РУКОВОДСТВО ПО ЭКСПЛУАТАЦИИ

# ОРБИТА 8015

## ПЕРСОНАЛЬНЫЙ КОМПЬЮТЕР

с Цветным монитором

# МАШИНА ВЫЧИСЛИТЕЛЬНАЯ ЭЛЕКТРОННАЯ ПЕРСОНАЛЬНАЯ ПЭВМ „ОРБИТА“ ПК 8015

## Руководство по эксплуатации

Книга 2

## БАЗОВЫЕ ПРОГРАММНЫЕ СРЕДСТВА

## СОДЕРЖАНИЕ

**Приложения:**

# 1. ВВОДНАЯ ЧАСТЬ

Данная книга Руководства содержит описание базовых программных средств ПЭВМ.

В разделах Руководства приведены основные характеристики интерпретатора языка БЕЙСИК. Даны основные режимы работы, описаны команды, встроенные функции и операторы языка. Функциональные особенности, а также описание программно-доступных элементов ПЭВМ представлены в приложениях 1—5. В приложении 6 приведены дополнительные возможности интерпретатора БЕЙСИК, имеющиеся в версиях базовых программных средств, начиная с 2.0 и старше.

Книга предназначена для пользователей ПЭВМ «Орбита» ПК 8015.

Язык БЕЙСИК ПЭВМ «Орбита» ПК 8015 идентичен языку БЕЙСИК ПЭВМ ПК 8010 и ПК 8020, входящих в состав КУВТ «Корвет».

Идентичность программного обеспечения позволяет использовать ПЭВМ «Орбита» для выполнения домашних заданий с переносом учебных программ с КУВТ «Корвет» на «Орбиту».

# 2. ОБЩИЕ СВЕДЕНИЯ О ПРОГРАММИРОВАНИИ НА ЯЗЫКЕ БЕЙСИК

## 2.1. Режимы работы

Работа с интерпретатором языка БЕЙСИК возможна в одном из двух режимов: командном и программном.

Сообщение «Ок» на экране означает, что БЕЙСИК находится в командном режиме. В этом режиме можно ввести строку, состоящую из операторов языка БЕЙСИК, длиной не более 255 символов. После нажатия клавиши ↵ БЕЙСИК анализирует строку и выполняет указанные в ней действия.

В программный режим БЕЙСИК переходит, получив команду RUN в командном режиме. В программном режиме каждая строка начинается с номера, указывающего на последовательность выполнения операторов. Выход из программного режима осуществляется в случаях, если:

1) выполнена последняя строка программы;

2) встречен оператор STOP или END;

3) одновременно нажаты клавиши «УПР» и «С» или клавиша «СТОП»;

4) обнаружена неперехватываемая ошибка.

### 2.2. Формат строки

Программные строки имеют следующий формат:

nnnnn оператор [: оператор . . .] ↵,

где nnnnn — номер строки (от одной до пяти цифр) 0 — 65529.

В одной строке может быть несколько операторов, они должны разделяться двоеточием (:), и общее число символов в строке не должно превышать 255.

### 2.3. Алфавит языка

Программа на языке БЕЙСИК записывается с использованием следующих символов:

прописные и строчные буквы латинского алфавита от A до Z и буквы русского алфавита от А до Я;

арабские цифры от 0 до 9;

специальные символы + — = * / ^ ( ) ! # % @ & . , ; : " ' ? \ [ ] + < > ;

символы пропусков: пробел, горизонтальная табуляция, возврат каретки, перевод строки.

### 2.4. Описание данных

Данные в языке БЕЙСИК подразделяются на два вида: переменные и константы. С каждым видом данных связано понятие типа. Тип данного определяет его внутреннее представление в памяти. БЕЙСИК оперирует с данными следующих типов: строковые; шестнадцатиричные, восьмиричные и десятичные целые; с плавающей точкой одинарной и двойной точности.

### 2.5. Константы

Константы — значения, не изменяемые во время выполнения программы. Существует два типа констант: числовые и строковые (символьные).

Строковые константы — последовательность символов не более 255, заключенная в кавычки.

*Пример:*

«STRING СТРОКА 1, 4, 5 [@AT]» *.

Числовая константа — положительное или отрицательное число. Числовые константы могут быть целыми, с фиксированной или плавающей точкой.

Целое число — число между минус 32768 и +32767 включительно.

---

* Здесь и дальше по тексту символы «,» соответствуют символу ".

*Пример:*

—5 или 10

Число с фиксированной точкой — положительное или отрицательное число, содержащее точку на постоянном месте.

*Пример:*

—25.91, +4.02 или 790.7

Число с плавающей точкой — положительное или отрицательное число, представленное в экспоненциальной форме.

Константа с плавающей точкой состоит из знаковой или фиксированной целой части (мантиссы), за которой следует буква E и (необязательно) целое число со знаком (порядок).

*Пример:*

4E + 20

Константы с плавающей точкой могут быть одинарной и двойной точности. Константа двойной точности использует букву D вместо E.

*Пример:*

+4.6 D —35

Числовые константы могут быть записаны в десятичном, шестнадцатиричном, восьмиричном и двоичном формате.

Шестнадцатиричное число — число, содержащее до четырех цифр с префиксом &H.

*Пример:*

&H76, &HABC или &H32 FE

Восьмиричное число — число, содержащее до шести символов с префиксом &O.

*Пример:*

&O1057 или &O123456

Двоичное число — число, содержащее до шестнадцати цифр с префиксом &B.

*Пример:*

&B10101101 или &B1111

Числовые константы могут быть записаны как целое число одинарной и двойной точности. Константы в шестнадцатиричном, восьмиричном или двоичном формате запоминаются в двух байтах памяти и интерпретируются как целые числа.

Числа двойной точности запоминаются как шестнадцать цифр, и печатается шестнадцать цифр. Числа одинарной точности запоминаются как семь цифр, и печатается семь цифр, хотя только шесть цифр будут точными.

Константы одинарной точности записываются:
семью или меньше цифрами, или в экспоненциальной форме, используя E, или с завершающим знаком !

*Примеры:*

Числа одинарной точности

46.8
—1.08E +6
3546.54
454.2 !

Константы двойной точности записываются: восьмью или больше цифрами, или в экспоненциальной форме, используя D, или с завершающим знаком #.

*Примеры:*

Числа двойной точности

46.7556678923
—1.08456D +7
3546.54#
123478.65438

## 2.6. Переменные

Переменная — это величина, к которой обращаются по имени и которая может принимать различные значения.

Имена переменных могут содержать любое количество символов, но распознавание происходит по первым двум символам, причем первый из них должен быть латинской буквой, а остальные — латинскими буквами или цифрами. Специальный символ, идентифицирующий тип переменной, должен быть последним символом в имени.

*Пример:*

```
10 ABC = 1
20 ABD = 2
30 PRINT ABC, ABD
RUN
2        2
OK
```

Ключевое слово не может быть именем или частью имени переменной. Имя переменной определяет тип переменной (строковая или числовая, определенной точности). Специальный символ, идентифицирующий тип переменной, указан в табл. 1.

Таблица 1

| Символ | Тип переменной | Пример |
|---|---|---|
| $ | Строковая | NI$ = «BASIC» |
| % | Целочисленная | NI% = 12 |
| ! | Одинарной точности | NI! = —1.84 |
| # | Двойной точности | NI# = +2.23780523 |

Если имя переменной не содержит специального символа, то по умолчанию принимается числовая переменная одинарной точности. Имена переменных, различающихся только специальным символом, соответствуют разным переменным, например, имена A¤, A%, A! (или A), A# присваиваются различным переменным. Для определения типа переменной могут быть использованы операторы DEFINT, DEFSNG, DEFDBL и DEFSTR. Тип переменной определяет ее внутренее представление. Целочисленная переменная занимает два байта, переменная одинарной точности — четыре байта, переменная двойной точности — восемь байт.

Размер строкового пространства при загрузке Бейсика составляет 100 байт, поэтому при работе с длинными строковыми переменными необходимо в начале программы использовать оператор CLEAR, например, CLEAR 1000.

### 2.7. Массивы

В языке БЕЙСИК рассматриваются переменные, а также наборы переменных, образующие массивы. Массив — это упорядоченная последовательность величин, обозначенная одним именем (или набор переменных, отличающихся индексом). Отдельные величины, образующие массив, называются элементами массива. Элементы массива именуются с помощью имени массива и индекса, заключенного в скобки, и образуют переменные с индексом или индексированные переменные. Индекс указывает положение элемента в массиве, причем первый элемент имеет индекс ноль. При обращении к массиву индекс может задаваться любым выражением. Количество индексов имени переменной массива соответствует размерности массива (до 4 индексов). Перед первым обращением к массиву должна быть определена его размерность оператором DIM.

*Пример* одномерного массива B# (I):

B# (0)
B# (1)
B# (2)
B# (3)

*Пример* двумерного массива A (I, J):

| | | | |
|---|---|---|---|
| A (0, 0) | A (0, 1) | A (0, 2) | A (0, 3) |
| A (1, 0) | A (1, 1) | A (1, 2) | A (1, 3) |
| A (2, 0) | A (2, 1) | A (2, 2) | A (2, 3) |

## 2.8. Перевод чисел из одного типа в другой

Если числовое значение одной точности присваивается числовой переменной другой точности, то число будет запоминаться с точностью, описанной в имени переменной.

Округление осуществляется, когда присваивается любое значение более высокой точности переменной меньшей точности. Этим эффектом обладает не только оператор присваивания, но и вычисление функций и операторов.

*Пример:*

```
10 C% = 55.88
20 PRINT C %
RUN
56
OK
```

Во время вычисления выражения все операнды в арифметических операциях или операциях отношений переводятся в ту же точность, что и операнд с наибольшей точностью. Результат арифметических операций возвращается с этой точностью.

Логические операторы переводят свои операнды в целые числа и возвращают целочисленный результат. Операнды могут быть в диапазоне от минус 32768 до +32767, иначе появится ошибка ПЕРЕПОЛНЕНИЕ.

## 2.9. Выражения, типы операций

Выражения представляют собой компактную запись, указывающую, какие операции надо производить над данными, чтобы получить требуемое значение. Порядок вычисления выражения определяется приоритетом используемых операций. В языке БЕЙСИК выражение состоит из операндов, соединенных знаком арифметических, логических операций или операций отношения и круглыми скобками. Операндами выражений являются любые переменные, числовые и строковые константы. Также в выражении могут присутствовать встроенные функции.

