



*О. В. Григорьев*

★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★

**БЕЙСИК для  
ZX  
SPECTRUM**



О.В.ГРИГОРЬЕВ

Бейсик

для

**"ZX-SPECTRUM"**



ББК 32.97  
Г83

Издание подготовлено малым предприятием  
"Канон" совместно с издательством "Лига".

Г-83 Григорьев О.В. "Бейсик для ZX-SPECTRUM".  
/ Художник Вельбой В.В. - Свердловск: "Лига", 1991.  
- 144 стр.

В данной книге описан алгоритмический язык Бейсик, предназначенный для решения задач прикладных программ на микро-ЭВМ "ZX-SPECTRUM". Изложены основные приемы и особенности программирования на этом языке. Книга рассчитана на читателей, имеющих опыт работы на ЭВМ.

ISBN 5-85258-005-8

© Григорьев О.В., 1991 г., составление.



---

# Глава 1

---

## Введение

Если вы читаете эту книгу впервые или открыли ее на этой странице, то вы должны иметь представление о том, что команды бейсика выполняются непосредственно (интерпретация), операторы имеют номер 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 только в ZX SPECTRUM.

Запуск бейсика начнем с повторения некоторых моментов, изложенных во вводной книге, но рассмотрим их значительно более подробно, уяснив себе, что можно делать, а чего нельзя.

Что бы вы ни делали, старайтесь в своей деятельности использовать компьютер. Если у вас возник вопрос "что будет, если я сделаю так и так?", тогда ответ очень прост: введите эти фразы в компьютер и вы увидите сами.

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

В конце этой книги имеется несколько приложений, они содержат сведения по организации памяти, по операциям 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, функции и т.п.).



Для того, чтобы реализовать все эти функции и команды, некоторые клавиши клавиатуры имеют 5 и более значений, получаемых либо путем выбора 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.

Режим 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 ввода 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 к знакоместу сразу же после двоеточия ":" (за исключением двоеточия в строке), если не изменен режим, то нажатие следующей клавиши будет интерпретироваться как ключевое слово, написанное на клавиатуре, 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 другой клавиши воспринимается как вспомогательный символ, изображенный на клавише, а в случае нажатия клавиши 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-нибудь другой клавиши. В этом режиме нажатие клавиши 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 **SIMBOL 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** может вызывать появление определенных пользователем графических символов.

Если некоторая клавиша удерживается более чем 2 или 3 сек., это вызывает 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 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 нажатие 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 курсора вверх, вниз.



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

В определенных состояниях клавиши CAPS SHIFT и SPACE действуют как BREAK, останавливая компьютер с выдачей сообщения 'D' или 'L' это распознается: а) в конце выполняющегося 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 заполняется полностью, он сворачивается на одну строку, компьютер останавливается с выдачей сообщения: "SCROLL ? ". Ответ: N, SPACE или STOP вызовет останов программы с выдачей сообщения: " D BREAK - CONT REPEATS ", нажатие любой другой из клавиш разрешает свертку экрана.

Нижняя часть экрана используется для ввода команд, строк программы и вводимых данных, а также для вывода сообщений системы.



---

## Глава 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 две строки программы вычисления суммы двух чисел:

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

так, чтобы на экране появилось:

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

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 называется программным курсором, а строка на которую он указывает, называется текущей, это обычно последняя введенная строка, но Вы имеете возможность переместить программный курсор выше или



ниже, используя соответствующие клавиши управления курсором: - установите его в строку 20. Когда Вы нажмете клавишу EDIT, то в нижней части экрана появится копия текущей строки, в нашем случае копия строки 20. Нажмите и удерживайте клавишу перемещения курсора вправо до тех пор, пока курсор [L] не переместится в конец строки и затем введите: " +B " (без нажатия ENTER), строка в нижней части экрана примет вид:

```
20 PRINT A+B
```

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

```
10 LET A=10  
15 LET B=15  
20 PRINT A+B
```

[K]

запустите программу, нажав RUN и ENTER и получите на экране сумму. Выполните программу 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 перенесена 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 - служить именем оператора при ссылке к нему из другого места программы (в GO TO N).

Команда LIST без операндов выдаёт 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 градусы 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 возникнет вопрос, что же делает строка 10? Компьютер игнорирует эту строку. Это комментарии(REMARK или REMINDER). Все что следует после REMARK 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 нажмете ENTER, компьютер выведет результат и снова перейдет в ожидание следующего числа, этот переход обеспечивается в строке 60 в операторе GO TO 40



Если на запрос очередного числа ответить STOP, то компьютер остановится с выдачей сообщения : 'H STOP IN INPUT 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 'O OK') номер последней выполняемой строки и продолжает выполнение именно с этой строки. Просмотрите внимательно оператор PRINT в строке 50; запятая в нем очень важна. Запятые в операторе PRINT используются для указания того, что вывод, 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 значение 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 еще несколько программных строк:

```
100 REM THIS POLITE PROGRAM RE-  
MEMBER 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 последнюю программу, надо ввести команду:

```
RUN 100
```



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

N\$

(удалив кавычки) это сделает оператор INPUT в строке 110 подобным оператору

LET N\$=N\$

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

Действие команды 'RUN 100' подобно действию оператора 'GO TO', но имеются и отличия. 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 программы BREAK, которая остановит вычисления 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 произошел останов. Ввод команды CONTINUE после сообщения 'L BREAK INTO PROGRAM' продолжит выполнение со следующего оператора.

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

N\$

(удалив 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 OK,0:1', а затем введете CONTINUE, то увидите, что программа завершится нормально. Как уже отмечалось, сообщение 'L BREAK INTO PROGRAM' особое, так как выдача после него CONTINUE не вызывает повторение команды, вызвавшей останов.

Все приведенные в этой главе утверждения PRINT, LET, INPUT, RUN, LIST, GO TO, CONTINUE, NEW и REM могут быть использованы либо как операторы в программе, либо как команды, хотя RUN, LIST, CONTINUE и REM 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 вычисления. Оператор, реализующий это имеет форму:

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

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

```
10 REM GUESS THE NUMBER (уга-  
    дывание числа)  
20 INPUT A:CLS  
30 INPUT "GUESS THE NUMBER",B  
    (угадайте число)  
40 IF A=B THEN PRINT "THIS IS COR-  
    RECT": 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 IF оператор 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 до 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, выражения  $1 < 2$ ,  $-2 < 1$ ,  $-3 < 1$  истинны, а выражения  $1 < 0$ ,  $0 < 2$  ложны. Строка программы 40 сравнивает числа 'A' и 'B'. и, если они равны, завершает работу, выполняя команду STOP. При этом будет выдано сообщение '9 STOP, STATEMENT, 40:3', показывающее, что команда STOP была выдана в 3-ем операторе в 40-й строке.

Знаки условия набирают на клавиатуре следующим образом:

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



---

# Глава 4

---

## Циклы

Краткое содержание: *FOR, NEXT, TO, STEP*

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

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

Получилась большая и не очень оптимальная программа. Можно решить эту задачу более рационально, если ввести счетчик и оператор *GO TO*

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

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



Наша программа при использовании этих операторов будет выглядеть так:

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

Здесь 'C' управляющая переменная цикла должна иметь имя в одну букву. 'C' последовательно принимает значения 1, 2, 3, 4, 5 (предел - конечное значение 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 приращ.'
```

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 TO 'C'
```

последовательно принимает 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 несколько раз используемых в различных местах программы. Такие части оформляются как подпрограммы, которые могут вызываться в любом месте программы.

Для этого используются операторы *GO SUB* (*GO TO SUBROUTINE*) и *RETURN* в форме: *GO SUB N* где 'N' номер первой строки в подпрограмме. Этот оператор подобен *GO TO* с той разницей, что при использовании *GO SUB* компьютер запоминает следующий после *GO SUB* оператор, которому и передается управление после выполнения программы. Делается это посредством помещения номера оператора (адреса возврата) в специальную область памяти, называемую *GO SUB* стек. *RETURN* выбирает верхний адрес возврата из *GO SUB* стека и продолжает выполнение программы с оператора, следующего после оператора с этим номером.

Приведем пример :

```
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 U=1 TO X
520 LET S=S+U
530 NEXT U
540 RETURN
```

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



---

# Глава 6

---

## Операторы READ, DATA, RESTORE

Краткое содержание: *READ, DATA, RESTORE*

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

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

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

Каждый оператор DATA - это список значения - числовых или строчных - разделенных запятыми. Вы можете вводить их где угодно в программе, так как компьютер игнорирует их, за исключением тех случаев, когда их использует оператор READ. Вы должны представлять выражения для всех операторов DATA в программе, что позволит вам вводить совместно формы одной длинны списка выражений (списка DATA). Сначала компьютер выбирает первое значение из списка DATA для величины из операторов READ, в следующий раз он берет второе значение из списка DATA и, таким образом, выбираемые последовательно операторы READ обрабатываются с использованием списка DATA (если оказалось, что список просмотрен до конца, то возникает ошибка). Заметьте для себя, что является бесполезным введение оператора DATA в виде управляющего оператора, так



как оператор READ не обнаружит его. Оператор DATA выполняется только в программе.

Можно посмотреть, как эти требования выполняются в приведенной программе: строка 10 дает указание компьютеру читать три значения данных и присвоить их переменным A, B, C.

Строка 20 говорит о том, что надо вывести (PRINT) три значения. В строке 30 оператор DATA задает значения для A, B, C.

Строка 40 заканчивает программу. Информация об операторе DATA может быть частью FOR... NEXT цикла. Например:

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

Когда выполняется эта программа, то можно увидеть, как READ оператор перебирает весь список DATA. Оператор DATA может содержать также и строчные переменные.

Например:

```
10 READ D$
20 PRINT "THE DATE IS", D$
30 DATA "JONE IST, 1982"
40 STOP
```

Это простой способ получения выражения 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 оператора DATA, в строке, указанной в операторе RESTORE. Вообще-то, вы можете не указывать номер строки в операторе RESTORE, и, в этом случае, указатель данных становится на первый оператор в программе оператор DATA.

Попробуйте выполнить такую программу:

```
15 DATA G
25 READ X, Y
32 DATA -7, 2.9
```



```
35 PRINT X+Y
45 READ P, Q
55 PRINT P*Q
65 DATA 149.21
```

Вы можете проверить, что эта программа получает результат -1 для суммы  $X+Y$  и 432.709 для произведения  $P*Q$ .

Дополните эту программу строкой

```
40 RESTORE 32
```

Теперь в результате работы этой программы сумма будет равна -1, а произведение -20.3. Это происходит из-за того, что оператор RESTORE сбрасывает указатель данных на начало оператора DATA в строке 32.



---

# Глава 7

---

## Арифметические операции

Краткое содержание: *выражения, операции +, -, \*, / условные выражения, имена переменных*

Выражение состоит из переменных, констант, знаков операций и круглых скобок. Вы видели несколько примеров, в которых ZX SPECTRUM может оперировать числами. Также можно выполнять четыре арифметические операции: +, -, \*, /.

Пример:

LET TAX= SUM\*15/100

Отсюда видно, что вычисления могут быть комбинированными. Комбинации такого типа, как SUM \* 15/100 называются выражениями. Выражение- это самый короткий путь для указания компьютеру на то, что вычисления надо делать одно за другим. В нашем примере выражение SUM\*15/100 указывает: возьми значение переменной с именем "SUM", умножь его на 15 и затем раздели на 100. Если вы не можете еще этого сделать, мы рекомендуем 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 выполняются, будут выполняться сложение и вычитание по порядку также слева направо.

Для задания приоритета в компьютере ZX SPECTRUM используются числа от 1 до 16. Например, операция '\*', и '/' имеет приоритет 8, а '+' и '-' -6. Этот порядок вычисления является жестким, но его можно изменить при помощи скобок. Выражение в скобках вычисляется первым, а затем



подставляется в общее выражение как одно число. В сомнительных случаях рекомендуется ставить скобки. Лишние скобки не влияют на выполнение программы.

Вы можете использовать операцию сложения "+" для сцепления строк (конкатенации) 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
MAKING A MISTAKE
NOW WE ARE SIX
NOWWEARESIX
```

Последние два имени указывают NOWWEARESIX на одну и ту же переменную

Примеры недопустимых имен переменных:

```
2001 (начинается с цифры)
3 BEARS (начинается с цифры)
m*a*c*x - знак '*' не буква и не цифра
FOTHERINGTON-TOMAS (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 2.34E18
```

Помните, что оператор PRINT дает лишь 8 значащих цифр числа. Попробуйте выполнить еще:

```
PRINT 4294967296, 4294967295-429E7
```

и вы увидите, что компьютер может воспринять только число 4294967300 и 4294967300.



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

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

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

Еще один более наглядный пример:

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

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

Числа 5E9+1 и 5E9+2 для компьютера равны. Наибольшее целое которое может воспринять компьютер, равно 2\*\*32-1 (4 294 957 295).

Строка"" без единого символа называется пустой или нуль строкой, не путайте ее с пробелом. Наберите:

```
PRINT" HAVE YOU FINISHED"FINNE-  
GANS WAKE" YET?"
```

Когда вы нажмете клавишу ENTER, вы получите мерцающий знак вопроса, указывающий ошибочное место в строке. Когда при интерпретации всей строки компьютер найдет двойную кавычку, открывающую " FINNEGANS WAKE", то сочтет ее закрывающей кавычкой для строки " HAVE YOU FINISHED" и затем не сможет вывести "FINNEGANS WAKE". Здесь надо помнить специальное правило: если вы хотите вывести кавычки внутри строки, они должны удваиваться. например: PRINT" HAVE YOU FINISHED"" FINNEGANS WAKE""YET?" В данном случае будет выведена фраза, обрамленная двойными кавычками " FINNEGANS WAKE".



# Глава 8

## Строки символов

Краткое содержание: сечения, использование *TO*

Примечание: эти операции отсутствуют в стандартном бейсике

Пусть имеется строка символов, тогда ее подстрокой будет некоторая последовательность символов из этой строки, так "STRING" является подстрокой от "BIGGEN STRING", а "B STRING" и "BIG STRING" не является. 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) = "BCDEF",

если опущено 'начало', то по умолчанию подразумевается 1, если 'конец', то 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" (*TO* 6) =  
"ABCDEF"

последнее выражение можно было бы записать и так

"ABCDEF" ( ),

что тоже верно, можно опускать и слово *TO*:

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

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

"ABCDEF" (5 *TO* 7)

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



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

```
'ABCDEF' (8 TO 7) = ""
```

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

'начало' и 'конец' не могут быть отрицательными, иначе выдается сообщение 'B INTEGER OUT OF RANGE', Следующая программа иллюстрирует эти правила:

```
10 LET A$ = "ABCDEF"
```

```
20 FOR N=1 TO 6
```

```
30 PRINT A$ (N TO 6)
```

```
40 NEXT N
```

```
50 STOP
```

Можно также присваивать значение подстроке, попробуйте:

```
10 LET A$ = "I AM THE ZX SPECTRUM"
```

```
20 PRINT A$
```

```
30 LET A$ (5 TO 8) = "*****"
```

```
40 PRINT A$
```

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

Если вы теперь выполните:

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

вы увидите что будут выведены дополнительные пробелы, так '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 вход 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, функции будут вычислены перед выполнением любых операций, однако вы можете изменить этот порядок, применяя скобки.

Функция *STR\$* преобразует число в символьный вид, подобно 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, т.к. *STR \$ 100.0000 = " 100"*

Функция *VAL* обратится в функцию *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 понадобится учетверение или даже увосьмирение.

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

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

Сделайте

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

и затем выведите все следующие значения

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

Некоторые из них работают, а некоторые нет, проанализируйте эти ответы.

Функция SGN - это так называемая математическая функция сигнум(знак), и аргумент и результат его числовые, результат равен: 1, если аргумент положителен;

0, если аргумент равен 0;

-1, если аргумент отрицателен. Функция ABS преобразует аргумент в положительное число:

```
ABS -3,2 = ABS 3,2 = 3,2
```

Функция INT (от 'INTEGER PART' - целая часть) преобразует дробное число к целому отбрасыванием дробной части:

```
INT 3,9 = 3
```

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

```
INT -3,9 = - 4
```



Функция SQR вычисляет корень квадратный от числа, например:

$$\text{SQR } 4 = 2$$

$$\text{SQR } 0.25 = 0.5$$

$$\text{SQR } 2 = 1.4142136 (\text{приблизительно})$$

Если аргумент отрицательный, то выдается сообщение:

'A INVALID ARGUMENT'.

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

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

```
10 DEF FN S(X) = X*X:
```

```
REM THE SQUARE OF X DEF вводится в  
соответствующем режиме (SIMBOL  
SHIFT и1), теперь функция может  
использоваться в программе.
```

```
20 PRINT FN S(2)
```

```
30 PRINT FN S(3+4)
```

```
40 PRINT 1+ INT FN S(LEN " CHICREN  
"/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:

```
DEF FN R(X)= INT(X+0.5): REM CIVES X  
RONDED TO THE NEAREST INTEGER
```

и можете затем попробовать ввести :

```
FN R(2.9)=3
```

```
FN R(2.4)=2
```

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

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

Введите и выполните следующее:

```
10 LET X=0: LET Y=0: LET A=10
```

```
20 DEF FN P(X,Y)=A+X*Y
```

```
30 DEF FN Q ( ) =A+X*Y
```

```
40 PRINT FN P(2.3), FN Q( )
```

Есть одна тонкость в этой программе, во-первых, функция FN Q не использует аргументов, но скобки при этом должны обязательно использоваться. Во-вторых, операторы

DEF не выполнимые, компьютер после выполнения строки 10, просто переходит к выполнению строки 40, помните, что DEF может быть только оператором, но не командой. В третьих, 'X' и 'Y' - имена целых переменных в программе и в то же время имена аргументов в функции FN P

Функция FN P использует в вычислении результата значения аргументов 'X', 'Y' и переменной 'A', не являющейся аргументом. Так, когда вычисляется FN P(2,3), значение 'A' равно 10, как определено в программе, а значения 'X' и 'Y' соответственно 2 и 3, так как они аргументы и результат будет  $10+2*3=18$ .

При вычислении FN Q() участвуют только переменные программы, так как аргументов нет, и ответ в этом случае будет  $10+0*0$ . Теперь изменим строку 20 на 20 DEF FN P(X,Y)=FNQ() В этом случае FN P(2,3) будет возвращать значение 10.

