и к первой. Другие команды TRDOS для выбора дисковода, копирования, изменения заголовков и т.д. в равной степени используют оба способа доступа. Единственное отличие состоит в типе признака файла, который является "*" вместо "CODE" или "DATA", т.е.:

```
COPY "B:PHONE","A:PHONE"
ERASE "NAMEOLD"
```

Подобно файлам с массивами данных файлы последовательного и прямого доступа редко управляются из TRDOS.

ФАЙЛЫ ПОСЛЕДОВАТЕЛЬНОГО ДОСТУПА

Файл последовательного доступа может быть открыт командой "WRITE" или "READ", но не той и другой одновременно. Открытие файла командой "WRITE":

```
10 LET DOS=15619
20 RANDOMIZE USR DOS; REM; OPEN 4, "TEST", W
30 PRINT 4, "THIS IS A TEST LINE"
40 RANDOMIZE USR DOS; REM;CLOSE 4
```

Отметим использование DOS вместо 15619 и закрытия файла. Символьная строка в 30 строке программы будет записана в файл TEST на диске, если файл не будет закрыт, данные будут утеряны. Любой закрытый файл может быть повторно открыт для последующей записи и чтения, изменения и повторной записи под другим именем. Открытие файла командой "READ":

```
OPEN STREAM NUMBER,"FILENAME",R.
```

Когда открыт канал для READ ключевое слово INPUT в SOS используется для чтения данных в этом канале:

```
10 LET DOS=15619
20 RANDOMIZE USR DOS1; REM; OPEN 7, "TEST", R
30 INPUT
```

Данные из файла TEST будут записаны в переменную A$.

ФАЙЛЫ ПРЯМОГО ДОСТУПА

Файл прямого доступа, это файл с данными, который содержит набор записей, пронумерованных с 0 и далее. Длина записей (максимум 254 байта каждая) выбирается и определяется при первом открытии файла. Запись сохраняется как одна строка. Любая строка длиной менее, чем указанная, сохраняется и заканчивается ENTER (код ASCII 13), оставшиеся байты не определяются. Любая строка, которая длиннее, чем указано, будет "обрезана" до указанной длины и сохранена без ENTER. Когда файл прямого доступа открыт впервые, TRDOS определит 16 секторов (примерно 4K) в этом файле, тем самым предполагается, что длина файла 4K. Открытие файла прямого доступа:

```
OPEN STREAM NUMBER; RECORD NUMBER; VERIABLE NAME.
```

Чтобы сделать запись в файл (префикс RANDOMIZE не требуется для команд PRINT и INPUT):

```
PRINT STREAM NUMBER; RECORD NUMBER; VERIABLE NAME.
```

Например:

```
100 RANDOMIZE USR 15619; REM; OPEN 8, "ADDRESS" RND, 100
150 PRINT 8; 72, D$
```

Канал открыт для файла ADRESS, в котором на каждую запись отводится 100 байт, содержимое D$ помещается в запись номер 72 в этом файле. В отличие от файла последовательного доступа Вы можете читать и записывать в то время, когда

файл открыт. Чтобы прочитать запись из файла:

INPUT STREAM NUMBER; (RECORD NUMBER), VERIABLE NAME.

Например:

```
100 RANDOMIZE USR 15619; REM; OPEN 12, "TEST" RND, 20
110 PRINT 12; 20, A$
    :
    :
570 INPUT 12; (15); A$
    :
    :
990 RANDOMIZE USR 15619; REM; CLOSE 12
999 END
```

В строке 570 запись номера 15 (запомните, что это в самом деле 16-я запись) читается и помещается в переменную A$. Когда все операции с файлом закончены, канал должен быть закрыт, как показано в строке 990. Содержимое файла будет утеряно, если питание выключить до закрытия канала. Запись может включать в себя более, чем одну переменную.

17.14. НЕПОСРЕДСТВЕННОЕ ЧТЕНИЕ/ЗАПИСЬ СЕКТОРА

В добавление к трем типам файлов с данными, описанных в двух предыдущих пунктах, TRDOS также имеет возможность непосредственного чтения/записи сектора файла на диске (1 сектор содержит 256 байт).

КОМАНДА PEEK

Команда PEEK позволяет читать любую часть файла на диске и перемещать данные в оперативную память. Вы можете только читать сектор, а данные будут перемещаться в любом направлении в оперативной памяти:

PEEK "FILENAME" BUFFER ADDRESS, SECTOR NUMBER.

иначе:

PEEK "RECORD" 30023,5.

В этом примере читается 5 сектор файла RECORD и его содержимое записывается в оперативную память с адреса 30023.

КОМАНДА POKE

Команда POKE позволяет Вам записывать сектор с данными из оперативной памяти в файл на диске. Синтаксис аналогичен команде PEEK:

POKE "FILENAME" BUFFER ADDRESS, SECTOR NUMBER.

иначе:

POKE "B: OLD" 30024,10.

В этом примере содержимое области оперативной памяти, начинающейся с адреса 30024 и длиной в один сектор, записывается в 10 сектор файла "OLD" на диске "B". Так как эта команда изменяет содержимое файла на диске, ее надо использовать с осторожностью.

17.15. ПРОГРАММИРОВАНИЕ В МАШИННЫХ КОДАХ

Ниже описано включение в TRDOS программ в машинных кодах. Для программы требуется три элемента:

(1) эквивалент машинного кода команды Бейсика TRDOS.

(2) программа в машинных кодах для выполнения (1).

(3) программа в машинных кодах для повторения сохранения системы в ее первоначальном состоянии перед вызовом и выполнением команды.

Действительное расположение в памяти зависит от программы в целом. В нижеследующем примере расположение программы будет следующим: 49000 для SAVE, 49500 для LOAD и 50000 для команд, вызывающих их. Таким образом, пункт (1) будет выполняться при положении 49000 или 49500, а пункты (2) и (3) - в положении 50000.

Пример:

АДРЕС	КОД	БЕЙСИК	КОММЕНТАРИИ
49000	234	REM	Коды, как в положении "A"
49001	58	:	описания "SPECTRUM"
49002	248	SAVE	
49003	34	"	
49004	69	E	
49005	120	X	
49006	97	A	Имя файла "EXAMPLE"
49007	109	M	
49008	112	P	
49009	108	L	
49010	101	E	
49011	34	"	
49012	13	ENTER	Всегда заканчивается ENTER

Код для LOAD начинается с 49500 и он в точности такой же, как показанный выше, кроме того, что адрес 49502 будет содержать 239 (LOAD) вместо 248 (SAVE). Эти две программы LOAD и SAVE могут быть размещены в любом месте, но вызывающая программа, которую мы помещаем в 50000, требует изменения в адресах 50007-50008 (записать программный адрес) и 50025-50026 (загрузить программный адрес, чтобы отметить новые адреса). Для переразмещения вызывающая программа требует сама по себе повторного ассемблирования. По этой причине ниже приводится только мнемоника Z-80.

```
CHADD EQU 23645       размещение переменной CHADD SOS
ORD XXXXX             XXXXX-адрес в этом коде
LD HL,(CHADD)         начало сохранения верного CHADD
LD (TEMP), HL         временное сохранение CHADD
LD HL 49000           адрес программы SAVE
LD (CHADD),HL         CHADD сейчас указывает на нашу программу
CALL 15363            ввод TRDOS SAVE через CHADD
JP BACK               переход к точке программы, из
                      которой была вызвана вся данная программа

LD HL,(CHADD)
LD (TEMP), HL         программа для LOAD сейчас
LD HL, 49500          повторяет рассмотренную выше
LD (CHADD),HL         программу с изменением адресов
CALL 15363
BACK LD HL, (TEMP)    начало восстановления CHADD
LD (CHADD),HL         восстановление первоначальной CHADD
RET                   возвращение туда, откуда Вы пришли
TEM                   метка для помещения в память на
                      временное хранение
```

Вся программа, включающая программы и SAVE, и LOAD с "возвратом к точке входа", которым заканчивается программа, занимает только 47 байт.

