BASIC

Владельцам SPECTRUM

ОПИСАНИЕ КОМАНД И
ОСНОВЫ ПРОГРАММИРОВАНИЯ
ПРИМЕРЫ ПРОГРАММ

SPECTRUM ZX

ВНИМАНИЕ!

ЭТО ИМЕННО ТО,
ЧТО ВАМ НЕОБХОДИМО!

Содержание Стр

Глава 1
Введение

Если вы читаете эту книгу впервые или открыли ее на этом листе, то вы должны иметь представление о том, что команды бейсика выполняются непосредственно, операторы начинаются с номера строки и сохраняются в памяти компьютера. Вы должны также представлять себе, что такие команды, как PRINT, LET и INPUT, используются во всех компьютерах, имеющих бейсик, а такие команды, как BORDER, PAPER и BEEP, используются в ZX SPECTRUM.

Запуск бейсика начнем с повторения некоторых моментов, изложеных во вводной части, но рассмотрим их значительно более полно, уяснив, что можно делать, а что нельзя.

Что бы вы не делали, старайтесь в своей деятельности использовать компьютер, если у вас возник вопрос «что будет, если я скажу так и так?», тогда ответ для вас очень прост: введите эти фразы в компьютер и вы увидите!

Всякий раз, когда в этой книге вы встретите предложение что-нибудь ввести в компьютер и выполнить на нем, спрашивайте сами себя: "Что я могу сделать вместо этого?" и попробуйте это проделать, чем больше собственных программ вы напишете, тем лучше вы будете понимать компьютер.

В конце этой книги имеется несколько приложений, они содержат сведения по организации памяти, как компьютер оперирует с числами, а также несколько примеров программ, иллюстрирующих возможности ZX SPECTRUM.

Клавиатура

В ZX SPECTRUM клавиши содержат не только одиночные символы (буквы, цифры и т. д.), но также составные символы (ключевые слова, названия функций и т. п.) и все то, что вводится с клавиатуры не посимвольно, для того, чтобы реализовать все эти функции и команды. Некоторые клавиши клавиатуры имеют 5 и более значений, получаемых либо путем набора соответствующего регистра (т. е. путем нажатия клавиш 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.

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 оператора THEN, или же к знакоместу сразу же после «:» (за исключением двоеточия в строке). Если не изменен режим, то нажатие следующей клавиши будет интерпретироваться как ключевое слово, написанное на клавише, либо как цифра.

Режим курсора (L)(для букв) появляется обычно во всех других случаях. Если он не меняется, то нажатие следующей клавиши будет интерпретировано как основной символ на клавише. В большинстве случаев это буквы. 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. Режим курсора (C) (для заглавных букв) -это вариант режима (L), в котором все буквы появляются на экране как заглавные.

Нажатие клавиши CAPS LOCK приводит к смене курсора (L) на (C) или наоборот. Режим курсора (E) (расширение) используется для получения дополнительных символов (обычно знаков). Курсор (E) появляется после одновременного нажатия обеих клавиш смены режима и сохраняется до нажатия какой-либо из них. В этом режиме нажатие дает один символ или знак, если режим сохраняется, и другой, если одновременно нажата одна из клавиш смены режима.

Одновременное нажатие цифровых клавиш с клавишей смены режима SYMBOL SHIFT вызывает появление знака, в противном случае они дают появление символов, управляющих цветом.

Режим курсора (G) возникает после нажатия клавиши GRAPHICS (CAPS SHIFT и 9) и сохраняется до тех пор, пока не будет нажата клавиша CAPS SHIFT одна или совместно с 9.

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

Если некоторая клавиша удерживается в нажатом состоянии более, чем 2 или 3 секунды, это вызовет повторение ее действия, 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 вопроса(?).

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, вывод осуществляется в верхнюю часть экрана и сохраняется до ввода строки программы, либо нажатия клавиши ENTER при наличии пустой строки, либо нажатия клавиш перемещения вверх, вниз.

В нижнюю часть экрана выводятся также сообщения об ошибках, которые сохраняются там до нажатия любой из клавиш (это индицируется переходом в режим (K).

В определенных состояниях клавиши CAPS SHIFT и SPASE действуют как BREAK, останавливая компьютер с выдачей сообщений «D» или «L», это распознается:
а) в конце выполняющего оператора программы; б) после завершения операции на принтере или магнитофоне.

Экран телевизора

Экран содержит 24 строки по 32 символа в каждой и делится на две части. Верхняя часть экрана (22 строки) служит для отображения листинга и вывода результатов работы программы. Когда верхняя часть экрана заполнится полностью, он сворачивается на одну строку, компьютер останавливается с выдачей сообщения « SCROLL?». Ответ N, SPACE или STOP вызывает остановку программы с выдачей сообщения «D BREAK-CONT REPEATS», нажатие любой другой из клавиш разрешает свертку.

Нижняя часть экрана используется для ввода команд, строк программы и вводимых данных, а также вывода сообщений системы. (подробнее смотри приложение А)

Г л ава 2

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 невозможно 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, новы имеете возможность переместить программный курсорвыше или ниже,

используя соответствующие клавиши управления курсором. Установите его в строку 20. Когда вы нажмете клавишу EDIT, то в нижней части экрана появится копия текущей строки, в нашем случае копия строки 20. Нажмите и удержите клавишу перемещения курсора вправо до тех пор, пока курсор (L) 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 :

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

[K]
```

Запустив программу, нажав RUN и ENTER, 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 номеров строк -это служить именем оператора при ссылке к нему 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 увидите, что заголовок выводится в строке 20, и у вас возникает 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 вами значения переменной F. Вы можете ввести это значение в режиме (L). Наберите число и нажмите ENTER, компьютер выведет результат и снова перейдет в ожидание следующего числа. Этот переход

обеспечивается в строке 60 оператором GO TO 40. Если на запрос очередного числа ответить STOP, то компьютер остановится с выдачей сообщения: ' H STOP IN INPUT IN LINE 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 ,0 OK') номер последней выполнявшейся строки и продолжает выполнение именно этой строки. Посмотрите внимательно оператор PRINT в строке 50. Запятая в нем очень важна, запятые в операторе PRINT используются для указания того, что вывод, следующий после запятой, 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. 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 несколько программных строк:

```
100 REM TNIS 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 должно указываться строковыми кавычками. Если их опустить, то компьютер попытается эти переменную с таким же 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 кусором < 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 начнется 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 работу с выдачей сообщения ,L BREAK INTO PROGRAM'. Команда BREAK может быть использована и во время выполнения операции на магнитофоне или принтере. В этом случае выдается сообщение «D BREAK-CONT REPEATS», команда CONTINUE в этом случае (как и в большинстве других), вызовет повторение оператора , в котором произошла остановка. 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 кавычки).
```

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 ок, 0:1», а затем введите CONTINUE, то увидите, что программа завершится нормально. Как уже отмечалось, сообщение «L BREAKINTO PROGRAMM» особое,

так как выдача после него CONTINUE не вызывает повторение команды, вызывающей останов.

Все приведенные в этой главе утверждения: PRINT, LET, INPUT, RUN, LIST, GO, CONTINUE, NEW, 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

Глава 3
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, реализующйй это имеет форму: IF-некоторое истинное или ложное выражение, THEN-некоторое действие.

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

```
10 REM GUESS TNE NUMBER (УТАДАЙТЕ ЧИСЛО)
20 INPUT A:CLS
30 INPUT ,'GUESS THE NUMBER", B (УТАДАЙТЕ ЧИСЛО)
40 IF A = B  THEN PRINT ,'THIS IS CORRECT" :STOP
50 IF B < A THEN PRINT ,'THIS IS TOO SMALL, TRY AGAIN"
60 IF B > A THEN PRINT ,'THIS IS TOO BIG, TRY AGAIN"
70 GO TO 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, 30 :3' показывающее, что команда STOP была выдана в 3-ем операторе в 30-й строке.

Знаки условия набирают на клавиатуре следующим образом:

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

Глава 4
Циклы

Краткое содержание: FOR, NEXT, то, 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 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 THET 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, 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, таким образом, общая форма оператора FOR выглядит следующим образом:

FOR ‚упр. Перем. , = 'нач. Знач. , TO‚предел' STEP ‚шаг прираш. , здесь ‚начальное значение', ‚предел', ‚шаг приращения' есть выражения, принимающие числовое значение. Итак, 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 M
30 PRINT M;»:»;N;» «;
40 NEXT N
50 PRINT
60 NEXT M
```

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

Глава 5
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 1 строки в подпрограмме. Этот оператор подобен GO TO N с той разницей, что при использовании GO SUB компьютер запоминает следующий после GO SUB номер 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).

Глава 6
Оператор **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**

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

Это простой способ получения выражений из DATA списка: старт и выполнение от начала до тех пор, пока не будет достигнут конец. Однако вы можете использовать и программный переход для DATA списков, используя оператор RESTORE. 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. Вообще-то вы можете не указывать номер 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, Z, начиная с первого значения в DATA. Выполните программу без строки 30, и вы увидите, что из этого получится.

Глава7
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 арифметические операции: + , -, *, / (помните, что * умножение, а / деление), 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 краткий путь для указания компьютеру на то, что вычисления надо делать одно за другим. В нашем примере выражение:

```
SUM*15/100
```

указывает: возьмите значение переменной с именем „SUM", умножьте его на 15 и затем разделите на 100. Если вы не можете еще этого сделать, мы рекомендуем просмотреть 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 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
NOW WE ARE SIX    ЭТИ ДВА ИМЕНИ УКАЗЫВАЮТ
NOWWEARESIX       НА ОДНУ И ТУ ЖЕ ПЕРЕМЕННУЮ
```

Примеры недопустимых имен переменных:

```
2001 НАЧИНАЕТСЯ С ЦИФРЫ
3 WEARS —/—/—/—/—/—
M*A*S*H ЗНАК * НЕ БУКВА И НЕ ЦИФРА
POTHERINGTON-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 8 значащих цифр числа. Попробуйте выполнить еще:

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

и вы увидите, что компьютер может воспринять только цифры 4292967295.

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

Выполните:

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

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

Еще один наглядный пример:

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

Погрешность составляет около 1, а с прибавлением 1, фактически округляется до 2. Числа 5E9 + 1 и 5E9 + 2 для компьютера равны. Наибольшее целое, которое воспринимает он, 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, то 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 сможет вывести "FINNEGANS WAKE" . Здесь надо помнить специальное правило:если вы хотите вывести кавычки внутри строки, они должны удваиваться, например:

```
PRINT ,'HAVE YOU FINNISHED""FINNISHED WAKE" YET?"
```

В данном случае будет выведена только фраза, обрамленная двойными кавычками ,"FINNISHED WAKE"

Глава 8
Строки символов

Краткое содержание: сечения, использование TO

Примечание: эти операции отсутсвуют 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 ,'B 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 то:

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

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

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

вызывает сообщение '3 SUBSCRIPTWRONG', 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 то 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. Если теперь вы выполните:

```
LET A$() = "HELLO  THERE"
```

и

```
PRINT A$;". "
```

Вы увидите, 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"
```

Глава 9
Функции

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

Функции - это зашитые в бейсик 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 - VAL$. И аргументом и результатом ее является строка символов. Она работает как VAL, примененная дважды, раскрывает 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
```

Функция SQR вычисляет корень квадратный от числа, например:

```
SQR 4 = 2
SQR 0. 25 = 0. 5
SQR 2 = 1. 4142136
```

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 аргумент 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 (SYMBOL 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:

```
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 как, когда вычисляется FN P(2, 3), значение 'A' равно 10, как и определено в программе, а значения 'X', '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.

Примечание:

Функция может иметь до 26 числовых аргументов и в тоже время до 26 строковых.

Глава 10
Математические функции

Краткое содержание : **, P1, 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 обратным к ним:ASN, ACS, ATN. ** и EXP

Вы можете возвести число в некоторую степень путем многократногоумножения его само на себянеобходимое числораз, это, обычно, изображается записьючисла, обозначающегостепень, справа вверху от числа, обозначающего основание, то такую форму записи трудно реализовать в компьютере, поэтому там используется специальный символ (направленная вверх стрелка, в данном описании замененная двумя звездочками:'**').

Например, степень 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 в 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 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 точно 100% от того, что имели вначале, 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, т. е. 1. 1 5**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 математическу функцию. Это значение называется ‚E'. ZX SPECTRUM 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 первых его 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**LOGAX = 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

Допустим имеется некоторый круг, вы можете найти его периметр (длину окружности), умножив его диаметр на число, называемое PI.

Подобное ‚e', PI представляется бесконечной десятичной дробью, его начало:

3. 141592653589. . . Слово ‚пи' в ZX SPECTRUM обозначает это число.

Выполните, например:

```
PRINT PI
```

SIN, COS, TAN, ASN, ACN, 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 - абсцисса. необходимо помнить, что если ТОЧКА НАХОДИТСЯ СЛЕВА ОТ ОСИ Y, то COS ОТРИЦАТЕЛЬНЫЙ, А ЕСЛИ ТОЧКА ПОД ОСЬЮ X,- то отрицательный SIN.