В языке БЕЙСИК используются следующие типы операций (приведены в порядке приоритета):

1) арифметические операции;
2) операции отношений;
3) логические операции;
4) строковые операции.

Ниже будут подробно рассмотрены все эти типы операций.

### 2.9.1. Арифметические операции

Арифметические операции указаны в табл. 2.

Таблица 2

| Знак операции | Операция | Пример |
|---|---|---|
| ^ | Возведение в степень | X^Y |
| — | Отрицательное значение | —X |
| * | Умножение | X * Y |
| / | Деление с плавающей точкой | X/Y |
| \ | Целочисленное деление | X \ Y |
| MOD | Арифметический модуль | X MOD Y |
| + | Сложение | X + Y |
| — | Вычитание | X — Y |

Операции приведены в порядке приоритета.

Операция целочисленного деления обозначается символом\. Перед выполнением операции деления операнды округляются до целых чисел, которые должны лежать в диапазоне от —32768 до + 32767. Частное от деления округляется до целого отбрасыванием дробной части. Если значение операнда лежит вне указанного диапазона, то выдается сообщение об ошибке.

*Пример:*

10\4 = 2

25.68\6.99 = 3

Операция арифметического модуля обозначается MOD. Она вычисляет целое значение, которое является остатком от целочисленного деления.

*Пример:*

13.4 MOD 3.8 = 1, т. е. 13\4 = 3 с остатком 1

25.68 MOD 6.99 = 5, т. е. 26\7 = 3 с остатком 5

### 2.9.2. Операции отношения

Операции отношения указаны в табл. 3.

Таблица 3

| Знак операции | Операция | Пример |
|---|---|---|
| = | Равенство | X = Y |
| < > | Неравенство | X < > Y |
| < | Меньше | X < Y |
| > | Больше | X > Y |
| < = | Меньше или равно | X < = Y |
| > = | Больше или равно | X > = Y |

Операции отношения приведены в порядке приоритета. Результат сравнения ИСТИНА (—1) или ЛОЖЬ (0).

### 2.9.3. Логические операции

Логические операции имеют дело с комбинациями значений ИСТИНА — ЛОЖЬ и в результате возвращают значение ИСТИНА или ЛОЖЬ. Считается, что операнд логического оператора или результат операции ИСТИНА, если он не равен нулю.

Логические операции:

NOT — логическое отрицание;
AND — конъюнкция;
OR — дизъюнкция;
XOR — исключающее ИЛИ;
IMP — импликация;
EQV — эквивалентность.

NOT — логическое отрицание

| X | NOT X |
|---|---|
| И | Л |
| Л | И |

AND — конъюнкция

| X | Y | X AND Y |
|---|---|---|
| И | И | И |
| И | Л | Л |
| Л | И | Л |
| Л | Л | Л |

OR — дизъюнкция

| X | Y | X OR Y |
|---|---|---|
| И | И | И |
| И | Л | И |
| Л | И | И |
| Л | Л | Л |

XOR — исключающее ИЛИ

| X | Y | X XOR Y |
|---|---|---|
| И | И | Л |
| И | Л | И |
| Л | И | И |
| Л | Л | Л |

IMP — импликация

| X | Y | X IMP Y |
|---|---|---|
| И | И | И |
| И | Л | Л |
| Л | И | И |
| Л | Л | И |

EQV — эквивалентность

| X | Y | X EQV Y |
|---|---|---|
| И | И | И |
| И | Л | Л |
| Л | И | Л |
| Л | Л | И |

Логические операции выполняются путем преобразования их операндов в шестнадцатибитовые, знаковые, в дополнительном коде целые числа в диапазоне от минус 32768 до +32767. Если операнд не лежит в этом диапазоне, то выдается сообщение об ошибке ПЕРЕПОЛНЕНИЕ.

Логическая операция выполняется с целыми числами поразрядно (побитно), т. е. каждый бит результата зависит от значения соответствующих битов двух операндов.

Логические операции могут быть использованы для проверки машинных слов на битовый набор. Например, оператор AND может быть использован для «маскирования» всех битов, кроме одного бита байта состояния машинного порта ввода-вывода. Оператор OR может быть использован, чтобы «соединить» два байта для создания двоичного значения. Следующие примеры помогут продемонстрировать, как работают логические операторы.

| | |
|---|---|
| 63 AND 16 = 16 | 63 = двоичное 111111, 16 = двоичное 10000, поэтому 111111 AND 100000 = 10000 (десятичное 16); |
| 15 AND 14 = 14 | 15 = двоичное 1111, 14 = двоичное 1110, поэтому 1111 AND 1110 = 1110 (десятичное 14); |
| —1 AND 8 = 8 | — 1 = двоичное 1111111111111111, 8 = двоичное 1000, поэтому 1111111111111111 AND 1000 = 0000000000001000 (десятичное 8); |
| 4 OR 2 = 6 | 4 = двоичное 100, 2 = двоичное 10, поэтому 100 OR 10 = 110 (десятичное 6); |
| 10 OR 10 = 10 | 10 = двоичное 1010, поэтому 1010 OR 1010 = 1010 (десятичное 10); |
| —1 OR —2 = —1 | — 1 = двоичное 1111111111111111, — 2 = двоичное 1111111111111110, поэтому 1111111111111111 OR 1111111111111110 = 1111111111111111; |
| NOT X = —(X + 1) | дополнительный код любого целого это обратный код числа плюс 1. |

Часто логические операции используются в условном операторе IF.

*Пример:*

```
IF (A = 1) OR (B = 5) THEN GOTO 350
IF A = (B OR C) THEN GOSUB 800
IF ERR = 13 AND ERL = 400 THEN PRINT : GOTO 5000
```

### 2.9.4. Строковые операции

Строковые операции: конкатенация, сравнение, встроенные строковые функции.

Операция конкатенации (сцепления) обозначается символом '+'. В результате операции конкатенации образуется строка, левая часть которой равна первому операнду, а правая часть — второму.

*Пример:*

```
10 CO¤ = «ПРИМЕР»
20 TP¤ = «ЗАДАЧИ»
30 PN¤ = TP¤ + «НА ЯЗЫКЕ БЕЙСИК»
40 PRINT CO¤ + PN¤
RUN
ПРИМЕР ЗАДАЧИ НА ЯЗЫКЕ БЕЙСИК
Ok
```

Строки можно сравнивать, используя операции отношений так же, как с числами. Строковое сравнение выполняется путем посимвольного сравнения строк по кодам КОИ-8. Если все коды двух строк совпадают, то строки равны. Если коды различны, то строка с меньшим кодом меньше. Если во время сравнения конец одной строки встретился раньше, то эта строка меньше.

*Пример:*

«AA» < «AB»
«FILE NAME» = «FILE NAME»
«X &» > «X#»
«S» > «C»
«kq» > «KG»
B¤ < «9/10/85», где B¤ = «8/10/85»

Все строковые константы, используемые в выражениях, должны быть заключены в кавычки.

### 2.10. Ввод и редактирование программ

Все программные строки в языке БЕЙСИК начинаются с номера (от 0 до 65529).

При вводе строки программист может либо непосредственно набрать ее номер, либо использовать автоматическую нумерацию строк с помощью команды AUTO.

Номер строки определяет ее положение в программе. Поэтому в любое место программы, если не исчерпана отведенная для нее область памяти, можно добавить строку.

Если номер вводимой строки совпадает с номером уже существующей, то новая строка замещает старую.

Для удаления программной строки достаточно нажать вместе клавиши «УПР» и «U», если клавиша «↵» еще не была нажата, в противном случае нужно ввести номер и нажать клавишу « ↵ ». С помощью команды DELETE удаляется группа строк. Для стирания всей программы из памяти нужно ввести команду NEW.

При наборе программ можно работать в верхнем и нижнем регистрах.

Буквы латинского алфавита, если они не заключены в кавычки, автоматически переводятся в верхний регистр. Буквы русского алфавита можно использовать в качестве констант в кавычках.

Просмотр текста на экране осуществляется с помощью оператора LIST.

Одновременное нажатие клавиш
УПР и S приостанавливает,
УПР и Q продолжает,
УПР и C прекращает вывод текста на экран.

## 2.11. Встроенный редактор

Интерпретатор языка БЕЙСИК имеет встроенный редактор, используемый для изменения отдельных символов или целых программных строк. Команды встроенного редактора используются для пересылки курсора, ввода, стирания или нахождения текста в пределах редактируемой строки. Большинству команд редактора предшествует целое число, которое указывает, сколько раз должна выполняться данная команда. Когда число не указано, по умолчанию оно принимается равным 1.

Команды редактора могут быть разделены на группы согласно следующим функциям:

1) пересылка курсора;
2) ввод текста;
3) стирание текста;
4) нахождение текста;
5) замена текста;
6) конец и возобновление режима редактирования.

Введем следующие обозначения:

«см» — произвольный символ;
«текст» — строка символов произвольной длины;
[n] — целое число, по умолчанию 1;
**$** — нажатие клавиши «ПРФ» с кодом 1BH.

### 2.11.1. Пересылка курсора

Пересылка курсора осуществляется нажатием клавиши «пробел», что вызывает пересылку курсора вправо. Нажатие клавиши «пробел» n раз пересылает курсор на n позиций вправо. После каждого нажатия клавиши «пробел» появляется следующий символ редактируемой строки.

### 2.11.2. Команды ввода текста

I «текст» **$** команда вводит текст с текущей позиции курсора. Вводимые символы печатаются на терминале. Чтобы закончить ввод, следует нажать клавишу «ПРФ». Если во время команды ввода нажать клавишу возврата каретки, то прекращается ввод текста и редактирование строки. Если при вводе символа общая длина строки превышает 255 символов, символ не печатается;

X «текст» $ команда вводит текст в конец редактируемой строки, пересылает курсор в конец строки и включает режим ввода. Нажатие клавиш «ПРФ» или возврата каретки заканчивает ввод.

### 2.11.3. Команды стирания текста

[n] D команда стирает символы справа от курсора. Стертые символы печатаются в обратных косых черточках, и курсор располагается справа от последнего стертого символа. Если справа от курсора меньше символов, то стирается вся оставшаяся строка;

H «текст» команда стирает все символы справа от курсора, а затем автоматически включает режим ввода.