17.16. СООБЩЕНИЕ ОБ ОШИБКАХ

СТРОЧНЫЕ СООБЩЕНИЯ

Когда Вы вводите команду из TRDOS, она будет выполняться. Если же эту команду TRDOS не может воспринять (например, LOAD, REN, FORMAT и т.д.), TRDOS проигнорирует ее. Если имеется синтаксическая ошибка или ошибка при выполнении команды, то сообщение об этом появится на экране. Сообщения об ошибках и о том, чем они вызваны, приведены ниже.

(1) NO DISK

Нет диска, диск не размечен или не закрыт дисковод. Команда отображается на экране после "A>". Вы можете вставить диск или закрыть дисковод и ввести ENTER для выполнения той же команды.

(2) NO FILE(S)

TRDOS не может найти файл на диске.

(3) "ERROR"

Это сообщение появится, если есть синтаксическая ошибка в команде.

(4) OUT OF MEMORY

Это сообщение появится при загрузке программы с диска и в случае недостаточного объема ОЗУ, и при использовании команды MOVE, если нет необходимых 4K рабочего пространства в памяти. Обычно перезапуск компьютера помогает преодолеть эти проблемы.

(5) FILE EXIST

Файл с таким же именем и типом, как и тот, который Вы хотите сохранить, уже существует на диске.

(6) OVERWRITE EXISTING FILE ? (Y/N)

Это сообщение появится, когда копируются все файлы с одного диска на другой. Уже есть такой файл:

Y - переписать его;

N - игнорировать его.

(7) WRITE ERROR
TRK XX SEC YY
RETRY, ABORT, IGNORE ?

Диск имеет сбой на дорожке XX в секторе YY

ввести "R" - еще раз попытаться

"A" - исключить эту операцию и вернуться в TRDOS

"I" - игнорировать и продолжать дальше

(8) WRITE PROTECT
TRK 0 SEC 1
RETRY, ABORT, IGNORE ?

Диск защищен от записи, есть три возможности (см. выше)

(9) VERIFY ERROR

Файл на диске не такой, как в памяти.

КОДЫ ОШИБОК

Все сообщения об ошибках появляются только из TRDOS. Если команда поступила из SOS или из программы в машинных кодах, то сообщения не появятся на экране, однако они хранятся как код в регистровой паре BC Z80. Ключ к коду:

0 - нет ошибок
1 - нет файлов
2 - файл существует
3 - нет пространства памяти
4 - сбой в каталоге
5 - номер записи переполнен
6 - нет диска
7 - дисковые ошибки
8 - синтаксическая ошибка
10 - канал уже открыт
11 - нет файла на диске
12 - канал не открыт

Чтобы получить код ошибки, установите переменную равной команде TRDOS. Переменная примет значение кода ошибки при завершении команды TRDOS. Пример 1:

```
LET A=USR 15619; REM; CAT
```

Пример 2:

```
10 CLEAR 65367
20 LET ERR=USR 15619; REM; LOAD "CORY"CODE
25 REM ERROR CODE IS RETURNED IN TEN WARIABLE ERR
30 IF ERR=1 THEN CLS: PRINT AT 10, 1; ""CORY""CODE
   NOT ON DISK"; STOP
40 RANDOMIZE USR 32768
50 RANDOMIZE USR 15616
```

17.17. СВЕДЕНИЯ О TRDOS

1. TRDOS занимает 112 байт оперативной памяти.
2. Без TRDOS память пользователя начинается с
   а) адреса 23755, без связи с интерфейсом 1;
   б) адреса 23813, со связью с интерфейсом 1;

При использовании TRDOS пользовательская память начинается с:
   а) адреса 23867, без связи с интерфейсом 1;
   б) адреса 23925, со связью с интерфейсом 1.

Чтобы записать/загрузить/запустить программу, которая использует байты между адресом 23755 и указанным в процедуре пункта 17.11. (MAGIC BUTTON) адресом 23925 надо выполнить то, что предписывает упомянутый пункт.

3. TRDOS для сохранения использует секторы на диске. Если количество байтов превышает 256, используется следующий сектор, и так, пока весь файл не запишется. Запись 522 байтов занимает 3 сектора, 3 сектор будет занимать только 10 байтов. Только эти 10 байтов будут загружаться с диска, оставшиеся пустые 246 байт не загружаются и не портят другие данные, которые могут находиться в памяти SOS.

4. В добавление к 112 байтам оперативной памяти TRDOS также использует 256-байтный буфер. При доступе к диску этот буфер размещается в памяти динамически. При выполнении многих команд TRDOS сначала перемещается Бейсик-программа (если таковая имеется) для создания буфера, по окончании команды Бейсик-программа перемещается в свое первоначальное положение. Эта операция "ПРОЗРАЧНА" для пользователя.

5. Команда MOVE требует 4K (минимум) оперативной памяти "SPECTRUM" в качестве рабочего пространства. Если команда MOVE выполняется вместе с программой, которая все еще находится в памяти, необходимо очистить компьютер, затем выполнить команду MOVE. Очистка компьютера может быть осуществлена возвратом в SOS и вводом USR 0 или использованием клавиши перезагрузки на системном блоке.

Приложение 1

ПОЛНЫЙ НАБОР СИМВОЛОВ

дес. код	символ	шестн. код	ассемблер. мнемоника	CBH...	EDH...
0	не использ.	00	NOP	RLC B	
1	не использ.	01	LD BC,NN	RLC C	
2	не использ.	02	LD (BC),A	RLC D	
3	не использ.	03	INC BC	RLC E	
4	не использ.	04	INC B	RLC H	
5	не использ.	05	DEC B	RLC L	
6	PRINT упр.	06	LD B,N	RLC(HL)	
7	EDIT	07	RLCA	RLC A	
8	курс.Влево	08	EX AF,AF"	RRC B	
9	курс.Вправо	09	ADD HL,BC	RRC C	
10	курс.Вниз	0A	LD A,(BC)	RRC D	
11	курс.Вверх	0B	DEC BC	RRC E	
12	DELETE	0C	INC C	RRC H	
13	ENTER	0D	DEC C	RRC L	
14	число	0E	LD C,N	RRC(HL)	
15	не использ.	0F	RRCA	RRC A	
16	INC упр.	10	DJNZ DIS	RL B	
17	PAPER упр.	11	LD DE,NN	RL C	
18	FLASH упр.	12	LD (DE),A	RL D	
19	BRIGHT упр.	13	INC DE	RL E	
20	INVERSE упр	14	INC D	RL H	
21	OVER упр.	15	DEC D	RL L	
22	AT упр.	16	LD D,N	RL (HL)	
23	TAB упр.	17	RLA	RL A	
24	не использ.	18	JR DIS	RR B	
25	не использ.	19	ADD HL,DE	RR C	
26	не использ.	1A	LD A,(DE)	RR D	
27	не использ.	1B	DEC DE	RR E	
28	не использ.	1C	INC E	RR H	
29	не использ.	1D	DEC E	RR L	
30	не использ.	1E	LD E,N	RR (HL)	
31	не использ.	1F	RRA	RR A	
32	пробел	20	JR NZ,DIS	SLA B	
33	!	21	LD HL,NN	SLA C	
34	"	22	LD (NN),HL	SLA D	
35	#	23	INC HL	SLA E	
36	доллар	24	INC H	SLA H	
37	%	25	DEC H	SLA L	
38	&	26	LD H,N	SLA(HL)	
39	'	27	DAA	SLA A	
40	(	28	JR Z,DIS	SRA B	
41	)	29	ADD HL,HL	SRA C	
42	*	2A	LD HL,NN	SRA D	
43	+	2B	DEC HL	SRA E	
44	,	2C	INC L	SRA H	
45	-	2D	DEC L	SRA L	
46	.	2E	LD L,N	SRA(HL)	
47	/	2F	CPL	SRA A	
48	0	30	JR NC,DIS		
49	1	31	LD SP,NN		
50	2	32	LD (NN),A		
51	3	33	INC SP		
52	4	34	INC (HL)		