Некоторые версии языка бейсик имеют функции

LEFT\$, 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: \*\*, PI, EXP, LN, SIN, COS, TAN, ASN, ACN, ATN

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

### \*\* и EXP

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

$$2^{**1}=2$$

$$2^{**2}=2*2=4 \text{ (два в квадрате)}$$

$$2^{**3}=2*2*2=8 \text{ (два в кубе)}$$

Таким образом, запись  $A^{**B}$  означает: умножь 'A' само на себя 'B' раз, но это предполагает, что 'B' положительное целое число.

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

$$A^{**B+C}=A^{**B}*A^{**C}$$

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



$$A^{**}0=1$$

$$A^{**}(-B)=1/(A^{**}B)$$

$$A^{**}(1/B)=\text{корню } B\text{-ой степени из } A$$

$$A^{**}(B*C)=(A^{**}B)^{**}C$$

Полезно помнить, что:

$$A^{**}(-1)=1/A$$

$$A^{**}(1/2)=\text{SQR } A$$

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 А, кстати, не должно быть отрицательным.

Другой типичный пример использования этой операции, это вычисление дохода. Предположим, что вы вложили часть своих денег в общественное строительство, которое приносит вам 15% годовых, после года вы будете иметь уже не точно 100% от того, что имели в начале, а плюс 15% дохода, что составит 115%. Для вычисления другим способом, вы умножаете вашу сумму денег на 1,15 и получаете тот же результат. В конце следующего года вы снова получите прибыль, что в сумме составит

$$1.15*1.15=1.15^{**}2=1.3225$$

от вашей первоначальной суммы. В итоге после Y лет вы будете иметь в  $1.15^{**}Y$  раз больше денег. Выполним:

```
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 умножения этого числа самого на себя, называется экспоненциальным законом. Предположим вы написали:

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

Здесь А определено в операторе LET, его значение передается для вычисления степени.

Имеется определенное значение А, которое делает функцию FN А иллюстрирующей специальную математическую функцию, это значение называется 'е'.



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

$$\text{EXP } X = e^{**} X$$

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

PRINT EXP 1

так как  $\text{EXP } 1 = e^{**} 1 = e$ , 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  $A^{**}\text{LOGA} X = X$  так же верно как и  $\text{LOGA}(A^{**}X) = X$ ).

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:

$$\text{LOGA } X = \text{LN } X / \text{LN } A$$

PI

Допустим имеется некоторый круг, вы можете найти его периметр (длину окружности), умножив его диаметр на число, называемое PI. Подобно 'e', PI представляется бесконечной десятичной дробью, начало 3,141592653569...

Слово 'PI' в ZX SPECTRUM обозначает это число.

Выполните, например:

PRINT PI

SIN, COS, TAN, ASN, ACN, ATN

Тригонометрические функции измеряют те случаи, когда точка перемещается по 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 косинус отрицательный, а если точка находится под осью X, то отрицательный синус. Необходимо также помнить, что:

$$\text{SIN}(A + 2 * \text{PI}) = \text{SIN } A$$

$$\cos(A+2\pi)=\cos A$$

Имеются и другие тригонометрические функции:

TAN- тангенс; ASN- арксинус;

ACS- арккосинус; ATN- арктангенс.

Помните, в ZX SPECTRUM тригонометрические функции вычисляют в радианах. Для перевода из градусов в радианы необходимо число разделить на 180 и умножить его на  $\pi$ , а для обратного преобразования необходимо разделить на  $\pi$  и умножить на 180.



# Глава 11

## Случайные числа

Краткое содержание: *RANDOMIZE, RND.*

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

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

```
10 PRINT RND
```

```
20 GO TO 10
```

вы увидите как меняется результат.

Фактически RND не абсолютно случайное число, а выбирается в определенной последовательности длиной в 65536 чисел, поэтому обычно говорят что RND - псевдослучайное число.

Для получения случайного числа в интервале отличным от 0, можно использовать выражение, например:

```
1,3+0,7*RND
```

даст интервал от 1,3 до 2.

Для получения случайных целых чисел используйте функцию

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

где N любое число например:

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

даст 1,2,3,4,5,6. пусть имеется программа:

```
10 REM DICE THROWING PROGRAM  
   (выбрасывание кости)
```

```
20 CLS
```

```
30 FOR N=1 TO 2
```

```
40 PRINT 1+INT(RND*6); " ";
```

```
50 NEXT N
```

```
60 INPUT A$: GO TO 20
```

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

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

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

После каждого выполнения RANDOMIZE 1 случайная последовательность будет начинаться с числа 0,0022735596. В утверждении RANDOMIZE вы можете использовать любые числа в интервале от 1 до 65635. 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 дополнение большинство версий бейсика использует 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 COLN=INT(RND*2)
30 IF COLN=0 THEN LET
  HEADS=HEADS+1
40 IF COLN=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 из чисел, каким-то образом описывающих 10 человек. Для записи их в память компьютера, вы должны будете занести переменную на каждого человека. Это не удобно, так как приходится обращаться к данным, называя каждый раз новую переменную, например BLOGGS1, BLOGGS2 и т.д. до BLOGGS 10. Как это неудобно, вы можете убедиться из программы:

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

Имеется специальный аппарат для подобного случая, это применение массивов, переменные в массиве являются его элементами, обладают общим именем и различаются только номером, записываемым после имени (индексом).

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

Элементы массивов называют индексруемыми переменными. Перед использованием массива необходимо зарезервировать под него память. Это делается в операторе DIM (от английского DIMENSION). В нашем случае это будет оператор DIM B(10), который определяет массив с именем B и размерностью 10 (т.е 10 индексруемых 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, 5
```

Можно также объявлять массивы с более чем одной размерностью.

Например в двумерном массиве первый индекс можно сравнить с номером строки, а второй с позицией в строке. Такой массив описывает страницу, если ввести третье измерение для номера страницы, то массив будет описывать книгу в виде: (номер страницы, номер строки, номер столбца) Объявим двумерный массив 'C' с размерностью 3 и 5:

`DIM C(3,5)`

что даст  $3 \times 5 = 15$  индексируемых переменных:

`C(1,1), C(1,2).....C(1,5)`

`C(2,1), C(2,2).....C(2,5)`

`C(3,1), C(3,2).....C(3,5)`

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 и добавлением до полной длины пробелами. Имя строкового массива образуется добавлением справа к имени специального символа \$ (перечеркнутой буквы S).

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 сечения массивов:

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

Помните, что в строковых массивах все строки имеют иную 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: логическое выражение,  $<$ ,  $>$ ,  $<=$ ,  $>=$ ,  $=$ ,  $<>$ , AND, OR, NOT.

Результатом вычисления любого арифметического выражения является число. Однако, возможны логические выражения. Любое логическое выражение может быть истинным или ложным. Понятия "истина" или "ложь" считаются результатом логического выражения.

В логическом выражении кроме знаков арифметических операций используются логические операции, а для сравнения значений величин - знаки отношений.

К знакам отношений относятся:  $<$  (меньше),  $>$  (больше),  $<=$  (меньше или равно),  $>=$  (больше или равно),  $=$  (равно),  $<>$  (не равно).

Если заданное отношение между операндами выполняется, то результат операции считается истинным, если не выполняется, то ложным.

В языке имеются специальные логические операции AND (и - логическое умножение), OR (или - логическое сложение), и NOT (не - логическое отрицание).

В логических выражениях принят следующий приоритет выполнения операций:

- 1)  $*$ ,  $/$
- 2)  $+$ ,  $-$
- 3)  $<$ ,  $>$ ,  $<=$ ,  $>=$ ,  $=$ ,  $<>$
- 4) NOT
- 5) AND
- 6) OR

Для изменения порядка выполнения операций применяют круглые скобки.

Примеры логических выражений:

| Выражение | Результат                                  |
|-----------|--------------------------------------------|
| $10 > 3$  | Истинный                                   |
| $12 < 7$  | Ложный                                     |
| $A = B$   | Истинный, если значение A равно значению B |

Ложный, если значение А не равно значению В

$D \leq (C + 17)$  Истинный, если значение D меньше или равно значению  $C+17$ ;

Ложный, если значение D больше значения  $C+17$ .

Логическое сложение - даст истинный результат, если один из операндов, либо оба являются истинными.

Логическое умножение - даст истинный результат только в том случае, если оба операнда истинные. Иначе результат операции будет ложным.

Операция отрицания - даст истинный результат, если операнд был ложным, и, наоборот, даст ложный результат, если операнд был истинным.

Результаты логических операций сведены в таблицу, где первый операнд обозначен буквой А, второй операнд - буквой В.

| А       | В       | А OR В  | А AND В | NOT А   |
|---------|---------|---------|---------|---------|
| истинно | истинно | истинно | истинно | ложно   |
| истинно | ложно   | истинно | ложно   | ложно   |
| ложно   | истинно | истинно | ложно   | истинно |
| ложно   | ложно   | ложно   | ложно   | истинно |

Числовой результат логической операции равен 1, если выражение истинно, и равен 0, если выражение ложно.

В операторе 'IF условие THEN ....', само условие может быть числовым выражением. Если его значение после вычисления равно 0, то считается, что ложь, если другое значение (включая и 1), то считается что это истина.

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

$X\$ \text{ AND } Y$  имеет значение  $X\$$ , если  $Y=1$ , и "", если  $Y=0$  где "" - пустая строка.

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

```
10 INPUT " TYPE IN TWO STRINGS",  
   A$,B$  
20 IF A$>B$ THEN LET C$=A$: LET  
   A$=B$: LET B$=C$  
30 PRINT A$, " ;(" AND A$)+(" 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 кодировку и наоборот служат две функции CODE и CHR\$.

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

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

Следующая программа выводит весь отображаемый символьный набор:

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

Все эти символы (кроме знака фунта и 'с' в кружочке) образуют код ASCII (AMERICAN STANDART CODES FOR INFORMATION INTERCHANGE).

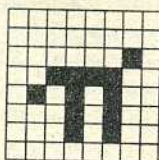
Следующие символы не входят в ASCII, но используются в ZX SPECTRUM, первые из них это 15 черно-белых значков, называемых графическими символами и используемых для изображения рисунков. Их можно ввести с клавиатуры используя так называемый, графический режим, если вы нажмете GRAPHICS (CAPS SHIFT 9), то курсор изменяется 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 кла-

виша '0' функцию DELETE. После графических символов на клавиатуре располагаются символы алфавита от А до У. Графические значения этих клавиш могут определяться самим пользователем, а затем использоваться в графическом режиме. Определение графики этих клавишей проиллюстрируем на примере определения буквы греческого алфавита 'пи'.

1. Каждый символ представляется точками в матрице 8Х8, поэтому мы вначале 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 'Р', так, чтобы при нажатии клавиши в графическом режиме выдавался символ 'пи'



3. Запрограммируем это изображение, каждый определяемый пользователем символ запоминается в памяти восемью 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 символов 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, закрепленного 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 подробнее будем говорить в главе 24.

## ГРАФИЧЕСКИЕ СИМВОЛЫ

символ      код набора      символ      код набора

|                                                                                    |           |                                                                                    |             |
|------------------------------------------------------------------------------------|-----------|------------------------------------------------------------------------------------|-------------|
|  | 128 [G] 8 |  | 143 [G] SH8 |
|------------------------------------------------------------------------------------|-----------|------------------------------------------------------------------------------------|-------------|

|                                                                                     |           |                                                                                     |             |
|-------------------------------------------------------------------------------------|-----------|-------------------------------------------------------------------------------------|-------------|
|  | 129 [G] 1 |  | 142 [G] SH1 |
|-------------------------------------------------------------------------------------|-----------|-------------------------------------------------------------------------------------|-------------|

|                                                                                     |           |                                                                                     |             |
|-------------------------------------------------------------------------------------|-----------|-------------------------------------------------------------------------------------|-------------|
|  | 130 [G] 2 |  | 141 [G] SH2 |
|-------------------------------------------------------------------------------------|-----------|-------------------------------------------------------------------------------------|-------------|

|                                                                                     |           |                                                                                     |             |
|-------------------------------------------------------------------------------------|-----------|-------------------------------------------------------------------------------------|-------------|
|  | 131 [G] 3 |  | 140 [G] SH3 |
|-------------------------------------------------------------------------------------|-----------|-------------------------------------------------------------------------------------|-------------|

|                                                                                   |           |                                                                                   |             |
|-----------------------------------------------------------------------------------|-----------|-----------------------------------------------------------------------------------|-------------|
|   | 132 [G] 4 |   | 139 [G] SH4 |
|  | 133 [G] 5 |  | 138 [G] SH5 |
|  | 134 [G] 6 |  | 137 [G] SH6 |
|  | 135 [G] 7 |  | 136 [G] SH7 |

Вернемся к знакам. Мы еще не поговорили о не напечатанных первых 32 знаках с кодами от 0 до 32, это управляющие символы, они не отображаются. Вместо них на экране телевизора отображается знак '7'. 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 тот же 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 - это 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.

С 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 расположены после заглавных, так, что 'а' следует после 'Z'.

```
CHR$ 3+" ZOOLOGIKAL GARDENS"
```

```
CHR$ 8+"AARDYARK HURTING"
```

```
"(ARENTHETICAL REMARK)"
```

```
"100"
```

```
"129.95 INC,YAT"
```



```

"AASYOGEL"
"AARDYARK"
"PRINT"
"ZOO"
"LINTERPOLATIONJ"
"AARDYARK"
"AASYOGEL"
"ZOO"
"ZOOLOGU"

```

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 встретятся несовпадающие символы, либо пока одна из строк не кончится. Она и будет меньшей, в противном случае строки считаются равными.

Отношения =, <, >, <=, >=, <> применяются к строкам символов также как и к числам: знак '<' означает 'находится впереди в кодовой таблицы', а '>' - 'находится позади', так, что выражения

```

"AAMAN" < "AARDYARK"
"AARDYARK" > "AAMAN"

```

оба истины.

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 LETT C$=A$:LET
   A$=B$:LETB$=C$
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$: REM 6
  PIECES
20 FOR F=1 TO 7 :REM READ PIECE
  INTO 8 BUTES
30 READ A: POKE USR$ P$+1,A
40 NEXT F
50 NEXT N
100 REM BISHOP
110 DATA " B ", 0, D, BIN 00101000,
  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 ROCK
170 DATA "R ",0, BIN 01010100, B, C
180 DATA C,B,B,0
190 REM QUEEN
200 DATA " Q ",0, BIN 01010100, BIN
  00101000,D
210 DATA BIN 01101100,B,B,0
220 REM 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

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

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

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

PRINT AT 11,16;"\*"

выведет '\*' 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, номер строки и столбца, но их значения закрываются в скобки. Оператор SCREEN\$ сообщает вам какой символ находится на экране в указанной позиции. Выполнив: PRINT SCREEN\$ (11,16) мы получим '\*' выведенную предыдущим оператором.

В качестве возвращаемого значения могут использоваться алфавитно-цифровые символы, специальные символы, пробелы, линии, нарисованные с помощью операторов: PLOT, DRAW, CIRCLE, определяемые пользователем символы и графические символы, возвращаемые как пустая

строка. То же, когда функция OVER используется для получения комбинированных знаков.

### TAB столбец

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

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

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

выведет на экран оглавление для первой страницы книги. Рассмотрим пример, иллюстрирующий уменьшение по модулю 32 параметра TAB:

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

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

### Несколько замечаний:

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

2) нельзя использовать для вывода две нижние строки экрана т.к. они используются для получения оператором INPUT данных, последняя используемая строка -21.

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

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

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

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

и далее выполнить PRINT некоторое количество раз, или вариант с остановом вывода, для просмотра текста. Чтобы убедиться в этом выполним:

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



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

Оператор INPUT используется для ввода различных значений, например:

```
INPUT "NOW  
OLO 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 "HO OLD ARE YOU?" и "AGE" оба являются параметрами INPUT, однако существуют и некоторые отличия.

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

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

Пример:

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

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

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

Когда "THIS IS LINE 2" будет выведено, нижние строки станут сдвигаться вверх, освобождая место для нумерации .  
Выполним

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

Когда информация начнет смещаться в область действия операторов PRINT произойдет свертка экрана. Еще одним

параметром оператора INPUT является LINE, он не предназначен для ввода строчных переменных, рассмотрим пример: INPUT LINE A\$ Если ввести какую либо строчную переменную без строковых кавычек, то ее значение будет присвоено A\$. Заметим, что мы не можем использовать параметр LINE для числовых переменных.

Управляющие символы CHR\$22 и CHR\$23 выполняют функции подобные параметрам TAB и AT. Их преимущество состоит в том, что можно использовать имена переменных, а для TAB и AT это невозможно, эти управляющие символы обрабатываются как числа. Аналогом AT является управляющий символ CHR\$22, первое значение определяет строку, второе столбец: PRINT CHR\$22+CHR\$1+CHR\$C то же что и PRINT AT 1,C; как значения параметров рассматриваются только CHR\$1 и CHR\$C (CHR\$22 не учитывается).

Аналогом TAB является управляющий символ CHR\$23, значения задаваемых им параметров находятся в пределах от 0 до 65535

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

то же, что и

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

Вы можете использовать POKE для остановки компьютера, запрашивающего свертку, выполнив: POKE 23692,255 Компьютер станет сворачивать экран без запроса 255 раз, прежде чем запросит свертку.

Так, например, запустите:

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

и следите, сколько сверток сделал компьютер. Стартуем следующую программу проверяющую знание таблицы умножения

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



Можно несколько изменить программу, так чтобы не зная 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  
  A$= "TYPE IT PROPERLY, AS  
  NUMBER,": GO TO 30
```

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

---

# Глава 16

---

## Цвета

Краткое содержание: *INK,PAPER,FLASH,BRIGHT,IN  
VERSE,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 COL-
   OURED SPACES
50 NEXT C. NEXT N:   NEXT M
60 FOR M=0 TO 1: BRIGHT M: PAPER 7
70 FOR C=0 TO 3
80 INK C:PRINT C;" ";
90 NEXT C:PAPER 0
100 FOR C=4 TO 7
110 INK C: PRINT C;" ";
120 NEXT C:NEXT M
130 PAPER 7: INK 0: BRIGHT 0
```

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

Ниже дана кодировка цветов:

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



Для черно-белого телевизора этот ряд представляет собой последовательность перехода серых полутонов от черного до белого. Для использования цветов уясним строение графического экрана. Он состоит из 768 позиций (24 строки по 32 знакоместа), каждая из которых представляет из себя матрицу 5 на 8 пикселей. Вспомним

0-белая точка

1-черная точка

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

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

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

3) яркость: 0-обычная, 1-повышенная

4) мерцание: 0-постоянно, 1-мерцание.