### 2.11.4. Команды нахождения текста

[n] S «см» команда находит позицию n-го нахождения символа, и курсор останавливается перед ним. Символ в текущей позиции курсора не рассматривается. Если «см» не найден, то курсор остановится в конце строки. Все символы, рассмотренные во время поиска, печатаются;

[n] K «см» подобна команде S, за исключением того, что все рассмотренные во время поиска символы стираются. Курсор устанавливается перед «см», и все стертые символы печатаются в обратных косых черточках.

### 2.11.5. Команды замены текста

[n] C «текст» команда изменяет следующие n символов на n символов текста. После того, как n-й новый символ напечатан, осуществляется возврат в режим редактирования.

### 2.11.6. Команды конца и возобновления режима редактирования

«возврат каретки» по этой команде печатается остаток строки, сохраняются сделанные изменения и производится возврат на уровень команд;

E команда сохраняет сделанные изменения и возвращает на уровень команд. Остаток строки не печатается;

Q команда возвращает на уровень команд без сохранения изменений, которые были сделаны во время редактирования;

L команда печатает остаток редактируемой строки и устанавливает курсор в начало строки, оставаясь в режиме редактирования;

A команда позволяет начать редактирование с начала. Восстанавливает первоначальную строку и устанавливает курсор в начало строки.

### 2.11.7. Переход в режим редактирования

Для того, чтобы изменить неверную строку в программе, нужно набрать:

ED IT «номер»

где «номер» — номер изменяемой строки.

Интерпретатор перейдет в режим редактирования, на экране будет высвечен номер строки и можно будет использовать перечисленные выше команды редактора.

*Пример:*

```
ED IT 200
200...
```

В случае, если при выполнении программы встретится синтаксическая ошибка, режим редактирования включится автоматически.

*Пример:*

```
ОШИБКА СИНТАКСИСА В 30
30...
```

Иногда бывает необходимо исправить ошибки в строке, вводимой с клавиатуры. Для этого нужно перейти в режим редактирования, нажав «УПР-А». В результате такого нажатия на экране появится восклицательный знак, и, используя команды редактора, можно исправить строку.

### 2.11.8. Использование поля дополнительных клавиш для ввода команд редактирования

Команды встроенного редактора могут вводиться как с основной клавиатуры, так и из поля дополнительных клавиш.

Соответствие клавиш основной клавиатуры дополнительным клавишам приведено в табл. 4.

Таблица 4

| Клавиши дополнительной клавиатуры | | Клавиши основной клавиатуры | |
|---|---|---|---|
| ⇥ | . | L | |
| ⇤ | 0 | A | |
| ▽ | 2 | K | |
| ⇒ | 3 | ␣ | (пробел) |
| МЕНЮ | | | |
| MODE | 5 | C | |
| ▷ | 6 | ␣ | (пробел) |
| ↖ | 7 | Q | |
| △ | 8 | S | |
| ↘ | 9 | X | |
| ИЗ | DEL | D | |
| ВЗ | INS | I | |
| СТРН | CLS | H | |

## 3. ОПИСАНИЕ ИНСТРУКЦИЙ ЯЗЫКА БЕЙСИК

### 3.1. Функция ABS

Y = ABS (X)

Функция возвращает абсолютное значение числа X. Абсолютное значение всегда положительно или равно нулю.

*Пример:*

```
PRINT ABS (7* (—5))
35
Ok
```

### 3.2. Функция ASC

Y = ASC (X$)

Функция возвращает код КОИ-8 первого символа строки X$. Результат функции — числовое значение. Если строка X$ пустая, то возвращается сообщение об ошибке НЕВЕРЕН ВЫЗОВ ФУНКЦИИ.

*Пример:*

```
10 X$ = «TEST»
20 PRINT ASC (X$)
RUN
84
Ok
```

### 3.3. Функция ATN

Y = ATN (X)

Функция возвращает арктангенс значения X. Результат функции возвращается в радианах в диапазоне от минус PI/2 до PI/2, где PI = 3.141593. Выражение X может быть задано в любом числовом формате, но вычисления функции ATN всегда выполняются с одинарной точностью. Для перевода радиан в градусы результат вычисления умножается на 180/PI.

*Пример:*

```
10 INPUTX
20 PRINT ATN (X)
RUN
? 3
1.24905
Ok
10 PI =3.141593
20 RADIANS = ATN (1)
30 DEGREES = RADIANS * 180/PI
40 PRINT RADIADS, DEGREES
RUN
.785398          45
Ok
```

### 3.4. Команда AUTO

AUTO [номер] [,[шаг]]

где [номер] — номер, который будет использован как начальный при нумерации строк;

[шаг] — значение, которое будет прибавляться к каждому номеру строки для получения следующего номера строки.

Команда AUTO автоматически генерирует номер строки после каждого нажатия клавиши возврата каретки. Команда AUTO обычно используется при введении программ. Нумерация начинается с НОМЕРА, для получения следующего номера строки к предыдущему номеру будет добавляться шаг. Если оба параметра опущены, то по умолчанию принимается AUTO 10, 10. Если за НОМЕРОМ следует запятая, а шаг не определен, то принимается шаг, определенный в предыдущей команде AUTO. Если НОМЕР опущен, а шаг определен, то нумерация начинается с 0. Точка (.) может быть использована на месте НОМЕРА строки, чтобы указать текущую строку. Если AUTO генерирует номер строки, который уже существует в программе, то после сгенерированного номера появится * , чтобы предупредить о том, что любая введенная строка будет замещать уже существующую строку. AUTO останавливается нажатием клавиши «СТОП». Строка, в которой нажимается клавиша «СТОП», не сохраняется. После нажатия клавиши «СТОП» интерпретатор возвращает управление на уровень команд.

*Пример:*

```
AUTO
```

эта команда генерирует номера строк: 10, 20, 30, 40...;

```
AUTO 100, 50
```

эта команда генерирует номера строк: 100, 150, 200, 250...;

```
AUTO 500, 50
```

эта команда генерирует номера строк: 500, 550, 600, 650...;

```
AUTO, 20
```

эта команда генерирует номера строк: 0, 20, 40, 60..

### 3.5. Оператор BEEP

BEEP

Оператор BEEP заставляет звучать динамик с частотой 800 Гц за 1/4 с. Операторы BEEP и PRINT CHR$ (7) имеют одинаковый эффект.

*Пример:*

```
10 REM, если X вне диапазона
20 IF X < 20 THEN BEEP
```

### 3.6. Функция BIN$

Y$ = BIN$ (n)

Функция возвращает строку, которая представляет собой двоичное значение десятичного аргумента. Значение аргумента n может изменяться от минус 32768 до 65535. Если n — отрицательное число, то справедливо равенство

$$BIN\$(-n) = BIN\$(65535-n).$$

Аргумент n не должен быть строковым, иначе будет выдаваться сообщение ОШИБОЧНЫЙ ТИП. Если значение аргумента лежит вне диапазона, то выдается сообщение об ошибке ПЕРЕПОЛНЕНИЕ.

*Пример:*

```
10 PRINT BIN$ (—255)
20 PRINT BIN$ (9999)
RUN
1111111100000001
10011100001111
Ok
```

### 3.7. Функция CDBL

Y = CDBL (X)

Функция CDBL переводит X в число двойной точности.

*Пример:*

```
10 A = 454.67
20 PRINT A, CDBL (A)
RUN
454.67          454.6700134277344
```

### 3.8. Функция CHR□

Y□ = CHR□ (n)

где n — числовое значение в диапазоне 0—255.

Функция CHR□ переводит код КОИ-8 в его символьный эквивалент. Функция CHR□ возвращает строку из одного символа с кодом КОИ-8. CHR□ используется для получения символа на экране или на печатающем устройстве. Если n больше 255, то выдается сообщение НЕВЕРЕН ВЫЗОВ ФУНКЦИИ.

*Пример:*

```
PRINT CHR□ (65); CHR□ (32); CHR□ (70)
A F
Ок
```

### 3.9. Функция CINT

Y = CINT (X)

Функция CINT переводит выражение X в целое число округлением дробной части. Если X не лежит в диапазоне от минус 32768 до +32767, то появляется сообщение об ошибке ПЕРЕПОЛНЕНИЕ.

*Пример:*

```
PRINT CINT (45.67)
46
Ок
PRINT CINT (—2.89)
—3
Ок
```

### 3.10. Оператор CIRCLE

CIRCLE [STEP] (X, Y), радиус [,[цвет][,[начало] [,[конец] [,аспект]]]]

где X, Y — координаты центра эллипса;

радиус — числовое выражение, определяющее радиус эллипса;

цвет — значение цвета в диапазоне от 0 до 7;

начало, конец — угловые параметры дуги, заданные в радианах;

аспект — числовое выражение.

Оператор CIRCLE рисует эллипс, дугу или сектор на экране дисплея с центром (X, Y) и заданным радиусом.

Координаты центра могут быть заданы в абсолютной или относительной форме. Точки, заданные вне экрана, не рисуются.

Если цвет отсутствует, то используется цвет переднего фона, заданный оператором COLOR. Аспект — это отношение длины отрезка, заданного на оси X в эллипсе, к длине отрезка, заданного на оси Y. По умолчанию аспект равен единице (оператор CIRCLE рисует эллипс). Если аспект меньше 1.3333, то

заданный радиус является Y-радиусом (на экране дисплея рисуется эллипс, вытянутый по оси Y). Если аспект больше 1.3333, то заданный радиус является X-радиусом эллипса (на экране дисплея рисуется эллипс, вытянутый по оси X).

Начало и конец задаются в диапазоне от 0 до 2*PI, где PI = 3.141593. Они позволяют определить начало и конец дуги эллипса. Если перед началом или концом указан минус, то рисуется прямая, соединяющая центр с началом или концом дуги. По умолчанию начало равно нулю, а конец — 2*PI.

*Пример:*

```
5 PI = 3.141593
10 CIRCLE (250, 100), 100, 3, —1.4*PI, —1.6*PI, 1.0
```

### 3.11. Команда CLEAR

CLEAR [выражение]

где выражение — счетчик байтов, который устанавливает размер строкового пространства.

Команда устанавливает все числовые переменные в ноль и очищает строковые переменные, а также устанавливает размер строкового пространства.

CLEAR освобождает всю память, используемую для данных, не стирая текущей программы. После CLEAR массивы не определены, числовые переменные принимают нулевые значения, строковые переменные очищаются, а любая информация, определенная оператором DEF, теряется.

*Пример:*

```
CLEAR
CLEAR 2000
```

### 3.12. Оператор CLS

CLS

Оператор CLS очищает алфавитно-цифровое запоминающее устройство (АЦЗУ) в цвет заднего фона и стирает текстовую информацию. Оператор CLS возвращает текстовый курсор в верхнюю левую позицию (1, 1).