Необходимо так же помнить, что:

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

Имеются и другие тригонометрические функции:

```
TAN - ТАНГЕНС
ASN - арксинус
ACN - АРККОСИНУС
ATN - АРКТАНГЕНС.
```

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

Глава 11
Случайные числа

Краткое содержание: RANDOMIZE, RND.

В этой главе описывается функция RND и ключевое слово RANDOMIZE. их не надо путать, хотя оба 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 увидите 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 DICE TNRCWING 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. 0022735596.

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.

В дополнении, 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.

Глава 12
Массивы

Краткое содержание: DIM

Допустим у вас имеется список 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, BLOGGGS2 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, 3, 11, 1, 0, 6, 3
```

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

Элементы массива называют индексируемыми переменными. Перед использованием массива надо 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 более чем с одной размерностью например в двухмерном 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 двухмерный 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 (в данном описании заменен на ,$').

Допустим вам необходимо объявить 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 объявлен 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 «12345678907» можно использовать также сечение 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 объявлен одномерный массив, то он определяет массив символов: DIM A$(10).

Глава 13
Логические операции

Краткое содержание: AND, OR, NOT.

Если мы взглянем на описанную в третьей главе 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), NOT(нет). Некоторое выражение «и», некоторое другое выражение истинны, если истинны 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 истинны, если истинно хотя бы одно из этих выражений. «НЕ» выражение истинно, если ложно само выражение и наоборот.

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

Условие « < > » обратно в логическом смысле условию « = », то есть

```
A < > B ТО ЖЕ, ЧТО И NOT A = B NOT A < > B ТО ЖЕ, 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 истинно.

2. В операторе «IF условие THEN. . . », само условие может быть числовым выражением, если его 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 имеет значение 1, если Y^ = 0 и X, если Y = 0 то 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 итерации будет выводиться большее из двух чисел A или B.

Пример использования OR:

```
LET TOTAL PRICE = PRICE LESS TAX * N (1. 15 OR V$ = "ZERO RATED").
```

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

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

Глава 14
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 а. Для преобразования из символьной формы во внутреннюю кодовую и наоборот служат две функции **CODE и CHR$.**

CODE применяется к строке симолов 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>.

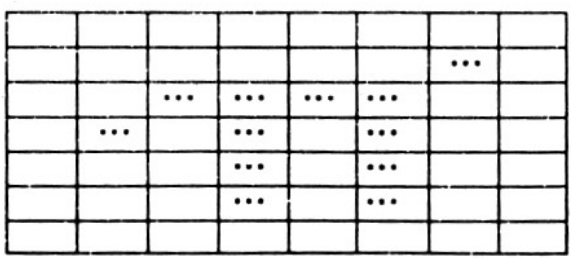
Теперь цифровые клавиши с 1 по 8 выдают графические символы, обозначенные на клавишах, а если при этом удерживать **SHIFT**, то они будут выдавать инверсные символы, т. е. черное становится белым, а белое черным.

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 а до U. 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 х 8, поэтому мы в начале начертим диаграмму, приведенную на рисунке. Мы оставим по одной клетке по периметру символа для отделения его от соседних знаков.

2. Закрепим данный символ за клавишей «P», так, чтобы при нажатии клавиши в графическом режиме выдавался символ «ПИ».



3. Запрограммируем это изображение. Каждый определяемый пользователем символ запоминается в памяти восьмью знаками, по одному на каждый ряд. Вы можете 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»

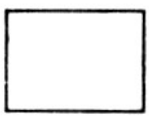
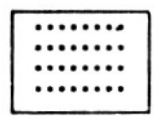
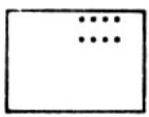
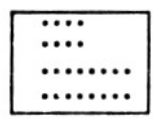
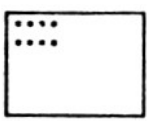
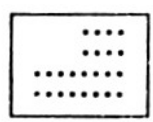
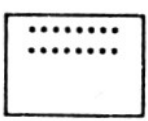
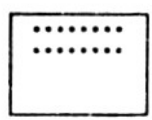
USR -функция преобразования строки симолов в адрес первого байта в строке. Строковый аргумент может содержать единственный символ, который будет обозначать символ, определяемый пользователем. Имеются и другие применения функции USR с числовым аргументом, но об этом позже.

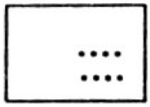
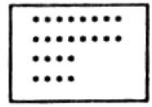
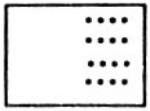
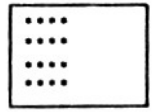
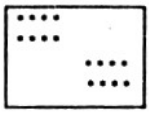
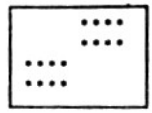
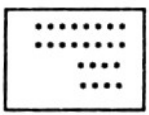
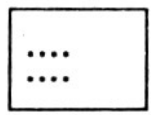
Поясним все сказанное программой:

```
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 области памяти, но об этом в Главе 24.

Графические Символы

Символ	Код набора	символ	код набора
	128 <G> 8		143 <G> SHB
	129 <G> 1		142 <G> SH!
	130 <G> 2		141 <G> SH©
	131 <G> 3		140 <G> SH3

Символ	код набора	символ	код набора
	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. Мы еще не поговорили о ненапечатанных первых 32 знаках с кодами от 0 до 32. Это управляющие символы. Они не отображаются, вместо них на телевизоре отображается знак «?», назначение этих символов описано в приложении А.

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

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

```
PRINT 1;CHR$ 6;2
```

даст результат, что и оператор:

```
PRINT 1, 2
```

но это не совсем корректное использование. Вернее будет сделать

```
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.

Телевизором также используются символы с кодами 16 и 23, но об этом поговорим в главах 15, 16. Все символы расположены в кодовой таблице в алфавитном порядке по возрастанию кодов, причем все прописные буквы располагаются после заглавных, так, что «A» следует после «Z».

```
CHR$ 3 + "ZOOLOGICAL GARDENS"
CHR$ 8 + "AARDVARK HUNTING"
"(PARENTHETICAL REMARK)"
"100"
"129.95 INC. VAT"
"AASVOGEL"
"AARDVARK"
"PRINT"
"ZOO"
"[INTERPOLATION]"
```

«AARDVARK»
«AASVOGEL»
«ZOO»
«ZOOLOGY»

Существует правило, по которому сортируются две строки. Сначала сравниваются первые символы, если они различаются, то строка, содержащая символ с меньшим кодом является «меньшей», а если равны , то выбирается для сравнения следующая пара символов. Так до тех пор, пока встречаются несовпадающие символы, либо пока одна из строк не кончится, она будет и меньшей, в противном случае строки считаются равными.

Отношения =, <, >, < =, > =, < > применяются к строкам символов так же, как и к числам: знак « < »находится впереди в кодовой таблице», а « > »-находится позади . Так, что выражения «AAMAN» < «AARDVARK» «AARDVARK» > «AAMAN» оба истинны.

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; (ROCK)
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$:REM 6 PIECES
 20 FOR F = 0 TO 7:REM READ PIECES INTO 8 BYTES
 30 READ A:POKE USR P$ + 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 001101100, B, B, 0
220 REM POWN
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
```

Глава 15

Дополнительные сведения об операторах «PRINT» и «INPUT»

Краткое содержание: CLS, PRINT- параметры: их отсутствие вообще.
Выражение (числового или строчного типа);
TAB- числовое выражение, AT- числовое выражение
PRINT разделители: ';' ',' ';'
INPUT- параметры,
LINE строчная переменная,
Свертка, SCREEN$.

Выражение, значение которого выводится в операторе **PRINT**, называется **PRINT**-выражением, оно разделяется запятыми или точкой с запятой, называемыми PRINT-разделителями. В операторе PRINT возможно отсутствие каких-либо параметров, в этом случае ставятся две запятые подряд.

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

печатается '*' в середине экрана.

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. Оператор SCREEN$: его действие противоположно действию оператора PRINT AT, он использует те же параметры, но их значения заключаются в скобки (номер строки, номер столбца).

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

Мы возвратимся на ',', выведенную предыдущим оператором. Как символы вывода могут использоваться: алфавитно-цифровые символы, специальные символы, пробелы. Линии, нарисованные с помощью операторов: PLOT, DRAW, CIRCLE и определяемые пользователем с помощью графических символов, возвращаются как пустая строка, однако имеется и другое применение, когда функция OVER используется для производства (построения) 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).

Заметим, что 'TAB 33' равнозначно 'TAB 1'

К примеру:

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

выведет на экран для первой страницы книги оглавление

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

Мы увидим, что значения параметра TAB получаются в результате вычитания по модулю из 32. 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 пробел, как конец оператора) при этом необходимо следить, чтобы позиция вывода не переместилась в часть экрана, предназначенную для вывода сообщения.

2. Мы не можем использовать две нижних строки экрана (22 и 23), т.к. они используются для получения оператором **INPUT** данных, а последняя используемая строка - 21.

3. Мы можем использовать параметр AT для установки позиции вывода в то место, где уже имеется выведенная информация, при этом новый символ уничтожает старый. еще одним оператором, используемым совместно с PRINT является CLS. он производит очиску экрана подобно операторам CLEAR и RUN, при заполнении всего экрана происходит его свертка, в этом можно убедиться проделав:

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

и далее PRINT 99 некоторое количество раз. При выводе текста на экран происходит останов вывода, для просмотра текста, чтобы убедиться в этом выполним:

```
CLS; FOR N = 1 TO 22: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 и ‚A') или SPASE(BREAK). Компьютер остановит программу и выдаст сообщение: D BREAK-CONT REPEATS.

INPUT операторы используются 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' являются параметрами INPUT. Однако существуют и значительные различия.

1. Параметры INPUT-переменные, которые вы вводите сами, INPUT может начинаться с буквы, которая является вводимой переменной.

2. Вы можете вводить 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 ат в INPUT-операторе:

```
10 INPUT «THIS IS LINE 1», A$;ат 0, 0;«THIS LINE IS 0", A$;
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 будут сдвигаться вверх через намеченную линию. Выполним:

```
10 FOR N = 0 TO 19:PRINT AT N, 0;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 примером оператора INPUT является LINE, он предназ-начен для ввода строчных переменных. Рассмотрим пример:.____(без строковых кавычек INPUT LINE A$!

Если ввести какую-либо строчную переменную, 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 задавать более чем два значения, а для TAB и AT это не возможно. Эти управляющие символы обрабатываются как числа. аналогом ат является управляющий символ CHR$22, первое значение определяет строку, второе столбец.

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

то же, что и

```
PRINT AT 1, C
```

Как значения параметров рассматриваются только CHR$1 и CHR с (CHR$22 не учитывается). Аналогом тав 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 остановки компьютера, запрашивающего свертку, выполнив:

```
POKE 23692, 255
```

Компьютер станет сворачивать экран без запроса 255 раз, прежде чем запросит свертку, как в примере:

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

и следите сколько сверток сделает компьютер. Стартуем следующую программу, проверяющую таблицу 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.":GOTO30
```

Для исключения подсказки поменяйте ‚C$' в строке 30 на ‚LINE C$'

Глава 16

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" ";REM 4 COLOVRED 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 на 32), каждая из которых представляет матрицу 8 на 8 пикселей. Вспомним из 14 главы :

0 - белая точка
1 - черная точка

Позиция символа (знакоместо) также рассматривается с этих позиций :INK - цвет тона, PAPER - цвет фона, т.к. знакоместо состоит из INK и PAPER (рисунок на бумаге). Можно также говорить о INK и PAPER. Все это имеет следующую кодировку: обычной и повышенной яркости, а также мерцающих и немерцающих. Кодировка:

1. Для знакоместа (8 на 8 пикселей) форму символа определяют чистые и закрашенные точки (0S и 1S), 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 пиксела 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, FLUSH.

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.
FLUSH - 0 или 1 / 1 - вкл, 0 - выкл.
```

Отметим, что использование других 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 установит цвета фона (т.к. такого цвета нет), а поможет выяснить значение предыдущего PAPER. Операторы INK 8, BRIGT 8, FLUSH 8 выдадут значение этих атрибутов.

«9» может использоваться только для INK и PAPER как средство «контраст». Цвета INK и PAPER, которые вы используете, должны быть контрастны друг другу, так как к белому цвету подходят темные тона: черный, синий, красный, фиолетовый; к черному цвету-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 то 1000
PRINT N;NEXT N
```

Цвет фона будет контрастен цвету тона в любой выводимой позиций. Цветной телевизор построен на следующем принципе:человеческий глаз способен воспринимать только три первичных цвета- синий, красный и зеленый. Другие цвета образуются из их сочетаний, к примеру, фиолетовый цвет образуется как комбинация синего с красным(код фиолетового тона «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 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 DATA LINE = 200
60 CO SUB 1000
70 LET DATA LINE = 210
80 CO 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 DATA LINE
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 функция, 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), т.е. в обычном виде мы пишем черным по белому, а 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 «O»;CHR$8;»«««;
40 NEXT N
```

Отметим, что управляющий символ CHR$8 определяет одно место.

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, что при этом их действие распространяется только до конца PRINT. В примере

```
PRINT   PAPER 6;»X»;:PRINT «Y»
```

только ‚X' будет выведена на желтый фон.