Заметим, что для одного знакоместа в 64 пикселя мы не можем установить более одного цвета для фона и одного цвета для тона. Это же относится и к яркости, и к мерцанию. Цвет, яркость и мерцание задаются для знакоместа (а не для отдельного пикселя) и являются его атрибутами. Для изменения этих атрибутов предназначены операторы : INK, PAPER, BRIGHT, FLASH. Выполним: PAPER 5 на экране будет голубой фон (т.к. 5- код голубого цвета). Формат операторов:

PAPER число от 0 до 7 INK число от 0 до 7 BRIGHT число 0 или 1 (0-выкл. 1-вкл.) FLASH число 0 или 1 (0-выкл. 1-вкл.) Отметим что использование чисел больших, чем указывалось выше, допустимо, но дает другой эффект. К примеру '8' может использоваться во всех четырех операторах как средство, позволяющее определить значение ранее установленных атрибутов. Так PAPER 8 не изменит цвета фона (т.к. такого цвета нет), а поможет выяснить значение предыдущего PAPER. Операторы INK 8, BRIGHT 8, FLASH 8 выдадут значения этих атрибутов. "9" может использоваться только для INK и PAPER, как средство "контраста". Цвета INK и PAPER, которые вы используете, должны быть контрастны друг другу. Так к белому цвету подходят темные тона: черный, синий, красный, фиолетовый; к черному цве-

т/ подходят светлые тона: зеленый, голубой, желтый, белый.

Выполним :

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

Можно запустить программу выдающую на экран дисплея цветные полосы:

```
INK 9:PAPER 6:PRINT AT 0,0::FOR N=1  
TO 1000:  
PRINT N:: NEXT N
```

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

```
10 BORDER 0:PAPER 0: INK 7: CLS  
20 FOR A=1 TO 6  
30 PRINT TAB 6: INK 1;" 00...00":REM 18  
INK SQUARES  
40 NEXT A  
50 LET DATALINE=200  
60 GO SUB 1000  
70 LET DATALINE=210  
80 GO SUB 1000  
90 STOP  
200 DATA 2,3,7,5,4  
210 DATA 2,2,6,4,4  
1000 FOR A=1 TO 6  
1010 RESTORE DATALINE  
1020 FOR B=1 TO 5  
1030 READ C: PRINT INK C: "0...0":: REM  
6 INK SQUARES  
1040 NEXT B: PRINT :NEXT A  
1050 RETURN
```



Существует функция ATTR, позволяющая определить какие атрибуты были заданы для позиций 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 используется значения параметров '0' и '1'. Если вы дадите: INVERSE 1, то выводимый символ изменит свою обычную форму (вывод будет осуществляться в негативном изображении). В обычном виде мы пишем черным по белому, в инверсном- белым по черному. Оператор : OVER 1 устанавливает режим расширенного вывода. В обычном режиме при выводе символа на знакоместо, там стирается все выведенное ранее, при расширенном выводе можно накладывать символы друг на друга, это позволяет выводить составные символы, например стилизованные тексты. 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 возвращает на одну позицию.

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

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

только 'X' будет выведен на желтый фон.

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 FLASH 1; INK 1; "WHAT IS YOUR
NUMBER ?";N
```

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

CHR\$16 соответствует INK  
CHR\$17---//--- PAPER  
CHR\$18---//--- FLASH \*  
CHR\$19---//--- BRIGHT  
CHR\$20---//--- INVERSE  
CHR\$21---//--- OVER

Так

PRINT CHR\$16+CHR\$9;

то же, что и

PRINT INK 9;

можно пользоваться либо управляющими символами, либо операторами. Их можно ставить как после номера строки, так и в конце строки. Для удобства можно пользоваться в расширенном режиме [E] цифрами. Цифры от 0 до 7 устанавливают цвет INK, если CAPS SHIFT нажата, и цвет PAPER, если не нажата. Если нажать цифру в E-режиме, то будут выведены CHR\$17 и CHR\$ код цвета, если в это время была нажата CAPS SHIFT, то будут выведены: CHR\$16 и CHR\$ Код цвета. Если вы захотите уничтожить вводимое, нажмите DELETE два раза, после первого раза на экране высветится знак вопроса, или что-нибудь еще, не пугайтесь и нажмите DELETE еще раз. Управление курсором тоже не работает обычным способом до тех пор пока курсор не выйдет к предыдущему управляющему символу.

Действие в расширенном режиме [E];

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

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

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

CAPS SHIFT с 9 дает CHR\$18 и CHR\$1 -мерцающее

в [L]- режиме:

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

CAPS SHIFT с 4 дает CHR\$20 и CHR\$1- инверсный (негативный) вывод

Функция ATTR имеет следующий формат:

ATTR ('строка', 'столбец') значения двух параметров функции подобно 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 ярко-



сти, желтый фон, синий тон  $128+0+(8*6)+1=177$  Проверим это, выполнив:

```
PRINT AT 0,0;FLASH 1;PAPER 6; INK 1;
```

```
" ";
```

```
ATTR(0,0)
```

Упражнения:

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

здесь "/" перечеркивает "B". Этим способом можно вывести слова 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.

4) введём вручную, или с помощью оператора LOAD программу для ввода шахматных фигур из главы 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 COL-
    OURS OF WHITE AND BLA 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$(8)="RNBQKBNR":REMBIG
    LINE 210
300 REM DISPLAU BOARD
310 FOR N=1 TO 8:FOR M=1 TO 8
320 LET BC = CODE " " THEN GO TO
    350:REM SPACE
330 IF BC>CODE "Z" THEN INK PB:LET
    BC=BC-32:REM LO CASE FOR BLACK
340 LET BC=BC+79:REM CONVERT TO
    GRAPHICS
350 PRINT AT 5+N,9+M;CHR$BC
360 NEXTM:NEXT N
400 PAPER 7: INK 0

```



# 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  $22 \times 32 = 704$  символьные позиции. Как вы уже поняли из 16 главы, каждая символьная позиция представляется квадратом  $8 \times 8$  точек, называемых пикселями.

Пиксель задается двумя числами, его координатами, первое задает координату X, т.е. удаление (в пикселях) до левой границы экрана), второе задает координату Y, удаление от нижней границы экрана. Координаты записываются в скобках, так (0,0), (255,0), (0,175) и (255,175) задают соответственно нижний левый, нижний правый, верхний левый и верхний правый углы экрана.

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

Например программа:

```
10 PLOT INT(RND*256),INT(RND*176)
20 INPUT A$
30 GOTO 10
```

будет высвечивать некоторый случайный пиксель при каждом нажатии ENTER.

Есть и более интересные программы, например следующая программа вычерчивает график функции  $\sin X$ , для X в интервале от 0 до  $2 \times \pi$ :

```
10 FOR N=0 TO 255
20 PLOT N,86+80*SIN(N/128*PI)
30 NEXT N
```

или программа:

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

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

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

Пользуйтесь диаграммой из главы 15. Помощь при построении изображений вам могут оказать операторы DRAW и CIRCLE. Оператор DRAW чертит линию, заданную в форме DRAW X,Y.

Началом линии является пиксель, на котором завершился один из предыдущих операторов PLOT, DRAW или CIRCLE (этот пиксель называется текущей PLOT- позицией. Операторы RUN, CLEAR, CLS, и NEW устанавливают ее в левый нижний угол экрана). Таким образом оператор DRAW задает длину и направление вычерчивания линии, но не ее 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 ввиду, что цвет устанавливается для целой символьной позиции и не может быть задан для отдельного пикселя. Следующая программа демонстрирует это:

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

Вы можете использовать в операторах PLOT и DRAW управляющие символы PAPER, INK, FLASH, BRIGHT, INVERSE и OVER так же, как и в операторах PRINT и INPUT. Управляющие символы записываются между клю-



чевым словом координатами и оканчиваются запятой или точкой с запятой (смотри строку 50).

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

DRAW X,Y,A если 'A' положительно, то дуга вычерчивается влево, а если отрицательно, то вправо. При 'A' равном  $2\pi$  вычерчивается полная окружность, например:

```
10 PLOT 100,100: DRAW 50,50, $\pi$ 
```

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

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

CIRCLE X,Y, радиус. Как и в операторах PLOT и DRAW вы можете указать в этом операторе различные цвета. Функция POINT возвращает характеристики цвета заданного пикселя, например строка программы

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

выведет:

```
PAPER 7:INK0
```

Допускается также задавать управляющие символы INVERSE и в операторе PLOT, по умолчанию предполагаются равными 0 (отключено), но вы можете задать и 1, при этом: PLOT INVERSE 1 - устанавливает для заданного пикселя цвет фона PLOT OVER 1 - изменяет цвет пикселя на противоположный, если был цвет тона, то становится цвет фона и наоборот. PLOT INVERSE 1; OVER 1; - сохраняет цвет пикселя без 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
```

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

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

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

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

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

```
10 PLOT 0,87:DRAW 255,0
20 PLOT 127,0:DRAW 0,175
30 INPUT S,B$
40 FOR F=0 TO 255
50 LET X=(F-128)*S/128:LET Y=VAL B$
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, 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 течении 'N' телевизионных кадров (50 кадров в секунду в Европе или 60 кадров в Америке). 'N' может быть вплоть до 65535, что составляет 22 минуты, если  $N=0$ , то это означает, что оператор *PAUSE* не имеет ограничения 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 FIST WE DRAW THE CLOCK  
   FASE  
20 FOR N = 1 TO 12  
30 PRINT AT 10-  
   10*COS(N/6*PI),16+10*SIN(N/6*PI);N  
40 NEXT N  
50 REM NOW WE START THE CLOCK  
60 FOR T=0 TO 200000:REM THIS THE  
   TIME IN SECONDS  
70 LET A=T/30*PI:REM A IS THE ANGLE  
   OF THE SECOND HAND IN RADIANS  
80 LET SX=80*SIN A:LET SY=80*COS A  
200 PLOT 128,88:DRAW OVER  
   1: SX,SY:REM DRAW SECOND HAND  
210 PAUSE 42  
220 PLOT 128,88:DRAW OVER  
   1: SX,SY:REM ERASE SECOND HAND  
400 NEXT T
```

эти часы остановятся, проработав приблизительно 55,5 часов, что задается в операторе с номером 60. Оператор 210 производит отсчет времени. Казалось бы здесь должен быть оператор PAUSE 50 (Европа), для точного отсчета одной секунды, но тогда бы мы не учли время, затраченное на выполнение остальных операторов программы. Рассматриваемый вариант часов обеспечивает 2%-ю точность или иными словами уход на полчаса в день.

Возможны и более точные способы измерения времени. Для этого можно использовать содержимое специальных областей памяти, в этом случае данные из памяти могут быть вызваны с помощью функции PEEK. Подробно это рассмотрено в главе 25, здесь же в качестве примера рассмотрим выражение:

```
(65536*PEEK 23674+256*PEEK 23673  
+PEEK 23672)/50
```

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

```
10 REM FIRST WE DRAW THE CLOCK  
   FASE  
20 FOR X=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 T0=INT((65536*PEEK  
   23674+256*PEEK 23673+PEEK  
   23672)/50):REM NUMBER OF SE-  
   KOND SINC START  
100 REM NOW WE START THE CLOCK  
110 LET T1=FN T0  
120 LET A=T1/30*PI:REM A IS THE  
   ANGLE OF THE SECOND HAND IN  
   RADIANS  
130 LET SX=72*SIN A:LET SY=72*COS A  
140 PLOT 131,91:DRAW OVER 1:  
   SX,SY:REM DRAW HAND  
200 LET T=FN T0  
210 IF T<=T1 THEN GOTO 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:GOTO 120

```

Эти часы обеспечивают точность 0,001 % или уход на 10 секунд в день, однако, это возможно при условии, что вы не использовали оператор ВЕЕР, ввод/вывод на магнитофон и принтер. Все эти операции увеличивают погрешность.

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

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

$$(65536*0 + 256*255 + 255)/50 = 1310,7$$

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

Пока производится вычисление выражения, компьютер может оценить значение ячейки РЕЕК 23674 как 0 до завершения циклического переноса, в результате получим:

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

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 LANDER OF AND Y
20 DEF FN U ( ) = (65536*РЕЕК 23674 +
    256*РЕЕК 23673 + РЕЕК 23672)/50:
    REM TIME, MAY BU 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 компь-

ютер был включен. Например, надо установить 10 часов вечера, вы посчитали, что  $12*60*50=180000$  50-х долей секунды и значит  $180000 = 65536*27+256*119+64$  Для присвоения трем ячейкам значения 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 имеет эта клавиша в режиме маркера , или пустая строка.

Выполните программу которая использует эту функцию:

```
10 IF INKEY$= " " THEN GOTO 10
20 IF INKEY$= " " THEN GOTO 20
30 PRINT INKEY$;
40 GOTO 10
```

Помните, что функция INKEY\$ не будет подобно INPUT ждать Вас.

Если Вы не выполните ВВОД то считайте, что ваш шанс упущен.

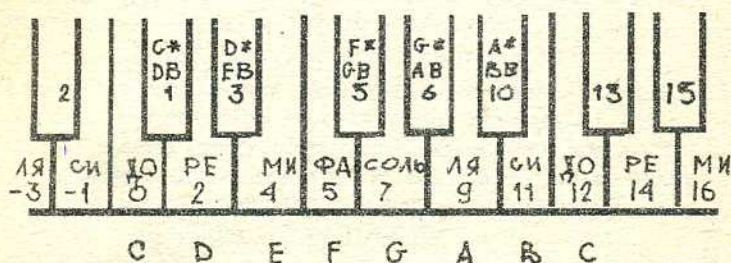


## Глава 19

## Программирование звуков

Краткое содержание: *BEER*

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:

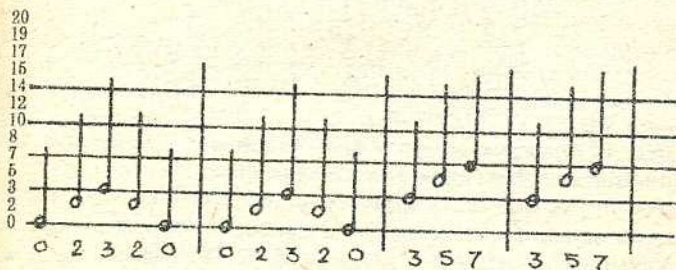


Для получения более высоких или более низких нот, вы должны прибавить или отнять 12 для каждой октавы вверх или вниз. Например:

```
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:BEEP1,0  
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 Моцарта, ту часть, когда гоблины 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 .5KEY+2: BEEP  
1,KEY+0

Теперь, при выполнении этой 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 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 изучать ее прямо на компьютере, например фрагмент программы:

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

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

Фрагмент программы:

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

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

```
20 BEEP.5,0:BEEP.5,2.039:BEEP  
.5,3.86:BEEP.5,4.9  
BEEP.5,7.02:BEEP.5,8.84:BEEP  
.5,10.88:BEEP.5,10.80: 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 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, подключая их к выходным разъемам 'MIG' 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 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
```

а 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 их 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 загружает в нее новые, считанные с магнитофона;

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

За каждой из этих команд следует ключевое слово - имя программы, определенное первоначально в команде *SAVE*. Пока компьютер ищет указанную программу, он выводит



имена всех программ, уже прочитанные с ленты. Имеется 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 вновь считана по команде LOAD (но не MERGE), она автоматически установится на строку с указанным номером и сама инициирует свое выполнение.

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

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

**SAVE STRING DATA ARRAY NAME()**

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

**LOAD STRING ARRAY NAME()**

нельзя использовать оператор MERGE.

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

**"CHARACTER ARRAY:"**

и далее имя этого массива.

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

**SAVE "PICTURE" CODE 16384,6912**

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

**LOAD "PICTURE" CODE**

После CODE можно указать числа:

**LOAD "PICTURE" CODE START,  
LENGTH**

LENGTH (длина) - определяет сколько данных (байтах) надо загрузить с ленты. Если длина больше, чем записано на ленту, то выдается сообщение "R TAPE LOADING

ERROR" (ошибка загрузки с ленты). Этот параметр можно опустить и тогда компьютер считает все данные, которые записаны на ленте.

START (начало) - указывает адрес, с которого должны загружаться данные и может быть отличным от адреса, указанного в SAVE, вы можете опустить этот параметр в команде LOAD. выражение CODE 16364,6912 можно заменить на SCREEN\$:

SAVE " PICTURE" SCREEN\$

и затем

LOAD " PICTURE" SCREEN\$

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

В заключение, везде, где указывается имя файла на ленте, используются только первые 10 символов.

Существует четыре 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 в виде

NUMBER 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 NUMBER

и начинает её выполнять

2. Байты.



## SAVE NAME CODE START LENGHT

записывает на ленту "LENGHT" 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").

Команда VERIFY проверяет (сравнивает) информацию в памяти и на ленте, может выдавать сообщение

R TAPE LOADING ERROR",

#### 1. Программа и переменные.

## VERIFY NAME

#### 2. Байты.

## VERIFY NAME CODE START LENGTH

Если данных в файле "NAME" более, чем указано в "LENGHT", то выдается сообщение об ошибке "R".

## VERIFY NAME CODE START

здесь осуществляется сравнение данных на ленте с данными в памяти, начиная с адреса, с которого записывается на ленту первый байт данных.

## VERIFY NAME SCREEN\$

или эквивалентно

## VERIFY NAME CODE 16384,6912

однако это будет проверка уже проверенного файла.

### 3. Массивы.

## VERRIFY NAME DATA LETTER ( )

## VERIFY NAME DATA LETTER\$ ( )

Команда LOAD загружает новые данные с ленты, стирая старые данные в памяти.

#### 1. Программа и переменные.

## LOAD NAME

может выдавать сообщение

" 4 OUT OF MEMORY",

если нет места для новой программы. В этом случае старая программа не уничтожается.