53	5	35	DEC (HL)			
54	6	36	LD (HL).N			
55	7	37	SCF			
56	8	38	JR C.DIS	SRL B		
57	9	39	ADD HL.SP	SRL C		
58	:	3A	LD A.(NN)	SRL D		
59	;	3B	DEC SP	SRL E		
60	<	3C	INC A	SRL H		
61	=	3D	DEC A	SRL L		
62	>	3E	LD A.N	SRL(HL)		
63	?	3F	CCF	SRL A		
64	@	40	LD B,B	BIT 0,B	IN B,(C)	
65	A	41	LD B,C	BIT 0,C	OUT (C),B	
66	B	42	LD B,D	BIT 0,D	SBC HL,BC	
67	C	43	LD B,E	BIT 0,E	LD (NN),BC	
68	D	44	LD B,H	BIT 0,H	NEG	
69	E	45	LD B,L	BIT 0,L	RETN	
70	F	46	LD B,(HL)	BIT 0,(HL)	IM 0	
71	G	47	LD B,A	BIT 0,A	LD L,A	
72	H	48	LD C,B	BIT 1,B	IN C,(C)	
73	I	49	LD C,C	BIT 1,C	OUT C,(C)	
74	J	4A	LD C,D	BIT 1,D	ADD HL,BC	
75	K	4B	LD C,E	BIT 1,E	LD BC,(NN)	
76	L	4C	LD C,H	BIT 1,H		
77	M	4D	LD C,L	BIT 1,L	RETI	
78	N	4E	LD C,(HL)	BIT 1,(HL)		
79	O	4F	LD C,A	BIT 1,A	LD R,A	
80	P	50	LD D,B	BIT 2,B	IN D,(C)	
81	Q	51	LD D,C	BIT 2,C	OUT (C),D	
82	R	52	LD D,D	BIT 2,D	SBC HL,DE	
83	S	53	LD D,E	BIT 2,E	LD (NN),DE	
84	T	54	LD D,H	BIT 2,H		
85	U	55	LD D,L	BIT 2,L		
86	V	56	LD D,(HL)	BIT 2,(HL)	IM 1	
87	W	57	LD D,A	BIT 2,A	LD A,L	
88	X	58	LD E,B	BIT 3,B	IN E,(C)	
89	Y	59	LD E,C	BIT 3,C	OUT (C),E	
90	Z	5A	LD E,D	BIT 3,D	ADC HL,DE	
91	[	5B	LD E,E	BIT 3,E	LD DE,(NN)	
92	/	5C	LD E,H	BIT 3,H		
93	]	5D	LD E,L	BIT 3,L		
94	стрелка врх	5E	LD E,(HL)	BIT 3,(HL	IM 2	
95	-	5F	LD E,A	BIT 3,A	LD A,R	
96	фунт-стерл.	60	LD H,B	BIT 4,B	IN H,(C)	
97	A (строчн)	61	LD H,C	BIT 4,C	OUT (C),H	
98	B (строчн)	62	LD H,D	BIT 4,D	SBC HL,HL	
99	C (строчн)	63	LD H,E	BIT 4,E	LD (NN),HL	
100	D (строчн)	64	LD H,H	BIT 4,H		
101	E (строчн)	65	LD H,L	BIT 4,L		
102	F (строчн)	66	LD H,(HL)	BIT 4,(HL)		
103	G (строчн)	67	LD H,A	BIT 4,A	RRD	
104	H (строчн)	68	LD L,B	BIT 5,B	IN L,(C)	
105	I (строчн)	69	LD L,C	BIT 5,C	OUT(C),L	
106	J (строчн)	6A	LD L,D	BIT 5,D	ADC HL,HL	
107	K (строчн)	6B	LD L,E	BIT 5,E	LD HL,(NN)	
108	L (строчн)	6C	LD L,H	BIT 5,H		
109	M (строчн)	6D	LD L,L	BIT 5,L		
110	N (строчн)	6E	LD L,(HL)	BIT 5,(HL)		
111	O (строчн)	6F	LD L,A	BIT 5,A	RID	

112	P (строчн)	70	LD (HL),B	BIT 6,B	IN F,(C)
113	Q (строчн)	71	LD (HL),C	BIT 6,C	
114	R (строчн)	72	LD (HL),D	BIT 6,D	SBC HL,SP
115	S (строчн)	73	LD (HL),E	BIT 6,E	LD(NN),SP
116	T (строчн)	74	LD (HL),H	BIT 6,H	
117	U (строчн)	75	LD (HL),L	BIT 6,L	
118	V (строчн)	76	HALT	BIT 6,(HL)	
119	W (строчн)	77	LD (HL),A	BIT 6,A	
120	X (строчн)	78	LD A,B	BIT 7,B	IN A,(C)
121	y (строчн)	79	LD A,C	BIT 7,C	OUT (C),A
122	Z (строчн)	7A	LD A,D	BIT 7,D	ADC HL,SP
123	фигур.ск.лев	7B	LD A,E	BIT 7,E	LD SP,(NN)
124	верт.черта	7C	LD A,H	BIT 7,H	
125	фигур.ск.прв	7D	LD A,L	BIT 7,L	
126	дефис	7E	LD A,(HL)	BIT 7,H	
127	с в окр.	7F	LD A,A	BIT 7,A	
128	О О	80	ADD A,B	RES 0,B	
	О О				
129	О ж	81	ADD A,C	RES 0,C	
	О О				
130	ж О	82	ADD A,D	RES 0,D	
	О О				
131	ж О	83	ADD A,E	RES 0,E	
	О О				
132	О О	84	ADD A,H	RES 0,H	
	О ж				
133	О ж	85	ADD A,L	RES 0,L	
	О ж				
134	ж О	86	ADD A,(HL)	RES 0,(HL)	
	О ж				
135	ж ж	87	ADD A,A	RES 0,A	
	О ж				
136	О О	88	ADC A,B	RES 1,B	
	ж О				
137	О ж	89	ADC A,C	RES 1,C	
	ж О				
138	ж О	8A	ADC A,D	RES 1,D	
	ж О				
139	ж ж	8B	ADC A,E	RES 1,E	
	ж О				
140	О О	8C	ADC A,H	RES 1,H	
	ж ж				
141	О ж	8D	ADC A,L	RES 1,L	
	ж ж				
142	ж О	8E	ADC A,(HL)	RES 1,(HL)	
	ж ж				
143	ж ж	8F	ADC A,A	RES 1,A	
	ж ж				
144	(A)	90	SUB B	RES 2,B	
145	(B)	91	SUB C	RES 2,C	
146	(C)	92	SUB D	RES 2,D	
147	(D)	93	SUB E	RES 2,E	
148	(E)	94	SUB H	RES 2,H	
149	(F)	95	SUB L	RES 2,L	
150	(G)	96	SUB (HL)	RES 2,(HL)	
151	(H)	97	SUB A	RES 2,A	
152	(I)	98	SBC A,B	RES 3,B	
153	(J)	99	SBC A,C	RES 3,C	
154	(K)	9A	SBC A,D	RES 3,D	