[K], [L], [C]			[G]		[E]			реж
НЕТ	SYMBOL	CAPS	НЕТ	ЛЮБОЙ	НЕТ	CAPS	SYM-BOL	рег
1	1	EDIT	▪▪▪▪ ▪▪▪▪	▪▪▪▪ ▪▪▪▪ ▪▪▪▪▪▪▪▪ ▪▪▪▪▪▪▪▪	ФОН ГОЛУБОЙ	ТОН ГОЛУБОЙ	DEF FN	
2		CAPS LOCK	▪▪▪▪ ▪▪▪▪	▪▪▪▪ ▪▪▪▪ ▪▪▪▪▪▪▪▪ ▪▪▪▪▪▪▪▪	ФОН КРАСНЫЙ	ТОН КРАСНЫЙ	FN	
3		TRUE VIDEO	▪▪▪▪▪▪▪▪ ▪▪▪▪▪▪▪▪	▪▪▪▪▪▪▪▪ ▪▪▪▪▪▪▪▪	ФОН ФИОЛЕ-ТОВЫЙ	ТОН ФИОЛЕ-ТОВЫЙ	LINE	
4	$	INVERSE VIDEO	▪▪▪▪ ▪▪▪▪	▪▪▪▪▪▪▪▪ ▪▪▪▪▪▪▪▪ ▪▪▪▪ ▪▪▪▪	ФОН ЗЕЛЕНЫЙ	ТОН ЗЕЛЕНЫЙ	OPEN	
5	%	КУРСОР ВЛЕВО	▪▪▪▪ ▪▪▪▪ ▪▪▪▪ ▪▪▪▪	▪▪▪▪ ▪▪▪▪ ▪▪▪▪ ▪▪▪▪	ФОН СИНИЙ	ТОН СИНИЙ	CLOSE	
6	&	КУРСОР ВНИЗ	▪▪▪▪ ▪▪▪▪ ▪▪▪▪ ▪▪▪▪	▪▪▪▪ ▪▪▪▪ ▪▪▪▪ ▪▪▪▪	ФОН ЖЕЛТЫЙ	ТОН ЖЕЛТЫЙ	MOVE	

7	'	КУРСОР ВВЕРХ	▪▪▪▪▪▪▪▪ ▪▪▪▪▪▪▪▪ ▪▪▪▪ ▪▪▪▪	▪▪▪▪ ▪▪▪▪	ФОН БЕЛЫЙ	ТОН БЕЛЫЙ	ERASE
8	(	КУРСОР ВПРАВО		▪▪▪▪▪▪▪▪ ▪▪▪▪▪▪▪▪ ▪▪▪▪▪▪▪▪ ▪▪▪▪▪▪▪▪	НОРМАЛ ЯРКОСТЬ	БЕЗ МЕРЦАНИЯ	POINT
9	)	ГРАФИ-ЧЕСКИЙ РЕЖИМ	ГРАФИЧ. ВЫХОД	ГРАФИЧ ВЫХОД	ПОВЫШ ЯРКОСТЬ	С МЕРЦА-НИЕМ	CAT
10	-	DELETE	DELETE	DELETE	ФОН ЧЕРНЫЙ	ЗАКРАШ ЧЕРНЫЙ	FORMAT

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 мерцания 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 FLUSH 1;INK 1;»WANT IS YOUR NUMBER ?»;N
```

Возможно изменение цветов и с помощью управляющих символов, подобных управляющим символам для AT и TAB в главе 15.

```
CHR$16 СООТВ.  INK
CHR$17   -     FLASH
CHR$19   -     BRIGHT
CHR$20   -     INVERSE
CHR$21   -     OVER
```

так

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

то же, что и

```
PRINT INк 9;
```

Можно пользоваться управляющими символами либо операторами. Их можно ставить как после номера строки, так и в конце строки. Для удобства можно пользоваться расширением клавиатуры с цифрами. Цифры от 0 до 7 устанавливают цвет INK, если CAPS SHIFT нажата, и цвет PAPER, если CAPS SHIFT не нажата. При нажатии цифры в E-режиме будут выведены CHR$17 и CHR код цвета. Если в это время была нажата CAPS SHIFT, то будут выведены CHR$17 и CHR код цвета.

Если вы хотите уничтожить вводимое, нажмите DELETE два раза: после первого раза на экране высветится ,?' или что-нибудь еще- не пугайтесь и нажмите DELETE еще раз. Управление курсором тоже не работает до тех пор, пока курсор не выйдет к предыдущему управляющему символу. Так же в E-режиме :

8 аналогично CHR$19 и CHR$0 - нормальная яркость
9 аналогично CHR$19 и CHR$1 - повышенная яркость
CAPS SHIFT и 8 дает CHR$18 и CHR$0 - немерцающее поле
CAPS SHIFT и 9 дает CHR$18 и CHR$1 - мерцающее поле

В L - режиме :

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

CAPS SHIFT и 4 дает CHR$20 и CHR$1 - инверсный вывод
ATTR - это функция, 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;FLUSH 1;PAPER 6;INk 1;» «;ATTR(0, 0)
```

Упражнение :

```
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.

Введем вручную или с помощью оператора LOAD программу для ввода шахматных фигур из гл.14, и введем программу разбивки экрана под шахматную доску:

```
5 REM DRAW BLANKBOARD
10 LET BB = 1:LET BW = 2:REM RED AND BLUE FOR BOARD
15 PAPER BW:INK BB:CLS
20 PLOT 79, 128:REM BORDER
30 DRAW 65, 0:DRAW 0, -65
40 DRAW -65, 0:DRAW 0, 65
50 PAPER BB
60 REM BOARD
70 FOR N = 0 TO 3:FOR M = 0 TO 3
80 PRINT AT 6 + 2*N, 11 + 2*M;» «
90 PRINT AT 7 + 2*N, 10 + 2*M;» «
100 NEXT M:NEXT N
```

```
110 PAPER 8
120 LET PW = 6:LET PB = 5:REM COLOURS OF WHITE AND BLACK PIECES
200 DIM B$(8, 8):REM POSITIONS OF PIECES
205 REM SET AP INITIAL POSITION
210 LET B$(1) = »RNBQKBNR»:REM LITTLE LINE 240
220 LET B$(2) = »PPPPPPPP»:REM LITTLE LINE 230
230 LET B$(7) = »PPPPPPPP»:REM BIG LINE 220
240 LET B$(8) = »RNBQKBNR»:REM BIG LINE 210
300 REM DISPLAY BOARD
310 FOR N = 1 TO 8:FOR M = 1 TO 8
320 LET BC = CODE B$(N, M): INK PW
325 IF BC = CODE « « THEN GO TO 350:REM SPACE
330 IF BC > CODE «Z» THEN INK PB:LET BC = BC-32:REM LOWER
 CASE FOR BLACK
340 LET BC = BC + 79:REM CONVERT TO GRAPHICS
350 PRINT AT 5 + N, 9 + M;CHR$BC
360 NEXT M:NEXT N
```

Глава 17
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 22 x 32 = 704 символьные позиции. Как вы уже поняли из 16 главы, каждая символьная позиция представляется квадратом 8 x 8 точек, называемых пикселями. Пиксель задается его координатами. Первое число задает координату X, то есть удаление (в пикселях) до левой границы экрана. Координаы записываются в скобках, так (0, 0), (255, 0), 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 синус(X) для X в интервале от нуля до 2*PI:

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

Или программа :

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

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

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

Пользуйтесь диграммой из главы 15.

Помощь при построении изображений вам могут оказать операторы: DROW, 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, CL и NEW устанавливают ее в левый нижний угол экрана.). Таким образом оператор DRAW задает длину и направ-

ление вычерчивания линии, но не начальную точку. Поэкспериментируйте 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 BORDER0: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, INк, FLUSH, 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(см. 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 «А»-положительно, то дуга вычерчивается влево, а если отрицательно- то вправо. При «А» равном 2*PI вычерчивается полная окружность.

Например:

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

вычертит полуокружность с начальной точкой (100, 100) и конечной точкой (150, 150)

Вычерчивание начинается в направлении юго-восток и заканчивается 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, стока программы:

```
CLS:PRINT POINT(0, 0):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 нулю(отключено), но вы можете задать и единицу, при этом: PLOT INVERSE 1 - устанавливает для заданного пикселя цвет фона. PLOT OVER 1 - изменяет цвет пикселя на противоположный, если был закрашивающий цвет, то становится цвет фона и наоборот. PLOT INVERSE 1;OVER 1; - сохраняет цвет пикселя без изменений, 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
```

что не изменит картинку, т.к. при черчении как вперед, так и назад используются одни и те же пиксели. Имеется способ получения необычных цветов в одном квадрате, с использованием определяемых оператором 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 строится 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:GOTO 100
70 IF NOT T THEN PLOT F, Y + 88:LET T = 1:GOTO 100
80 DRAW 1, Y-OLDY
100 LET OLDY = INT(Y + 5)
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.

Глава 18
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 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 (надо помнить, что CAPS SHIFT будет вызывать прерывание). Пример программы моделирования секундной стрелки 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 REM NOW WE START THE CLOCK
60 FOR T = 0 TO 200000:REM T IS THE TIME IN SECONDS
70 LET A = T/30*PI:REM A IS THE ENGLE 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 час. Это 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 2-х процентную 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 по мощью функции peek, подробно это рассмотрено в главе 25. Здесь же 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 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 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 ноль. Первой начинает отсчитывать ячейка PEEK 23672, каждые 1/50 сек. Ее содержимое увеличивается на 1. Когда в ячейке накопится величина, равная 255, то она сбрасывается в ноль а значение ячейки peek 23673 увеличивается на 1. Через каждые 256/50 сек. содержимое этой ячейки переходит из состояния 255 в ноль, 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 секунды ячейки будут содержать соответственно следующие значения: 1, 0, 0. Пока производятся вычисления 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 = 1800000 50-х долей секунды и значит

1800000 = 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 маркера <L>, или пустая строка.

Выполните программу, которая использует эту функцию:

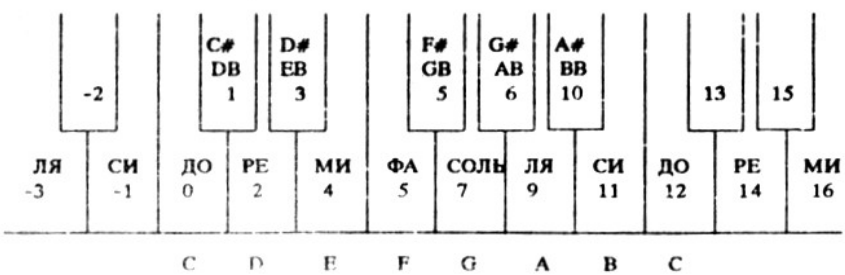
```
10 IF INKEY$ <> "" THEN GO TO 10
20 IF INKEY$ = "" THEN GO TO 20
30 PRINT INKEY$;
40 GO TO 10
```

Помните, что функция INKEY$ не будет подобно INPUT ждать вас. Если вы не выполните ввод, то считайте, что ваш шанс упущен.

Глава 19
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:



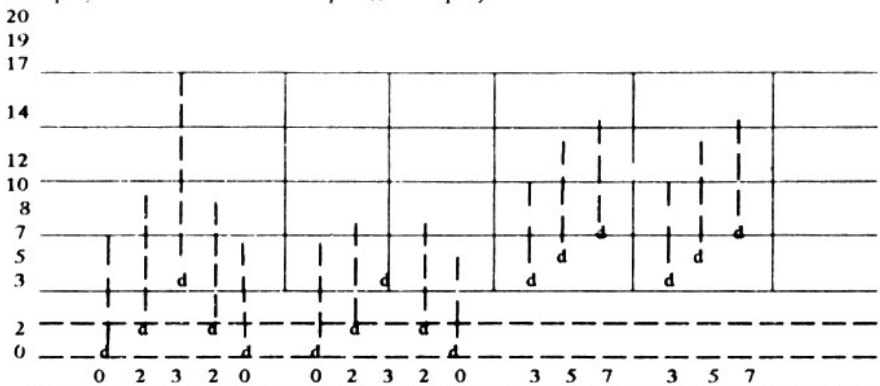
Для получения более высоких или более низких нот, вы должны прибавить или отнять 12 для каждой октавы вверх или вниз.

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 сообщением об ошибке ‚B'. Вы можете также паралельно 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 .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'.

Глава 20
Внешняя память на магнитной ленте

Краткое содержание : LOAD, SAVE, VERIFY, MERGE

Основные команды работы с магнитофоном SAVE, LOAD 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. Программу « DICE » (« игральная кость ») из главы 11 запишем на ленту под именем «DICE».

А теперь введем и выполним следующую программу :