### 3.13. Оператор COLOR

COLOR [передний фон] [, задний фон]

где передний фон — значение целочисленного выражения в диапазоне 0—7;

задний фон — значение целочисленного выражения в диапазоне 0—7.

Оператор устанавливает цвета, используемые в графических операторах. Передний фон — цвет, используемый операторами в качестве неявного параметра при выводе графических изобра-

жений на экран дисплея. Задний фон — цвет, используемый в качестве неявного параметра в операторах PCLS, PRESET для стирания изображения или точки с экрана дисплея.

Если значения параметров выходят за пределы определенных диапазонов, то печатается сообщение НЕВЕРЕН ВЫЗОВ ФУНКЦИИ.

Задаваемые оператором COLOR цвета переднего и заднего фона с помощью таблицы LUT преобразуются в физические цвета.

*Пример:*

10 COLOR 7,0 : PCLS

При начальной инициализации устанавливаются следующие цвета:

0 — черный;
1 — синий;
2 — зеленый;
3 — голубой;
4 — красный;
5 — фиолетовый;
6 — коричневый;
7 — белый.

### 3.14. Команда COM

COM «текст»

Команда COM устанавливает признак заполнения почтового ящика локальной сети и заносит содержимое, находящееся внутри кавычек, в почтовый ящик. Максимальное количество символов текста, заключенного в кавычках, равно 240. Если не указана какая-либо пара кавычек или превышено число символов текста, то выдается сообщение ОШИБКА СИНТАКСИСА.

*Пример:*

COM «Иванов задание выполнил»

### 3.15. Команда CONT

CONT

Команда CONT продолжает выполнение программы после прерывания (нажатие клавиши «СТОП»). Команда CONT может быть использована для возобновления выполнения программы после нажатия клавиши «СТОП» или после выполнения оператора STOP. Выполнение продолжается с той точки, где осуществилось прерывание.

Если прерывание встречается после напоминания в операторе INPUT, то выполнение продолжается с перепечатки напоминания. Команда CONT обычно используется совместно с оператором STOP для устранения ошибок. Когда выполнение останавливается, можно проверить и изменить значения переменных, используя операторы в прямом режиме. Затем можно ис-

пользовать CONT для продолжения выполнения или можно использовать прямой режим GOTO, который восстанавливает выполнение с определенного номера строки. Если программа редактируется во время прерывания, то команда CONT не правомочна.

В следующем примере создается длинный цикл:

```
10 FOR A = 1 TO 50
20 PRINT A;
30 NEXT A
RUN
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22
23 24 25 26 27 28 29 «нажатие клавиши СТОП»»
СТОП В 20
Ок
CONT
30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50
Ок
```

### 3.16. Функция COS

Y = COS (X)

Функция COS возвращает косинус угла X.

X должен быть задан в радианах. Чтобы перевести градусы в радианы, результат вычисления надо умножить на PI/180, где PI = 3.141593. Вычисление косинуса выполняется с одинарной точностью.

*Пример:*

```
10 X = 2 * COS (.4)
20 PRINT X
RUN
1.84212
Ок
10 PI = 3.141593
20 PRINT COS (PI)
30 DEGREES = 180
40 RADIANS = DEGREES * PI/180
50 PRINT COS (RADIANS)
```

### 3.17. Функция CSNG

Y = CSNG (X)

Функция CSGN переводит выражение X в число одинарной точности, подобна функциям CINT и CDBL для перевода чисел в целое и двойной точности.

*Пример:*

```
10 A # = 975.3421222 #
20 PRINT A #; CSNG (A #)
RUN
975.3421222          975.342
Ок
```

### 3.18. Переменная CSRLIN

Y = CSRLIN

Переменная возвращает текущую вертикальную координату текстового курсора. Возвращаемое значение лежит в диапазоне от 1 до 16.

*Пример:*

```
100 IF CSRLIN > 10 THEN GOTO 300
```

### 3.19. Оператор DATA

DATA «список констант»

Оператор DATA запоминает числовые и строковые константы, которые будут считаны оператором READ. Константы разделяются запятыми (,). Оператор DATA — неисполняемый оператор и может быть расположен в любом месте программы. Оператор DATA может содержать столько констант, сколько их поместится на строке экрана дисплея. В программе может быть использовано любое количество операторов DATA. Данные, которые содержатся в операторах DATA, могут считываться как один продолжающийся список данных, невзирая на количество элементов в строке и на место, где расположены строки. Операторы READ имеют доступ к операторам DATA в порядке номеров строк. В списке констант оператора DATA не допускаются выражения. Числовые константы могут быть заданы в любом формате. Нет необходимости в операторе DATA заключать в кавычки строковые константы, если они не содержат запятых, двоеточий и пробелов. Тип переменной, заданной в операторе READ, должен соответствовать типу констант в операторе DATA, иначе появится сообщение об ошибке ОШИБКА СИНТАКСИСА.

Для того чтобы заново считать данные с начала списка операторов DATA, надо использовать оператор RESTORE.

### 3.20. Оператор DEFFN

DEFFN имя [(аргумент [, аргумент]...)] = определение

где имя — имя переменной. Это имя, которому предшествует FN, становится именем функции;

аргумент — имя переменной в описании функции, которое будет заменено значением при вызове функции;

определение — выражение, выполняющее операцию функции.

Этот оператор и именует функцию, которую записывает пользователь.

Определение функции ограничивается одной строкой. Имя переменной, используемое в описании функции, не должно появляться в списке аргументов. Если это сделано, то подставля-

ется значение аргумента при вызове функции. С другой стороны, используется текущее значение переменной. Аргументы в списке соответствуют один к одному значениям, которые передаются при вызове функции. Тип функции определяет тип возвращаемого значения (числовое или строковое).

Если тип определения не соответствует типу функции, то появляется сообщение об ошибке ОШИБОЧНЫЙ ТИП. Если функция числовая, то значение выражения определения переводится в число с точностью, описанной именем, перед возвратом в вызывающий оператор. Оператор DEFFN должен быть выполнен для определения функции перед ее вызовом, иначе появится сообщение об ошибке НЕТ ФУНКЦИИ ПОЛЬЗОВАТЕЛЯ. Функция может быть определена более одного раза. Действительным является самое последнее определение.

Нельзя использовать оператор DEFFN в прямом режиме.

*Пример:*

```
10 PI = 3.141593
20 DEFFN AREA (R) = PI * R ^ 2
30 INPUT «РАДИУС»; RADIUS
40 PRINT «ПЛОЩАДЬ КРУГА»; FNAREA (RADIUS)
RUN
РАДИУС ?2
ПЛОЩАДЬ КРУГА 12.5664
Ok
```

Строка 20 определяет функцию FNAREA, которая вычисляет площадь круга с радиусом R. Функция вызывается в строке 40.

*Пример* функции с двумя аргументами:

```
10 DEFFNMUD (X, Y) = X — (INT (X/Y) * Y)
20 A = FNMUD (7.4,4)
30 PRINT A
RUN
3.4
Ok
```

### 3.21. **Операторы** DEFINT, DEFSNG, DEFDBL **и** DEFSTR

DEF тип буква [— буква] [, буква [— буква]]
где тип — INT или SGN, или DBL, или STR.

Перечисленные операторы описывают типы переменных: целочисленная, одинарной точности, двойной точности, строковая переменная. Оператор DEF тип определяет, что переменные, имена которых начинаются с заданных букв, будут переменными этого типа.

По умолчанию все переменные без символов описания типа являются переменными одинарной точности. Операторы описа-

ния типа, если они используются, должны быть заданы в начале программы перед использованием какой-либо переменной, описанной в операторе DEF тип.

*Пример:*

```
30 DEFINT  XD — H : DEFDBL  L — P : DEFSTR  A : O =
   1#/3
40 PRINT O
50 ANIMAL = «CAT» : PRINT  ANIMAL : X = 10/3 : PRINT  X
RUN
.3333333333333333
CAT
3
Ok
```

### 3.22. Оператор DEFUSR

DEFUSR [цифра] = адрес

где цифра — любая цифра 0—9;

адрес — целочисленное выражение в диапазоне 0—65535.

Оператор DEFUSR определяет начальный адрес программы на машинном языке, которая позднее вызывается функцией USR. Цифра соответствует номеру программы USR, адрес которой определяется. Если цифра опущена, по умолчанию принимается DEFUSR 0. Значение адреса может задаваться в десятичном и шестнадцатиричном формате и является действительным начальным адресом подпрограммы USR. Для переопределения начального адреса подпрограммы в программе может появляться любое число операторов DEFUSR, так как разрешается доступ к подпрограммам столько раз, сколько необходимо.

*Пример:*

```
100 DEF USR 0 = &H9000
.
.
.
500 X = USR 0 (X + 2)
```

### 3.23. Команда DELETE

DELETE [номер строки] [— номер строки]

Команда стирает программные строки. Команда DELETE убирает определенную область строк из программы. После выполнения команды DELETE интерпретатор возвращается на уровень команд. Точка (.) может быть использована на месте номера строки для указания текущей строки. Если одна из двух строк в программе не существует, то появляется сообщение об ошибке НЕВЕРЕН ВЫЗОВ ФУНКЦИИ.

*Пример:*

DELETE 40 стереть строку 40;

DELETE 40—100 стереть все строки с 40 по 100 включительно;

DELETE-40 стереть все строки с начала программы до 40 включительно.

### 3.24. Оператор DIM

DIM переменная (индексы) [, переменная (индексы)...]

Оператор определяет максимальное значение для индексов переменных массива и соответственно резервирует память. Оператор DIM устанавливает все элементы заданного числового массива в начальное значение, равное нулю, а все элементы заданного строкового массива становятся пустыми.

Если имя переменной массива используется без предварительного описания оператором DIM, то максимальное значение ее индекса равно 10. Если используется индекс, который больше заданного максимума, то появляется сообщение об ошибке ИНДЕКС ВНЕ ДИАПАЗОНА. Минимальный индекс по умолчанию равен 0.

*Пример:*

```
10 DIM A(5), B□(5)
20 FOR I=1 TO 5: READ A(I), B□(I): PRINT A(I);
B□(I)
30 NEXT I
50 DATA 14.7, AGE1, 6.9, AGE2. 8, MONTH, 29
60 DATA DAY, 80, YEAR, 12.543, BIRTHDAY
```

### 3.25. Команда EDIT

EDIT номер строки

где номер строки — номер редактируемой строки.