155	(L)	9B	SBC A,E	RES 3,E	
156	(M)	9C	SBC A,H	RES 3,H	
157	(N)	9D	SBC A,L	RES 3,C	
158	(O)	9E	SBC A,(HL)	RES 3,D	
159	(P)	9F	SBC A,A	RES 3,E	
160	(Q)	A0	AND B	RES 4,B	LDI
161	(R)	A1	AND C	RES 4,C	CPI
162	(S)	A2	AND D	RES 4,D	INI
163	(T)	A3	AND E	RES 4,E	OUTI
164	(U)	A4	AND H	RES 4,H	
165	RND	A5	AND L	RES 4,L	
166	INKEY$	A6	AND (HL)	RES 4,(HL)	
167	PI	A7	AND A	RES 4,A	
168	FN	A8	XOR B	RES 5,B	LDD
169	POINT	A9	XOR C	RES 5,C	CPD
170	SCREEN$	AA	XOR D	RES 5,D	IND
171	ATTR	AB	XOR E	RES 5,E	OUTD
172	AT	AC	XOR H	RES 5,H	
173	TAB	AD	XOR L	RES 5,L	
174	VAL$	AE	XOR (HL)	RES 5,(HL)	
175	CODE	AF	XOR A	RES 5,A	
176	VAL	B0	OR B	RES 6,B	LDIR
177	LEN	B1	OR C	RES 6,C	CPIR
178	SIN	B2	OR D	RES 6,D	INIR
179	COS	B3	OR E	RES 6,E	OTIR
180	TAN	B4	OR H	RES 6,H	
181	ASN	B5	OR L	RES 6,L	
182	ACS	B6	OR (HL)	RES 6,(HL)	
183	ATN	B7	OR A	RES 6,A	
184	LN	B8	CP B	RES 7,B	LDDR
185	EXP	B9	CP C	RES 7,C	CPDR
186	INT	BA	CP D	RES 7,D	INDR
187	SQR	BB	CP E	RES 7,E	OTDR
188	SGN	BC	CP H	RES 7,H	
189	ABS	BD	CP L	RES 7,L	
190	PEEK	BE	CP (HL)	RES 7,(HL)	
191	IN	BF	CP A	RES 7,A	
192	USR	C0	RET NZ	SET 0,B	
193	STR$	C1	POP BC	SET 0,C	
194	CHR$	C2	JP NZ,NN	SET 0,D	
195	NOT	C3	JP NN	SET 0,E	
196	BIN	C4	CALL NZ,NN	SET 0,H	
197	OR	C5	PUSH BC	SET 0,L	
198	AND	C6	ADD A,N	SET 0,(HL)	
199	<=	C7	RST 0	SET 0,A	
200	>=	C8	RET Z	SET 1,B	
201	<>	C9	RET	SET 1,C	
202	LINE	CA	JP Z,NN	SET 1,D	
203	THEN	CB		SET 1,E	
204	TO	CC	CALL Z,NN	SET 1,H	
205	STEP	CD	CALL NN	SET 1,L	
206	DEF FN	CE	ADC A,N	SET 1,(HL)	
207	CAT	CF	RST 8	SET 1,A	
208	FORMAT	D0	RET NC	SET 2,B	
209	MOVE	D1	POP DE	SET 2,C	
210	ERASE	D2	JP NC,NN	SET 2,D	
211	OPEN#	D3	OUT (N),A	SET 2,E	
212	CLOSE#	D4	CALL NC,NN	SET 2,H	
213	MERGE	D5	PUSH DE	SET 2,L	

214	VERIFY	D6	SUB N	SET 2,(HL)	
215	BEEP	D7	RST 16	SET 2,A	
216	CIRCLE	D8	RET C	SET 3,B	
217	INK	D9	EXX	SET 3,C	
218	PAPER	DA	JP C,NN	SET 3,D	
219	FLASH	DB	IN A,(N)	SET 3,E	
220	BRIGHT	DC	CALL C,NN	SET 3,H	
221	INVERSE	DD	PREFIXES IN-	SET 3,L	
			STRUCTIONS		
			USING IX		
222	OVER	DE	SBC A,N	SET 3,(HL)	
223	OUT	DF	RST 24	SET 3,A	
224	LPRINT	E0	RET PO	SET 4,B	
225	LLIST	E1	POP HL	SET 4,C	
226	STOP	E2	JP PO,NN	SET 4,D	
227	READ	E3	EX (SP),HL	SET 4,E	
228	DATA	E4	CALL PO,NN	SET 4,H	
229	RESTORE	E5	PUSH HL	SET 4,L	
230	NEW	E6	AND N	SET 4,(HL)	
231	BORDER	E7	RST 32	SET 4,A	
232	CONTINUE	E8	RET PE	SET 5,B	
233	DIM	E9	JP (HL) ,	SET 5,C	
234	REM	EA	JP PE,NN	SET 5,D	
235	FOR	EB	EX DE,HL	SET 5,E	
236	GO TO	EC	CALL PE,NN	SET 5,H	
237	GO SUB	ED		SET 5,L	
238	INPUT	EE	XOR N	SET 5,(HL)	
239	LOAD	EF	RST 40	SET 5,A	
240	LIST	F0	RET P	SET 6,B	
241	LET	F1	POP AF	SET 6,C	
242	PAUSE	F2	JP P,NN	SET 6,D	
243	NEXT	F3	DI	SET 6,E	
244	POKE	F4	CALL P,NN	SET 6,H	
245	PRINT	F5	PUSH AF	SET 6,L	
246	PLOT	F6	OR N	SET 6,(HL)	
247	RUN	F7	RST 48	SET 6,A	
248	SAVE	F8	RET M	SET 7,B	
249	RANDOMIZE	F9	LD SP,HL	SET 7,C	
250	IF	FA	JP M,NN	SET 7,D	
251	CLS	FB	EI	SET 7,E	
252	DRAW	FC	CALL M,NN	SET 7,H	
253	CLEAR	FD	PREFIXES	SET 7,L	
			INSTRUCTIONS		
			USING IY		
254	RETURN	FE	CP N	SET 7,(HL)	
255	COPY	FF	RST 56	SET 7,A	

Приложение 2

ШЕСТНАДЦАТИРИЧНАЯ И ДВОИЧНАЯ СИСТЕМЫ СЧИСЛЕНИЯ

В компьютере ZX SPECTRUM используется шестнадцатиричная система счисления. При этом каждая шестнадцатиричная цифра записывается четырьмя двоичными (тетрада). Таким образом в одном байте может быть записано два шестнадцатиричных числа.