```
1 PRINT 1
2 PRINT 2
10 PRINT 10
20 LET X = 20
```

Затем осуществим ее проверку, заменив команду VERIFY «DICE» на команду MERGE «DICE». Вы увидите, что строки 1 и 2 сохраняются, а строки 10 и 20 заменяются на строки с этими номерами из программы «DICE», переменная X тоже сохраняется (проверьте PRINT X)

Теперь вы знаете четыре оператора для работы с кассетным магнитофоном :

SAVE - записывает программу и переменные на магнитофон;

VERIFY - проверяет программу и переменные в памяти компьютера по их копии на ленте;

LOAD - очищает память компьютера от всех программ и мзагружает в нее новые данные, считанные с магнитофона;

MERGE - подобно 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. Имеются две возможности для снятия справки с ленты :

Вариант 1. В операторах VERIFY, LOAD и MERGE вместо имени можно указать пустую строку. Тогда будет взят первый встретившийся файл.

Вариант 2. С использованием оператора SAVE :

SAVE STRING LINE NUMBER

Программа запишется на ленту так, что когда она будет считана по команде LOAD (но не MERGE), то она автоматически установится на строку (NUMBER) и сама инициирует свое выполнение.

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

Например, записать на ленту массив вы можете, используя команды SAVE и DATA таким образом :

SAVE STRING DATA ARRAY NAME ( )

Здесь «STRING» - имя, присваеваемое файлу данных, которое может состоять из букв или букв и символа «$» (перечеркнутая буква «S») для строковых данных это требование не важно.

Такие данные загружаются по команде :

LOAD STRING DATA ARRAY NAME ( )

Здесь нельзя использовать оператор MERGE.

Если загружается строковый массив, то после обнаружения его на ленте компьютер выдаст:

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.п.. 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 надо записать на ленту (6912 -объем 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

START (начало) - указывает адрес, с которого должны загружаться данные.

Он может быть отличен от адреса, указанного в SAVE. Вы можете опускать этот параметр в команде LOAD.

LENGTH (длина) - определяет, сколько данных в байтах надо загрузить с ленты. Если длина больше, чем записано на ленте, то выдается сообщение : «R TAPE LOADING ERROR» (ошибка загрузки с ленты). Этот параметр можно опустить -тогда компьютер считает все данные, записанные на ленте.

Выражение CODE 16384, 6912 можно заменить на SCREEN$:

SAVE «PICTURE» SCREEN$

Тогда при вводе необходимо:

LOAD «PICTURE» SCREEN$

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

«В заключении. Везде, где указывается имя файла на ленте, используются только 10 символов. Существует 4 типа информации, которые могут быть записаны на ленту:

-программы и переменные(совместно),

-числовые массивы;
-строковые массивы;
-непосредственно 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 :

«PROGRAMM:», «NAMBER ARRAY:», «CHARACTER ARRAY:», «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 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 NUMWER
и начинает выполнять ее.

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$()
записывают числовой или строковый массив (требование не относится к «Name»).

VERIFY

Команда VERIFY проверяет (сравнивает) информацию в памяти и на ленте. В случае несравнения выдается 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 «LENGTN», то выдается сообщение об ошибке «R».
VERIFY NAME CODE START
Здесь осуществится 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. VARIFY Name SCREEN$ или эквивалентно VARIFY NAME CODE 16384, 6912 однако это будет проверка уже проверенного файла.

3. Массивы. VERIFY Name DATA LETTER()

VARIFY NAME DATA LETTER$()

LOAD

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 указанно в «LENGTH», то выдается сообщение об ошибке R.

LOAD NAME CODE START

производит загрузку данных из «Name» в память, начиная с адреса «START».

LOAD NAME CODE

загружает дзин

3. Массивы.

LOAD NAME DATA LETTER()

LOAD Name DATA LETTER$()

Уничтожает в памяти массив с именем «LETTER» или «LETRER$», 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.

MERGE

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

или

LOAD «CHESS» CODE USR «A»

Глава 21
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. Пример:

```
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 SPASE).

Если вы задали операторы управления принтером без подключения 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 PRNT 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.

Вообще, при печати перевод строки осуществляется в следующих случаях:

1. При заполнении буфера строки;
2. После LPRINT, если это не конец оператора и в нем встретилась запятая или точка с запятой;
3. Если запятая, апостроф или «TAB» требуют новой строки;
4. При окончании программы, если остались невыведенные данные.

Глава 22
Другое переферийное оборудование

Имеется ряд других устройств, которые могут быть подключены к компьютеру ZX SPECTRUM. ZX MICRODRIVE-высокоскоростное устройство памяти, может быть использовано вместо кассетного магнитофона. Однако оно не может управляться командами SAVE, VARYFY, LOAD, MERGE, а лишь командами PRINT, LIST, INPUT и INKEY$.

При помощи этого устройства можно организовать сеть из нескольких компьютеров ZX SPECTRUM. Стандартным интерфейсом для ZX SPECTRUM является RS-232, посредством которого подключаются: клавиатура, принтер и любые другие устройства, отвечающие стандартам этого интерфейса. При работе с такими устройствами могут использоваться имеющиеся на клавиатуре дополнительные ключевые слова:OPEN$, CLOSE$, MOVE, ERASE, CAT, FORMAT.

Глава 23
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, представляются совокупностью адресов от 0 до 65536, каждый из которых адресует один байт.

Таким же образом можно адресовать и еще 65536 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.

OUT подобен оператору POKE:
OUT ADDRESS, VALUE
и записывает указанные данные в заданный порт вывода. ZX SPECTRUM оперирует 16-тиразрядными адресами, разряды которых мы будем обозначать буквой A:

A15, A14, A13, A12.....A1, A0.

Биты адреса A0, A1, A2, A3, A4 очень важны. Как правило они равны 1. Но если хотя бы один из них в нуле, это предписывает компьютеру некоторые действия. Не более чем один из этих 5 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 мы будем обозначать буквой D:

D7, D6, D5.....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 полуряду:

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 7,
IN 49150 считывает ряд от ENTER до H,
IN 32766 считывает ряд от SPASE 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 5 клавиш в данном полуряду. D0-для крайней клавиши, а D4-для ближней к центру. Состояние одного из этих битов 0 указывает, что соответствующая ему клавиша нажата. D6 принимает свое значение при чтении с разъема 'EAR'.

Входной порт 254 обеспечивает громкоговоритель(D4) и разъем 'MIC' (D3), а также установку цвета (D2, D1, D0).

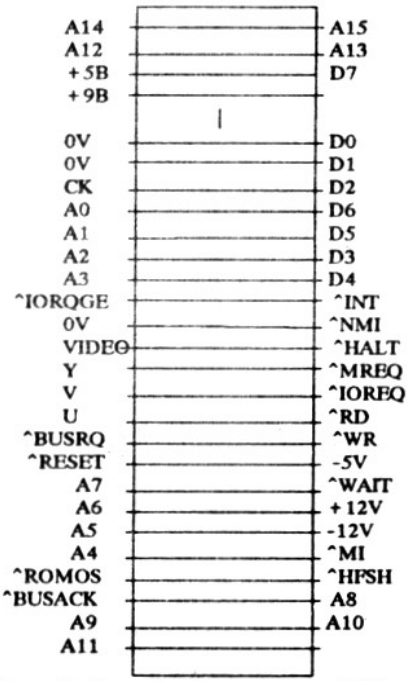
Порт 251 обеспечивает связь с принтером, как чтение, так и запись. Чтение-для проверки готовности принтера к работе. Порты 254, 247, 239 используются для связи с дополнительными устройствами, описанными в главе 22.

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
```

И понажимайте по одной клавише в каждом полуряду. После нажатия очередной клавиши введите BREAK, а затем NEXT N.

Ниже, на рисунке показано распределение контактов разъема:



Книги: 1. «Искусство схемотехники» П. Хоровиц, У. Хилл Мир, 1986, том 2, стр. 579-580
«Технические средства микропроцессорных систем»
2.Дж. Кофрон, Мир, 1983, стр. 334-340

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.

НЕ М - (машинный цикл). Выход. Активный сигнал-низкий. Указывает, что в текущем цикле осуществляется выборка КОП.

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

НЕ IORG- (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 генерируется после выдачи подтверждения прерывания, тем самым указывая, что вектор прерывания может быть помещен на шину данных.

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

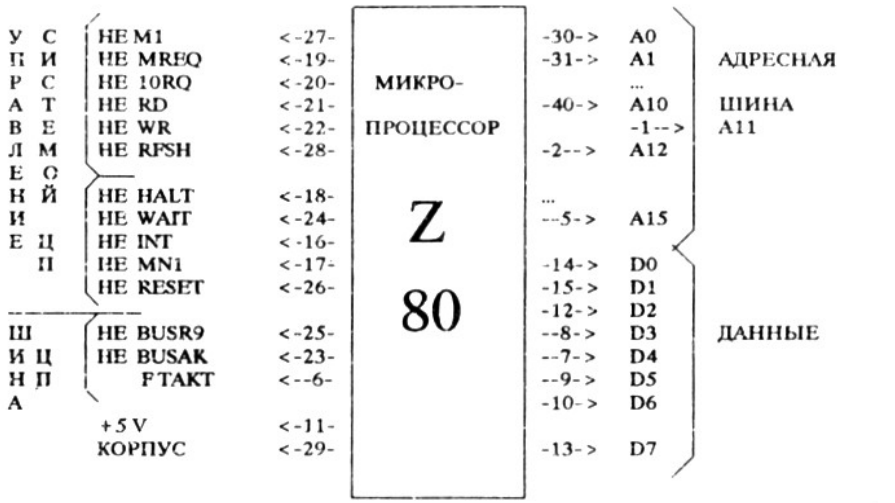


Рис. 22-1 Назначение выводов микропроцессора Z80, Z80a

Не 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 УВВ.

НЕ FRSH- (восстановление). Выход, активный уровень-низкий. Сигнал указывает, что младшие 7 разрядов шины адреса содержат адрес восстановления для ОЗУ и текущий сигнал не MREQ должен использоваться для восстановления динамической памяти.

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

НЕ 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.

НЕ 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.

НЕ 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).

НЕ RESET–(сброс). Вход. Активный уровень- низкий. При поступлении сигнала выполняются следующие действия:

а) сброс триггера разрешения прерывания IFF.

б) очистка счетчика команд и регистров I и R.

в) шины адресная и данных в состоянии высокого сопротивления.

г) для всех управляющих выходных сигналов устанавливается неактивный уровень.

НЕ BUSRQ–(запрос шин). 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) и переводит в состояние высокого сопротивления все шины и тристабильные выходы сигналов управления, после чего этими шинами могут управлять другие устройства.

НЕ BUSAK- (подтверждение перевода шин в состоянии высокого сопротивления.) выход. Активный уровень-низкий. Сигнал подается на запрашивающее внешнее устройство.

Глава 24
Память

Краткое содержание:**CLEAR**

Вся память компьютера разбита на байты, каждый из которых представим числом от 0 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 2 байта. На диаграмме показано распределение памяти компьютера ZX SPECTRUM:

0	3FFFH	4000H	7FFFH	8000H	FFFFH
ROM (ПЗУ)		RAM (ОЗУ) (системная область)		доступно для программ	
0	16383	16384	32767	32768	65535

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 их адресами:

```
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 CONTENS,

где «ADDRESS» и «NEW CONTENS»- числовые выражения. Например:

POKE 31000, 57

«NEW CONTENS» может принимать значения от -255 до + 255

Вся память подразделяется на области, предназначенные для хранения информации различного назначения, что показано на диаграмме:

область экрана TV	атрибуты	буфер принтера	системные переменные	План дополнительной памяти	
16384	22528	23296	23562	23734	CHANS

канальная информация	80H	программа на бейсике	переменные! программы	80H	редактируемые строки программы	NL
CHANS		PROG	VARS		E LINE	WORKSP

считанные данные	NL	рабоч. ласть	стек калькулятора	резерв	аппаратный стек	стек переходов к подпрог.
WORKSP			STKBOT	STKEND	SP	

7	3EH	определяемые пользователем символы
	RAM TOP	UDG P RAMT

Область телевизионного экрана содержит образ текущего кадра. Она доступна для операторов PEEK и POKE. Каждая позиция экрана представима матрицей 8*8 точек (один байт на каждый ряд из 8 точек). Однако эти 8 байт хранятся в памяти не вместе. Полный экран представляет собой 24 строки по 32 символа. Каждая строка экрана прописывается 8-ю 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 ДОП 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:

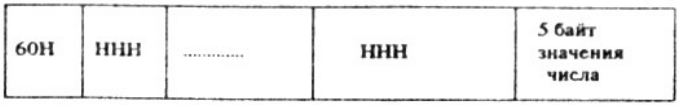
Старший байт		Младший байт			
2 байта		2 байта		текст	0DH
N стр.		Длина текста + ENTER			ENTER

Числовые константы в программе представлены в 2-ой форме, используя CHR$14 и следующие за ним 5 байт самого числа. Переменные имеют различные форматы представления в памяти.

Ниже представлен формат записи числа, имя которого состоит из одной буквы:

60H	1 байт порядка	бит знака	4 байта мантиссы
буква		значение числа	

Формат размещения числа, если имя имеет более, чем одну букву:



Формат размещения числового массива

Порядок элементов следующий:

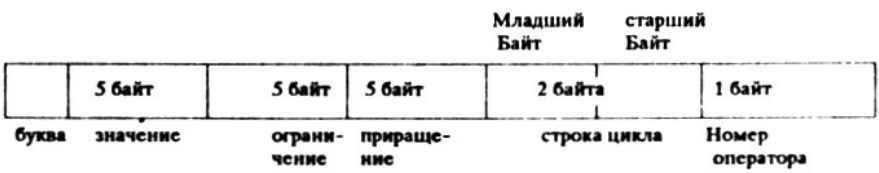
1. Элементы, для которых первая размерность равна 1.
2. Элементы, для которых первая размерность равна 2.
3. Элементы, для которых первая размерность равна 3 и т.д.