#### 2. Байты.

## LOAD NAME CODE START, LENGTH

если данных в файле "NAME" больше, чем указано в "LENGTH", то выдается сообщение R.

LOAD NAME CODE START

производит загрузку данных из "NAME" в память, начиная с адреса "START"

LOAD NAME CODE

загружает программу по адресу, с которого записывались данные на ленту в файл "NAME".

### 3. Массивы

LOAD NAME DATE LETTER0

или

LOAD NAME DATE LETTER\$0

уничтожает в памяти массив с именем "LETTER" или "LETTER\$", формирует новый массив и переписывает туда данные из файла "NAME", может выдать сообщение "4 OUT OF MEMORY" при нехватке памяти под массив, в этом случае старый массив не уничтожается.

Команда MERGE загружает новые данные с ленты, не уничтожая старые.

#### 1. Программа и переменные.

MERGE NAME

дописывает программу "NAME" к некоторой программе, находящейся в памяти, может выдать сообщение

"4 OUT OF MEMORY"

2. Байты. не поддерживается.

3. Массивы. не поддерживается.

Пример: записать на ленту информацию о 21-м определенном пользователем символе.

SAVE "CHESS" CODE USR "A", 21\*8

обратная загрузка

LOAD "CHESS" CODE

LOAD "CHESS" CODE USR "A"



---

# Глава 21

---

## Устройство печати

Краткое содержание: *LPRINT, LLIST, COPY.*

Эта глава описывает операторы бейсика, необходимые для работы с принтером ZX SPECTRUM. Два оператора *LPRINT* и *LLIST* подобны операторам *PRINT* и *LIST*, но с той разницей, что они работают не с телевизором, а с принтером. Попробуйте для примера выполнить следующую программу:

```
10 LPRINT "THIS PROGRAM."  
20 LLIST  
30 LPRINT " PRINTS OUT THE CHARAC-  
   TER SET"  
40 FOR N=32 TO 255  
50 LPRINT CHR$ N  
60 NEXT N
```

Оператор *COPY* позволяет распечатать экран телевизора. Например, по *LIST* текст программы будет выведен на экран, а затем по *COPY* его можно распечатать на принтере.

Вы всегда можете прекратить вывод на печать выдав  
**BREAK (CAPS SHIFT и SPACE).**

Если вы задали операторы управления принтером без подключения реального устройства, то вывода просто не будет и выполнение программы продолжится до следующего оператора.

Теперь попробуйте выполнить следующую программу:

```
10 FOR N=31 TO 0 STEP -1  
20 PRINT AT 31-N,N;  
   CHR$(CODE"0"+N);  
30 NEXT N
```

вы получите последовательность символов расположенных по диагонали экрана, начиная с правого верхнего угла. Теперь заменим в строке 20 ' AT 31-N,N' на 'TAB N' программа будет работать также как и прежде. Теперь заменим в строке 20 *PRINT* на *LPRINT*. Заметим, что развертки по

диагонали не получится, а заменив теперь 'TAB N' на 'AT 31-N,N' и сохранив LPRINT получим по одному символу на строку, что и требовалось получить.

Вообще при печати перевод строки осуществляется в следующих случаях:

- а) при заполнении буфера строки;
- б) после LPRINT, если это не конец оператора и в нем встретилась запятая или точка с запятой;
- в) если запятая, апостроф или TAB требуют новой строки;
- г) при окончании программы, если остались невыведенные данные.



---

## Глава 22

---

### Другое периферийное 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, VERIFY, LOAD и 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 и 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, представляется 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 адрес. Он использует один аргумент - адрес порта, и позволяет считать один байт из указанного порта. OUT подобен оператору POKE:

OUT адрес, данные и записывает указанные данные в заданный порт вывода.

ZX SPECTRUM оперирует шестнадцатиразрядными адресами, которые мы будем обозначать буквой A: A15, A14, A13....A1, A0.

Биты адреса A0, A1, A2, A3 и A4 очень важны. Как правило, они в 1, но если хотя бы один из них в 0, это предписывает компьютеру некоторые действия, не более чем один из этих пяти битов может быть в 0.

Биты A6 и A7 игнорируются, так что если вы знакомы с электроникой, то можете использовать их по своему усмотрению.

Биты A8, A9 используются иногда для получения дополнительной информации. Информационный байт мы будем обозначать буквой D:

D7, D6, D5, ..., D1, D0.

Теперь предоставим список адресов портов. Имеется целый ряд входных адресов для чтения с клавиатуры, а также входного разъема 'EAR'. Сама клавиатура разбита на 8 полуразрядов по 5 клавиш в ряду:

IN 65278 считывает ряд от CAPS SHIFT до V,



IN 65022 считывает ряд от A до G,  
 IN 64510 считывает ряд от Q до T,  
 IN 63486 считывает ряд от I до 5,  
 IN 61438 считывает ряд от 0 до 6,  
 IN 57342 считывает ряд от P до Y,  
 IN 49150 считывает ряд от ENTER до H,  
 IN 32766 считывает ряд от SPASE до B,

Эти адреса могут быть вычислены из выражения:  $254+256*(255-2**N)$  при N пробегающим от 0 до 7. В байте, считанном с клавиатуры, биты от D0 до D4 служат для обозначения пяти клавиш в данном полуяруду, D0 для крайней клавиши а D4 для той, что ближе к центру. Состояние одного из этих битов 0 указывает, что соответствующая ему клавиша нажата, D6 принимает свое значение при чтении с разъема 'EAR'.

Выходной порт 254 обеспечивает громкоговоритель (D4) и разъем 'MIC' (D3), а так же установку цвета (D2,D1,D0). Порт 251 обеспечивает связь с принтером, как чтение так и запись, чтение для проверки готовности принтера к работе.

Порты 254,247 и 239 используются для связи с дополнительными устройствами, описанными в главе 22.

Запустите следующую программу:

```
10 FOR N=0 TO 7:REM HELF=ROW
  NUMBER (номер полуяруда)
20 LET A=254+256*(256-2**N)
30 PRINT AT=0,0;IN A: GO TO 30
```

и понажимайте по одной клавише в каждом полуяруду. После нажатия очередной клавиши введите BREAK, А затем NEXT N.

Ниже, на рисунке показано распределение контактов разъема:

```
A14----!----A15
A12----!----A13
+5V----!----D7
+9V----!----
0V----!----D0
0V----!----D1
CK----!----D2
A0----!----D6
A1----!----D5
A2----!----D3
```

|        |      |   |      |       |
|--------|------|---|------|-------|
| A3     | ---- | ! | ---- | D4    |
| IORQGE | ---- | ! | ---- | INT   |
| INT 0V | ---- | ! | ---- | NMI   |
| VIDEO  | ---- | ! | ---- | HALT  |
| Y      | ---- | ! | ---- | MREQ  |
| V      | ---- | ! | ---- | IOREQ |
| U      | ---- | ! | ---- | RD    |
| BUSRQ  | ---- | ! | ---- | WR    |
| RESET  | ---- | ! | ---- | 5V    |
| A7     | ---- | ! | ---- | WAIT  |
| A6     | ---- | ! | ---- | +12V  |
| A5     | ---- | ! | ---- | - 12V |
| A4     | ---- | ! | ---- | M1    |
| ROMOS  | ---- | ! | ---- | RFSH  |
| BUSACK | ---- | ! | ---- | A8    |
| A9     | ---- | ! | ---- | A10   |
| A11    | ---- | ! | ---- |       |



# 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 памяти по определенному адресу от 0 до FFFFH (H - здесь и далее означает шестнадцатиричное представление числа). Сам адрес может быть записан в память как два байта. На диаграмме показано распределение памяти компьютера ZX SPECTRUM:

|              |       |                                     |       |                          |       |
|--------------|-------|-------------------------------------|-------|--------------------------|-------|
| 0            | 3FFFH | 4000H                               | 7FFFH | 8000H                    | FFFFH |
| ROM<br>(ПЗУ) |       | RAM (ОЗУ)<br>(системная<br>область) |       | доступно для<br>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 ROM (ПЗУ) с их адресами:

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

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

POKE адрес, данные

где "адрес" и "данные" - числовые выражения, например: POKE 31000,57 "данные" может принимать значения от -255 до +255.

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

| область<br>экрана<br>TV | аттрибу-<br>ты | буфер<br>принтера | систем-<br>ные<br>перемен-<br>ные | план<br>допол.<br>памяти |       |
|-------------------------|----------------|-------------------|-----------------------------------|--------------------------|-------|
| 16384                   | 22528          | 23296             | 23532                             | 23734                    | CHANS |

| каналь-<br>ная<br>инфор-<br>мация | 80H | прог-<br>рамма<br>на бей-<br>сике | пере-<br>менные<br>прог-<br>раммы | 80H | редак-<br>тируе-<br>мые<br>строки<br>прог-<br>раммы | NL |            |
|-----------------------------------|-----|-----------------------------------|-----------------------------------|-----|-----------------------------------------------------|----|------------|
| CHANS                             |     | PROG                              | VARs                              |     | E LINE                                              |    | WORKS<br>R |

| считан-<br>ные<br>данные | NL | рабо-<br>чая<br>об-<br>ласть | стек<br>каль-<br>кулято-<br>ра | резерв     | аппа-<br>рат-<br>ный<br>стек | стек<br>пере-<br>ходов<br>к<br>подпр. |  |
|--------------------------|----|------------------------------|--------------------------------|------------|------------------------------|---------------------------------------|--|
| WORKS<br>R               |    |                              | STKBO<br>T                     | STKEN<br>D | SP                           |                                       |  |

|        |     |                                       |
|--------|-----|---------------------------------------|
| 7      | 3EH | определяемые<br>пользователем символы |
|        | UDG | P RAMT                                |
| RAMTOR |     |                                       |

Область телевизионного экрана содержит образ текущего кадра, она доступна для операторов PEEK и POKE. Каждая позиция экрана представима матрицей 8x8 точек (один байт на каждый ряд из 8 точек) однако эти восемь байт хранятся в памяти не вместе.

Полный экран представляет собой 24 строки по 32 символа. Каждая строка экрана прописывается 8-мью строками развертки телевизионного экрана. Итого для записи одного экрана выполняется 192 сканирования. В памяти рядом хра-



няются 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 тд.) содержат адреса границ между системными областями памяти, но это не переменные бейсика, и их имена не распознаются компьютером.

Область планов дополнительной памяти используется только с MICRODRIVE.

В области канальной информации содержатся данные об устройствах ввода-вывода, а именно: клавиатуре (с нижней половиной экрана), верхней половине экрана в принтере.

Каждая строка в области бейсик программ имеет формат:

|              |                         |       |       |
|--------------|-------------------------|-------|-------|
| 2 байта      | 2 байта                 | текст | ODH   |
| номер строки | длина текста +<br>ENTER |       | ENTER |

Числовые константы в программе представлены в двоичной форме используя CHR\$14 и следующие за ним 5 байт самого числа.

Переменные имеют различные форматы представления в памяти. Ниже представлен формат записи числа, имя которого состоит из одной буквы.

|       |                   |           |                     |
|-------|-------------------|-----------|---------------------|
| 60H   | 1 байт<br>порядка | бит знака | 4 байта<br>мантиссы |
| буква | значение числа    |           |                     |

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

|         |         |                 |     |                             |
|---------|---------|-----------------|-----|-----------------------------|
| 60H     | NNN     | ...             | NNN | 5 байт<br>значения<br>числа |
| 1 буква | 2 буква | последняя буква |     |                             |

Формат размещения числового массива:

| nnn   | 2 байта                                                                      | 1 байт                             | 2 байта                         |  | 2 байта                                 |  |
|-------|------------------------------------------------------------------------------|------------------------------------|---------------------------------|--|-----------------------------------------|--|
| буква | общая<br>длина<br>элемен-<br>тов+ 1<br>на<br>каж-<br>дое из-<br>мере-<br>ние | номе-<br>ра раз-<br>мерно-<br>стей | первая<br>раз-<br>мер-<br>ность |  | послед-<br>няя<br>раз-<br>мер-<br>ность |  |

по 5 байтов на каждый элемент

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:

V(1,1), V(1,2), V(1,3), V(1,4), V(1,5), V(1,6),  
V(2,1)... V(3,6)

Формат размещения управляющих переменных для FOR-NEXT операторов :



|       |          |             |            |              |                 |
|-------|----------|-------------|------------|--------------|-----------------|
|       | 5 байт   | 5 байт      | 5 байт     | 2 байта      | 1 байт          |
| буква | значение | ограничение | приращение | строка цикла | номер оператора |

Формат размещения строки символов:

|       |                     |                                 |
|-------|---------------------|---------------------------------|
|       | 2 байта             | текст(может быть пустая строка) |
| буква | количество символов |                                 |

Формат размещения строкового массива:

|       |                                                       |                     |                 |
|-------|-------------------------------------------------------|---------------------|-----------------|
|       | 2 байта                                               | 1 байт              | 2 байта         |
| буква | общее число элементов<br>плюс 1 на каждую размерность | номера размерностей | 1-я размерность |

|  |                       |                       |
|--|-----------------------|-----------------------|
|  | 2 байта               | по 1 байту на 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 главе 5.

Байт, адресуемый по RAMTOP содержит верхний адрес, используемый бейсиком.

Даже оператор NEW, который очищает ОЗУ, не изменяет содержимого области, определяемых пользователем символов.

Вы можете изменить адрес RAMTOP в операторе CLEAR:

CLEAR NEW RAMTOP ,

по которому:

- 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 двум 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, изменяя число по адресу 23608. По умолчанию предполагается 64.

Числа (за исключением 0) могут записываться в показательной форме как:

$+M*2^{**e}$

где M - мантиса в интервале 0.5,...1 (не может быть 1);  
e - экспонента, положительное или отрицательное число.

Допустим, вы записали "N" в двоичной системе счисления. "M" - дробное и имеет двоичную точку (подобно десятичной точке), то тогда будет:

$1/2 - .1$

$1/4 - .01$

$3/4 - .11$  и т.д.

Наше число "N" 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 битов мантиссы во второй байт (мы помним, что первый бит это 1), вторые 8 битов в третий байт и так до пятого байта.

б) заменяем первый 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  $1/10$ .

$$1/10 = 4/5 \cdot 2^{**(-3)}$$

Мантисса будет:

.11001100110011001100110011001100B

(поскольку 33-й бит равен 1, мы должны округлить 32-й бит, записав 1 вместо 0), а экспонента равна -3.

Теперь, применив наши три правила, запишем 5 байт:

|               | знак |                                           |           |           |           |
|---------------|------|-------------------------------------------|-----------|-----------|-----------|
|               | чис- |                                           |           |           |           |
|               | ла   |                                           |           |           |           |
| 0111 11<br>01 | 0    | 100 1100                                  | 1100 1100 | 1100 1100 | 1100 1100 |
| -3+128        |      | 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 FFH для отрицательного;

в) 3 и 4 байты содержат младшие и старшие значащие биты числа (или число +131072, если оно отрицательное);

г) 5 байт равен 0.

# Глава 25

## Системные переменные

Байты памяти с 23532 до 23733 предназначены для специального использования. В них размещаются, так называемые системные переменные. Не надо путать их имена с именами переменных в программе. Компьютер не распознает ссылки к этим переменным из бейсик-программы по их именам. Имена используются только для мнемонического обозначения этих переменных в этом описании.

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

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

N - изменение переменной не приводит к длительному эффекту.

Число - число байтов в переменной (для двухбайтовых переменных младший байт первый).

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

10 POKE N,V - 256\*INT(V/256)

20 POKE N+1,INT(V/256)

Для просмотра нового значения можно использовать оператор:

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

| 3H | адрес | имя    | содержание                                                                                                             |
|----|-------|--------|------------------------------------------------------------------------------------------------------------------------|
| N8 | 23552 | KSTATE | используется при чтении с клавиатуры                                                                                   |
| N1 | 23560 | LAST K | запоминается вновь нажатая клавиша                                                                                     |
| I  | 23561 | REPDEL | время, в 50-ых долях секунды, в течение которого клавиша должна быть зафиксирована в нажатом состоянии, начальное зна- |



|     |       |         |                                                                                                                                        |
|-----|-------|---------|----------------------------------------------------------------------------------------------------------------------------------------|
| 1   | 23562 | REPPER  | чение 35, но может быть изменено.                                                                                                      |
| N2  | 23563 | DEFADD  | задержка в 50-х долях секунды, между последовательными опросами клавиш, начальное значение 5                                           |
| N1  | 23565 | K DATA  | адрес аргументов функции пользователя если они используются, иначе 0.                                                                  |
| N2  | 23566 | TV DATA | второй байт управления цветом с клавиатуры.                                                                                            |
| X38 | 23568 | STRMS   | байты цвета, АТ, ТАВ управления телевизором                                                                                            |
| 2   | 23606 | CHARS   | адреса подключенных каналов.                                                                                                           |
| 1   | 23608 | RASP    | адрес символьного набора - 256, обычно этот набор находится в ПЗУ, но может быть размещен и в ОЗУ с указанием CHARS адреса размещения: |
| 1   | 23609 | PIP     | продолжительность звукового сигнала                                                                                                    |
| 1   | 23610 | ERR NR  | длительность задержки устраняющей дребезг клавиатуры.                                                                                  |
| X1  | 23611 | FLAGS   | код сообщения -1, начальное значение 255 (для '-1'), т.е. PEEK 23610=255                                                               |
| X1  | 23612 | TV FLAG | управляющие флажки бейсика                                                                                                             |
| X2  | 23613 | ERR SP  | флажок телевизора                                                                                                                      |
| N2  | 23615 | LIST SF | адрес в аппаратном стеке, используемый как адрес возврата при ошибке.                                                                  |
| N1  | 23617 | MODE    | адрес возврата из автоматического листинга.                                                                                            |
| 2   | 23618 | NEW PPC | режим, спецификация [K], [L], [C], [E] или [G] курсора.                                                                                |
|     |       |         | номер строки, на которую должен быть сделан переход.                                                                                   |

|    |       |         |                                                                                                   |
|----|-------|---------|---------------------------------------------------------------------------------------------------|
| 2  | 23621 | PPC     | номер строки, оператор в которой выполняется.                                                     |
| 1  | 23623 | SUB PPC | порядковый номер выполняющегося оператора в строке.                                               |
| 1  | 23624 | DORDCR  | цвет рамки экрана, содержит атрибуты                                                              |
| 2  | 23625 | KEPPC   | количество текущих строк курсоров                                                                 |
| X2 | 23627 | VARs    | адреса переменных.                                                                                |
| N2 | 23629 | DEST    | адрес переменной в задании                                                                        |
| X2 | 23631 | CHANS   | адрес канала данных                                                                               |
| X2 | 23633 | CURGRL  | 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 POKE операторах. |
| 2  | 23647 | X PRI   | адрес символа следующего за маркером [?].                                                         |
| X2 | 23649 | WORR SP | адрес временной рабочей области                                                                   |
| X2 | 23651 | STK SOT | адрес "дна" программируемого стека                                                                |
| X2 | 23653 | STK END | адрес начала резервной области памяти                                                             |
| N1 | 23655 | BREG    | В -регистр калькулятора                                                                           |
| N2 | 23656 | MEM     | адрес области, используемой как память калькулятора (обычно MEMBOT, но не всегда).                |
| 1  | 3658  | FLAG S2 | старшие флажки                                                                                    |
| X1 | 23659 | DF SZ   | число строк (включая и одну чистую) в нижней части экрана                                         |



|    |       |         |                                                                               |
|----|-------|---------|-------------------------------------------------------------------------------|
| 2  | 23660 | S TOP   | количество верхних строк в программе 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 расстояния 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 MS (см. главу 18).               |
| 2  | 23675 | UDG     | адрес первого определяемого пользователем символа.                            |
| 1  | 23677 | COORDS  | X-координата точки графопостроителя                                           |
| 1  | 23678 |         | Y-координата точки графопостроителя                                           |
| 1  | 23679 | P POSN  | 33-позиционное число для позиционирования принтера.                           |
| 1  | 23680 | PR CC   | младший байт адреса позиций для LPRINT для печати.                            |
| 1  | 23681 |         | не используется                                                               |
| 2  | 23682 | ECHO E  | 33-позиционное и 24-строковое число (в нижней половине) конца входного буфера |
| 2  | 23684 | DF CC   | адрес принт-позиций в области экрана                                          |
| 2  | 23686 | DF CCL  | подобно DF CC для нижней части экрана                                         |
| X1 | 23688 | S POSN  | 33-позиционное число для PRINT позиций                                        |

|     |       |         |                                                                                                                                                |
|-----|-------|---------|------------------------------------------------------------------------------------------------------------------------------------------------|
| X1  | 23689 |         | 23- строковое число для PRINT позиций                                                                                                          |
| X2  | 23690 | S POSNL | подобно S POSN для нижней части                                                                                                                |
| 1   | 23692 | SCR CT  | счетчик сверток: всегда на 1 больше числа сверток, которые должны проведены перед остановом со сверткой                                        |
|     |       |         | если вы установите это число больше чем на 1 (скажем 255), то экран будет сворачиваться без запроса к вам.                                     |
| 1   | 23693 | ATTR P  | сплошные цвета                                                                                                                                 |
| 1   | 23694 | MASK P  | используется для высвечивания цветов. Бит, установленный в 1, показывает, биты атрибутов берутся не из ATTR, 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                                                                                                         |