10-тичное	16-ричное	2-ичное(байт)
0	0	0000 0000
1	1	0000 0001
2	2	0000 0010
3	3	0000 0011
4	4	0000 0100
5	5	0000 0101
6	6	0000 0110
7	7	0000 0111
8	8	0000 1000
9	9	0000 1001
10	A	0000 1010
11	B	0000 1011
12	C	0000 1100
13	D	0000 1101
14	E	0000 1110
15	F	0000 1111
16	10	0001 0000
17	11	0001 0001
18	12	0001 0010
19	13	0001 0011
20	14	0001 0100
21	15	0001 0101
22	16	0001 0110
23	17	0001 0111
24	18	0001 1000
25	19	0001 1001
26	1A	0001 1010
27	1B	0001 1011
28	1C	0001 1100
29	1D	0001 1101
30	1E	0001 1110
31	1F	0001 1111
32	20	0010 0000
...	...	...
65535	FFFF	1111 1111

Два байта образуют машинное слово. Для записи двоичных кодов служит функция BIN. Например "BIN 0" запишет в память двоичный 0. "BIN 10" записывает число два и т.д. Для записи "-3" необходимо указать "-BIN 11",но не "BIN -11".

Приложение 3

СООБЩЕНИЯ

Они появляются в нижней части экрана, если компьютер остановился при выполнении некоторого оператора Бейсика, и указывают причину, вызвавшую останов. Сообщение содержит кодовый номер или букву, краткое содержание помогает найти ошибочную строку и ошибочный оператор в этой строке. (команда указывается как строка 0, оператор 1 располагается в строке первым, оператор 2 следует после первого или THEN и т.д.)

От состояния CONTINUE зависит очень многое в сообщениях. Обычно

продолжение начинается с оператора, специфицированного в предыдущем сообщении, но имеются исключения - сообщения 0,9 и D.

Код	Значение	Ситуация
1	2	3
0	OK (О' кей! порядок)	разное
	Успешное завершение или переход на строку с номером, большим чем имеется всего. Это сообщение не имеет строки или оператора, определенного для CONTINUE	
1	NEXT WITHOUT FOR (NEXT без FOR)	NEXT
	Управляющей переменной нет (не была определена в операторе FOR), но есть обычная переменная с тем же именем.	
2	VARIABLE NOT FOUND (переменная не найдена)	разное
	Для простой переменной выдается, если она используется без предварительного определения в операторах LET,READ или INPUT, или загружается с ленты или устанавливается в операторе FOR. Для индексируемой переменной сообщение выдается, если она не была предварительно определена в операторе DIM перед использованием или загрузкой с ленты.	
3	SUBSCRIPT WRONG (ошибочный индекс)	в индексной переменной или подстроке
	Индекс превышает размерность массива, либо ошибочное число задает индекс. Если индекс отрицательный или больше 65536, то выдается сообщение B.	
4	OUT OF MEMORY (вне памяти)	LET INPUT FOR DIM GO SUB LOAD MERGE
	В памяти недостаточно места для Ваших действий. Вы можете освободить себе память, удалив командные строки, используя DELETE затем удалить одну или две строки программы (с целью возврата их впоследствии), получить дополнительную память, маневрируя оператором CLEAR.	
5	OUT OF SCREEN (вне экрана)	INPUT PRINT AT
	Если INPUT оператор генерирует больше чем 23 строки в нижней половине экрана. Также встречается с PRINT AT 22,....	
6	NUMBER TOO BIG (число больше максимально допустимого)	арифметические операции
	В результате вычисления получилось число, больше, чем 10**38.	

7	RETURN WITHOUT GO SUB (RETURN без GO SUB) Встретилось больше операторов RETURN, чем было операторов GO SUB	RETURN
8	END OF FILE (конец файла)	операции с внешней памятью
9	STOP STATEMENT (оператор STOP) После этого сообщения CONTINUE не может повторить STOP, но может передать управление на следующий оператор.	STOP
A	INVALID ARGUMENT (ошибочный аргумент) Аргумент функции не допустим в данной версии	SQR LN ASN ACS USR со строковым аргументом
B	INTEGER OUT OF RANGE (переполнение целого) Выдается, когда аргумент с плавающей точкой округляется к целому. Для случая массивов смотри также сообщение 3.	RUN RANDOM POKE GO TO GO SUB LIST LLIST PAUSE PLOT CHR$ PEEK USR(с числовым аргументом)
C	NONSENSE IN BASIC (выражение не Бейсика) Текст (строка) не распознается Бейсиком как допустимое выражение	VAL VAL$
D	BREAK-CONT REPEATS (повторяется BREAK) Клавиша BREAK нажата во время действия периферийной операции. Действия CONTINUE после этого оператора обычные, те что указаны в операторе. Сравните с сообщением L.	LOAD SAVE VERIFY MERGE LPRINT LLIST COPY(только когда компьютер запросил свертку, а Вы ответили N.SPACE или STOP
E	OUT OF DATA (вне данных)	READ

	Попытка выдать READ, когда список данных в DATA кончился.	
F	INVALID FILE NAME (неверное имя файла) Оператор SAVE с пустой строкой вместо имени файла или с именем длиннее 10 символов.	SAVE
G	NO ROOM FOR LINE (нет места для строки) Недостаточно места в памяти для записи очередной строки программы	ввод строки в программу
H	STOP IN INPUT Некоторые введенные данные начинаются с оператора STOP, или была нажата INPUT LINE. Действие CONTINUE обычное.	INPUT
I	FOR WITHOUT NEXT (FOR без NEXT) Цикл FOR ни разу не выполнялся, не найден NEXT оператор.	FOR
J	INVALID I/O DEVICE (неверное устройство ввода-вывода)	в операции с внешними устройствами
K	INVALID COLOUR (неверный цвет) Специфицированное число имеет неверное значение	INK PAPER BORDER FLASH BRIGHT INVERS OVER а также после одного из перечисленных данных управляющих символов
L	BREAK INTO PROGRAMM (BREAK во время выполнения программы) Нажата клавиша BREAK; Это обнаруживается между двумя операторами. Строка и номер оператора в строке указывает на оператор, выполненый перед нажатием BREAK, но CONTINUE переходит к следующему оператору.	разное
M	RAMTOP NO GOOD (адрес RAMTOP не годен) Число, указанное для RAMTOP слишком велико или слишком мало.	CLEAR возможно RUN
N	STATMENT LOST (оператор отсутствует) Переход к оператору, которого уже нет	RETURN NEXT CONTINUE

O	INVALID STREAM (ошибочный поток данных) Определяемая пользователем функция не определена в операторе DEF FN.	в операциях ввода-вывода
P	FN WITHOUT DEF (FN без DEF)	FN
Q	PARAMETER ERROR (ошибка в параметре) Ошибочное число аргументов или один из них не того типа, который был описан.	FN
R	TAPE LOADING ERROR (ошибка загрузки с ленты) Файл на ленте найден, но не считывается.	VERIFY LOAD MERGE

Приложение 4

ПРИМЕРЫ ПРОГРАММ

Это приложение содержит некоторые примеры программ, демонстрирующие возможности ZX SPECTRUM.