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:



Формат размещения строки символов:

Формат размещения строкового массива:

	2 байта	1байт	2 байта
буква	общее число элементов + 1 на каждую размерность	номера размерностей	1-я размерность

	2 байта	по одному байту на каждый элемент
	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 гл.5

Байт, адресуемый по RAMTOP, содержит верхний адрес, используемый бейсиком. Даже оператор NEW, который очищает ОЗУ, не изменяет содержимое области определяемых пользователем символов.

Вы можете изменить адрес RAMTOP в операторе CLEAR:

CLEAR NEW RAMTOP, по которому:

а) очищаются все области переменных,

б) очищается область экрана (подобно CLS),

в) переустанавливается позиция PLOT в левый нижний угол экрана,

г) выполняется функция RESTORE,

д) очищается стек переходов и устанавливается новое значение RAMTOP

Функция RUN также выполняет действия CLEAR, хотя и не изменяет значение RAMTOP. Используя функцию CLEAR, вы можете смещать Ramtop, увеличивая область для бейсика, уменьшая тем самым область определяемых пользователем символов. Можно несколько увеличить доступную часть Ram (ОЗУ), используя функцию NEW. Например, выполнение NEW, затем CLEAR 23800 помогает компьютеру при переполнении ОЗУ. Все указанные действия могут приводить к 2 сообщениям об ошибке и выдаче звукового сигнала:

«4 MEMORY FULL» (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 от 0.5 до 1 (исключая 1);
E-экспонента, положительное или отрицательное число.

Допустим, вы записали м в двоичной системе счисления, М-дробное и имеет двоичную точку (подобно десятичной точке). Тогда будет:

1/2 ——> .1
1/4 ——> .01
3/4 ——> .11 и т.д.

Наше число М 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 битов в 3-тий байт и т.д. до 5-го байта;

б) заменяем 1-й бит второго байта, в котором записана 1, на знаковый бит (0 для + и 1 для -);

в) записываем в первый байт экспоненту + 128.

Например, мы хотим записать число 1/10.

$$1/10 = 4/5 * 2^{-3}$$

Мантисса будет .11001100110011001100110011001100в (поскольку 33-й бит = 1, мы должны округлить 32-ой бит, записав 1 вместо 0), а экспонента равна -3.

Теперь применив наши 3 правила, запишем 5 байт:

0111 1101	0	100 1100	1100 1100	1100 1100	1100 1101

-3 + 128 Знак числа Мантисса 4/5, исключая левый знаковый бит

Имеется альтернативный способ записи целого числа в интервале -65535...65535:

а) первый байт равен 0;

б) второй байт равен 0 для положительного числа и FFн для отрицательного;

в) 3-й и 4-й бэйты содержат младшие и старшие значащие биты числа (или число + 131037, если оно отрицательное);

д) 5-й байт равен 0.

Глава 25
Системные переменные

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, т.к. это может нарушить работу системы.

N- изменение переменной не приводит к длительному эффекту.

зн.	адрес	имя	Содержание
N8	23552	KSTATE	используется при чтении с клавиатуры
N1	23560	LAST K	запоминается вновь нажатая клавиша
1	23561	REPDEL	время в 50-х долях секунды, в течение которого клавиша должна быть зафикси- рована в нажатом состоянии. Начальное значение 35, но может быть изменено
1	23562	REPPER	задержка, в 50-х долях секунды, между последовательными опросами клавиш. Начальное значение -5.
N2	23563	DEFADD	адрес аргументов функций пользователя, если они используются, иначе 0.
N1	23565	K DATA	второй байт управления цветом с клавиатуры.
N2	23566	TVDATA	байты цвета, AT, TAB управления телевизором
x38	23568	STRMS	адреса подключенных каналов
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 флажки бейсика.
X1	23612	TV FLAG	флажок телевизора.
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; спецификация [K], [L], [C], [E] или [G] курсора.
2	23618	NEW PPC	номер строки, на которую должен быть сделан переход.
2	23621	PPC	номер строки, оператор в которой выполняется.
1	23623	SUB PPS	порядковый номер выполняющегося оператора в строке.
1	23624	DORDCR	цвет рамки экрана, содержит атрибуты.
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 выведенной команды.
2	23643	K CUR	адрес курсора.
X2	23645	CH ADD	адрес следующего интерпретируемого символа:символ аргумента в PEEK, NEWLINE или в 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 начала резервной области памяти.
N1	23655	BREG	B-регистр калькулятора.
N2	23656	MEM	адрес области, используемой как память калькулятора (обычно MEMBOT, но не всегда)
1	23658	FLAG52	старшие флажки.
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 флажки.
N2	23666	STR LEN	размер расстояний между строками.
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 мS (см. Главу 18).
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 в нижней части экрана.
X1 X1	23688 23689	S POSN	33-позиционное число для PRINT позиции 23-строковое число для PRINT позиции.
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 на 1 (скажем 255), то экран будет сворачиваться без запроса.
1	23693	ATTR P	сплошные цвета.
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	временный указатель цветов
N1	23696	MASK T	временный MASK P
1	23697	P FLAG	старшие флажки.
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 бейсик-системы.
2	23732	P-RAMT	адрес последнего байта физического ОЗУ

Число- число байтов переменной (для 2-х байтовых переменных младший байт - 1й). Например, необходимо изменить значение на V в 2-х байтовой 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)
```

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.

Глава 26
Использование машинных кодов

Краткое содержание :USR с числовым аргументом.

Эта глава описывает применение машинных команд микропроцессора Z80 [U880 (ГДР), 1810ВМ80 (СССР)].

Программы в машинных кодах пишут обычно на ассемблере с последующей трансляцией. (перечень мнемокодов команд микропроцессора Z80 приведен в приложении А). Транслятор с ассемблера встроен в компьютер ZX SPECTRUM.

Приведем пример программы:

```
ID BC, 99
RET
```

которая загружает в «BC» регистр число 99. Эта программа будет транслироваться в 4-х байтный машинный код:

Байты 1, 99, 0 для ID BC, 99 и
201 для RET

Следующим шагом является загрузка программы в компьютер. Для этого используется дополнительная память, получаемая между бейсик- областью и областью определяемых пользователем символов. Допустим, вы имели следующее распределение последней части ОЗУ:

	определяемые пользователем Символы	

RAMTOP = 32599 UDG = 32600 PRAMT = 32767

Если вы теперь выполните:
CLEAR 32499

то получите дополнительно 100 байтов памяти, начиная с адреса 32500.

100 байтов	определяемы пользователем Символы	

32500 UDG = 32600 PRAMT = 32767
RAMTOP = 32499

Для загрузки программы в машинных кодах вы можете выполнить следующую бейсик-программу:

```
10 LET A = 32500
20 READ N: POKE A, N
30 LET A = A + 1: GOTO 20
40 DATA 1, 99, 0, 201
```

(программа может завершиться с сообщением 'E OUT OF DATA', если переполняются отведенные вами 4 байта).

Для выполнения загруженных машинных кодов используется функция USR, но с числовым аргументом, определяющим начальный адрес. Если вы выполните:

```
PRINT USR 32500
```

то получите ответ : 99

Возврат в бейсик- программу осуществляется обычным образом по команде микропроцессора RET. В машинной программе вы не должны использовать регистры IY и I.

Вы можете записать вашу программу на ленту:

```
SAVE «NAME» CODE 32500, 4
```

Можно записать эту программу и так, что она будет автоматически выполняться после загрузки:

```
10 LOAD « » CODE 32500, 4
20 PRINT USR 32500
```

для чего надо сделать:
SAVE NAME LINE

а затем:

```
SAVE «XXXX» CODE 32500, 4
LOAD «NAME»
```

Это приведет к тому, что вначале будет загружена и автоматически выполнена бейсик-программа, которая, в свою очередь, загрузит и выполнит программу в машинных кодах.

По данному вопросу можно рекомендовать ознакомиться с книгой «Искусство схемотехники» П.Хоровиц, У.Хилл Мир, 1986, том 2, стр.579-580.

Далее приводятся 78 команд микропроцессора 8085, совместимых с микропроцессором Z80 (158 команд). (U880-ГДР, К1810ВМ-СССР)

Мнемокака	Действие	КОП	Цикл
1	2	3	4
	Пересылка, загрузка и запись		
MOV R, R	Переслать регистр в регистр	01RR RRRR	4[7]
MVI R, D	Переслать непоср. в регистр	00RR R110	7[10]
LXI RP,DD	Загрузить непоср. в два рег.	00PP 0001	10
STAX B	Запомнить A косвенно по BC	0000 0010	7
STAX D	Запомнить A косвенно по DE	0001 0010	7
LDAX B	Запомнить A косвенно по BC	0000 1010	7
LDAX D	Загрузить A косвенно по DE	0001 1010	7
STA DD	Запомнить A по адресу DD	0011 0010	13
LDA DD	Загрузить A по адресу DD	0011 1010	13
SHLD DD	Запомнить H, L по адресу DD	0010 0010	16
LHLD DD	Загрузить H, L по адресу DD	0010 1010	16
XCHG	Обменять DE И HL	1110 1011	4
	Приращение и уменьшение		
INR R	Приращение регистра	00RR R100	4[11]
DCR R	Уменьшение регистра	00RR R101	4[11]
INX RP	Приращение пары регистров	00PP 0011	6
DCX RP	Уменьшение пары регистров	00PP 1011	6
	Арифметические и логические		
ADD R	Прибавить регистр к A	1000 0RRR	4[7]
ADC R	Прибавить рег к A с переносом	1000 1RRR	4[7]
SUB R	Вычесть регистр из A	1001 0RRR	4[7]
SBB R	Вычесть с заемом	1001 1RRR	4[7]
ANA R	Регистр & A	1010 0RRR	4[7]
XRA R	Искл. ИЛИ регистра и A	1010 1RRR	4[7]
ORA R	Ррегистр A	1011 0RRR	4[7]

CMA R	Сравнить регистр и A	1011 1RRR	4[7]
ADI D	Прибавить непоср. данные к A	1100 0110	7
ACI D	прибавить непоср. с переносом	1100 1110	7
SUI D	Вычесть непоср. из A	1101 0110	7
SBI D	Вычесть непоср. с заемом	1101 1110	7
ANI D	Непоср. & A	1110 0110	7
XRI D	Искл. или непоср. и A	1110 1110	7
ORI D	Непоср. ! A	1111 0110	7
CPI D	Сравнить непоср. с A	1111 1110	7
DAD RP	Прибавить пару регистров к HL	00PP 1001	11
	Операции с накопителями и флагами		
RLC	Сдвинуть A влево	0000 0111	4
RRC	Сдвинуть A вправо	0000 1111	4
RAL	Сдвинуть A влево чер. разр.пер	0001 0111	4
RAR	Сдвинуть A вправо чр.разр.пер	0001 1111	4
DAA	Десятич. коррекция накопителя	0010 0111	4
CMA	Дополнение к накопителю	0010 1111	4
STC	Установить бит переноса	0011 0111	4
CMC	Обратить бит переноса	0011 1111	4
	I/O управление и операции со стеком		
IN D	Ввод из порта D	1101 1011	10
OUT D	Вывод в порт D	1101 0011	11
EI	Разрешение прерываний	1111 1011	4
DI	Запрещение прерываний	1111 0011	4
NOP	Нет операции	0000 0000	4
HLT	Останов	0111 0110	4
PUSH RP	Занести пару регистров в стек	11PP 0101	11
POP RP	Взять пару регистров из стека	11PP 0001	10
XTHL	Обменять HL с верхом стека	1110 0011	19
SPHL	Переслать HL в SP	1111 1001	6
	Передачи управления		
JMP DD	Безусловный переход	1100 0011	10
JCC DD	Перейти по условию CC	11CC C010	10
CALL DD	Безусловный вызов	1100 1101	17
CCC DD	Вызов по условию CC	11CC C100	17(10)
RET	Возврат после вызова	1100 1001	10
RCC	Возврат по условию CC	11CC C000	11(15)
RST N	Возобновление в ячейке 8*N	11NN N111	11
PCHL	Переслать HL в PC	1110 1001	4

Обозначения :

1. Поля данных.

D - один байт непосредственных данных (длина команды 2 байта)
DD - двухбайтовый адрес (длина команды 3 байта)
Все остальные команды имеют длину 1 байт.

2. Циклы.

N - число тактов, нужное для выполнения команды.
[N]- число тактов, когда R = M (доступ в память).
(N)- число тактов, если условие не выполнено.

3. Поля регистров.

«R»	RRR
B	000
C	001
D	010
E	011
H	100
L	101
M	110
A	111

[«M» = (HL)]

«RP»	PP
BC	00
DE	01
HL	10
PS	11
PSW	11

4. Коды условий.

«CC»	CCC	Условие
NZ	000	не нуль
Z	001	нуль
NC	010	нет переноса
C	011	перенос
PO	100	нечетный паритет
PE	101	четный паритет
P	110	положительное
M	111	отрицательное

Приложение А

Полный набор символов

Дес. код	Символ	Шестн. код	Ассемблер. мнемоника	CBH.	EDH.
1	2	3	4	5	6
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, H	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	FLUSH 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), H	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, E	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 IN
72	H	48	LD C, B	BIT 1, B	C, (C) OUT
73	I	49	LD C, C	BIT 1, C	C, (C) ADD
74	J	4a	LD C, D	BIT 1, D	HL, BC LD
75	K	4B	LD C, E	BIT 1, E	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 IN
80	P	50	LD D, B	BIT 2, B	D, (C)
81	Q	51	LD D, C	BIT 2, C	OUT D, (C)

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 вврх	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	в (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	RRD IN
103	g (строчн)	67	LD H, A	BIT 4, A	L, (C) OUT
104	н (строчн)	68	LD L, B	BIT 5, B	(C), L
105	i (строчн)	69	LD L, C	BIT 5, C	
106	j (строчн)	6A	LD L, D	BIT 5, D	ADC HL, HL
107	к (строчн)	6B	LD L, E	BIT 5, E	LD HL, (NN)
108	k (строчн)	6C	LD L, H	BIT 5, H	
109	м (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	HALт	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	ABC HL, SP
123	ф и г у р . С к . Л е в	7B	LD A, E	BIT 7, E	LD SP, (NN)
124	верт.Черта	7C	LD A, H	BIT 7, H	

125	фигур.Ск.Пр.	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	o o o o	80	ADD A, B	RES 0, B	
129	o ж o o	81	ADD A, C	RES 0, C	
130	ж o o o	82	ADD A, D	RES 0, D	
131	ж o o o	83	ADD A, E	RES 0, E	
132	o o o ж	84	ADD A, H	RES 0, H	
133	o ж o ж	85	ADD A, L	RES 0, L	
134	ж o o ж	86	ADD A, (HL	RES 0, (HL	
135	ж ж o ж	87	ADD A, A	RES 0, A	
136	o o ж o	88	ADc A, B	RES 1, B	
137	o ж ж o	89	ADc A, C	RES 1, C	
138	ж o ж o	8A	ADC C, D	RES 1, D	
139	ж ж ж o	8B	ADc A, E	RES 1, E	
140	ж ж o ж	8C	ADC A, H	RES 1, H	
141	o ж ж ж	8D	ADC A, L	RES 1, L	
142	ж o ж ж	8E	ADC A, (HL	RES 1, (HL	
143	ж ж ж ж	8F	ADC A, A	RES 1, A	
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, N	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, N	SET 2, H	
213	MERGE	D5	PUSH DE	SET 2, L	
214	VARIFY	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	FLUSH	DB	IN A, (N)	SET 3, E	
220	BRIGHT	DC	CALL C, NN	SET 3, H	
221	INVERSE	DD	PREFIXES IN-STRUCTIONS USING IX	SET 3, L	
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), H	SET 4, E	
228	DATA	E4	CALL PO, N	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, N	SET 5, H	
237	GO SUB	ED	XOR N	SET 5, L	
238	INPUT	EE	RST 40	SET 5, (HL	

239	LOAD	EF	RET P	SET 5, A	
240	LIST	F0	ROR AF	SET 6, B	
241	LET	F1	JP P, NN	SET 6, C	
242	PAUSE	F2	DI	SET 6, D	
243	NEXT	F3	CALL P, NN	SET 6, E	
244	POKE	F4	PUSH AF	SET 6, H	
245	PRINT	F5	OR N	SET 6, L	
246	PLOT	F6	RST 48	SET 6, (HL	
247	RUN	F7	RET M	SET 6, A	
248	SAVE	F8	LD SP, HL	SET 7, B	
249	RANDOMIZE	F9	JP M, NN	SET 7, C	
250	IF	FA	EI	SET 7, D	
251	CLS	Fв	CALL M, NN	SET 7, E	
252	DRAW	FC	PREFIXES	SET 7, H	
253	CLEAR	FD	INSTRUCTIONS USING IY	SET 7, L	
254	RETURN	FE	CP N	SET 7, (HL	
255	COPY	FF	RST 56	SET 7, A	

Приложение В
Сообщения

Они появляются в нижней части экрана, если компьютер остановился при выполнениии некоторого оператора бейсика, и указывает причину, вызвавшую останов. Сообще-ние содержит кодовый номер или букву. Краткое сообщение 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 сообщение начинается с оператора, специфицированного в предыдущем сообщении, но имеется исключения- сообщение 0, 9, D. (смотри также приложение С).

код	Значение	ситуация
1	2	3
0	OK (о'кей! порядок!) Успешное завершение или переход на строку с номером, большим, чем имеется всего. Это сообщение не меняет строки или оператора, определенного для CONTINUE	разное
1	NEXT WITHOUT FOR (NEXT без FOR) Управляющей переменной нет (не была определена в операторе FOR), но есть обычная переменная с тем же именем.	NEXT
2	VARIABLE NOT FOUND (переменная не найдена) 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 выдает	разное

	ся, если она не была предварительно определена в операторе DIM перед использованием или загрузкой с ленты.	
3	**SUBSCRIPT WRONG** (ошибочный адрес) Индекс превышает размерность массива, либо ошибочное число задает индекс, если индекс отрицательный или больше 65535, то выдается сообщение B.	в индексной переменной или подстроке
4	**OUT OF MEMORY** (вне памяти) В памяти недостаточно места для ваших действий. Вы можете освободить себе память, удалив командные строки, используя DELETE, затем удалить 1 или 2 строки программы (с целью возврата их впоследствии), получить дополнительную память, маневрируя оператором CLEAR.	LET, INPUT, FOR, DIM, GO SUB, LOAD, MERGE.
5	**OUT OF SCREEN** (вне экрана) Если оператор INPUT генерирует больше, чем 23 строки в нижней половине экрана. Также встречается с **PRINT AT 22, ...**	PRINT, PRINT AT
6	NUMBER TOO BIG (число больше макс. Допуст.) В результате вычислений получилось число больше 10**38.	арифметич. операции.
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 см. также сообщение 3.	RUN, RANDOMIZE, POKE, DIM. GO TO, GO SUB, LIST, LLIST, PAUSE, PLOT, CHR$, PEEK, USR(с целым аргументом)

C	NONSENSE IN BASIC (выражение не бейсика) Текст (строка) не распознается бейсиком как допустимое выражение.	VAL, VAL$
D	BREAK-CONT REPEATS Клавиша BREAK нажата во время действия перефирийной 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 комьютер запросил свертку, а вы ответили N, SPASE или STOP.
E	OUT OF DATA (вне данных) Попытка выдать READ когда список данных в DATA кончился.	READ
F	INVALID FILE NAME (неверное имя файла) Оператор SAVE с пустой строкой вместо имени 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 операциях с внешними устройствами
K	INVALID COLOUR (неверный цвет) Специфицированное число имеет неверное значение.	INK, PAPER, BORDER, FLUSH, BRIGHT, INVERSE, OVER, а также после одной из передач управл. символов

L	BREAK INTO PROGRAM (BREAK во время выполнения программы) Нажата клавиша BREAK;это обнаруживается между двумя операторами. Строка и номер оператора в строке указывает на оператор, выполняемый 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	STATEMENT LOST (оператор отсутствует) Переход к оператору, которого уже нет.	RETURN NEXT CONTINUE
O	INVALID STREAM (ошибочный поток данных)	в операциях ввода-вывода
P	FN WITHOUT DEF (FN без DEF) Определяемая пользователем функция не определена в операторе DEF FN.	FN
Q	PAREMETER 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 может быть считан.	VARIFY, LOAD, MERGE.

Приложение С (часть 1)
Описание микрокомпьютера ZX SPECTRUM

Клавиатура.

Каждая клавиша клавиатуры компьютера ZX SPECTRUM имеет многофункциональное назначение и позволяет вводить как отдельные символы, так и целые слова. Действие, производимое клавишей определяется частично переключающими клавишами (CAPS SHIFT и SYMBOL SHIFT) а частью режимом, в котором находится компьютер.

режим отображается курсором, мерцающей буквой, указывающей позицию, в которую будет вводиться очередной символ с клавиатуры. Возможны следующие режимы:

K-(для ключевых слов) KEYBORDS.

Этот режим автоматически сменяет режим L, если компьютер переходит в ожидание ввода команды или строки программы. Это может быть либо в начале строки, либо после THEN, либо после «:». И если не было нажатия переключающих клавиш, то нажатие любой клавиши будет интерпретироваться как ключевое слово (написанное на клавише) или цифра.

L-(для букв) LETTER.

Основной режим для компьютера. Если не было переключения регистров, то клавиша интер-

претируется как основной символ, нанесенный на эту клавишу. Для обоих режимов L и K при одновременном нажатии с клавишей клавиши SYMBOL SHIFT, клавиша будет интерпретироваться как вспомогательный символ, а при нажатии CAPS SHIFT с цифрой, клавиша будет интерпретироваться как управляющая функция, написанная на белом поле клавиши. Нажатие CAPS SHIFT с любой из клавиш не вызывает ключевого слова в режимах K и L.

C-(для заглавных букв) CAPITAL.

Режим представляет собой вариант режима L, в котором используются заглавные буквы. CAPS LOCK используется для перехода из режима L в C и обратно.

E-(для расширения) EXTEND.

Используется для получения дальнейших символов, главным образом знаков. Этот режим вводиться одновременным нажатием двух переключающих клавиш, с удержанием затем только одной клавиши. В этом режиме клавиша дает один символ или знак (изображенный на зеленом поле клавиши), если не нажата переключающая клавиша, или знак, изображенный на красном поле, если удерживается переключающая клавиша. Цифровые клавиши выдают знак, если нажимаются вместе с SYMBOL SHIFT, в противном случае они выдают последовательность управления цветом.

G- (для графики) GRAPHICS

Режим вводится после нажатия GRAPHICS (CAPS SHIFT и 9) и сохраняется до следующего нажатия этой клавиши. Цифровые клавиши будут выдавать мозаичные графические символы, сохраняя GRAPHICS и DELETE, а каждая алфавитная клавиша, кроме V, W, X, Y и Z, будет выдавать определенный пользователем графический символ. Если некоторая клавиша будет удерживаться более 2-3х сек., это вызовет повторение производимого ею действия. Ввод с клавиатуры производится в нижнюю часть экрана. Каждый символ(или составной знак) вставляются перед курсором. Курсор может быть переслан влево действием CAPS SHIFT и 5, а вправо CAPS SHIFT и 8. Символ перед курсором можно удалить командой: DELETE(CAPS SHIFT и 0. Целая строка может быть удалена вводом EDIT (CAPS SHIFT и 1) и последующим ENTER. Когда нажимается ENTER, выполняется набранная строка, либо она вводится в программу, либо она используется как входные данные для INPUT-оператора, либо в строке имеются синтаксические ошибки. В этом случае нажатием [?] происходит переход на ошибку.

Когда строки программы введены, листинг отображается в верхней части экрана. Более подробно этот процесс описан в Гл.2.

Последняя введенная строка называется текущей и отмечается символом [>], и ее можно изменить, используя клавиши перемещения курсора вверх и вниз (CAPS SHIFT и 6;CAPS SHIFT и 7).

Если введен EDIT (CAPS SHIFT и 1), то текущая строка переносится в нижнюю часть экрана и становится доступной для редактирования.

Если выполняется команда или целая программа, то результаты отображаются в верхней половине экрана и сохраняются до ввода строки программы, ввода пустой строки или нажатия клавиши управления курсором вверх или вниз. В нижней части выдаются сообщения и коцы, приведенные в приложении в. В сообщении указывается номер ошибочной строки (0 для команды) и позиция оператора в этой строке. Сообщение сохраняется на экране до нажатия любой клавиши (отображается переходом в режим K).

В определенных обстоятельствах CAPS SHIFT и SPACE действуют как BREAK, останавливая компьютер с сообщениями D или L, и при этом до останова:

а) завершается выполнявшийся оператор или

б) завершаются действия, выполняемые компьютером с магнитофоном или принтером.

Пример использования клавиатуры:

При курсоре [K]

LN	1. Простое нажатие клавиши	- на экране —> COPY
	2. SYMBOL SHIFT и клавиша	- на экране —> Z
		- курсор —> [L]
:	3. CAPS SHIFT и клавиша	- на экране —> :
Z	4. BREAK / SPACE и клавиша	- на экране —> Z
COPY	5. SYMBOL SHIFT и CAPS SHIFT	- курсор —> [E]
	При курсоре [E]	
BEEP	6. Простое нажатие клавиши	- на экране —> LN
	7. CAPS SHIFT и клавиша	- на экране —> BEEP

Экран телевизора

Экран телевизора содержит 24 строки по 32 позиции в каждой и делится на 2 части. Верхняя часть в 22 строки отображает листинг или вывод из программы. Когда вывод в верхней части достигает низа, необходима свертка на одну строку. При этом может захватываться строка, которую вам хочется сохранить. Компьютер в этом случае останавливается с запросом «SCROLL?». Если теперь нажать клавиши N, SPACE или STOP, то программа остановится с выдачей сообщения. «D BREAK = CONT REPORTS». Нажатие других клавиш разрешает свертку и продолжение выполнения.

Нижняя часть используется для ввода команд строк программы и входных INPUT- данных, а также для отображения сообщений. Нижняя часть экрана состоит из двух строк(верхняя из них чистая для расширения). При переполнении верхней строки осуществляется свертка.

Каждая позиция имеет атрибуты, определяющие ее как чистую (цвет фона), либо как закрашенную (основной цвет), с повышенной или пониженной яркостью, мерцающую или нет.

Доступны цвета: черный, голубой, красный, пурпурный (фиолетовый), зеленый, желтый, белый.