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(23765+N)
```

и вы дополнительно получите DAMP самой программы.



# Глава 26

## Использование машинных кодов

Краткое содержание: *USR с числовым аргументом*

Эта глава описывает применение машинных команд микропроцессора Z80 (U880 (ГДР), 1810BM80 (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<br>пользователем<br>символы |             |
| RAMTOP=32599 | UDG=32600                                | PRAMT=32767 |

если вы теперь выполните:

CLEAR 32499

то получите 100 байтов памяти, начиная с адреса 32500

|                       |                                           |             |
|-----------------------|-------------------------------------------|-------------|
| 100 байтов            | определяемые<br>пользователем-<br>символы |             |
| 32500<br>RAMTOP=32499 | UDC=32600                                 | PRAMT=32767 |

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 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 `IX`.

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.



# Приложение А

## Полный набор символов

| дес | символ           | ше<br>ст. | ассемблер     | СВН...      | EDH<br>... |
|-----|------------------|-----------|---------------|-------------|------------|
| код |                  | код       | мнемоника     |             |            |
| 1   | 2                | 3         | 4             | 5           | 6          |
| 0   | не ис-<br>польз. | 00        | NOP           | RLC B       |            |
| 1   | не ис-<br>польз. | 01        | LD BC,<br>NN  | RLC<br>C    |            |
| 2   | не ис-<br>польз. | 02        | LD (BC),A     | RLC<br>D    |            |
| 3   | не ис-<br>польз. | 03        | INC BC        | RLC<br>E    |            |
| 4   | не ис-<br>польз. | 04        | INC B         | RLC<br>H    |            |
| 5   | не ис-<br>польз. | 05        | DEC B         | RLC L       |            |
| 6   | PRINT<br>УПР.    | 06        | LD B, N       | RLC(H<br>L) |            |
| 7   | EDIT             | 07        | RLCA          | RLC<br>A    |            |
| 8   | курс. вле-<br>во | 08        | EX AF,<br>AF" | RRC<br>B    |            |
| 9   | курс.<br>вправо  | 09        | ADD HL,<br>BC | RRC<br>E    |            |
| 10  | курс вниз        | 0a        | LD A, (BC)    | RRC<br>D    |            |
| 11  | курсор<br>вверх  | 0в        | DEC BC        | RRC<br>E    |            |
| 12  | DELETE           | 0C        | INC C         | RRC<br>H    |            |
| 13  | ENTER            | 0D        | DEC C         | RRC<br>L    |            |

|    |                          |    |                |             |
|----|--------------------------|----|----------------|-------------|
| 14 | число                    | 0  | LD C, N        | RRC(H<br>L) |
| 15 | не ис-<br>польз.         | 0F | RRCA           | RRC<br>A    |
| 16 | INC упр.                 | 10 | DJNZ DIS       | RL B        |
| 17 | PAPER                    | 11 | LD DE,<br>NN   | RL C        |
| 18 | FLASH                    | 12 | LD (DE), A     | RL D        |
| 19 | упр.<br>BRIGHT           | 13 | INC DE         | RL E        |
| 20 | УПР.<br>INVERSE          | 14 | INC D          | RL H        |
| 21 | УПР.<br>OVER             | 15 | DEC D          | RL L        |
| 22 | УПР.<br>AT УПР.          | 16 | LD D, M        | RL<br>(HL)  |
| 23 | ТАВ                      | 17 | RLA            | RLA         |
| 24 | УПР.<br>не ис-<br>польз. | 18 | JR DIS         | RR B        |
| 25 | не ис-<br>польз.         | 19 | ADD HL,<br>DE  | RR C        |
| 26 | не ис-<br>польз.         | 1A | LD A, (DE)     | RR D        |
| 27 | не ис-<br>польз.         | 1B | DEC DE         | RR E        |
| 28 | не ис-<br>польз.         | 1C | INC E          | RR H        |
| 29 | не ис-<br>польз.         | 1D | DEC E          | RR L        |
| 30 | не ис-<br>польз.         | 1  | LD E, M        | RR<br>(HL)  |
| 31 | не ис-<br>польз.         | 1F | RRA            | RR A        |
| 32 | пробел                   | 20 | JR NZ, DIS     | SLA B       |
| 33 | !                        | 21 | LD HL,<br>NN   | SLA C       |
| 34 | "                        | 22 | LD (NN),<br>HL | SLA C       |
| 35 | #                        | 23 | INC HL         | SLA E       |
| 36 | \$                       | 24 | INC H          | SLA H       |
| 37 | %                        | 25 | DEC H          | SLA L       |



|    |   |    |               |             |
|----|---|----|---------------|-------------|
| 38 | & | 26 | LD H, N       | SLA<br>(HL) |
| 39 | ' | 27 | DAA           | SLA A       |
| 40 | ( | 28 | JR Z, DIS     | SRA B       |
| 41 | ) | 29 | ADD<br>HL,HL  | SRA C       |
| 42 | * | 2A | LD HL,NN      | SRA<br>D    |
| 43 | + | 2B | DEC HL        | SRA E       |
| 44 | . | 2C | INC L         | SRA<br>H    |
| 45 | - | 2D | DEC L         | SRA<br>E    |
| 46 | , | 2E | LD L, N       | SRA<br>(HL) |
| 47 | / | 2F | CPL           | SRA<br>A    |
| 48 | 0 | 30 | JR NC,<br>DIS |             |
| 49 | 1 | 31 | LD SP,<br>NN  |             |
| 50 | 2 | 32 | LD (NN),<br>A |             |
| 51 | 3 | 33 | INC SP        |             |
| 52 | 4 | 34 | INC (HL)      |             |
| 53 | 5 | 35 | DEC (HL)      |             |
| 54 | 6 | 36 | LD (HL),<br>N |             |
| 55 | 7 | 37 | SCF           |             |
| 56 | 8 | 38 | JR C,DIS      | SRL<br>B    |
| 57 | 9 | 39 | ADD HL,<br>SP | SRL<br>C    |
| 58 | : | 3A | LD A,<br>(NN) | SRL<br>D    |
| 59 | ; | 3B | DEC SP        | SRL<br>E    |
| 60 | < | 3C | INC A         | SZL<br>H    |
| 61 | = | 3D | DEC A         | SRL<br>L    |

|    |   |    |            |                  |                     |
|----|---|----|------------|------------------|---------------------|
| 62 | > | 3E | LD A, N    | SRL<br>(HL)      |                     |
| 63 | ? | 3F | CCF        | SRL              |                     |
| 64 | @ | 40 | LD B, B    | A<br>BIT<br>0, B | IN<br>B, (C<br>)    |
| 65 | A | 41 | LD B, C    | BIT<br>0, C      | OUT<br>(C, B)       |
| 66 | B | 42 | LD B, D    | BIT<br>0, D      | SBC<br>HL, B<br>C   |
| 67 | C | 43 | LD B, E    | BIT<br>0, E      | LD<br>(NN<br>) , BC |
| 68 | D | 44 | LD B, H    | BIT<br>0, H      | NEG                 |
| 69 | E | 45 | LD B, L    | BIT<br>0, L      | RET<br>N            |
| 70 | F | 46 | LD B, (HL) | BIT 0, (HL)      | IM<br>0             |
| 71 | G | 47 | LD B, A    | BIT<br>0, A      | LD<br>L, A          |
| 72 | H | 48 | LD C, B    | BIT<br>1, B      | IN<br>C, (C<br>) c  |
| 73 | I | 46 | LD C, C    | BIT<br>1, C      | OUT<br>C, (C<br>)   |
| 74 | J | 4A | LD C, D    | BIT<br>1, D      | ADD<br>HL, B<br>C   |
| 75 | K | 4B | LD C, E    | BIT<br>1, E      | LD<br>BC, (NN)      |
| 76 | L | 4C | LD C, H    | BIT<br>1, H      |                     |
| 77 | M | 4D | LD C, L    | BIT<br>1, L      | RET<br>I            |



|    |                   |    |               |                |                  |
|----|-------------------|----|---------------|----------------|------------------|
| 78 | N                 | 4E | LD C, (HL)    | BIT<br>1, (HL) |                  |
| 79 | O                 | 4F | LD C, A       | BIT<br>1, A    | LD<br>R, A       |
| 80 | P                 | 50 | LD D, B       | BIT<br>2, B    | IN<br>D, (C<br>) |
| 81 | Q                 | 51 | LD D, C       | BIT<br>2, C    | OUT<br>(C)       |
| 82 | R                 | 52 | LD D, D       | BIT<br>2, D    | SBC<br>HL, D     |
| 83 | S                 | 53 | LD D, E       | BIT<br>2, E    | LD<br>(NN<br>)   |
| 84 | T                 | 54 | LD D, H       | BIT<br>2, H    |                  |
| 85 | U                 | 55 | LD D, L       | BIT<br>2, L    |                  |
| 86 | V                 | 56 | LD<br>D, (HL) | BIT<br>2, (HL) | IM<br>1          |
| 87 | W                 | 57 | LD D, A       | BIT<br>2, A    | LD<br>A, L       |
| 88 | X                 | 58 | LD E, B       | BIT<br>3, B    | IN<br>E, (C<br>) |
| 89 | Y                 | 59 | LD E, C       | BIT<br>3, C    | OUT<br>(C)       |
| 90 | Z                 | 5A | LD E, D       | BIT<br>3, D    | ADC<br>RL, D     |
| 91 | [                 | 5B | LD DE,<br>(N) | BIT<br>3, E    |                  |
| 92 | /                 | 5C | LD E, H       | BIT<br>3, H    |                  |
| 93 | ]                 | 5D | LD E, L       | BIT<br>3, L    |                  |
| 94 | стрелка<br>вверх. | 5E | LD E, (HL)    | BIT<br>3 (HL)  | IM<br>2          |

|     |                |    |              |                      |                        |
|-----|----------------|----|--------------|----------------------|------------------------|
| 95  | -              | 5F | LD E,A       | BIT                  | LD                     |
| 96  | фунт<br>стерл. | 60 | LD H,B       | 3,A<br>BIT<br>4,B    | A,R<br>IN<br>H,(C<br>) |
| 97  | a              | 61 | LD H,C       | BIT                  | OUT                    |
| 98  | b              | 62 | LD H,D       | 4,C<br>BIT<br>4,D    | (C)<br>SBC<br>HL,N     |
| 99  | c              | 63 | LD H,E       | BIT<br>4,E           | LD<br>(NN<br>)         |
| 100 | d              | 64 | LD H,H       | BIT                  |                        |
| 101 | e              | 65 | LD H,L       | 4,H<br>BIT           |                        |
| 102 | f              | 66 | LD<br>H,(HL) | 4,L<br>BIT<br>4(HL)  |                        |
| 103 | g              | 67 | LD H,A       | BIT                  | RRD                    |
| 104 | h              | 68 | LD L,B       | 4,A<br>BIT<br>5,B    | IN<br>L,(C<br>)        |
| 105 | i              | 69 | LD L,C       | BIT<br>5,C           | OUT<br>(C),L           |
| 106 | j              | 6A | LD L,D       | BIT<br>5,D           | ADC<br>HL,R            |
| 107 | k              | 6B | LD L,E       | BIT<br>5,E           | LD<br>HL,(<br>N)       |
| 108 | l              | 6C | LD L,H       | BIT                  |                        |
| 109 | m              | 6D | LD L,L       | 5,H<br>BIT           |                        |
| 110 | n              | 6E | LD L,(HL)    | 5,L<br>BIT<br>5,(HL) |                        |
| 111 | o              | 6F | LD L,A       | BIT<br>5,A           | RID                    |



|     |                                                                                     |    |              |               |                  |
|-----|-------------------------------------------------------------------------------------|----|--------------|---------------|------------------|
| 112 | p                                                                                   | 70 | LD (HL),B    | BIT<br>6,B    | IN<br>F,(C<br>)  |
| 113 | q                                                                                   | 71 | LD (HL),C    | BIT<br>6,C    |                  |
| 114 | r                                                                                   | 72 | LD<br>(HL),D | BIT<br>6,D    | SBC<br>HL,S      |
| 115 | s                                                                                   | 73 | LD (HL),E    | BIT<br>6,E    | LD(N<br>N),S     |
| 116 | t                                                                                   | 74 | LD<br>(HL),H | BIT<br>6,H    |                  |
| 117 | u                                                                                   | 75 | LD (HL),L    | BIT<br>6,L    |                  |
| 118 | v                                                                                   | 76 | HALT         | BIT<br>6,(HL) |                  |
| 119 | w                                                                                   | 77 | LD (HL),A    | BIT<br>6,A    |                  |
| 120 | x                                                                                   | 78 | LD A,B       | BIT<br>7,B    | IN<br>A,(C<br>)  |
| 121 | y                                                                                   | 79 | LD A,C       | BIT<br>7,C    | OUT<br>(C)       |
| 122 | z                                                                                   | 7A | LD A,D       | BIT<br>7,D    | ADC<br>HL,S      |
| 123 | {                                                                                   | 7B | LD A,E       | BIT<br>7,E    | LD<br>SP,(R<br>) |
| 124 |                                                                                     | 7C | LD A,H       | BIT<br>7,H    |                  |
| 125 | }                                                                                   | 7D | LD A,L       | BIT<br>7,L    |                  |
| 126 | дефис                                                                               | 7E | LD A,(HL)    | BIT<br>7,H    |                  |
| 127 | ©                                                                                   | 7F | LD A,A       | BIT<br>7,A    |                  |
| 128 |  | 80 | ADD A,B      | RES<br>0,B    |                  |
| 129 |  | 81 | ADD A,C      | RES<br>0,C    |                  |

|         |                                                                                     |    |               |               |
|---------|-------------------------------------------------------------------------------------|----|---------------|---------------|
| 130     |     | 82 | ADD A,D       | RES<br>0,D    |
| 131     |    | 83 | ADD A,E       | RES<br>0,E    |
| 132     |    | 84 | ADD A,H       | RES<br>0,H    |
| 133     |    | 85 | ADD A,L       | RES<br>0,L    |
| 134     |    | 86 | ADD A,HL      | RES<br>0,(HL) |
| 135     |    | 87 | ADD A,A       | RES<br>0,A    |
| 136     |    | 88 | ADC A,B       | RES<br>1,B    |
| 137     |    | 89 | ADC A,C       | RES<br>1,C    |
| 138     |    | 8A | ADC A,D       | RES<br>1,D    |
| 139     |    | 8B | ADC A,E       | RES<br>1,E    |
| 140     |   | 8C | ADC A,H       | RES<br>1,H    |
| 141     |  | 8D | ADC A,L       | RES<br>1,L    |
| 142     |  | 8E | ADC<br>A,(HL) | RES<br>1,(HL) |
| 143     |  | 8F | ADC A,A       | RES<br>1,A    |
| 144 (A) |                                                                                     | 90 | SUB B         | RES<br>2,B    |



|         |    |               |               |     |
|---------|----|---------------|---------------|-----|
| 145 (B) | 91 | SUB C         | RES<br>2,C    |     |
| 146 (C) | 92 | SUB D         | RES<br>2,D    |     |
| 147 (D) | 93 | SUB E         | RES<br>2,E    |     |
| 148 (E) | 94 | SUB H         | RES<br>2,H    |     |
| 149 (F) | 95 | SUB L         | RES<br>2,L    |     |
| 150 (G) | 96 | SUB (HL)      | RES<br>2,(HL) |     |
| 151 (H) | 97 | SUB A         | RES<br>2,A    |     |
| 152 (I) | 98 | SBC A,B       | RES<br>3,B    |     |
| 153 (J) | 99 | SBC A,C       | RES<br>3,C    |     |
| 154 (K) | 9A | SBC A,D       | RES<br>3,D    |     |
| 155 (L) | 9B | SBC A,E       | RES<br>3,E    |     |
| 156 (M) | 9C | SBC A,H       | RES<br>3,H    |     |
| 157 (N) | 9D | SBC A,L       | RES<br>3,C    |     |
| 158 (O) | 9E | SBC<br>A,(HL) | RES<br>3,D    |     |
| 159 (P) | 9F | SBC A,A       | RES<br>3,E    |     |
| 160 (Q) | A0 | AND B         | RES<br>4,B    | LDI |
| 161 (R) | A1 | AND C         | RES<br>4,C    | CPI |
| 162 (S) | A2 | AND D         | RES<br>4,D    | INI |
| 163 (T) | A3 | AND E         | RES<br>4,E    | OUT |
| 164 (U) | A4 | AND H         | RES<br>4,H    | I   |
| 165 RND | A5 | AND L         | RES<br>4,L    |     |

|     |          |    |          |               |          |
|-----|----------|----|----------|---------------|----------|
| 166 | INKEY\$  | A6 | AND (HL) | RES<br>4,(HL) |          |
| 167 | PI       | A7 | AND A    | RES<br>4,A    |          |
| 168 | FN       | A8 | XOR B    | RES<br>5,B    | LDD      |
| 169 | POINT    | A9 | XOR C    | RES<br>5,C    | CPD      |
| 170 | SCREEN\$ | AA | XOR D    | RES<br>5,D    | IND      |
| 171 | ATTR     | AB | XOR E    | RES<br>5,E    | OUT<br>D |
| 172 | AT       | AC | XOR H    | RES<br>5,H    |          |
| 173 | TAB      | AD | XOR L    | RES<br>5,L    |          |
| 174 | VAL\$    | AE | XOR (HL) | RES<br>5,(HL) |          |
| 175 | CODE     | AF | XOR A    | RES<br>5,A    |          |
| 176 | VAL      | B0 | OR B     | RES<br>6,B    | LDIR     |
| 177 | LEN      | B1 | OR C     | RES<br>6,C    | CPIR     |
| 178 | SIN      | B2 | OR D     | RES<br>6,D    | INIR     |
| 179 | COS      | B3 | OR E     | RES<br>6,E    | OTIR     |
| 180 | TAN      | B4 | OR H     | RES<br>6,H    |          |
| 181 | ASN      | B5 | OR L     | RES<br>6,L    |          |
| 182 | ACS      | B6 | OR (HL)  | RES<br>6,(HL) |          |
| 183 | ATN      | B7 | OR A     | RES<br>6,A    |          |
| 184 | LN       | B8 | CP B     | RES<br>7,B    | LDD<br>R |
| 185 | EXP      | B9 | CP C     | RES<br>7,C    | CPD<br>R |



|     |       |    |          |               |      |
|-----|-------|----|----------|---------------|------|
| 186 | INT   | B9 | CP D     | RES           | INDR |
| 187 | SQR   | BA | CP E     | 7,D<br>RES    | OTD  |
| 188 | SGN   | BB | CP H     | 7,E<br>RES    | R    |
| 189 | ABS   | BD | CP L     | 7,H<br>RES    |      |
| 190 | PEEK  | BE | CP (HL)  | 7,L<br>RES    |      |
|     |       |    |          | 7,(HL)        |      |
| 191 | IN    | BF | CP A     | RES           |      |
| 192 | USR   | C0 | RET NZ   | 7,A<br>SET    |      |
| 193 | STR\$ | C1 | POP BC   | 0,B<br>SET    |      |
| 194 | CHR\$ | C2 | JP NZ,NN | 0,C<br>SET    |      |
| 195 | NOT   | C3 | JP NN    | 0,D<br>SET    |      |
| 196 | BIN   | C4 | CALL     | 0,E<br>SET    |      |
|     |       |    | NZ,NN    | 0,H<br>SET    |      |
| 197 | OR    | C5 | PUSH BC  | 0,L<br>SET    |      |
| 198 | AND   | C6 | ADD A,N  | 0,(HL)<br>SET |      |
|     |       |    |          | 0,(HL)        |      |
| 199 | <=    | C7 | RST 0    | SET           |      |
| 200 | >=    | C8 | RET Z    | 0,A<br>SET    |      |
| 201 | <>    | C9 | RET      | 1,B<br>SET    |      |
| 202 | LINE  | CA | JP Z,NN  | 1,C<br>SET    |      |
| 203 | THEN  | CB | SET 1,E  | 1,D           |      |
| 204 | TO    | CC | CALL     | SET           |      |
|     |       |    | Z,NN     | 1,H           |      |
| 205 | STEP  | CD | CALL NN  | SET           |      |
|     |       |    |          | 1,L           |      |

|     |         |    |                                   |               |
|-----|---------|----|-----------------------------------|---------------|
| 206 | DEF FN  | CE | ADC A,N                           | SET<br>1,(HL) |
| 207 | CAT     | CF | RST 8                             | SET<br>1,A    |
| 208 | FORMAT  | D0 | RET NC                            | SET<br>2,B    |
| 209 | MOVE    | D1 | POP DE                            | SET<br>2,C    |
| 210 | ERASE   | D2 | JP NC,NN                          | SET<br>2,D    |
| 211 | OPEN#   | D3 | OUT (N),A                         | SET<br>2,E    |
| 212 | CLOSE#  | D4 | CALL<br>NC,NN                     | SET<br>2,H    |
| 213 | MERGE   | D5 | PUSH DE                           | SET<br>2,L    |
| 214 | VERIFY  | D6 | SUB N                             | SET<br>2,(HL) |
| 215 | BEEP    | D7 | RST 16                            | SET<br>2,A    |
| 216 | CIRCLE  | D8 | RET C                             | SET<br>3,B    |
| 217 | INK     | D9 | EXX                               | SET<br>3,C    |
| 218 | PAPER   | DA | JP C,NN                           | SET<br>3,D    |
| 219 | FLASH   | DB | IN A,(N)                          | SET<br>3,E    |
| 220 | BRIGHT  | DC | CALL<br>C,NN                      | SET<br>3,H    |
| 221 | INVERSE | DD | PREFIXES<br>IN-<br>STRUCTIO<br>NS | SET<br>3,L    |
| 222 | OVER    | DE | USING IX<br>SBC A,N               | SET<br>3,(HL) |
| 223 | OUT     | DF | RST 24                            | SET<br>3,A    |



|     |              |    |               |                   |    |
|-----|--------------|----|---------------|-------------------|----|
| 224 | LPRINT       | E0 | RET FO        | SET<br>4,B        | 24 |
| 225 | LLIST        | E1 | POP HL        | SET<br>4,C        | 24 |
| 226 | STOP         | E2 | JP PC,NN      | SET<br>4,D        | 24 |
| 227 | READ         | E3 | EX<br>(SP),HL | SET<br>4,E        | 24 |
| 228 | DATA         | E4 | CALL<br>PO,NN | SET<br>4,H        | 24 |
| 229 | RE-<br>STORE | E5 | PUSH HL       | SET<br>4,L        | 24 |
| 230 | NEW          | E6 | AND N         | SET<br>4,(HL)     | 25 |
| 231 | BORDER       | E7 | RST 32        | SET<br>4,A        | 25 |
| 232 | CONTINUE     | E8 | RET PE        | SET<br>5,B        | 25 |
| 233 | DIM          | E9 | JP (HL)       | SET<br>5,C        | 25 |
| 234 | REM          | EA | JP PE ,NN     | SET<br>5,D        |    |
| 235 | FOR          | EB | EX DE,HL      | SET<br>5,E        | 25 |
| 236 | GO TO        | EC | CALL<br>PE,NN | SET<br>5,H        | 25 |
| 237 | GO SUB       | ED | SET 5,L       |                   |    |
| 238 | INPUT        | EE | XOR N         | SET<br>5,<br>(HL) |    |
| 239 | LOAD         | EF | RST 40        | SET<br>5,A        |    |
| 240 | LIST         | F0 | RET P         | SET<br>6,B        |    |
| 241 | LET          | F1 | POP AF        | SET<br>6,C        |    |
| 242 | PAUSE        | F2 | JP P,NN       | SET<br>6,D        |    |
| 243 | NEXT         | F3 | DI            | SET<br>6,E        |    |

|     |               |    |                                           |               |
|-----|---------------|----|-------------------------------------------|---------------|
| 244 | POKE          | F4 | CALL<br>P,NN                              | SET<br>6,H    |
| 245 | PRINT         | F5 | PUSH AF                                   | SET<br>6,L    |
| 246 | PLOT          | F6 | OR N                                      | SET<br>6,(HL) |
| 247 | RUN           | F7 | RST 48                                    | SET<br>6,A    |
| 248 | SAVE          | F8 | RET M                                     | SET<br>7,B    |
| 249 | RANDOMI<br>ZE | F9 | LD SP,HL                                  | SET<br>7,C    |
| 250 | IF            | FA | JP M,NN                                   | SET<br>7,D    |
| 251 | CLS           | FB | EI                                        | SET<br>7,E    |
| 252 | DRAW          | FC | CALL<br>M,NN                              | SET<br>7,H    |
| 253 | CLEAR         | FD | PREFIXES<br>INSTRUC-<br>TIONS<br>USING IY | SET<br>7,L    |
| 254 | RETURN        | FE | CP N                                      | SET<br>7,(HL) |
| 255 | COPY          | FF | RST 56                                    | SET<br>7,A    |



---

# Приложение В

---

## Сообщения

Они появляются в нижней части экрана, если компьютер остановился при выполнении некоторого оператора бейсика, и указывают причину, вызвавшую останов.

Сообщение содержит кодовый номер или букву. Краткое содержание помогает найти ошибочную строку и ошибочный оператор в этой строке, (команда указывается как строка 0, оператор 1 располагается в строке первым, оператор 2 следует после первого или THEN и т.д.)

От состояния CONTINUE зависит очень многое в сообщениях. Обычно продолжение начинается с оператора, специфицированного в предыдущем сообщении, но имеются исключения- сообщения 0,9 и D, (смотри также приложение С)

| КОД | значение                                                                                                                                                                                         | ситуация |
|-----|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------|
| 0   | ок (о' кей! порядок)<br>успешное завершение или<br>переход на строку с номером<br>большим чем имеется всего.<br>Это сообщение не имеет стро-<br>ки или оператора определен-<br>ного для CONTINUE | разное   |
| 1   | NEXT WITHOUT FOR<br>(NEXT без FOR)<br>управляющей переменной<br>нет (не была определена в<br>операторе FOR), но есть<br>обычная переменная с тем же<br>именем.                                   | NEXT     |
| 2   | VARIABLE NOT FOUND<br>(переменная не найдена)                                                                                                                                                    | разное   |

- для простой переменной выдается, если она используется без предварительного определения в операторах LET, READ или INPUT, или загружается с ленты или устанавливается в оператор FOR. Для индексируемой переменной, сообщение выдается, если она не была предварительно определена в операторе DIM перед использованием или загрузкой с ленты.
- 3 SUBSCRIPT WRONG (ошибочный индекс) индекс превышает размерность массива, либо ошибочное число задает индекс. Если индекс отрицательный или больше 65536, то выдается сообщение B.
- 4 OUT OF MEMORU (вне памяти) в памяти недостаточно места для ваших действий, вы можете освободить себе память удалив командные строки, используя DELETE затем удалить одну или две строки программы (с целью возврата их в последствии), получить дополнительную память маневрируя оператором CLEAR.
- 5 OUT OF SCREEN (вне экрана) если INPUT оператор генерирует больше чем 23 строки в нижней половине экрана, также встречается с PRINT AT 22,....
- в индексной переменной или подстроке
- LET  
INPUT  
FOR  
DIM  
GO SUB  
LOAD  
MERGE
- INPUT  
PRINT



|   |                                                                                                                                                            |                                                                                                                                   |
|---|------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------|
| 6 | NUMBER TOO BIG (число больше максимально допустимого) в результате вычисления получилось число больше 10**38.                                              | арифметические 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<br>ASN,<br>ACS, USR<br>строковый аргумент                                                                                 |
| B | INTEGER OUT OF RANGE (переполнение целого) выдаётся, когда аргумент с плавающей точкой округляется к целому. Для случая массивов смотри также сообщение 3. | RUN<br>RANDOM<br>POKE<br>DIM<br>GO TO<br>GO SUB<br>LIST<br>LLIST<br>PAUSE<br>PLOT<br>CHR\$<br>PEEK<br>USR (с числовым аргументом) |

|   |                                                                                                                                                                                           |                                                                                                                                                                                |
|---|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| C | NONSENSE IN BASIC (выражение не бейсика) текст (строка) не распознается бейсиком как допустимое выражение                                                                                 | VAL<br>VAL\$                                                                                                                                                                   |
| D | BREAK-CONT REPEATS<br>клавиша BREAK нажата во время действия периферийной операции. Действия CONTINUE после этого оператора обычные, те что указаны в операторе, сравните с сообщением L. | LOAD<br>SAVE<br>VERIFY<br>MERGE<br>LPRINT<br>LLIST<br>COPY<br>(только<br>когда<br>компьютер<br>попросил<br>свертку,<br>а вы<br>ответили<br>N,<br>SPACE<br>или<br>STOP)<br>READ |
| E | OUT OF DATA (вне данных)<br>попытка выдать READ, когда список данных в DATA кончился.                                                                                                     |                                                                                                                                                                                |