Первая из этих программ требует ввести дату и дает день недели, который соответствует этой дате.

```
 10 REM CONVERT DATE TO DAY
 20 DIM D$(7,6): REM DAYS OF WEEK
 30 FOR N=1 TO 7: READ D$(N): NEXT N
 40 DIM M(12): REM LENGTHS OF MONTHS
 50 FOR N=1 TO 12: READ M(N): NEXT N
100 REM INPUT DATE
110 INPUT "DAY?";DAY
120 INPUT "MONTH?";MONTH
130 INPUT "YEAR (20TH CENTURY ONLY)?";YEAR
140 IF YEAR<1901 THEN PRINT "20TH CENTURY STARTS
    AT 1901": GO TO 100
150 IF YEAR>2000 THEN PRINT "20TH CENTURY ENDS
    AT 2000": GO TO 100
160 IF MONTH<1  THEN GO TO 210
170 IF MONTH>12 THEN GO TO 210
180 IF YEAR/4-INT(YEAR/4)=0 THEN LET M(2)=29:
    REM LEAP YEAR
190 IF DAY>M(MONTH) THEN PRINT "THIS MONTH HAS ONLY";
    M(MONTH);" DAYS.": GO TO 500
200 IF DAY>0 THEN GO TO 300
210 PRINT "STUFF AND NONSENSE. GIVE ME A REAL DATE."
220 GO TO 500
300 REM CONVERT DATE TO NUMBER OF DAYS SINCE START OF
    CENTURY
310 LET Y=YEAR-1901
320 LET B=365*Y+INT(Y/4): REM NUMBER OF DAYS TO START
    OF YEAR
330 FOR N=1 TO MONTH-1: REM ADD ON PREVIOUS MONTH
340 LET B=B+M(N): NEXT N
350 LET B=B+DAY
400 REM SONVERT TO DAY OF WEEK
410 LET B=B-7*INT(B/7)+1
```

```
420 PRINT DAY;"/";MONTH;"/";YEAR
430 FOR N=6TO 3 STEP-1: REM REMOVE TRAILING SPACES
440 IF D$(B,N) <>" " THEN GO TO 460
450 NEXT N
460 LET E$=D$(B,TO N)
470 PRINT "IS A"; E$;"DAY"
500 LET M(2)=28: REM RESTORE FEBRUARY
510 INPUT "AGAIN?",A$
520 IF A$="N" THEN GO TO 540
530 IF A$<>"N" THEN GO TO 100
1000 REM DAYS OF WEEK
1010 DATA "MON","TUES","WEDNES"
1020 DATA "THURS","FRI","SATUR","SUN"
```

Эта программа устанавливает соответствие между ярдом, футом и дюймом:

```
10 INPUT "YARDS?", YD,"FEET?",FT,"INCHES?",IN
40 GO SUB 2000: REM PRINT THE VALUES
50 PRINT "  =  ";
70 GO SUB 1000: REM THE ADJUSTMENT
80 GO SUB 2000: REM PRINT THE ADJUSTED VALUES
90 PRINT
100 GO TO 10
1000 REM SUBROUTINE TO ADJUST YD.FT.IN TO THE NORMAL FORM
     FOR YARDS. FEET AND INCHES
1010 LET IN=36*YD+12*FT+IN: REM NOW EVERTH IS IN
     INCHES
1030 LET S=SGN IN: LET IN=ABS IN: REM WE WORK WITH IN
     POSITIVE. HOLDING ITS SIGN IN S
1060 LET FT=INT(IN/12): LET IN=(IN-12*FT)*S: REM NOW
     IN IS OK
1080 LET YD=INT(FT/3)*S: LET FT=FT*S-3*YD: RETURN
2000 REM SUBROUTINE TO PRINT YD.FT AND IN
2040 PRINT YD:"YD";FT;"FT";IN;"IN":: RETURN
```

Следующая программа рисует на экране "UNION FLAG"

```
  5 REM UNION FLAG
 10 LET R=2: LET W=7: LET B=1
 20 BORDER 0: PAPER B: INK W: CLS
 30 REM BLACK IN BOTTOM OF SCREEN
 40 INVERSE 1
 50 FOR N=40 TO 0 STEP -8
 60 PLOT PAPER 0;7,N: DRAW PAPER 0;241,0
 70 NEXT N: INVERSE 0
100 REM DRAW IN WHITE PARTS
105 REM ST. GEORGE
110 FOR N=0 TO 7
120 PLOT 104+N,175: DRAW 0,-35
130 PLOT 151-N,175: DRAW 0,-35
140 PLOT 151-N,48:  DRAW 0,35
150 PLOT 104+N,48:  DRAW 0,35
160 NEXT N
200 FOR N=0 TO 11
210 PLOT 0,139-N: DRAW 111,0
220 PLOT 255,139-N: DRAW -111,0
230 PLOT 255,84+N: DRAW -111,0
240 PLOT 0,84+N: DRAW 111,0
250 NEXT N
300 REM ST. ANDREW
```

```
310 FOR N=0 TO 35
320 PLOT 1+2*N,175-N: DRAW 32,0
330 PLOT 224-2*N,175-N: DRAW 16,0
340 PLOT 254-2*N,48+N: DRAW -32,0
350 PLOT 17+2*N,48+N: DRAW 16,0
360 NEXT N
370 FOR N=0 TO 19
380 PLOT 185+2*N,140+N: DRAW 32,0
390 PLOT 200+2*N,83-N: DRAW 16,0
400 PLOT 39-2*N,83-N: DRAW 32,0
410 PLOT 54-2*N,140+N: DRAW -16,0
420 NEXT N
425 REM FILL IN EXTRA BITS
430 FOR N=0 TO 15
440 PLOT 255,160+N: DRAW 2*N-30,0
450 PLOT 0,63-N: DRAW 3102*N,0
460 NEXT N
470 FOR N=0 TO 7
480 PLOT 0,160+N: DRAW 14-2*N,0
485 PLOT 255,63-N: DRAW 2*N-15,0
490 NEXT N
500 REM RED STRIPES
510 INVERSE 1
520 REM ST. GEORGE
530 FOR N=96 TO 120 STEP 8
540 PLOT PAPER R;7,N: DRAW PAPER R;241,0
550 NEXT N
560 FOR N=112 TO 136 STEP 8
570 PLOT PAPER R;N,168: DRAW PAPER R;0,-113
580 NEXT N
600 REM ST. PATRICK
610 PLOT PAPER R;170,140: DRAW PAPER R;70,35
620 PLOT PAPER R;179,140: DRAW PAPER R;70,35
630 PLOT PAPER R;199,83: DRAW PAPER R;56,-28
640 PLOT PAPER R;184,83: DRAW PAPER R;70,-35
650 PLOT PAPER R;86,83: DRAW PAPER R;-70,-35
660 PLOT PAPER R;72,83: DRAW PAPER R;-70,-35
670 PLOT PAPER R;56,140: DRAW PAPER R;-56,28
680 PLOT PAPER R;71,140: DRAW PAPER R;-70,35
690 INVERSE 0: PAPER 0: INK 7
```

Если Вы не англичанин, то можете изобразить свой флаг.