Команда EDIT выводит на экран дисплея строку для редактирования. На экране распечатывается редактируемая строка, и курсор устанавливается в позицию первого символа после номера строки. Затем строка может редактироваться. Если такой строки в программе нет, на экране появляется сообщение об ошибке НЕОПРЕДЕЛЕННЫЙ НОМЕР СТРОКИ. Для редактирования только что введенной строки можно использовать точку (.).

*Пример:*

```
EDIT 40
EDIT.
```

### 3.26. Оператор END

END

Оператор заканчивает выполнение программы и возвращает управление на уровень команд.

Оператор END может находиться в любом месте программы. Оператор END отличается от оператора STOP тем, что не вызывает сообщения СТОП.

*Пример:*

```
520 IFP > 100 THEN END ELSE GOTO 570
```

### 3.27. Оператор ERASE

ERASE имя массива [,имя массива]...

Оператор исключает массивы из программы. Если во время выполнения программы оказывается мало памяти, можно использовать оператор ERASE. После того как массивы удалены, пространство памяти, которое зарезервировано для массивов, может быть использовано для других целей. Оператор ERASE также может быть использован для того, чтобы заново определить размер массива, так как при попытке заново определить размер массива без предварительного выполнения оператора ERASE появляется сообщение об ошибке ДВОЙНОЕ ОПРЕДЕЛЕНИЕ.

*Пример:*

```
10 START = FRE (X)
20 DIM BIG (100, 100)
30 MIDLE = FRE (X)
40 ERASE BIG
50 DIM BIG (10, 10)
60 FINAL = FRE (X)
70 PRINT START, MIDLE,
```

### 3.28. Функции ERR и ERL

```
X = ERR
Y = ERL
```

Функция ERR возвращает код последней ошибки, а функция ERL возвращает номер строки, где ошибка обнаружилась. Функции ERR и ERL обычно используются в операторах IF... THEN, чтобы направить ветвь программы на подпрограмму, обрабатывающую ошибку. При использовании ERL в операторе IF... THEN надо убедиться, что номер строки находится справа от ERL, следующим образом:

IF ERL = номер строки THEN...

Номер строки должен находиться справа от ERL, чтобы быть перенумерованным командой «RENUM». Если оператор,

который вызывает программу, был оператором прямого режима, то ERL возвращает 65535. Так как нежелательно, чтобы этот номер строки изменился во время выполнения команды «RENUM», то при проверке ошибки в прямом режиме следует использовать форму

IF 65535 = ERL THEN...

Функции ERR и ERL могут быть заданы с использованием оператора ERROR.

*Пример:*

```
10 ON ERROR GOTO 80
20 A = 3
30 INPUT B
40 C = A/B
60 PRINT A, B, C
70 GOTO 30
80 IF (ERR = 11) AND (ERL = 40) THEN PRINT «ЗНАМЕНАТЕЛЬ РАВЕН НУЛЮ»
90 RESUME 30
```

### 3.29. Оператор ERROR

ERROR n

где n — целочисленное выражение в диапазоне 0—255.

Оператор ERROR моделирует встречу ошибки интерпретатором или распознает коды ошибок, которые определяет пользователь. Если n равно коду ошибки, то оператор ERROR будет моделировать встречу этой ошибки. Если оператором ON ERROR определена подпрограмма обработки ошибки, то будет осуществляться переход на эту подпрограмму, в противном случае будет печататься сообщение об ошибке и выполнение программы будет остановлено. Для определения собственного кода ошибки используется значение, которое отличается от любого значения, используемого интерпретатором. Рекомендуется использовать значения больше 200. Если ваша собственная ошибка определяется таким способом и она не обрабатывается в подпрограмме обработки ошибок, то будет печататься сообщение НЕОПРЕДЕЛЕННАЯ ОШИБКА и выполнение программы будет остановлено. Если значение лежит вне указанного диапазона, то выдается сообщение об ошибке НЕВЕРЕН ВЫЗОВ ФУНКЦИИ.

*Пример:*

```
10 T = 15
20 ERROR T
RUN
ДЛИННАЯ СТРОКА В 20
Ок
```

```
110 ON ERROR GOTO 400
120 INPUT «ЗАДАЙТЕ ЧИСЛО»; B
130 IF B > 5000 THEN ERROR 210
:
400 IF ERR = 210 THEN PRINT «ЧИСЛО БОЛЬШЕ,
ЧЕМ 5000»
410 IF ERL = 130 THEN RESUME 120
```

### 3.30. Функция EXP

Y = EXP (X)

Функция EXP вычисляет экспоненту, возвращает значение числа «е», возведенного в степень X. Аргумент X должен быть < 87.3365, иначе возникнет переполнение. В этом случае на экран дисплея выводится сообщение об ошибке ПЕРЕПОЛНЕНИЕ, в результате появится положительная машинная бесконечность и выполнение продолжится.

*Пример:*

```
10 X = 2
20 PRINT EXP (X—1)
RUN
2.71828
Ok
```

### 3.31. Функция FIX

Y = FIX (X)

Функция преобразует значение выражения X в целое число путем отбрасывания дробной части.

*Пример:*

```
PRINT FIX (58. 75)
58
Ok
PRINT FIX (—58. 75)
—58
Ok
```

### 3.32. Операторы FOR . . . NEXT

FOR переменная = выражение 1 TO выражение 2 [STEP выражение 3] NEXT [переменная] [, переменная . . .]

где переменная — целочисленная или одинарной точности переменная, которая будет использоваться как счетчик;

выражение 1 — начальное значение счетчика;

выражение 2 — конечное значение счетчика;

выражение 3 — шаг цикла.

Программные строки, следующие за оператором FOR, выполняются до тех пор, пока не встретится NEXT. Затем счетчик увеличивается на число, определенное выражением 3 в STEP. Если

шаг не задан, то он принимается равным единице. Затем выполняется проверка. Если значение счетчика не больше конечного значения выражения 2, то опять начинают выполняться программные строки, следующие за оператором FOR, и процесс повторяется. А если значение счетчика больше конечного значения выражения 2, то начинают выполняться программные строки, следующие за оператором NEXT. Если выражение 3 отрицательно, то каждый раз в цикле счетчик уменьшается и цикл выполняется, пока счетчик больше конечного значения.

Циклы FOR... NEXT могут быть вложены, т. е. один цикл FOR... NEXT может находиться (выполняться) внутри другого FOR... NEXT цикла. Каждый вложенный цикл должен иметь свои имена переменных в качестве счетчика. Оператор NEXT для внутреннего цикла должен выполняться раньше оператора NEXT для внешнего цикла.

Если вложенные циклы имеют одну точку конца, то может быть использован один оператор NEXT для всех переменных.

Вид оператора NEXT

NEXT переменная 1, переменная 2, переменная 3...

Эквивалент последовательности операторов:

```
NEXT переменная 1
NEXT переменная 2
NEXT переменная 3
```

Переменная в операторе NEXT может быть опущена, в этом случае оператор NEXT будет заканчивать самый последний FOR... NEXT цикл.

*Пример:*

```
10 J = 10 : K = 30
20 FOR I = 1 TO J STEP 2
30 PRINT I, : K = K + 10 : PRINT K
40 NEXT
RUN
1        40
3        50
5        60
7        70
9        80
Ok
```

### 3.33. Функция FRE

```
Y = FRE (X)
Y = FRE (X$)
```

Функция возвращает число байт памяти, не использованных интерпретатором. Так как строки в языке БЕЙСИК могут иметь переменную длину, то строковое пространство может стать фрагментированным. Функция FRE с любым строковым значением вызывает очистку перед возвратом числа свободных байт.

При очистке интерпретатор собирает все свои полезные данные и освобождает неиспользованные области памяти, которые были однажды использованы для строк. Интерпретатор будет автоматически делать очистку, когда она выполняется вне использованной рабочей области. Аргументы и функции FRE фиктивные.

### 3.34. Операторы GOSUB ... RETUPN

GOSUB номер строки

где номер строки — номер первой строки подпрограммы.

Оператор осуществляет переход к подпрограмме. В программе подпрограмма может вызываться любое число раз из самой программы или другой подпрограммы. Такие вложения подпрограмм ограничиваются только доступной памятью. Оператор RETURN возвращает управление обратно из подпрограммы к оператору, следующему за вызвавшим эту подпрограмму оператором.

Подпрограмма может содержать больше одного оператора RETURN для возврата из различных точек подпрограммы. Подпрограммы могут размещаться в любом месте программы. Часто используемые подпрограммы следует размещать в начале программы, чтобы ускорить ее выполнение. Для перехода к различным подпрограммам в зависимости от значения выражения используется оператор ON ... GOSUB.

*Пример:*

```
10 GOSUB 40
20 PRINT «ВОЗВРАТ ИЗ ПОДПРОГРАММЫ» : END
40 PRINT «РАБОТАЕТ»;
50 PRINT «ПОДПРОГРАММА»
60 RETURN
RUN
РАБОТАЕТ ПОДПРОГРАММА
ВОЗВРАТ ИЗ ПОДПРОГРАММЫ
Ок
```

### 3.35. Оператор GOTO

GOTO номер строки

Оператор осуществляет безусловный переход из нормальной программной последовательности к определенному номеру строки. Если номер строки — номер исполняемого оператора, то выполняется этот оператор и следующие за ним, если же это номер неисполняемого оператора (такого, как REM или DATA), то выполнение продолжается с первого исполняемого оператора. Оператор GOTO может быть использован в прямом режиме, чтобы произвести запуск программы с указанного номера строки.

Это может быть использовано при устранении ошибок. Для перехода к различным строкам в зависимости от значения выражения используется оператор ON ... GOTO.

*Пример:*

```
5 DATA 5, 7, 12
10 READ R
20 PRINT «R = » ; R,
30 A = 3.141 * R ^ 2
40 PRINT «AREA = » ; A
50 GOTO 5
RUN
R = 5          AREA = 78.525
R = 7          AREA = 153.909
R = 12         AREA = 452.304
ВНЕ DATA В 10
Ок
```

### 3.36. Функция HEX□

Y□ = HEX□ (n)

Функция возвращает строку, которая представляет собой шестнадцатиричное значение десятичного аргумента. Значение округляется до целого числа перед выполнением функции HEX□. Если n больше 65535, то выдается сообщение ПЕРЕПОЛНЕНИЕ.