| 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                                                                                                                                                                             | INPUT                                                                                                                                                                          |



|   |                                                                                                                                                                                                                                                                     |                                                                                                |
|---|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------|
|   | некоторые введенные данные начинаются с оператора STOP, или была нажата INPUT LINE, действие CONTINUE- обычное.                                                                                                                                                     |                                                                                                |
| 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 в операции ввода-вывода)                                                                                                                                                                                                    | в операциях с внешним устройством                                                              |
| K | INVALID COLOUR (неверный цвет) специфицированное число имеет неверное значение                                                                                                                                                                                      | INK, PAPER BORDER FLASH BRIGHT INVERS OVER а также после одной из передач управляющих символов |
| L | BREAK INTO PROGRAMM (BREAK во время выполнения программы) нажата клавиша BREAK; это обнаруживается между двумя операторами. Строка и номер оператора в строке указывает на оператор выполненный перед нажатием BREAK, но CONTINUE переходит к следующему оператору. | разное                                                                                         |
| M | RAMTOR NO COOD (адрес RAMTOP не годен)                                                                                                                                                                                                                              | CLEAR                                                                                          |

|   |                                                                                                                 |                         |
|---|-----------------------------------------------------------------------------------------------------------------|-------------------------|
|   | возможно число указанное для RAMTOP слишком велико или слишком мало.                                            | RUN                     |
| N | STATEMENT LOST (оператор отсутствует)                                                                           | RETURN                  |
|   | переход к оператору, которого уже нет                                                                           | NEXT                    |
| O | INVALID STREAM (ошибочный поток данных)                                                                         | CONTINUE                |
|   |                                                                                                                 | E                       |
| P | FN WITHOUT DEF (FN без DEF) определяемая пользователем функция не определена в операторе DEF FN.                | в операции ввода-вывода |
| Q | PARAMETER ERROR (ошибка в параметре) ошибочное число аргументов или один из них не того типа, какой был описан. | FN                      |
| R | TAPE LOADING ERROR (ошибка загрузки с ленты) файл на ленте найден, но не считывается.                           | VERIFY                  |
|   |                                                                                                                 | 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 частично режимом, в котором находится компьютер.

Режим отображается курсором, мерцающей буквой, указывающей позицию в которую будет вводится очередной символ с клавиатуры.

Возможны следующие режимы:

К- (для ключевых слов) KEYWORDS.

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 интерпретируется как основной символ, нанесённый на эту клавишу.

Для обоих режимов L и K при нажатии с клавишей одновременно и 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.

С- (для заглавных букв) CAPITAL



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

#### Е- (для расширения) EXTEND

Используется для получения дальнейших символов, главным образом знаков. Этот режим вводится 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.

#### Г- (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,N,X,Y и Z, будет выдавать определенный пользователем графический символ.

Если некоторая клавиша будет удерживаться более 2-3х секунд, это вызовет повторение производимого ею действия.

Ввод с клавиатуры производится в нижнюю часть экрана, каждый символ (или составной знак) вставляется 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 главе 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 результат отображается в верхней части экрана и сохраняется до ввода строки программы, ввода пустой строки или нажатия клавиши управления курсором вверх или вниз. В нижней части выдается сообщение и коды 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 К).

В определенных обстоятельствах CAPS SHIFT и SPACE действует 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   |
| :    |
| :    |
| Z    |
| COPY |
| BEEP |

1. простое нажатие клавиши --на экране--COPY

2. SYMBOL SHIFT и клавиша --на экране-- Z

--курсор --[L]

3. CAPS SHIFT и клавиша --на экране-- :

4. BREAK/SPACE и клавиша --на экране-- Z

5. SYMBOL SZIFT и CAPS SHIFT --курсор --[E]

при курсоре [E]:

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. Делится на две 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 программ 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 с пониженной яркостью, мерцающую или нет.

Доступны цвета: черный, голубой, красный, пурпурный (фиолетовый), зеленый, желтый и белый. Края экрана 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 две установки (постоянную и временную) в шести операторах:

PAPER, INK, FLASH, 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, не мерцающие.

Временные параметры устанавливаются командами:

PAPER, INK и т.д.

вставляемыми в операторы

PRINT, LPRINT, INPUT, PLOT, DRAW,  
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 PAPER и INK могут принимать значения от 0 до 9. Параметры от 0 до 7 определяют цвета выводимого символа:

- 0- черный (BLACK)
- 1- голубой (BLUE)
- 2- красный (RED)
- 3- фиолетовый (MAGENTA)



- 4- зеленый (GREEN)
- 5- синий (CYAN)
- 6- желтый (YELLOW)
- 7- белый (WHITE)

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

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

Параметры FLASH 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 изменений.

Параметры OVER и INVERSE могут принимать значения 0 и 1.

OVER 0 - новый символ затирает старый

OVER 1 - код старого символа и нового символа соединяются операция 'исключающего или', образуя новый символ (OVER PRINTING).

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 переводится на следующую строку.

### **Принтер**

Вывод на принтер осуществляется через буфер длиной 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 невведенные данные,

г) если встретились управляющие символы TAB или запятая, требующие 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, FLASH и BRIGHT.

При вводе BREAK принтер останавливается с выдачей сообщения 'B'. При отсутствии 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  $10^{**}38$ , наименьшее положительное число  $4*10^{**}(-39)$ . Числа имеют внутреннее представление как числа с плавающей (двоичной) точкой, с выделением одного бита на показатель степени "Е" (экспоненты) в интервале от 1 до 255, и четырех байтов на мантиссу "М" в интервале от 0,5 до 1 ( $M=1$ ), представляется числом  $M*2^{**}E(-128)$ .

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

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

Числовые переменные имеют имя произвольной длины, начинающейся 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 произвольный размер. Начальный индекс всегда 1.

Строки символов более гибкие в своей длине, имя строковой переменной в отличие от простой переменной заканчивается знаком 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 выражением' '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 с значением 'М', то результатом будет М-ый символ от S\$ (подстрока длиной 1).

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

Если  $0 \leq N < M$ , то результатом будет пустая строка, в любом другом случае выдается сообщение об ошибке '3'.

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

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



## Функции

| имя<br>функ-<br>ции | тип аргумента                                                                                            | действие (возвращаемое<br>значение)                                                                                                                                                                                                                                                                                         |
|---------------------|----------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ABS                 | число                                                                                                    | абсолютное значение                                                                                                                                                                                                                                                                                                         |
| ACS                 | число                                                                                                    | арккосинус в радианах,<br>выдает сообщение об<br>ошибке A, если X не ле-<br>жит в интервале от -1 до 1                                                                                                                                                                                                                      |
| AND                 | логическая<br>операция,<br>правый<br>операнд<br>всегда число,<br>слева может<br>быть:<br>- число, тогда: | $A \text{ AND } B = \begin{cases} A, & \text{если } B < > 0 \\ 0, & \text{если } B = 0 \end{cases}$                                                                                                                                                                                                                         |
|                     | - строка, тогда:                                                                                         | $A\$ \text{ AND } B = \begin{cases} A\$, & \text{если } B < > 0 \\ \text{" "}, & \text{если } B = 0 \end{cases}$                                                                                                                                                                                                            |
| ASN                 | число                                                                                                    | арксинус в радианах, вы-<br>дает сообщение A, если не<br>лежит в интервале от -1 до<br>1                                                                                                                                                                                                                                    |
| ATN                 | число                                                                                                    | арктангенс в радианах.                                                                                                                                                                                                                                                                                                      |
| ATTR                | два числовых<br>аргумента X и<br>Y, заключае-<br>мые в скобки                                            | число, двоичный код ко-<br>торого, представляет со-<br>бой атрибуты Y-ой пози-<br>ции X-ой строки экрана,<br>бит 7 (старший) равен: 1 -<br>для мерцающего поля, и 0<br>для немерцающего, биты<br>с 5 по 3 - цвет фона, биты<br>с 2 по 1 - цвет закрашива-<br>ния, выдает сообщение B,<br>если $0 > X > 23$ и $0 > Y > 31$ . |

|         |                  |                                                                                                                                                                                 |
|---------|------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| BIN     |                  | это не обычная функция, за BIN записывается последовательность нулей и единиц представляющая собой двоичное представление числа, которое записывается в память.                 |
| CHR\$   | число            | символ, чей код представим числом X, округленным к ближайшему целому.                                                                                                           |
| CODE    | строка символов  | код первого символа в строке X (или 0, если X - пустая строка).                                                                                                                 |
| COS     | число в радианах | косинус X                                                                                                                                                                       |
| EXP     | число            | е 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 ( $0 \leq X \leq \text{FFFFH}$ ), загружается пара регистров BC и выполняется команда ассемблера IN A(C).              |
| INKEY\$ | нет              | чтение с клавиатуры, возвращает символ введенный с клавиатуры ( в режиме [L] и [C]), если было действительное нажатие клавиши, или пустую строку в противном случае.            |
| INT     | число            | округление к ближайшему меньшему целому                                                                                                                                         |
| LEN     | строка           | длина строки символов                                                                                                                                                           |



|              |                                                           |                                                                                                                                                                                                   |
|--------------|-----------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| LN           | число                                                     | натуральный логарифм, выдает сообщение А, если $x < 0$ .                                                                                                                                          |
| NOT          | число                                                     | 0, если $X < 0$ , 1, если $X = 0$ . Операция имеет четвертый приоритет                                                                                                                            |
| OR           | логическая операция. Оба операнда числа                   | 1, если $B < 0$<br>$A \text{ OR } B =$<br>А, если $B = 0$<br>операция имеет второй приоритет                                                                                                      |
| PEEK         | число                                                     | значение байта в памяти по адресу X, округленному к ближайшему целому                                                                                                                             |
| PI<br>POINT  | нет<br>два числовых аргумента X и Y, заключенных в скобки | число пи (3,14159265...)<br>1, если точка экрана с координатами (X,Y) закрашена. 0, если эта точка имеет цвет фона. Выдает сообщение В, 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 $0 < Y < 1$ .            |
| SCRE<br>N \$ | 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 $X < 0$<br>0, если $X = 0$<br>1, если $X > 0$                                                                                                                                            |
| SIN          | число в радианах                                          | синус X                                                                                                                                                                                           |

|       |                    |                                                                                                                                                                        |
|-------|--------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| SQR   | число              | корень квадратный . Выдает сообщение А, 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 (не числовое) значение. Возможны и другие ошибки. |
| VAL\$ | строка<br>символов | вычисляет X как строковое выражение, выдает сообщение C, если X содержит синтаксическую ошибку или дает не строковое ( числовое) выражение.                            |

## Операции

### Префиксные:

- число отрицательное значение  
инфиксные ( двухоперандовые):

+ сложение для чисел, конкагенция для строк

- вычитание

\* умножение

/ деление

\*\* возведение в степень ( стрелка вверх), сообщение если

левый операнд отрицательный

= равенство



|                     |                                |
|---------------------|--------------------------------|
| > больше            | оба операнда должны быть       |
| < меньше            | одного типа, результат равен 1 |
| >= больше или равно | если сравнение истинно и не    |
| <= меньше или равно | равно 0, если нет.             |
| <> не равно         |                                |

Функции и операции имеют следующие приоритеты:

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

## Операторы

Принятые обозначения:

A - одна буква;

V - переменная;

X, Y, Z - числовые выражения;

M, N - числовые выражения, которые округляются к ближайшему целому

E - некоторое выражение;

F - выражение, имеющее строковое значение;

S - последовательность операторов, разделенных двоеточием;

C - последовательность символов управления цветом.

Каждый заканчивается запятой или точкой с запятой.

Цветовой символ имеет форму операндов; PAPER, INK, FLASH, BRIGHT, INVERSE или OVER.

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

Все операторы кроме INPUT, DEF и DATA могут использоваться и как команды и в программах.

Команда или строка программы может содержать несколько операторов, разделенных двоеточием (':'). Нет ограничений на положение оператора в строке, хотя есть некоторые ограничения в IF и REM.

Все операторы языка сведены в следующую таблицу:

|                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|-------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| оператор<br>BEEP<br>X,Y | действие оператора<br>воспроизводит звук длительностью X сек и высотой Y полутонов вверх от основного тона до (или вниз, если Y отрицательное).                                                                                                                                                                                                                                                                                                                                                 |
| BORDER<br>M             | устанавливает цвет рамки (бордюра) экрана, выдает сообщение об ошибке K, если $0 > M > Y$ ,                                                                                                                                                                                                                                                                                                                                                                                                     |
| BRIGHT<br>M             | устанавливает яркость выводимого символа: 0 - для обычной яркости, 1 - для повышенной яркости, 8 - сохраняет существующую яркость                                                                                                                                                                                                                                                                                                                                                               |
| CAT<br>CIRCLE<br>X,Y,Z  | без MICRODRIVE не работает<br>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#<br>CLS           | без MICRODRIVE не работает,<br>(CLEAR SCREEN) очищает файл экрана,                                                                                                                                                                                                                                                                                                                                                                                                                              |
| CONTINU<br>E            | 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,<br>Если сообщение возникло в командной строке, то CONTINUE вызовет попытку повторить командную строку и перейдет в цикл, если было сообщение 0:1, дает сообщение 0, если было 0:2, или дает сообщение N, если было 0:3 или более, в качестве CONTINUE ис- |



|                                           |                                                                                                                                                                                                                                                                                                                                                               |
|-------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| COPY                                      | пользуется ключевое слово CONT на клавиатуре, 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<br>E1,E2,E3,.                        | часть списка данных, должна располагаться в программе,                                                                                                                                                                                                                                                                                                        |
| ..<br>DEF<br>FNA(A1,A<br>2,...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 аргумента, без MICRODRIVE не работает                                                                                                                         |
| DELETE<br>F<br>DIM<br>A(N1,N2,.<br>.. NK) | уничтожает массив с именем "A" и устанавливает числовой массив "A" с "K" измерениями и присваивает всем его элементам значение 0,                                                                                                                                                                                                                             |
| DIM<br>A\$(N1,N2,<br>... NK)              | уничтожает массив или строку с именем "A\$" и устанавливает символьный массив с "K" измерениями и присваивает всем его элементам значения". Массив может быть представлен как массив строк фиксированной длины NK, с K-1 размерностью, сообщение 4 выдастся, если недостаточно места для размещения массива, массив не определен до описания в операторе DIM, |
| DRAW<br>X,Y<br>DRAW<br>X,Y,Z              | то же самое что и DRAW X,Y,0 чертит прямую линию<br>изображает линию от текущей графической позиции в точку с приращением X,Y по дуге в Z радиан, выдает сообщение B при выходе за пределы экрана, без MICRODRIVE не работает                                                                                                                                 |
| ERAZE<br>FLASH N                          | определяет, будет ли символ мерцающим или с постоянным свечением, N=1 для мерцания, N=8, для сохранения                                                                                                                                                                                                                                                       |

|                                              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|----------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| FOR A=X<br>TO Y<br>FOR A=X<br>TO Y<br>STEP Z | предыдущего состояния, N=0 для постоянного свечения,<br>FOR A=X TO Y STEP 1<br>уничтожает скалярную переменную "A" и устанавливает управляющую переменную "X", предел "Y", шаг приращения "Z", закидывает 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 NEXT A или выдача сообщения 1, если нет NEXT (см. NEXT), сообщение 4 выдается, если недостаточно места для размещения управляющей переменной,<br>без MICRODRIVE не работает |
| FORMAT<br>F<br>GO SUB<br>N                   | проталкивает строку с оператором GO SUB в стек для использования, затем как GO TO N выдается сообщение 4, если не все программы завершились с RETURN.                                                                                                                                                                                                                                                                                                                                                                                                                         |
| GO TO N                                      | продолжает выполнение программы со строки N. Если строка N опущена, то с первой строки после этой.                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| IF X<br>THEN S                               | если 'X' истинно (не равно 0), то выполняется 'S'. 'S' включает все операторы до конца строки. Форма 'IF X THEN номер строки' недопустима                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| INK N                                        | устанавливает цвет закрашивания (т.е. цвет, которым будут изображаться символы на цвете фона). 'N' в интервале от 0 до 7 указывает цвет. N=8-оставить цвет без изменения, N=9-увеличение контраста. Выдаст 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 так же, как и в операторе PRINT, за исключением того, что они все выводятся в нижнюю часть экрана. В случае б) компьютер останавливается и ждет ввода некоторого выражения с клавиатуры, значение которого будет присвоено переменной, ввод осуществляется обычным образом, а синтаксические ошибки выдаются мерцающим [?]. для строкового выражения вводной буфер устанавливается для размещения двух таких строк (который при необходимости может быть увеличен), если первый вводимый символ STOP, то программа останавливается с сообщением Н. Случай в) подобен случаю б) с той лишь разницей, что вводимая информация представляет собой строковый литерал неограниченной длины. 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 вводится в обычном виде с прорисовкой цветом закрашивания (INK) и цвете фона (PAPER), если  $N=1$ , цветовое решение изображения символа меняется на обратное, смотри приложение В, выдает сообщение К если 'N' не 0 или 1. 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 осуществля-