Следующая программа - это игра в слова. Первый игрок вводит слово, а второй его отгадывает.

```
  5 REM HANGMAN
 10 REM SET UP SCREEN
 20 INK 0: PAPER 7: CLS
 30 LET X=240: GO SUB 1000: REM DRAW MAN
 40 PLOT 238,128: DRAW 4,0: REM MOUTH
100 REM SET UP WORD
110 INPUT W$: REM WORD TO GUESS
120 LET B=LEN W$: LET V$=" "
130 FOR N=2 TO B: LET V$=V$+" "
140 NEXT N: REM V$=WORD GUESSED SO FAR
150 LET C=0: LET D=0: REM GUESS   MISTAKE COUNTS
160 FOR N=0 TO B-1
170 PRINT AT 20,N;"-";
180 NEXT N: REM WRITE-"S INSTEAD OF LETTERS
200 INPUT "GUESS A LETTER: ";G$
210 IF G$="" THEN GO TO 200
220 LET G$=G$(1): REM 1ST LETTER oNLy
230 PRINT AT 0,C;G$
```

```
240 LET C=C+1: LET U$=V$
250 FOR N=1 TO B: REM UPDATE GUESSED WORD
260 IF W$(N)=G$ THEN LET V$(N)=G$
270 NEXT N
280 PRINT AT 19,0:V$
290 IF V$=W$ THEN GO TO 500: REM WORD GUESSED
300 IF V$<>U$ THEN GO TO 200: REM GUESSED WAS RIGHT
400 REM DRAW NEXT PART OF GALLOWS
410 IF D=8 THEN GO TO 600: REM HANGED
420 LET D=D+1
430 READ X0,Y0,X,Y
440 PLOT X0,Y0: DRAW X,Y
450 GO TO 200
500 REM FREE MAN
510 OVER1: REM RUB OUT MAN
520 LET X=240: GO SUB 1000
530 PLOT 238,128: DRAW 4,0: REM MOUTH
540 OVER 0: REM REDRAW MAN
550 LET X=146: GO SUB 1000
560 PLOT 143,129: DRAW 6,0,PI/2: REM SMILE
570 GO TO 800
600 REM HANG MAN
610 OVER 1: REM RUB OUT FLOOR
620 PLOT 255,65: DRAW -48,0
630 DRAW 0,-48: REM OPEN TRAPDOOR
640 PLOT 238,128: DRAW 4,0: REM RUB OUT MOUTH
650 REM MOVE LIMBS
655 REM ARMS
660 PLOT 255,117: DRAW -15,-15: DRAW -15,15
670 OVER 0
680 PLOT 236,81: DRAW 4,21: DRAW 4,-21
690 OVER 1: REM LEGS
700 PLOT 255,66: DRAW -15,15: DRAW -15,-15
710 OVER 0
720 PLOT 236,60: DRAW 4,21: DRAW 4,-21
730 PLOT 237,127: DRAW 6,0,-PI/2: REM FROWN
740 PRINT AT 19,0;W$
800 INPUT "AGAIN?";A$
810 IF A$="" THEN GO TO 850
820 LET A$=A$(1)
830 IF A$="N" THEN STOP
840 IF A$(1)="N" THEN STOP
850 RESTORE : GO TO 5
1000 REM DRAW MAN AT COLUMN X
1010 REM HEAD
1020 CIRCLE X,132,8
1030 PLOT X+4,134: PLOT X-4,134: PLOT X,131
1040 REM BODY
1050 PLOT X,123: DRAW 0,-20
1055 PLOT X,101: DRAW 0,-19
1060 REM LEGS
1070 PLOT X-15,66: DRAW 15,15: DRAW 15,-15
1080 REM ARMS
1090 PLOT X-15,117: DRAW 15,-15: DRAW 15,15
1100 RETURN
2000 DATA 120,65,135,0,184,65,0,91
2010 DATA 168,65,16,16,184,81,16,-16
2020 DATA 184,156,68,0,184,140,16,16
2030 DATA 204,156,-20,-20,240,156,0,-16
2020 DATA 184,156,68,0,184,140,16,16
2030 DATA 204,156,-20,-20,240,156,0,-16
```

СИСТЕМА КОМАНД МИКРОПРОЦЕССОРА Z80

	0	1	2	3	4	5	6	7
0	NOP	LD BC,dddd	LD (BC),A	INC BC	INC B	DEC B	LD B,dd	RLCA
1	DJNZ dd	LD DE,dddd	LD (DE),A	INC DE	INC D	DEC D	LD D,dd	RLA
2	JR NZ,dd	LD HL,dddd	LD (ad),HL	INC HL	INC H	DEC H	LD H,dd	DAA
3	JR NC,dd	LD SP,dddd	LD (adr),A	INC SP	INC (HL)	DEC (HL)	LD (HL),dd	SCF
4	LD B,B	LD B,C	LD B,D	LD B,E	LD B,H	LD B,L	LD B,(HL)	LD B,A
5	LD D,B	LD D,C	LD D,D	LD D,E	LD D,H	LD D,L	LD D,(HL)	LD D,A
6	LD H,B	LD H,C	LD H,D	LD H,E	LD H,H	LD H,L	LD H,(HL)	LD H,A
7	LD (HL),B	LD (HL),C	LD (HL),D	LD (HL),E	LD (HL),H	LD (HL),L	HALT	LD (HL),
8	ADD A,B	ADD A,C	ADD A,D	ADD A,E	ADD A,H	ADD A,L	ADD A,(HL)	ADD A,A
9	SUB B	SUB C	SUB D	SUB E	SUB H	SUB L	SUB (HL)	SUB A
A	AND B	AND C	AND D	AND E	AND H	AND L	AND (HL)	AND A
B	OR B	OR C	OR D	OR E	OR H	OR L	OR (HL)	OR A
C	RET NZ	POP BC	JP NZ,addr	JP addr	CALL NZ,add	PUSH BC	ADD A,dd	RST 00
D	RET NC	POP DE	JP NC,addr	OUT (dd),A	CALL NC,add	PUSH DE	SUB dd	RST 10
E	RET PO	POP HL	JP PO,addr	EX (SP),HL	CALL PO,add	PUSH HL	AND dd	RST 20
F	RET P	POP AF	JP addr	DI	CALL P,addr	PUSH AF	OR dd	RST 30

	8	9	A	B	C	D	E	F
0	EX AF,AF'	ADD HL,BC	LD A,(BC)	DEC BC	INC C	DEC C	LD C,dd	RRCA
1	JR dd	ADD HL.DE	LD A,(DE)	DEC DE	INC E	DEC E	LD E,dd	RRA
2	JR Z,dd	ADD HL,HL	LD HL,(ad)	DEC HL	INC L	DEC L	LD L,dd	CPL
3	JR C.dd	ADD HL,SP	LD A,(adr)	DEC SP	INC A	DEC A	LD A,dd	CCF
4	LD C,B	LD C,C	LD C,D	LD C,E	LD C,H	LD C,L	LD C,(HL)	LD C,A
5	LD E,B	LD E,C	LD E,D	LD E,E	LD E,H	LD E,L	LD E,(HL)	LD E,A
6	LD L,B	LD L,C	LD L,D	LD L,E	LD L,H	LD L,L	LD L,(HL)	LD L,A
7	LD A,B	LD A,C	LD A,D	LD A,E	LD A,H	LD A,L	LD A,(HL)	LD A,A
8	ADC A,B	ADC A,C	ADC A,D	ADC A,E	ADC A,H	ADC A,L	ADC A,(HL)	ADC A,A
9	SBC A.B	SBC A.C	SBC A,D	SBC A,E	SBC A,H	SBC A,L	SBC A,(HL)	SBC A,A
A	XOR B	XOR C	XOR D	XOR E	XOR H	XOR L	XOR (HL)	XOR A
B	CP B	CP C	CP D	CP E	CP H	CP L	CP (HL)	CP A
C	RET Z	RET	JP Z.addr	* см. ниже	CALL Z.addr	CALL addr	ADC A,dd	RST 08
D	RET C	EXX	JP C.addr	IN A,(dd)	CALL C,addr	* см. ниже	SBC A,dd	RST 18
E	RET PE	JP (HL)	JP PE,addr	EX DE,HL	CALL PE,add	* см. ниже	XOR dd	RST 28
F	RET M	LD SP,HL	JP M,addr	EI	CALL M,addr	* см. ниже	CP dd	RST 38

Все команды, использующие регистровую пару IX, начинаются с кода DD, а те, которые используют регистровую пару IY -начинаются с кода FD.