Края могут быть установлены в определенный цвет использованием оператора BORDER.

Каждая позиция подразделяется на 8x8 точек, а графика символов обеспечивается индивидуальным определением каждой точки. Атрибуты каждой позиции настраиваются при записи символа или при установке точки (PIXEL). Способ настройки определяется параметрами вывода, имеющими 2 установки (постоянную и временную)в 6 операторах:

PAPER, INK, FLUSH, BRIGHT, INVERSE и OVER. Постоянные параметры для верхней части экрана устанавливаются в операторах PAPER, INK и т.д. Обычно они имеют черный цвет для закрашенной точки (INK) и белый для фоновой (PAPER), нормальную яркость, не мерцающие, не инверсные. Постоянные параметры для нижней части экрана используют цвет рамки (BORDER COLOUR) как цвет фона (незакрашенный), с черным или белым цветом, нормальную яркость, немерцающие.

Временные параметры устанавливаются командами:

PAPER, INK и т.д.,

Вставляемыми в операторы

PRINT, LPRINT, INPUT, PLOT, DRAW и CIRCLE,

а также

PAPER, INK и тому подобными управляющими символами, когда они выводятся на

телевизор.

Временные параметры сохраняются до конца действия оператора PRINT или других.

Параметры от 0 до 7 определяют цвета выводимого символа:

0-черный (BLACK)
1-голубой (BLUE)
2-красный (RED)
3-фиолетовый (MAGNETA)
4-зеленый (GREEN)
5-синий (GYAN)
6-желтый (YELLOW)
7-белый (WHITE)

Параметр 8 определяет, что цвет должен оставаться при выводе без изменения.

Параметр 9 (контрастность) определяет, что цвет должен стать либо белым, либо черным для выделения его от других цветов.

Параметры FLUSH и BRIGHT могут принимать значения 0, 1 или 8.

Параметр 1 указывает, что включается повышенная яркость и мерцание.

Параметр 0 указывает, что повышенная яркость и мерцание отключаются.

Параметр 8 указывает, что все остается без изменения. Параметры OVER и INVERSE могут принимать значения 0 или 1.

OVER 0 - новый символ затирает старый

OVER 1 - код старого символа и нового символа соединяются операцией ‚исключающего ИЛИ', образуя новый символ (OVERPRINTING)

INVERSE 0 - новый символ печатается в неинверсном (позитивном) виде

INVERSE 1 - новый символ печатается в инверсном (негативном) виде

Когда на телевизор передается управляющий символ TAB, то два старших байта используются для спецификации tab STOP N (первый байт является старшим). Это обеспечивается прогоном от 32 до ‚N' (указанным в TAB) и затем выводом нужного количества пробелов для смещения текущей позиции вывода в колонку ‚N'.

Если на вывод передается запятая как управляющий символ, то выводится нужное количество пробелов для перевода текущей позиции вывода в позицию 0 или 16.

Если передается управляющий символ ENTER, то позиция вывода передается на следующую строку.

Принтер

Вывод на принтер осуществляется через буфер длинной в 32 символа. Очередная строка выдается из буфера на принтер в следующих случаях:

а) когда окончен вывод одной строки и вывод переходит к другой строке,

б) при передаче в буфер символа ENTER,

в) при завершении программы, если еще остались другие невыведенные данные,

г) если встретились управляющие символы TAB или запятая, тебующие перевода строки.

Управляющие символы tab и запятая производят вывод пробелов при работе с телевизором. Управляющий символ at изменяет позицию вывода, используя число, задающее позицию.

Принтер также правильно реагирует на управляющие символы INVERSE, OVER (и операторы с тем же именем), но не воспринимает PAPER, INK, FLUSH и BRIGHT. При вводе BREAK принтер останавливается с выдачей сообщения ‚B'. При отсутсвии принтера вывод просто не осуществляется.

Приложение С (часть 2)

Язык программирования бейсик
Справочное пособие

Все числа в системе могут иметь точность 9 или 10 знаков. Наибольшее число 10**38, а наименьшее положительное число 4*10**(-39). Числа имеют внутреннее представление как числа с плавающей (двоичной) точкой, выделением одного байта на показатель степени ‚e' (экспоненты) в интервале от 1 до 255, и четырех байтов на мантиссу ‚м' в интервале от 0.5 до 1 (M^ = 1). Это представляется числом м*2** е(- 128).

Поскольку 1/2 < = м < 1, старший 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). 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. числовые массивы имеют имена длиной в одну букву, которая может быть такой же, как имя скалярной переменной. Эти массивы могут иметь произвольное количество измерений и 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 собой 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 'числовым выражение' 'ТО ' другим 'числовым 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, ..., индекс N) (сечение).

В случае *), строка выражения имеет значение S$.

Если сечение массива пусто , то S$ считается подстрокой от самой себя.

Если сечение представляется числовым выражением со 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' (умалчивоемое 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 операторLET). Если часть строки записывается в строковой литерал, она должна удваиваться.

Функции языка Бейсик для SPECTRUM ZX приведены в Таблице на стр 70...72

имя функции	тип аргумента	действие (возвращаемое значение)
1	2	3
ABS	число	абсолютное значение
ACS	число	арккосинус в радианах; выдает сообщение об ошибке A, если н 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$, если B< >0; "", если B=0	
ASN	число	арксинус в радианах; выдает сообщение A, если X не лежит в интервале от -1 до 1.
ATN	число	арктангенс в радианах.
ATTR	два числовых аргумента X и Y, заключаемые в скобки	число, двоичный код которого, представляет собой атрибуты Y-ой позиции X-ой строки экрана. Бит 7 (старший) равен 1 для мерцающего поля, и 0 для немерцающего. Биты с 5 по 3- цвет фона биты с 2 по 1 - цвет закрашивания. Выдает сообщение B, если 0< =X< =23 и 0< =X< =31.
BIN		это необычная функция. За BIN записывается последовательность нулей и единиц, представляющая собой двоичное представление числа, которое записывается в память.
CHR$	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
LN	число	натуральный логарифм. Выдает сообщение A, если X < = 0.
NOT	число	0, если X < > 0, 1, если X = 0. Операция имеет четвертый приоритет.
OR	логическая операция. оба операнда числа.	Г 1, если B < > 0 A OR B = < L 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 и Y, заключенных в скобки	1, если точка экрана с координатами (X, Y) закрашена. 0, если эта точка имеет цвет фона. Выдает сообщение B, если не выполняются условия 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, заключенных 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 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.

USR	число	вызывает подпрограмму 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 нестроковое (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 Г
> больше : оба операнда должны быть одного
< меньше : типа. Результат равен 1, если
>= больше или равно : сравнение истино и равно 0,
<= меньше или равно : если нет.
<> не равно L

Функции и операции имеют следующий приоритет:

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;
B - некоторое выражение;

F - выражение, имеющее строковое значение;
S - последовательность операторов, разделенных двоеточием „:"
C - последовательность символов управления цветом.

Каждый заканчивается „ , или „;". Цветовой символ имеет форму операндов:
PAPER, INK, FLUSH, 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 некоторые ограничения в IF и REM.

Все операторы языка сведены в следующую таблицу:

Оператор	Действие оператора
1	2
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 (бордюра) экрана. Выдает сообщение об ошибке K, если 0 > M > Y.
BRIGHT M	Устанавливает яркость выводимого символа 0-для обычной яркости; 1-для повышенной яркости; 8-сохраняет существующую яркость.
CAT	Без MICRODRIVE не работает.
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 файл экрана.
CONTINUE	Продолжает выполнение программы, начатой ранее и остановленной 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 01, дает сообщение 0, если было 02, или дает сообщение N, если было 03 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.
DATA E1, E2, E3, ...	Часть списка данных. Должна располагаться в программе.
F FNA(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 FNA(), 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 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.
DRAW X, Y, Z	Изображает линию от текущей графической позиции в точку с приращениями X, Y по дуге в Z радиан. Выдает сообщение в при выходе за пределы экрана.
ERAZE	Без MICRODRIVE не работает.
FLUSH 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	FOR A = X to Y STEP 1
FOR A = X TO Y STEP Z	Уничтожает скалярную переменную ‚A' и устанавливает управляющую пепеременную ‚X', предел ‚Y', шаг приращения ‚Z', зацикливает адрес, указанный в утверждении после FOR оператора. Проверяет, если начальное значение больше (если STEP > = 0) или меньше (если STEP < 0), чем предел, то происходит переход к утверждению NEXTa или выдача сообщения 1, если нет (см. NEXT). Сообщение 4 выдается, если недостаточно места для размещения управляющей переменной.
FORMAT F	Без MICRODRIVE не работает.
GO SUB N	Проталкивает строку с оператором GO SUB в стек для использования затем как GO TO N. Выдается сообщение 4, если не все подпрограммы завершились с RETURN.
GO TO N	Продолжает выполнение программы со строки ‚N'. Если ‚N' опущено, 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' не допустима.
INK N	Устанавливает цвет закрашивания (т.е. цвет, которым будут изображаться символы на цвете фона). ‚N' в интервале ои 0 до 7 указывает цвет. N = 8 - оставить цвет без изменений, 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 операторе 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 также, 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 знаком вопроса [?]. Для строкового выражения вводной буфер устанавливается для размещения двух таких строк (который при необходимости может быть увеличен). Если первый вводимый символ STOP, то программа останавливается с сообщением H. Случай В) подобен случаю Б) с той лишь разницей, что вводимая информация представляет собой строковый литерал неограниченной длины, и STOP в этом случае не сработает. для останова вы должны нажать клавишу , курсор вниз'.
INVERSE N	Символ управления инверсией выводимого символа. Если N = 0, символ выводится в обычном виде с прорисовкой цвета закрашивания (INк) на цвете фона (PAPER). Если N = 1, то цветовое решение изображения символа меняется на обратное. Смотри приложение В. Выдает сообщение K, если ‚N' не 0 или 1.
LET V = E	Присваивает значение ‚E' переменной ‚V'. Ключевое слово 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 дополнением пробелами до фиксированной длины.
LIST	То же, что и LIST 0.
LIST N	Записывает текст программы в верхнюю часть экрана, начиная с первой строки, меньшей, чем ‚N', и делает ‚N' текущей строкой.
LLIST	То же, что и LIST 0
LLIST N	Подобно LIST, но вывод осуществляется на принтер.
LOAD F	Загружает программу и переменные.
LOAD F DATA ()	Загружает числовой массив.

LOAD F CODE M, N	Загружает старшие ‚N' байтов, начиная с адреса ‚M'.
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 экрана и загружает его с кассетного магнитофона. Смотри главу 20.
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 которых совпадают номера или имена с такими же на ленте.
MOVE F1, F2	Без MICRODRIVE не работает
NEW	аммирования бейсик, уничтожая старую программу, переменные и используемую память, включая и байт адреса в системной переменной RAMBOT, 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 больше значения ‚предел', или STEP < 0, а значение ‚A' меньше, чем значение ‚предел', то происходит переход к оператору цикла.
OPEN#	Без MICRODRIVE не работает.
OUT M, N	Выводит байт ‚N' в порт ‚M'. Операция выполняется на уровне микропроцессора (загружает в регистровую пару BC адрес ‚M', а регистр A-'N' и выполняет команду ассемблера OUT (C) < A). 0 < = M < = 65535, -255 < = N < = 255, иначе выдается сообщение B.
OVER N	Управляющий символ надпечатывания по выведенной строке. Если N = 0, то выводимый символ затирает существующий в данной позиции. Если N = 1, то новый символ соединяется со старым, образуя закрашивающий цвет, при условии, что старый символ имел указание цвета, отличное от старого, или цвет фона, если оба указывают на один и тот же цвет (либо фона, либо закрашивания, сложение по модулю 2). смотри приложение в.
PAPER N	Подобен INK, но управляет цветом фона.
PAUSE N	Останавливает выполнение программы и задерживает изображение на экране на ‚N' кадров (50 кадров в сек. - 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 первого нажатия клавиши.
PLOT C;M, N	Выводит точку закрашивающего цвета (обработанную OVER и

	INVERSE) с координатами (ABS(M), ABS(N)) смещает графическую (PLOTPOSITION) позицию. Если цветной символ ‚C' не специфицирован иначе, то цвет закрашивания в позиции, где расположена эта точка, изменяется на текущий сплошной закрашивающий цвет, и другие указания (цвет фона, мерцание, яркость) остаются без изменения. 0 < = ABS(M) < = 65535, 0 < = ABS(N) < = 175, иначе–сообщение B.
POKE M, N	Записывает значение ‚N' в байт памяти по адресу ‚M'. 0 < = M < = 65535, -255 < = N < = 255, иначе сообщение B.
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', и апостроф порождает символ ENTER. В конце оператора PRINT, если он не заканчивается точкой с запятой, запятой или апострофом, автоматически выводится символ ENTER. PRINT-символом может быть а) пустая строка; б) числовое выражение. если значение выражения отрицательное, то выводится знак минус. если x < = 10**-5 или x > = 10**13, вывод осуществляется в показательной форме. Мантисса представляется 8 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. в строке возможны пробелы до и после символов. управляющие символы вызывают определяемое ими действие. Не отражаемые на экране символы выводятся как ‚7' г) AT M, N вывод в строку ‚M', позицию ‚N' д) tab N вывод управляющего символа TAB с последующими 2-мя байтами ‚N' (первый байт-старший). Вызывает tab–останов. е) цветовой символ в форме PAPER, INK, FLUSH, 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 запускает сокращение RAND (см. Клавишу). Выдает сообщение B, если ‚N' не лежит интервале от 0 до 65535.
READ V1, V2, ..., Vk	Присваивает переменным одна за другой значения, последовательно представленные в списке DATA.
REM ...	Не выполняется.'...' может быть последовательностью символов (исключая ENTER). Может включать двоеточие (,') для указания отсутствия операторов в строке с REM.

RESTORE N	Перезаписывает указатель данных в первый оператор DATA в строке меньшей, чем ‚N'. Следующий оператор READ начнет считывание отсюда.
RETURN	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, когда операторы GO SUB не сбалансированы операторами RETURN .
RUN	То же самое, что и RUN 0.
RUN N	CLEAR, а затем GO TO N .
SAVE F	Записывает на ленту программы 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 ()	Запись на ленту строкового массива
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. смотри главу 20.
STOP	Останавливает выполнение программы с выдачей сообщения 9. CONTINUE (продолжение) будет осуществлятся со следующего оператора.
VERIFY	То же, что и LOAD, за исключением того, что данные загружаются в ОЗУ, но сравниваются с находящимися там. выдает сообщение B, если обнаружен хотя бы один не совпадающий байт.
	Дополнение
LOAD F DAT$()	Загружает строковый масссив
RESTORE	То же самое, что и RESTORE 0

Приложение D

Примеры программ

Приложение D содержит некоторые примеры программ, демонстрирующие возможности ZX SPEKTRUM.

Первая из этих программ требует ввести дату и дает день недели, который соотвует 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(MGNTH) THEN PRINT «THIS MONTH HAS ON    LY»;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 = 6 TO 3 STEP-1: REM REMUVE 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», «STAUR», «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 EVRYTHING 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
2010    PRINT YD;»YD»;FT;»FT»;IN;»IN»;:      RETURN
```