*Пример:*

```
10 INPUT X
20 A□ = HEX□ (X)
30 PRINT X; «DECIMAL IS»; A□; «HEXADECIMAL»
RUN
? 32
32 DECIMAL IS 20 HEXADECIMAL
Ок
RUN
? 15
15 DECIMAL IS F HEXADECIMAL
Ок
```

### 3.37. Оператор IF

IF выражение THEN задание [ELSE задание]

IF выражение GOTO номер строки [ELSE задание]

где задание — оператор или последовательность операторов, или номер строки для перехода.

Оператор принимает решение относительно программного потока, основанное на значении выражения. Если выражение — ИСТИНА (не ноль), то выполняется часть THEN или часть GOTO. Если результат выражения — ЛОЖЬ (ноль), то части THEN и GOTO игнорируются и выполняется часть ELSE, если она существует, или выполнение продолжается со следующего за оператором IF номера строки. Операторы IF ... THEN ... ELSE могут быть вложены.

*Пример:*

IF X > Y THEN PRINT «X БОЛЬШЕ Y» ELSE IF Y > X THEN PRINT «X МЕНЬШЕ Y» ELSE PRINT «X РАВНО Y»

Если оператор не содержит одинаковое количество частей THEN и ELSE, то каждая часть ELSE соответствует части THEN, закрывая ближайший оператор IF.

*Пример:*

IF A = B THEN IF B = C THEN PRINT «A = C» ELSE PRINT «A < > C» не будет печататься A < > C, когда A < > B.

### 3.38. Функция INKEY ¤

Y¤ = INKEY¤

Функция INKEY ¤ опрашивает клавиатуру на нажатие какой-либо клавиши и возвращает нажатый символ. Если ни одна клавиша не нажимается, то возвращается пустая строка и осуществляется переход к выполнению следующего оператора программы. Значение функции INKEY ¤ не высвечивается на экране дисплея.

*Пример:*

10 PRINT «Нажмите любую клавишу для выхода из программы»

20 A = INKEY¤: IF A¤ = "" THEN 20 ELSE STOP

### 3.39. Оператор INPUT

INPUT «напоминание»; список переменных

Оператор осуществляет ввод с клавиатуры во время выполнения программы. Когда выполняется оператор INPUT, программа делает остановку, на экране появляется знак вопроса, указывающий, что программа ожидает данные. Если задана строка напоминания, то она высвечивается перед знаком вопроса. Затем требуемые данные вводятся с клавиатуры. Вводимые данные присваиваются переменным, заданным в списке. Количество элементов данных должно соответствовать количеству переменных в списке. Элементы данных отделяются запятыми. Переменные в списке могут быть числовыми и строковыми.

Тип каждого вводимого элемента должен соответствовать типу, определенному именем переменной. Нет необходимости заключать в кавычки строковые данные. Если на запрос оператора INPUT вводятся данные с типом, не соответствующим типу переменной, то на экране дисплея появляется сообщение ?ПОВТОРИТЕ ВВОД. Присваивание введенных значений не выполняется до тех пор, пока не будет задан правильный ответ.

Если на запрос оператора INPUT вводится больше элементов данных, то появляется сообщение ?ЛИШНИЕ ДАННЫЕ, лишние данные игнорируются и выполнение программы продолжается.

Если вводится меньше элементов данных, то появляется сообщение ?? до тех пор, пока не будет введено необходимое число данных. Если же вводится возврат каретки, то все оставшиеся переменные обнуляются.

*Пример:*

```
10 INPUT X
20 PRINT X; «В КВАДРАТЕ»; X ^ 2 : END
RUN
? 5
5 В КВАДРАТЕ 25
Ок
```

### 3.40. Функция INSTR

Z = INSTR ([n,] X¤, Y¤)

где n — числовое выражение в диапазоне 1—255;

X¤, Y¤ — строковые переменные, строковые выражения или строковые константы.

Функция возвращает позицию начала строки Y¤ в строке X¤. Необязательное смещение n устанавливает позицию для начала просмотра. Функция INSTR возвращает 0 в случаях:

n > LEN (X¤);

X¤ — пустая строка;

Y¤ — не может быть найдена.

Если Y¤ пустая строка, то INSTR возвращает n или 1. Если n вне диапазона, то возвращается сообщение об ошибке НЕВЕРЕН ВЫЗОВ ФУНКЦИИ.

*Пример:*

```
10 A¤ = «ABCDEB»
20 B¤ = «B»
30 PRINT INSTR (A¤, B¤), INSTR (4, A¤, B¤)
RUN
2     6
Ок
```

### 3.41. Функция INT

Y = INT (X)

Функция возвращает наибольшее целое число, которое меньше или равно значению выражения X.

*Пример:*

```
PRINT INT (99.89)
99
Ок
```

```
PRINT INT (— 12.11)
—13
Ок
```

### 3.42. Функция LEFT¤

Y¤ = LEFT¤(X¤, n)

где n в диапазоне 0—255.

Функция возвращает n самых левых символов строки X¤. Если n > LEN (X¤), то возвращается строка X¤. Если n = 0, возвращается пустая строка. Если n больше 255, то выдается сообщение об ошибке НЕВЕРЕН ВЫЗОВ ФУНКЦИИ.

*Пример:*

```
10 A¤ = «BASIC — ПРОГРАММА»
20 B¤ = LEFT ¤(A¤, 5)
30 PRINT B ¤
RUN
BASIC
Ок
```

### 3.43. Функция LEN

Y = LEN (X¤)

Функция возвращает количество символов строки X¤.

*Пример:*

```
X¤ = «BASIC — ПРОГРАММА» : PRINT LEN (X¤)
15
Ок
```

### 3.44. Оператор LET

[LET] переменная = выражение

Этот оператор присваивает переменной значение выражения. Слово LET необязательно. Если числовые значения присваиваются строковым переменным или строковые значения присваиваются числовым переменным, то появляется сообщение об ошибке ОШИБОЧНЫЙ ТИП.

*Пример:*

```
110 LET D = 12                  или  110 D = 12
120 LET E = 12 ^ 2                   120 E = 12 ^ 2
130 LET F¤ = «STROKA»                130 F¤ = «STROKA»
140 LET SUM = (D + E) /5             140 SUM = (D + E)/5
```

### 3.45. Оператор LINE

LINE [[STEP] (X0, Y0)]—[STEP] (X1, Y1) [,[цвет] [,B[F]]]

где STEP — позволяет указать координаты относительно последней отображаемой точки;

X0, Y0 — координаты начальной точки;

X1, Y1 — координаты конечной точки;
цвет — цвет в диапазоне от 0 до 7.

Оператор рисует на экране дисплея линию или прямоугольник. По умолчанию, если цвет не задан, используется цвет переднего фона.

Параметр B указывает, что рисуется прямоугольник заданным цветом, а параметр F указывает на то, что прямоугольник заполняется этим цветом.

Использование параметра B заменяет четыре оператора:

```
10 LINE (X0, Y0) — (X1, Y0)
20 LINE (X1, Y0) — (X1, Y1)
30 LINE (X0, Y1) — (X1, Y1)
40 LINE (X0, Y1) — (X0, Y0)
```

Точки, имеющие координаты, выходящие за пределы экрана, не рисуются.

Оператор в формате LINE — (X1, Y1) рисует линию от последней отображаемой точки к точке (X1, Y1) цветом переднего фона.

*Пример:*

```
10 LINE (0,200) — (511, 100), 7, B
20 LINE (0,100) — (511, 200), 2
30 LINE (511, 100) — (0,200), 4
```

### 3.46. Оператор LINE INPUT

LINE INPUT «напоминание»; строковая переменная
где напоминание — строковая константа, которая появляется на экране перед вводом данных.

Оператор присваивает строку данных (длиной до 254 символов) строковой переменной без использования разграничителей. Знак вопроса (?) не печатается, если он не является частью строки напоминания. Все введенные данные от конца напоминания до возврата каретки присваиваются строковой переменной. LINE INPUT можно остановить нажатием клавиш «УПР-С». Интерпретатор возвращает управление на уровень команд, появится подсказка Ok. Для возобновления выполнения программы вводится команда CONT.

### 3.47. Команда LIST

LIST [строка] [—[строка]].
где строка — номер строки в диапазоне 0—65529.

Команда распечатывает на экране дисплея программу, находящуюся в памяти. Вывод программы на экран может быть прерван нажатием клавиши «СТОП». Если номера строк опущены, то распечатывается вся введенная программа. Если задан только первый номер строки, то распечатывается только строка

с этим номером. Если заданы только первый номер строки и тире, то распечатываются программные строки, начиная с этого номера и до конца программы. Если задан только второй номер строки, то распечатывается программа, начиная с начала и до заданного номера строки включительно. Если заданы оба номера строки, то распечатываются программные строки в заданном диапазоне, включая строки с заданными номерами. Для указания текущей строки может быть использована точка.

*Пример:*

LIST распечатывает всю введенную программу;
LIST 35 распечатывает строку с номером 35 на экране дисплея;
LIST 100—200 распечатывает строки с 100 до 200 включительно;
LIST . распечатывает текущую строку.

При выполнении команд LIST и EDIT числа, представленные в двоичном виде, на экране дисплея высвечиваются в восьмиричном виде.

*Пример:*

```
10 A = & B01110100 : B = &O164
20 PRINT A, B
RUN
116         116
```

### 3.48. Команда LOAD

LOAD «CAS : имя файла»

Команда LOAD загружает программу с НКМЛ. Имя программы не должно быть более шести символов. Если имя файла не указано, то загружается текущая программа, т. е. та программа, которая лежит в данный момент под считывающей головкой. Если имя программы не совпадает с заданным, то эта программа пропускается и происходит поиск следующей программы на НКМЛ. Если имя файла определено не по правилам, то появляется сообщение об ошибке, и загрузка не выполняется.

Команда LOAD стирает все переменные и программные строки, находящиеся в текущий момент в памяти перед загрузкой программы.

*Пример:*

LOAD «CAS : ENT»

загружает программу ENT.

### 3.49. Оператор LOCATE

LOCATE [позиция] [, [строка] [, курсор]]

где позиция — номер позиции в строке. Числовое выражение в диапазоне от 1 до 64;

строка — номер строки экрана. Числовое выражение в диапазоне от 1 до 16;

курсор — значение, определяющее, виден курсор или нет, 0 — для выключенного, 1 — для включенного курсора.

Оператор устанавливает текстовой курсор в заданную позицию, включает и выключает текстовой курсор. Операторы INPUT и PRINT на экране дисплея начинают размещать символы по определенным позициям.