В нижеследующей таблице для регистровой пары IY замените код DD на код FD. а мнемонику IX - на IY.

DD 09	ADD IR,BC	!	DD 70 d	LD (IR+d),C
DD 19	ADD IR,DE	!	DD 72 d	LD (IR+d),D
DD 21 +dddd	LD IR+dddd	!	DD 73 d	LD (IR+d),E
DD 22 addr	LD (addr),IR	!	DD 74 d	LD (IR+d),H
DD 23	INC IR	!	DD 75 d	LD (IR+d),L
DD 29	ADD IR,IR	!	DD 77 d	LD (IR+d),A
DD 2A addr	LD IR,(addr)	!	DD 7E d	LD A,(IR+d)
DD 2B	DEC IR	!	DD 86 d	ADD A,(IR+d)
DD 34 d	INC (IR+d)	!	DD 8E d	ADC A,(IR+d)
DD 35 d	DEC (IR+d)	!	DD 96 d	SUB (IR+d)
DD 38 d+dd	LD (IR+d),+dd	!	DD 9E d	SBC A,(IR+d)
DD 39	ADD IR,SP	!	DD A6 d	AND (IR+d)
DD 46 d	LD B,(IR+d)	!	DD AE d	XOR (IR+d)
DD 4E d	LD C,(IR+d)	!	DD B6 d	OR (IR+d)
DD 56 d	LD D,(IR+d)	!	DD BE d	CP (IR+d)
DD 5E d	LD E,(IR+d)	!	DD CB d 06	RLC (IR+d)
DD 66 d	LD H,(IR+d)	!	DD CB d 0E	RRC (IR+d)
DD 6E d	LD L,(IR+d)	!	DD CB d 16	RL (IR+d)
DD CB d 1E	RR (IR+d)	!	DD CB d B6	RES 6,(IR+d)
DD CB d 26	SLA (IR+d)	!	DD CB d BE	RES 7,(IR+d)
DD CB d 2E	SRA (IR+d)	!	DD CB d C6	SET 0,(IR+d)
DD CB d 3E	SRL (IR+d)	!	DD CB d CE	SET 1,(IR+d)
DD CB d 46	BIT 0,(IR+d)	!	DD CB d D6	SET 2,(IR+d)
DD CB d 4E	BIT 1,(IR+d)	!	DD CB d DE	SET 3,(IR+d)
DD CB d 56	BIT 2,(IR+d)	!	DD CB d E6	SET 4,(IR+d)
DD CB d 5E	BIT 3,(IR+d)	!	DD CB d EE	SET 5,(IR+D)
DD CB d 66	BIT 4,(IR+d)	!	DD CB d F6	SET 6,(IR+d)
DD CB d 6E	BIT 5,(IR+d)	!	DD CB d FE	SET 7,(IR+d)
DD CB d 76	BIT 6,(IR+d)	!	DD E1	POP IR
DD CB d 7E	BIT 7,(IR+d)	!	DD E3	EX (SP),IR
DD CB d 86	RES 0,(IR+d)	!	DD E5	PUSH IR
DD CB d 8E	RES 1,(IR+d)	!	DD E9	JP (IR)
DD CB d 96	RES 2,(IR+d)	!	DD F9	LD SP,IR
DD CB d 9E	RES 3,(IR+d)	!		
DD CB d A6	RES 4,(IR+d)	!		
DD CB d AE	RES 5,(IR+d)	!		

Команды с префиксами ED и CB

ED 40	IN B,(C)	!	ED 67	RRD
ED 41	OUT (C),B	!	ED 68	IN L,(C)
ED 42	SBC HL,BC	!	ED 69	OUT (C),L
ED 43 addr	LD (addr),BC	!	ED 6A	ADC HL,HL
ED 44	NEG	!	ED 6B addr	LD HL,(addr)
ED 45	RETN	!	ED 6F	RLD
ED 46	IM 0	!	ED 72	SBC HL,SP
ED 47	LD I,A	!	ED 73 addr	LD (addr),SP
ED 48	IN C,(C)	!	ED 78	IN A,(C)
ED 49	OUT (C),C	!	ED 79	OUT (C),A
ED 4A	ADC HL,BC	!	ED 7A	ADC HL,SP
ED 4B addr	LD BC,(addr)	!	ED 7B addr	LD SP,(addr)
ED 4D	RETI	!	ED A0	LDI
ED 50	IN D,(C)	!	ED A2	INI
ED 51	OUT (C),D	!	ED A3	OUTI
ED 52	SBC HL,DE	!	ED A8	LDD
ED 53 addr	LD (addr),DE	!	ED A9	CPD
ED 56	IM 1	!	ED AA	IND
ED 57	LD A,I	!	ED AB	OUTD
ED 58	IN E,(C)	!	ED B0	LD IR
ED 59	OUT (C),E	!	ED B1	CPIR

ED 5A	ADC HL,DE	!	ED B2	INIR
ED 5B addr	LD DE,(addr)	!	ED B3	OTIR
ED 5E	IM 2	!	ED B8	LDDR
ED 5F	LD A,R	!	ED B9	CPDR
ED 60	IN H,(C)	!	ED BA	INDR
ED 61	OUT (C),H	!	ED BB	OTDR
ED 63 addr	LD (addr),HL	!		
CB 0+R0	RLC R	!	CB 8+R0	RES 0,(HL)
CB 0+R1	RRC R	!	CB 8+R1	RES 1,(HL)
CB 1+R0	RL R	!	CB 9+R0	RES 2,(HL)
CB 1+R1	RR R	!	CB 9+R1	RES 3,(HL)
CB 2+R0	SLA R	!	CB A+R0	RES 4,(HL)
CB 2+R1	SRA R	!	CB A+R1	RES 5,(HL)
CB 3+R0	SRL R	!	CB B+R0	RES 6,(HL)
CB 3+R1	SRR R	!	CB B+R1	RES 7,(HL)
CB 4+R0	BIT 0,R	!	CB C+R0	SET 0,(HL)
CB 4+R1	BIT 1,R	!	CB C+R1	SET 1,(HL)
CB 5+R0	BIT 2,R	!	CB D+R0	SET 2,(HL)
CB 5+R1	BIT 3,R	!	CB D+R1	SET 3,(HL)
CB 6+R0	BIT 4,R	!	CB E+R0	SET 4,(HL)
CB 6+R1	BIT 5,R	!	CB E+R1	SET 5,(HL)
CB 7+R0	BIT 6,R	!	CB F+R0	SET 6,(HL)
CB 7+R1	BIT 7,R	!	CB F+R1	SET 7,(HL)

Группа R0: 0 - B, 1 - C, 2 - D, 3 - E, 4 - H, 5 - L, 6 - (HL), 7 - A.

Группа R1: 8 - B, 9 - C, A - D, B - E, C - H, D - L, E - (HL), F - A.

Для получения кода команды необходимо подставить код группы (R0 или R1) в младшую тетраду второго байта.