ПРОГРАММА МОДЕЛИРУЕТ ВЫБРАСЫВАНИЕ МОНЕТЫ ДЛЯ ИГРЫ В «КИТАЙКУ»:

```
5 RANDOMIZE
10 FOR M = 1 TO6: REM FOR 6 THROWS
20 LET C = 0: REM INITIALIZE COIN TOTAL TO 0
30 FOR N = 1 TO 3: REM FOR 3 coINS
40 LET C = C + 2 + INT(2 * RND)
50 NEXT N
60 PRINT « «;
70 FOR N = 1 TO 2: REM      1ST FOR THE THROWN HEXAGRAM,
                            2ND FOR THE CHANGES
80 PRINT «—»;
90 IF C = 7 THEN PRINT «–»;
100 IF C = 8 THEN PRINT « «;
110 IF C = 6 THEN PRINT «X»;:LET C = 7
120 IF C = 9 THEN PRINT «0»;:LET C = 8
130 PRINT «—»;
140 NEXT N
150 PRINT
160 INPUT A$
170 NEXT M: NEW
```

Для запуска программы введите ее в компьютер, запустите на выполнение, а затем нажмите клавишу ENTER 5 раз для получения двух гексаграмм. Посмотрите «китайскую книгу изменений». Текст будет описывать ситуацию и последовательность соответствующих этому действий, а вы должны оценить глубину параллелей между ней и вашей собственной жизнью. Нажмите клавишу ENTER шестой раз. Программа будет обнуляться - это избавит вас от легкомысленного использования результатов.

Многие пользователи найдут тексты всегда более вероятными, нежели они сами могут это предполагать.

Следующая программа - игра «ящеры». Вы задумываете название некоторого животного, а компьютер пытается его отгадать, задавая вам вопросы, на которые вы должны отвечать «да» или «нет». Если компьютер нем был ранее знаком с таким животным, то он просит вас задать ему наводящие вопросы, которые помогут ему найти правильный ответ или он попросит вас предложить ему задать название нового животного:

```
 5 REM  PANGOLINS
10 LET  NQ = 100: REM NUMBER OF QUESTIONS AND ANIMALS
15 DIM  Q$(NQ, 50): DIM  A(NQ, 2): DIM R$(1)
20 LET  QF = 8
30 FOR  N = 1 TO QF/2-1
40 READ Q$(N):  READ A(N, 1): READ A(N,  2)
50 NEXT N
60 FOR  N = N TO QF-1
70 READ Q$(N):  NEXT N
100 REM START PLAYING
110 PRINT «THINK OF AN ANIMAL.», «PRESS  ANY KEY TO CONTINUE.»
120 PAUSE 0
130 LET  C = 1: REM START WITH 1ST QUESTION
140 IF A(C, 1) = 0 THEN GO TO 300
150 LET  P$ = Q$(C): GO SUB 910
160 PRINT «?»: GO SUB 1000
170 LET  IN = 1: IF R$ = »Y»  THEN GO TO 210
180 IF R$ = »Y» THEN GO TO 210
```

```
190 LET IN = 2: IF R$ = »N» THEN GO TO 210
200 IF R$ < > »N» THEN GO TO 150
210 LET C = A(C, IN): GO TO 140
300 REM ANIMAL
310 PRINT «ARE YOU THINKING OF»
320 LET P$ = Q$(C): GO SUB 900: PRINT «?»
330 GO SUB 1000
340 IF R$ = »Y» THEN GO TO 400
350 IF R$ = »Y» THEN GO TO 400
360 IF R$ = »N» THEN GO TO 500
370 IF R$ = »N» THEN GO TO 500
380 PRINT «ANSWER ME PROPERLY WHEN I'M», «TALKING TO YOU.»: GO TO 300
400 REM QUESSED IT
410 PRINT «I THOUQHT AS MUCH.»: GO TO 800
500 REM NEW ANIMAL
510 IF QF > NQ-1 THEN PRINT «I'M SURE YOUR ANIMAL IS VERY», «INTERESTING,
        BUT I DON'T HAVE», «ROOM FOR IT JUST NOW.»: GO TO 800
520 LET Q$(QF) = Q$(C): REM MOVE OLD ANIMAL
530 PRINT «WHAT IS IT, THEN?»: INPUT Q$(QF + 1)
540 PRINT «TEEL ME A QUSTION WHICH DIST-», «INQUISHES BETWEEN»
550 LET P$ = Q$(QF): GO SUB 900: PRINT «AND»
560 LET P$ = Q$(QF + 1): GO SUB 900:PRINT « «
570 INPUT S$: LET B = LENS$
580 IF S$(B) = »?» THEN LET B = B-1
590 LET Q$(S) = S$(TO B): REM INSERT QUESTION
600 PRINT «WHAT IS ANSWER FOR»
610 LET P$ = Q$(QF + 1): GO SUB 900: PRINT «?»
620 GO SUB 1000
630 LET IN = 1:LET IO = 2: REM ANSWERS FOR NEW AND OLD ANIMALS
640 IF R$ = »Y» THEN GO TO 700
650 IF R$ = »Y» THEN GO TO 700
660 LET IN = 2: LET IO = 1
670 IF R$ = »N» THEN GO TO 700
680 IF R$ = »N» THEN GO TO 700
690 PRINT «THAT'S NO GOOD.»: GO TO 600
700 REM UPDATE ANSWERS
710 LET A(C, IN) = QF + 1: LET A(C, IO) = QF
720 LET QF = QF + 2: REM NEXT FREE ANIMAL SPACE
730 PRINT « THAT FOOLED ME.»
800 REM ADAIN?
810 PRINT «DO YOU WANT ANOTHER GO?»: GO SUB 1000
820 IF R$ = »Y» THEN GO TO 100
830 IF R$ = »Y» THEN GO TO 100
840 STOP
900 REM PRINT WITHOUT TRAILING SPACES
905 PRINT « «;
910 FOR N = 50 TO 1 STEP -1
920 IF P$(N) < > » « THEN GO TO 940
930 NEXT N
940 PRINT P$(TO N);: RETURN
1000 REM GET REPLY
1010 INPUT R$: IF R$ = » « THEN RUTURN
2000 REM INITIAL ANIMALS
2010 DATA «DOES IT LIVE IN THE SEA», 4, 2
2020 DATA «IS IT SCALY», 3, 5
```

```
2030 DATA«DOES IT EAT ANTS», 6, 7
2040 DATA «A WHALE», «AWLANCMANGE», «A PANGOLIN», «AN AN
```

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  BLAK 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, 174-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 OVER 1: 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 HENG 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 A4 = »» 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
2040 DATA 184, 156, 68, 0, 184, 140, 16, 16
2050 DATA 204, 156, -20,   -20, 240, 156, 0, -16
```

Приложение Е
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 0001	
17	12	0001 0010	
18	13	0001 0011	
19	14	0001 0100	
20	15	0001 0101	
21	16	0001 0110	
22	17	0001 0111	
23	18	0001 1000	
24	19	0001 1001	
25	1A	0001 1010	
26	1B	0001 1011	
27	1C	0001 1100	
28	1D	0001 1101	
29	1E	0001 1110	
30	1F	0001 1111	
31	20	0010 0000	

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'. Число не может превышать 65535, т.е. занимать более шестнадцати двоичных разрядов (битов).

Приложение F
Использование кириллицы

MOVE, ERASE

В оригинальном исполнении ROM микрокомпьютера ZX SPECTRUM не содержит знакогенератора русских букв. Однако в предлагаемом описании речь идет о доработке стандартной прошивки ПЗУ 573 РФ4 в части, касающейся использования букв русского алфавита. Для перехода на «русский регистр» и обратно используются не задействованные в стандартном SPECTRUM функции ‚MOVE' и ‚ERASE'.Кроме того, использование кирилицы в SPECTRUM подчиняется обычным для такого применения ограничениям, т.е. русские буквы могут появиться в программе только в строковых константах. Это значит, ч т о применение букв русского алфавита вне областей, ограниченных строковымидвойными кавычками, интерпретируется бейсиком как синтаксическая ошибка.

Применение для переключения языкового регистра не используемых, но стандартных функций MOVE и ERASE, делает программно-совместимыми стандартный и модифицированный ROM., т.е. программы, написанные с использованием русского алфавита (но только в области констант), будут нормально выполняться и нестандартном SPECTRUMe (разумеется без отображения кирилицы на экране). Для перехода к русскому алфавиту необходимо ввести функцию MOVE (в режиме «E» курсора нажать SYMBOL SHIFT и 6). После набора текста на русском языке необходимо переключить регистр обратно- ввести ERASE (в режим «E» курсора нажать SYMBOL SHIFT и 7)

Таблица соответствия

Букв русского алфавита клавишам SPECTRUM

Латынь	Как вводится	Русский
A	ПРОСТЫМ НАЖАТИЕМ КЛАВИШИ	А
B	-	Б
W	-	В
G	-	Г
D	-	Д
E	-	Е
V	-	Ж
Z	-	З
I	-	И
J	-	Й
K	-	К
L	-	Л
M	-	М
N	-	Н
O	-	О
P	-	П
R	-	Р
S	-	С
T	-	Т
U	-	У
H	-	Х
C	-	Ц
A	«E»-SYMBOL SHIFT	Ч (БОЛЬШОЕ)
H	SYMBOL SHIFT	ч (МАЛОЕ)
F	«E»-SYMBOL SHIFT	Ш (БОЛЬШОЕ):
Y	«E»-SYMBOL SHIFT	ш(МАЛОЕ)
G	«E»-SYMBOL SHIFT	Щ (БОЛЬШОЕ):
U	«E»-SYMBOL SHIFT	щ(МАЛОЕ)
Y	-	
X	-	Ь
S	«E»-SYMBOL SHIFT	Э (БОЛЬШОЕ)
D	«E»-SYMBOL SHIFT	э (МАЛОЕ)
X	SYMBOL SHIFT	Ю (БОЛЬШОЕ)
2	SYMBOL SHIFT	ю (МАЛОЕ)
Q	-	Я

Переход от простых букв к строчным и наоборот осуществляется обычным образом (CAPS SHIFT и 2) для букв, вводимых простым нажатием клавиши. Для дополнительных букв, которым не хватает клавиш SPECTRUMa, используются разные клавиши для прописных и строчных букв, а также режим «E» курсора и клавиша SYMBOL SHIFT (см. таблицу).