Чтобы выключить курсор, можно использовать LOCATE,, 0. Для включения курсора используется LOCATE,, 1. Любые значения, заданные вне указанного диапазона, в результате вызовут сообщение об ошибке НЕВЕРЕН ВЫЗОВ ФУНКЦИИ.

*Пример:*

```
10 REM ПЕРЕСЛАТЬ КУРСОР В НАЧАЛЬНУЮ ПОЗИЦИЮ В ВЕРХНИЙ ЛЕВЫЙ УГОЛ
20 LOCATE 1, 1
30 REM ТЕКСТОВОЙ КУРСОР СДЕЛАТЬ НЕВИДИМЫМ, ПОЗИЦИЯ ОСТАЕТСЯ НЕИЗМЕННОЙ
40 LOCATE,, 0
```

### 3.50. Оператор LUT

LUT X (i)

где X (i) — имя первого элемента целочисленного массива, содержащего элементы X (i), X (i + 1),..., X (i + 15).

Оператор программирует таблицу цветов. Элементы массива с индексами от i до i +15 содержат номера физических цветов, а их порядковый номер соответствует логическим номерам цветов.

При начальной инициализации логические номера цветов в диапазоне 0—7 соответствуют цветам графики (без алфавитно-цифровых символов), а в диапазоне 8—15 — цветам алфавитно-цифровой информации, расположенной поверх графической информации.

Соответствие номеров физическим цветам при начальной инициализации представлено в операторе COLOR.

*Пример:*

```
10 CLS : DIM X% (15) : PSET (0, 0), 0
20 FOR I = 0 TO 7 : X % (I) = I
30 LINE — STEP (60, 30), I, B F : NEXT
40 FOR I = 8 TO 15 : X % (I) = 15 : NEXT
50 FOR T = 0 TO 100
60 FOR I = 1 TO 7
70 FOR L = 1 TO 50 : NEXT
80 X % (I) = I—1 : LUT X % (0) : X % (I) = I : NEXT
90 LUT X % (0)
100 NEXT
```

### 3.51. Оператор и функция MID ¤

Y¤ = MID¤(X¤, n[, m])

как оператор

MID¤(X¤, n[, m]) = Y¤

где n — целочисленное выражение 1—255;

m — целочисленное выражение 0—255.

Функция MID¤ возвращает требуемую часть заданной строки. Оператор MID¤ замещает часть одной строки другой строкой. Функция возвращает строку длиной m символов из строки X¤, начиная с n-го символа. Если m опущено или в строке меньше m символов справа от n-го символа, то возвращаются все самые правые символы, начиная с n-го. Если m = 0 или n > LEN (X¤), то MID¤ возвращает пустую строку. Символы в строке X¤, начиная с n-го, замещаются символами строки Y¤. m — число символов строки Y¤, которые будут использованы. Если m опущено, то используется вся строка Y¤. Длина строки X¤ не изменяется в зависимости от того, задано число или нет. Например, если X¤ длиной 4 символа, а Y¤ длиной 5 символов, то после замещения X¤ будет содержать первые 4 символа строки Y¤. Если n или m вне диапазона, то появляется сообщение об ошибке НЕВЕРЕН ВЫЗОВ ФУНКЦИИ.

*Пример:*

```
10 A¤ = «СЕЙЧАС»
20 B¤ = «УТРО ВЕЧЕР ПОЛДЕНЬ»
30 PRINT A¤, MID¤(B¤, 6,5)
RUN
СЕЙЧАС ВЕЧЕР
Ok
```

### 3.52. Команда MOTOR

MOTOR [ON/OFF]

где ON — включение двигателя магнитофона;

OFF — выключение двигателя магнитофона.

Команда включает или выключает двигатель НКМЛ.

### 3.53. Команда NEW

NEW

Команда стирает программу, находящуюся в текущий момент в памяти, и очищает все переменные. Команда NEW обычно используется для освобождения памяти перед введением новой программы. Интерпретатор всегда возвращается на уровень команд после выполнения команды.

### 3.54. Функция OCT ¤

Y¤ = OCT¤(n)

Функция возвращает строку, представляющую собой восьмиричное значение десятичного аргумента, n — округляется перед выполнением.

*Пример:*

```
PRINT OCT□(24)
30
Ok
```

### 3.55. Оператор ON ERROR

ON ERROR GOTO номер строки

Оператор позволяет перехватывать ошибки (не прерывать программу) и указывает начало подпрограммы их обработки. После выполнения этого оператора все обнаруженные ошибки будут приводить к переходу на указанную строку.

Для отказа от обработки ошибки можно выполнить оператор ON ERROR GOTO 0. Перехват ошибки в подпрограмме обработки ошибок невозможен. Оператор RESUME используется для выхода из подпрограммы обработки ошибок.

*Пример:*

```
10 ON ERROR GOTO 100
20 LOCATE 0,6 : PRINT «ВЫВОД СООБЩЕНИЯ»
30 END
100 IF ERR = 5 THEN PRINT «ОШИБКА ПОЗИЦИОНИРОВАНИЯ»
110 RESUME
```

### 3.56. Операторы ON ... GOSUB и ON ... GOTO

ON n GOSUB номер строки [, номер строки]...

ON n GOTO номер строки [, номер строки]...

Эти операторы осуществляют переход на один из заданных номеров строк в зависимости от вычисленного выражения n. Значение n определяет, какой номер строки в списке будет использован для перехода. Если необходимо, n округляется до целого числа. В операторе ON... GOSUB каждый номер строки в списке должен быть номером первой строки подпрограммы, поэтому необходимо иметь в подпрограмме оператор RETURN, чтобы вернуться в строку, следующую за оператором ON... GOSUB. Если значение n равно 0 или больше количества номеров строк в строке (но меньше или равно 255), то продолжается выполнение программы со следующего исполняемого оператора. Если значение отрицательно или больше 255, то появляется сообщение об ошибке НЕВЕРЕН ВЫЗОВ ФУНКЦИИ.

*Пример:*

```
100 ON L GOTO 150, 300, 450, 500
110 ON A GOSUB 1300, 1400
.
.
1300 REM НАЧАЛО ПОДПРОГРАММЫ A = 1
.
.
1390 RETURN
```

### 3.57. Оператор PAINT

PAINT [STEP] (X, Y) [,цвет закраски [,цвет границы]]

где STEP — позволяет указывать координаты относительно последней точки;

X, Y — координаты точки в пределах области, которая будет заполняться. Координаты могут быть заданы в абсолютной и относительной форме. Эта точка будет использоваться как начальная;

цвет закраски — цвет, которым будет закрашена область экрана, целое число в диапазоне 0—7;

цвет границы — цвет границы рисунка, подлежащего закраске, лежит в диапазоне 0—7.

Оператор PAINT заполняет область экрана выбранным цветом. Координаты могут быть заданы в абсолютной и относительной форме.

По умолчанию цвет границы равен цвету закраски.

Начальная точка (X, Y) в операторе PAINT должна быть внутри рисунка, который закрашивается. Если заданная точка уже имеет цвет границы, то оператор PAINT не будет действовать. Если цвет закраски опущен, то используется цвет переднего фона, установленный оператором COLOR.

PAINT может окрашивать любой рисунок, но «зубчатые» границы на рисунке будут увеличивать количество стекового пространства, требуемого оператором PAINT.

*Пример:*

```
20 CIRCLE (250, 100), 100,2,,,0.5
30 PAINT (250, 100), 1,2
```

### 3.58. Оператор PCLS

PCLS

Оператор очищает графическое запоминающее устройство (ГЗУ) в цвет заднего фона, стирает графическую информацию.

### 3.59. Функция PEEK

Y = PEEK (n)

где n — целое число в диапазоне 0—65535.

Функция возвращает байт, считанный из указанного адреса памяти. Возвращенное значение будет целым десятичным числом в диапазоне 0—255.

PEEK — дополнительная функция к оператору POKE. Если значение числа n вне указанного диапазона, то выдается сообщение об ошибке.

*Пример:*

```
10 POKE 24535,31
20 Y = PEEK (24535)
```

```
30 PRINT Y
RUN
31
Ok
```

### 3.60. Функция POINT

Y = POINT (X, Y)

где X, Y — координаты точки в абсолютной форме.

Функция POINT возвращает число от 0 до 7, определяющее цвет заданной точки на экране дисплея. Если точка задана вне диапазона экрана, то возвращается минус 1.

*Пример:*

```
10 LINE (100, 100) — (200, 200), 4, BF
20 I = 50
30 IF POINT (I, I) <>0 THEN PRESET (I, I) ELSE
PSET (I, I)
```

### 3.61. Оператор POKE

POKE n, m

где n — число в диапазоне 0—65535;

m — число в диапазоне 0—255.

Оператор записывает байт данных m по адресу памяти n. Дополнительной функцией к оператору POKE является PEEK. PEEK и POKE полезны для эффективного хранения данных, загрузки подпрограмм на машинном языке, передачи аргументов и результатов. Интерпретатор не делает никакой проверки адресов. Если значения n и m лежат вне указанных диапазонов, то выдается сообщение об ошибке.

*Пример:*

```
10 POKE 106,0
```

### 3.62. Функция POS

Y = POS (n)

Функция возвращает текущую горизонтальную позицию курсора. Самая левая позиция — I, n — фиктивный аргумент.

*Пример:*

```
IF POS (0) > 60 THEN PRINT CHR¤(13)
```

### 3.63. Оператор PRINT

PRINT [список выражений]

Оператор распечатывает данные на экране дисплея. Если список выражений опущен, то распечатывается строка пробелов. Выражения в списке могут быть числовыми и (или) строковыми. Строковые константы должны быть заключены в кавычки. Позиция каждого напечатанного элемента определяется знаками препинания, используемыми для разделения элементов списка.

Интерпретатор разделяет строку на зоны печати по четырнадцать символов каждая. Запятая в списке выражений заставляет печататься следующее значение с начала следующей зоны. Точка с запятой заставляет печататься следующее значение со следующей позиции с учетом пробелов при печати числовых данных. Если запятая или точка с запятой заканчивают список выражений, то следующий оператор PRINT начинает печать в той же строке через заданное число пробелов. Если длина печатаемой строки больше ширины экрана, то печать продолжается на следующей физической строке. За печатаемыми числами всегда следует пробел: положительным числам предшествует пробел, отрицательным — знак минус. Числа одинарной точности могут быть представлены шестью или меньше цифрами в формате с фиксированной точкой, но с меньшей точностью, чем если бы они могли быть представлены в формате с плавающей точкой. Они выводятся в формате с фиксированной точкой или целочисленном.