LET V=E

|          |                                                                                                                                                                                                                                    |
|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 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 LLIST 0.                                                                                                                                                                                                              |
| LLIST N  | подобно LIST, но вывод осуществляется на принтер.                                                                                                                                                                                  |
| LOAD F   | загружает программу и переменные                                                                                                                                                                                                   |
| LOAD F   | загружает числовой массив                                                                                                                                                                                                          |
| DATA 0   |                                                                                                                                                                                                                                    |
| LOAD F   | загружает строковый массив                                                                                                                                                                                                         |
| DAT\$0   |                                                                                                                                                                                                                                    |
| LOAD F   | загружает старшие 'N' байтов, начиная с адреса 'M'.                                                                                                                                                                                |
| CODE     |                                                                                                                                                                                                                                    |
| M,N      |                                                                                                                                                                                                                                    |
| LOAD F   | загружает байты, начиная с адреса 'M'.                                                                                                                                                                                             |
| CODE M   |                                                                                                                                                                                                                                    |
| LOAD F   | загружает байты по тому же адресу с которого они были разгружены                                                                                                                                                                   |
| CODE     | аналогично LOAD F CODE 16384,6912.                                                                                                                                                                                                 |
| LOAD F   | Очищает файл экрана и загружает с кассетного магнитофона. Смотри главу 20.                                                                                                                                                         |
| SCREEN\$ |                                                                                                                                                                                                                                    |
| 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     | без MICRODRIVE не работает.                                                                                                                                                                                                        |
| F1,F2    |                                                                                                                                                                                                                                    |
| NEW      | запускает по новой систему программирования бейсик, уничтожая старую программу и переменные и использующую 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<br>в) если $STEP \geq 0$ , а значение 'A' стало больше значения 'предел', или STEP, а значение 'A' меньше чем значение 'предел', то происходит переход к оператору цикла. Сообщение 2 выдается, если не найдена переменная 'A'. Сообщение 1 выдается, если 'A' не является управляющей переменной цикла.                                          |
| OPEN#<br>OUT M,N | без MICRODRIVE не работает<br>выводит байт 'N' в порт 'M'. операция выполняется на уровне микропроцессора (загружает в регистровую пару BC адрес 'M', а в регистр A - 'N', выполняет команду ассемблера OUT (C),A)<br>$0 \leq M \leq 65535$ ,<br>$-255 \leq N \leq 255$ , иначе выдается сообщение В.                                                                                                                      |
| 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 нового, 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 нажатия левой клавиши. $0 \leq N < 65535$ , иначе выдается сообщение В, при $N=0$ время задержки не учитывается и продолжается до первого нажатия клавиши.                                                                                                                       |
| PLOT<br>C;M,N    | выводит точку закрашивающего цвета ( обработанная OVER и INVERSE) с ко-                                                                                                                                                                                                                                                                                                                                                    |

POKE

M,N

PRINT....

ординатами (ABS(M),ABS(N)), смещая графическую (PLOTPOSITION) позицию если цветовой 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 изменений.  $0 \leq \text{ABS}(M) \leq 255$ ,  $0 \leq \text{ABS}(N) \leq 175$ , иначе сообщение В.

записывает значение 'N' в байт памяти по адресу 'N'.  $0 \leq M \leq 65535$ .  $255 \leq N \leq 255$ , иначе сообщение В.

где '...' последовательность PRINT символов, разделенных запятыми или точками с запятой или апострофом, которые выводятся в экраный файл для отображения на экране телевизора. Точка с запятой сама действий не вызывает, а используется для разграничения символов, запятая рождает управляющий символ 'запятая', а апостроф порождает символ ENTER. В конце оператора PRINT, если он заканчивается точкой с запятой, запятой или апострофом, автоматически выводится символ ENTER. PRINT-символом может быть: а) пустая строка (т.е. ничего); б) числовое выражение. Если значение выражения отрицательно то выводится знак минус. Если

$X \leq 10^{**}(-5)$  или  $X \geq 10^{**}(13)$  вывод осуществляется в показательной форме, мантисса представляется 8-мью 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-мью значащими цифра-



ми; в) строковое выражение. В строке возможны пробелы до и после символов. Управляющие символы вызывают определяемое ими действие, не отражаемое на экране символы выводятся как '?'; г) AT M,N вывод в строку 'M', позицию 'N'; д) TAB N вывод управляющего символа TAB с последующими двумя байтами 'N' (первый байт старший), вызывает TAB-останов; е) цветовой символ в форме PAPER, INK, FLASH, BRIGHT, INVERSE или OVER оператора.

то же, что и RANDOMIZE 0

RANDOMI  
ZE  
RANDOMI  
ZE N

устанавливает системную переменную SEED, используемую для вычисления очередного значения функции RND. если N0, то SEED принимает значение N. Если N=0, то SEED принимает значение другой системной переменной FRAMES, подсчитывающей кадры, отображаемые на экране, что обеспечивает вполне случайное число. Оператор допускает сокращение RAND (см. клавишу), выдает сообщение B, если 'N' не лежит в интервале от 0 до 65535.

READ  
V1, V2,  
...VK  
REM...

присваивает переменным одной за другой значения последовательно представленные в списке DATA. не выполняется. '...' может быть последовательностью символов (исключая ENTER), может включать двоеточие (':') для указания отсутствия операторов в строке с REM.

RE-  
STORE  
RE-  
STORE N

то же самое, что и в RESTORE 0.

RETURN

перезаписывает указатель данных в первый оператор DATA в строке с меньшим, чем 'N'. Следующий оператор READ начнет считывание отсюда.

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, и затем GO TO N                                                                                                                                                                |
| SAVE F    | записывает на ленту программу и переменные.                                                                                                                                           |
| SAVE F    | записывает на ленту программу и переменные таким образом, что при загрузке программа автоматически выполняется со строки 'N'                                                          |
| LINE N    | запись на ленту числового массива                                                                                                                                                     |
| SAVE F    |                                                                                                                                                                                       |
| DATA ()   | запись на ленту строкового массива \$                                                                                                                                                 |
| SAVE F    |                                                                                                                                                                                       |
| DATA\$ () |                                                                                                                                                                                       |
| SAVE F    | записывает на ленту 'N' байтов начиная с адреса 'M'.                                                                                                                                  |
| CODE      | аналогично SAVE F CODE 16384,6912.                                                                                                                                                    |
| SAVE F    | Выдает сообщение F, если 'F' пустая строка или имеет длину более 10, смотри главу 20                                                                                                  |
| SCREEN\$  |                                                                                                                                                                                       |
| STOP      | останавливает выполнение программ с выдачей сообщения 9, CONTINUE (продолжение) будет осуществляться со следующего оператора                                                          |
| VERIFY    | то же, что и LOAD, за исключением того, что данные не загружаются в ОЗУ, но сравниваются с находящимися там. Выдается сообщение B, если обнаружится хотя бы один не совпадающий байт. |



---

# Приложение D

---

## Микропроцессор Z80 фирмы ZILOG

### 1. Введение

Архитектура процессора Z80 основывается на архитектурных принципах микропроцессора 8080 и позволяет выполнять все 78 команд этого микропроцессора, а также 80 дополнительных команд. Всего микропроцессор Z80 имеет 696 кодов операций (в отличие от 244 кодов операций микропроцессора 8080). К числу его особенностей относятся: использование для питания лишь одного источника напряжения +5 В, наличие однофазного внешнего генератора, 17 внутренних регистров и встроенная схема регенерации ОЗУ. Предусмотрены три способа реакции на прерывания, имеется второй набор из 8-разрядных регистров, которые соответствуют восьми регистрам микропроцессора 8080. Все схемы синхронизации находятся внутри микропроцессора, за исключением отдельного внешнего генератора. Адресная шина устроена таким образом: что адреса регенерируемых ячеек динамического ОЗУ появляются в линиях ее младшей половины. В минимальный комплект вычислительной системы входят процессор, источник синхронизирующих импульсов и внешняя память.

В микропроцессоре Z80 повторены все регистры микропроцессора 8080 и дополнительно к его 8-разрядным регистрам A, F, B, C, D, E, H, L имеются также A', F', B', C', D', E', H', L' и еще несколько регистров специального назначения. Эти дополнительные регистры включают два 16-разрядных индексных регистра (IX и IY), 8-разрядный регистр вектора прерываний (I) и 8-разрядный регистр регенерации памяти (R). Кроме того, имеется один 16-разрядный СК и один 16-разрядный указатель стека. Особенностью микропроцессора Z80 является его способность регенерировать ОЗУ автоматически при помощи регистра R.

Микропроцессор Z80 имеет два идентичных 8-разрядных флаговых регистра, в которых предусмотрены четыре проверяемых и два непроверяемых флажка. Четыре

проверяемых флажка - это признак переноса, признак нуля, признак отрицательного числа, признак четности - переполнения. Два непроверяемых разряда - это признак полупереноса (аналогичен дополнительному признаку переноса в микропроцессоре 8080) и признак вычитания. Последний используется для коррекции результатов операций с двоично-десятичными числами, помогая определить вид предыдущей операции. Для операции сложения и вычитания коррекция различна.

Поясним теперь назначение некоторых из выводов и сигналов микропроцессора Z80:

**BUSRQ/** - вход, на который подается запрос перехода в состояние высокого импеданса не только адресной шины и шины данных, но и линий запроса  $\overline{3U}$ , запроса  $\overline{UBB}$  и т.д. с тем, чтобы другие устройства могли свободно использовать шину;

**BASAK/** - выход, высокий уровень которого указывает на то, что вышеперечисленные линии установлены в состояние высокого импеданса;

**MI/** - выход, низкий уровень сигнала указывает на то, что микропроцессор находится в состоянии выборки кода операции при выполнении некоторой команды;

**MREQ/** - выход. Указывает на то, что адресная шина несет действительный адрес для операции чтения или записи в память;

**RFSH/** - выход. Указывает на то, что младшие 7 разрядов адресной шины несут адрес регенерации для динамического ОЗУ: для выполнения регенеративного чтения всей динамической памяти должен использоваться текущий сигнал **MREQ/**;

**HALT/** - указывает на то, что была выполнена команда **HALT**.

## II. Способы адресации

Микропроцессор Z80 имеет 16-разрядную адресную шину, обеспечивающую возможность доступа к памяти программ емкостью 64К. Предусмотрены следующие различные способы адресации:

1. **Выборка команды.** Содержимое СК пересылается на адресную шину. На выходе **MREQ/** появляется сигнал низкого уровня, который разблокирует микросхему для обеспечения доступа к памяти программ. Низким уровнем сигнала **RD/** разрешается вывод данных из памяти на шину



данных. После декодирования соответствующей команды 7-разрядный адрес регенерации выводится на адресную шину и низкий сигнал RFSH/ указывает на необходимость регенерации динамического ОЗУ.

2. Непосредственная адресация. В этом режиме ячейка, следующая за кодом операции, содержит операнд.

3. Расширенная непосредственная адресация. Ячейки, следующие за кодом операции, содержат 16-разрядный операнд.

4. Модифицированная адресация нулевой страницы. Имеется команда, называемая рестартом, которая используется для иницирования перехода на подпрограмму из возможных адресов нулевой страницы. В ячейках с этими адресами могут размещаться часто используемые подпрограммы, доступа к которым может осуществляться с помощью однобайтных команд.

5. Относительная адресация. В этом режиме задается однобайтовое смещение в диапазоне от +127 до -127 относительно текущего адреса-плюс 2, что дает возможность обращения к расположенной поблизости подпрограмме с помощью команды длиной в два байта и позволяет создавать перемещаемые программы.

6. Расширенная адресация. Полный 16-разрядный адрес назначения определяется двубайтным операндом. Этим способом можно обеспечить обращение к подпрограмме или переход по любому адресу памяти.

7. Индексная адресация. Смещение прибавляется к числу в одном из двух индексных регистров, образуя действительный адрес памяти. Этот способ полезен для работы с просмотрными таблицами.

8. Регистровая адресация. Код операции содержит данное, определяющее конкретный адрес.

9. Неявная адресация. Регистр адреса, например, аккумулятор, однозначно определяется данной командой.

10. Регистровая косвенная адресация. 16-разрядная пара регистров, например, HL, содержит адрес операнда.

11. Побитовая адресация. Три разряда кода операции определяют конкретный разряд ячейки памяти, над которым производится данная операция.

### III. Система команд.

Беглый анализ системы команд микропроцессора Z80 позволяет сделать следующие выводы:



1. Данные могут перемещаться непосредственно из регистра в регистр, из регистра в память или из одних ячеек памяти в другие. Содержимое ячеек можно изменять, не прибегая к временному запоминанию промежуточных результатов.

2. С помощью команд EX и EXX может производиться взаимная замена основных и вспомогательных регистров (R и R'), что даст возможность разделения регистровых данных между различными процедурами.

3. Имеется полный набор операций для работы со стеками. Указатель стека (SP) используется как для запоминания извлеченных данных, так и для хранения адресов возврата подпрограмм. Стек работает по принципу "последним пришел - первым обслужен" и в пределах емкости памяти системы может быть сколь угодно большим. Таким образом, возможности организации вложений подпрограмм и хранения данных в стеке по существу не ограничены.

4. Предусмотрены операции над 16-разрядными числами, такие, как DAD (сложение содержимого спаренных регистров), INX и DCX (увеличение и уменьшение содержимого спаренных регистров на 1), LHLD (непосредственная загрузка спаренных регистров) и ряд других; эти операции облегчают выполнение арифметических операций над 16-разрядными числами и управление режимами адресации.

#### IV. Ввод - вывод.

##### Программируемый ввод-вывод

В микропроцессоре Z80 реализован режим ввода-вывода с отдельным адресным пространством для портов. Младшие восемь разрядов адресной шины используются для выбора заданного УВВ (из 256 возможных). Последовательность операций ввода следующая:

1. Адрес порта помещается в младшие восемь разрядов адресной шины.

2. На выходах IORQ/ и RD/ устанавливается низкий уровень сигнала, чтобы инициировать операцию ввода на внешнем устройстве.

3. В целях предоставления устройству ввода-вывода необходимого времени для ответа на инициализирующий сигнал (т.е. для пересылки данных на шину) предусматривается состояние ожидания.



#### 4. ЦП принимает 8-разрядное данное с шины данных.

#### Прерывание по вводу-выводу

Процессор Z80 предусматривает несколько режимов прерывания:

1. Немаскируемое прерывание. В этом режиме ЦП выполняет рестарт с ячейки 0066H. Запрос немаскируемого прерывания принимается процессором в любой момент времени.

2. Маскируемое прерывание. Существует в трех видах, каждый из которых доступен для программиста:

а) вид 0: прерывание этого вида возникает автоматически при появлении сигнала RESET; реакция на него идентична реакции на прерывание в процессоре 8080A. Прерывающее устройство может выдать любую команду на шину данных (обычно, команду перехода) и ЦП выполнит ее;

б) вид 1: этот вид прерываний инициирует рестарт с ячейки 0038H подобно реакции на немаскируемое прерывание, хотя и отличается адресом перехода;

в) вид 2: при этом виде прерывания прерывающее устройство посылает в ЦП 8-разрядное слово, которое указывает адрес стандартной программы обработки прерывания, хранящийся в памяти программ. Старшие восемь разрядов этого адреса помещаются программистом в регистр 1. Это наиболее эффективный способ организации прерывания, поскольку один 8-разрядный аргумент может определять целую подпрограмму, расположенную в любом месте памяти.

#### V. Временные диаграммы.

Минимальное число нужных для выполнения команды тактов равно 4. Это количество времени требуется для того, чтобы извлечь из памяти КОП и декодировать его. Если обращения к памяти не нужны, например, при пересылке данных из регистра в регистр при выполнении арифметических действий над сдержимым регистров, выполнение команды на этом заканчивается; если нужны последовательные операции с регистрами, количество требуемых периодов может увеличиться на несколько единиц (например, INX требует 6 периодов, A DAD - 11). Далее, каждое обращение к памяти требует трех дополнительных периодов, так что, например, команда

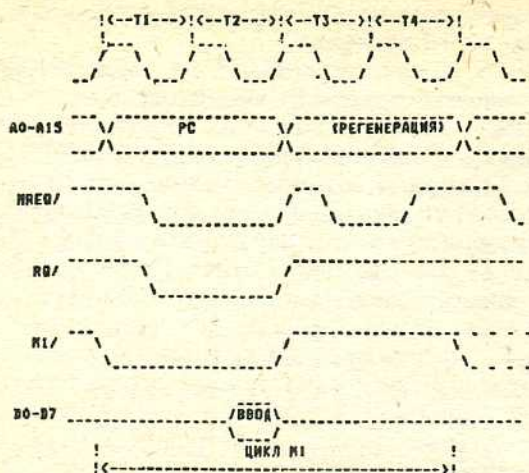
ADDM, которая должна осуществлять считывание из памяти, длится  $4+3=7$  периодов.

Прежде всего в начале такта T1 на адресную шину помещается адрес требуемой команды в программном счетчике (PC). Спустя полпериода (тактового), когда состояние шины установилось, уровень на линиях MREQ/ и RD/ становится низким, разрешая считывание из памяти. После этого памяти нужно полтора тактовых периода, чтобы выдать данные, которые фиксируются в начале T3. В течение T3 и T4 адресная шина используется для адреса регенерации динамической памяти (всего 7 бит) и уровень RFSH/ остается низким, чтобы это разрешить; тем временем ЦП производит декодирование полученной команды. Выход M1/ активен в течение первой половины операции извлечения КОП и только в это время, так что его можно использовать для идентификации начала выполнения каждой команды.

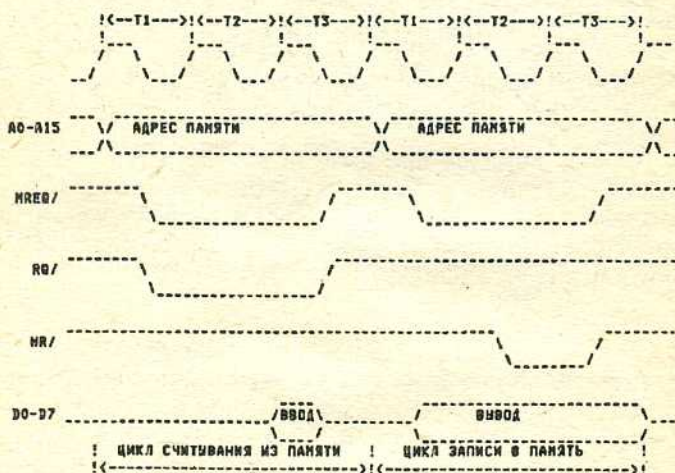
Использование линий MREQ/, RD/ и шин при считывании из памяти аналогично их использованию при извлечении КОП. При записи в память данные присутствуют на шине одновременно с разрешающим состоянием сигнала MREQ/, но строб WR/ задерживается, чтобы обеспечить установочное время ("время доступа") памяти.

Форма сигналов при вводе и выводе идентична форме при считывании и записи из памяти, за исключением того, что здесь ЦП приспособляется к большим установочным временам различных устройств В/В и вместо линии MREQ/ активизируется IORQ/.





СИНХРОНИЗАЦИЯ ПРОЦЕДУРЫ ИЗВЛЕЧЕНИЯ КОП КОМАНДЫ (ЦИКЛ M1)



ЦИКЛЫ СЧИТЫВАНИЯ ИЗ ПАМЯТИ И ЗАПИСИ В НЕЕ

# РАСПОЛОЖЕНИЕ ВЫВОДОВ ПРОЦЕССОРА Z80

|              |   |   |           |
|--------------|---|---|-----------|
| 27!          | ! | ! | !30       |
| -----+ M1/   | ! | ! | A0 +----- |
| 19!          | ! | ! | !31       |
| -----+MREQ/  | ! | ! | A1 +----- |
| 20!          | ! | ! | !32       |
| -----+IORQ/  | ! | ! | A2 +----- |
| 21!          | ! | ! | !33       |
| -----+ RD/   | ! | ! | A3 +----- |
| 22!          | ! | ! | !34       |
| -----+ WR/   | ! | ! | A4 +----- |
| 28!          | ! | ! | !35       |
| -----+RFSH/  | ! | ! | A5 +----- |
| 18!          | ! | ! | !36       |
| -----+HALT/  | ! | ! | A6 +----- |
| 24!          | ! | ! | !37       |
| -----+WAIT/  | ! | ! | A7 +----- |
| 16!          | ! | ! | !38       |
| -----+ INT/  | ! | ! | A8 +----- |
| 17!          | ! | ! | !39       |
| -----+ NMI/  | ! | ! | A9 +----- |
| 26!          | ! | ! | !40       |
| -----+RESET/ | ! | ! | A10+----- |
| 25!          | ! | ! | !1        |
| -----+DUSRR/ | ! | ! | A11+----- |
| 23!          | ! | ! | !2        |
| -----+DUSAK/ | ! | ! | A12+----- |
| 6!           | ! | ! | !3        |
| -----+ F     | ! | ! | A13+----- |
| 11!          | ! | ! | !4        |
| -----+ +SD   | ! | ! | A14+----- |
| 29!          | ! | ! | !5        |
| -----+ DBQ   | ! | ! | A15+----- |
| !            | ! | ! | !14       |
| !            | ! | ! | D0 +----- |
| !            | ! | ! | !15       |
| !            | ! | ! | D1 +----- |
| !            | ! | ! | !12       |
| !            | ! | ! | D2 +----- |
| !            | ! | ! | !8        |
| !            | ! | ! | D3 +----- |
| !            | ! | ! | !7        |
| !            | ! | ! | D4 +----- |
| !            | ! | ! | !9        |
| !            | ! | ! | D5 +----- |
| !            | ! | ! | !10       |
| !            | ! | ! | D6 +----- |
| !            | ! | ! | !13       |
| !            | ! | ! | A7 +----- |



## Оглавление

|                                                                        |     |
|------------------------------------------------------------------------|-----|
| Глава 1. Введение .....                                                | 3   |
| Глава 2. Основы программирования на языке<br>бейсик .....              | 7   |
| Глава 3. Условия .....                                                 | 13  |
| Глава 4. Циклы .....                                                   | 15  |
| Глава 5. Подпрограммы .....                                            | 17  |
| Глава 6. Операторы READ, DATA, RESTORE .....                           | 18  |
| Глава 7. Арифметические операции .....                                 | 21  |
| Глава 8. Строки символов .....                                         | 24  |
| Глава 9. Функции .....                                                 | 26  |
| Глава 10. Математические функции .....                                 | 30  |
| Глава 11. Случайные числа .....                                        | 34  |
| Глава 12. Массивы .....                                                | 36  |
| Глава 13. Логические операции .....                                    | 38  |
| Глава 14. Набор символов .....                                         | 40  |
| Глава 15. Дополнительные сведения об<br>операторах PRINT и INPUT ..... | 46  |
| Глава 16. Цвета .....                                                  | 51  |
| Глава 17. Графика .....                                                | 58  |
| Глава 18. Указания .....                                               | 62  |
| Глава 19. Программирование звуков .....                                | 66  |
| Глава 20. Внешняя память на магнитной ленте .....                      | 69  |
| Глава 21. Устройство печати .....                                      | 74  |
| Глава 22. Другое периферийное оборудование .....                       | 76  |
| Глава 23. Ввод и вывод .....                                           | 77  |
| Глава 24. Память .....                                                 | 80  |
| Глава 25. Системные переменные .....                                   | 87  |
| Глава 26. Использование машинных кодов .....                           | 92  |
| Приложение А. Полный набор символов .....                              | 94  |
| Приложение В. Сообщения .....                                          | 108 |
| Приложение С (часть 1).. Описание<br>микрокомпьютера ZX SPECTRUM ..... | 114 |
| Приложение С (часть 2).. Язык<br>программирования бейсик .....         | 120 |
| Приложение D. Микропроцессор Z80 фирмы<br>ZILOG .....                  | 136 |