*Пример:*

```
10 INPUT X
20 PRINT X; «В КВАДРАТЕ»; X ^ 2; «, A »;
30 PRINT X; «В КУБЕ»; X ^ 3
RUN
? 9
9 В КВАДРАТЕ 81, А 9 В КУБЕ 729
Ок
RUN
? 21
21 В КВАДРАТЕ 441, А 21 В КУБЕ 9261
Ок
10 FOR X = 1 TO 5
20 J = J + 5
30 K = K + 10
40 PRINT J; K;
50 NEXT X
RUN
5 10 10 20 15 30 20 40 25 50
Ок
```

### 3.64. **Оператор** PRINT USING

PRINT USING X□, список выражений [ ; , ]

Оператор печатает строки и числа, используя заданный формат.

Список выражений состоит из строковых или числовых выражений, которые будут печататься. Выражения в списке разделяются запятыми или точками с запятой. X□ — строковая константа или переменная, которая состоит из специальных символов формата. Эти символы определяют формат и поля печатаемых символов.

Для печати строк применяются следующие символы формата:

! — первый символ в заданной строке должен быть напечатан;

\ n пробелов \ — 2 + n символов из строки должны быть напечатаны. Если обратные слеши (\\) не содержат пробелов, то будут напечатаны два символа, если содержат один пробел, то три символа, и т. д. Если строка длиннее, чем поле, то лишние символы игнорируются. Если поле длиннее, то строка будет вводиться слева, а справа добавляться пробелами.

*Пример:*

```
10 A$ = «LOOK» : B$ = «OUT»
30 PRINT USING «!»; A$; B$
50 PRINT USING «\␣␣\»; A$; B$
70 PRINT USING «\␣␣\␣»; A$; B$; «!!»
RUN
LO
LOOK OUT
LOOK OUT !!
Ok
```

Для печати чисел используются следующие символы формата:

**#** — используется для представления каждой числовой позиции. Числовые позиции всегда заполнены. Если число, которое будет печататься, содержит меньше цифр, чем определенных позиций, то число будет вводиться в поле справа и ему будут предшествовать пробелы. Точка может быть введена в любую позицию поля. Если строка формата определяет, что цифры должны предшествовать точке, то перед точкой обязательно будут печататься цифры (если необходимо, будут печататься нули). При необходимости числа округляются.

*Пример:*

```
PRINT USING «##.##»; .78
 0.78
PRINT USING «##.##»; D87.654
 87.65
PRINT USING «##.##»; 10.2,5.3,66.789,.234
 10.20    5.30    66.79    0.23
```

\+ в начале (конце) строки формата используется для печати знака числа («+» или «—») перед (после) числом;

— в конце строки формата используется для печати отрицательного числа со знаком минус.

*Пример:*

```
PRINT USING «+ ##.##»; —68.95,2.4,55.6, — .9
 —68.95    +2.40    +55.60    —0.90
```

```
PRINT USING «##.##—»; 68.95,22.449,—7.08
 68.95   22.45   7.08—
```

**— двойная звезда ** в начале строки формата используется для заполнения звездочками передних пробелов в числовом поле и для определения позиций при использовании больше двух цифр.

*Пример:*

```
PRINT USING «**##.#»; 12.39,—0.9
 *12.4  *—0.9
```

$$ — двойной знак $$ для печати знака $ непосредственно слева от числа; $$ определяет позиции более двух цифр, одна из которых $. Экспоненциальный формат не может быть использован со знаком $$. Отрицательные числа не могут быть использованы, если знак «—» не указан справа.

*Пример:*

```
PRINT USING «$$####.##»; 456.78
 $456.78
```

**$ — этот знак в начале строки формата комбинирует действия вышеназванных двух символов формата. Передние пробелы будут заполняться звездочками, а знак $ будет печататься перед числом. Знак **$ определяет позиции более трех цифр, одна из которых $.

*Пример:*

```
PRINT USING «**$##.##»; 2.34
 ***$ 2.34
```

, — запятая, которая находится слева от точки в строке формата, используется для печати запятой слева от каждой третьей цифры слева или справа от точки. При использовании экспоненциального формата запятая не применяется.

*Пример:*

```
PRINT USING «####,.##» ; 1234.5
1,234.50
PRINT USING «####.,##» ; 1234.5
1234.50,
```

^^^^ — этот знак может быть размещен после символов цифровой позиции для определения экспоненциального формата. Этот знак резервирует пространство для E + nn или D + nn, которые будут печататься. Значащие цифры вводятся слева. Если задан начальный или конечный знак «+» или «—», то одна цифровая позиция будет использована слева или справа от точки для печати пробела или знака «—».

*Пример:*

```
PRINT USING «##.##^^^^»; 234.56
```

```
2.35E + 02
PRINT USING «###^^^^—»;—88888
.889E + 05—
PRINT USING «+.##^^^^»;123
+ .12E + 03
```

Если число, которое должно печататься, больше определенного числового поля, то в начале числа печатается знак %. Если при округлении число превышает поле, то знак % будет печататься в начале округленного числа.

*Пример:*

```
PRINT USING «##.##»;111.22
%111.22
PRINT USING «.##»;.999
%1. 0 0
```

Если число заданных цифр превышает 24, то появляется сообщение об ошибке НЕВЕРЕН ВЫЗОВ ФУНКЦИИ.

*Пример:*

```
PRINT USING «ЭТО ПРИМЕР ##»; 1
ЭТО ПРИМЕР 1
```

### 3.65. Оператор PSET

PSET [STEP] (X, Y) [, цвет]

где STEP — позволяет указывать координаты относительно последней точки;
X, Y — координата точки;
цвет — цвет, целое число в диапазоне от 0 до 7.

Оператор на экране дисплея рисует точку в определенной позиции, заданной координатами (X, Y). Координата точки может быть задана в абсолютной или относительной форме. Если в операторе PSET пропущен цвет, то используется цвет переднего фона, установленный оператором COLOR.

### 3.66. Оператор PRESET

PRESET [STEP] (X, Y) [, цвет]

где STEP — позволяет указывать координаты относительно последней точки;
X, Y — координаты точки;
цвет — цвет, целое число в диапазоне 0—7.

Оператор PRESET аналогичен оператору PSET с той разницей, что в качестве опущенного параметра цвета принимается цвет заднего фона.

Если значение цвета больше 7, появляется сообщение об ошибке НЕВЕРЕН ВЫЗОВ ФУНКЦИИ. Если в операторах PSET и PRESET координаты точек заданы вне области экрана дисплея, то не производится никакого действия и не выводится сообщений об ошибках.

*Пример:*

```
10 REM НАРИСОВАТЬ ДИАГОНАЛЬ
20 FOR 1 = 0 TO 100
30 PSET (I, I)
40 NEXT
50 REM СТЕРЕТЬ ЛИНИЮ, УСТАНОВИТЬ ЦВЕТ КАЖДОЙ ТОЧКИ В НОЛЬ
60 FOR I = 100 TO 0 STEP-1
70 PRESET (I, I)
80 NEXT
```

### 3.67. Оператор READ

READ список переменных

Оператор считывает значения из оператора DATA и присваивает их переменным.

Переменные в списке разделяются запятыми.

Типы переменных в READ должны соответствовать типам констант в DATA, иначе появляется сообщение ОШИБКА СИНТАКСИСА. Один оператор READ может иметь доступ к одному и более операторам DATA или несколько операторов READ могут иметь доступ к одному оператору DATA. Если число переменных оператора READ превышает число элементов в операторе (операторах) DATA, то появляется сообщение об ошибке ВНЕ DATA. Если число переменных оператора READ меньше числа элементов оператора (операторов) DATA, то следующие READ будут начинать чтение данных с первого несчитанного оператора DATA. Если больше нет операторов READ, то лишние данные операторов DATA игнорируются. Чтобы заново прочитать операторы DATA с начала, используется оператор RESTORE.

*Пример:*

```
10 PRINT «СТРАНА», «ГОРОД», «ИНДЕКС»
20 READ C□, S□, Z
30 DATA СССР, МОСКВА, 115470
40 PRINT C□, S□, Z
RUN
СТРАНА          ГОРОД          ИНДЕКС
СССР            МОСКВА         115470
Ок
```

### 3.68. Оператор RELOC

RELOC (X, Y)

где X, Y — координата начальной точки.

Оператор устанавливает начало координат при работе с графическими операторами.

*Пример:*

```
10 RELOC (100, 100)
20 LINE (0, 0) — (200, 100), 4, BF
```

### 3.69. Оператор REM

REM комментарий

где комментарий — любая последовательность символов.

Этот оператор вводит в программу комментарии. Оператор REM — неисполняемый оператор, однако он увеличивает время выполнения и занимает пространство в памяти. Операторами GOTO и GOSUB может быть осуществлен переход на операторы REM, и выполнение программы будет продолжаться с первого после REM исполняемого оператора.

*Пример:*

```
120 REM вычисление скорости
130 FOR I = 1 TO 20
140 SUM = SUM + V (I)
150 NEXT I
160 A = 0 : REM ИНИЦИАЛИЗАЦИЯ A
170 PRINT SUM
180 REM программа вычисления СКОРОСТИ
```

### 3.70. Команда RENUM

RENUM [новый номер] [, [старый номер] [, шаг]]

где новый номер — номер первой строки, которая будет использоваться в качестве новой последовательности номеров строк. По умолчанию 10;

старый номер — номер строки текущей программы, с которого начнется перенумерация. По умолчанию номер первой строки программы;

шаг — шаг, используемый в новой последовательности. По умолчанию 10.

Команда перенумеровывает строки программы. RENUM также изменяет все номера строк в операторах GOTO, GOSUB, THEN, ELSE, ON ... GOTO, ON ... GOSUB. Если после какого-либо из этих операторов появится несуществующий в программе номер строки, то выдается сообщение об ошибке НЕОПРЕДЕЛЕННЫЙ НОМЕР СТРОКИ. Обращение к некорректному номеру строки не изменяется командой RENUM.

*Пример:*

```
RENUM      300,,50
Ok
```

### 3.71. Оператор RESTORE

RESTORE [номер строки]

Оператор разрешает операторам DATA считываться заново с определенной строки. После того, как выполнится оператор RESTORE, следующий оператор READ имеет доступ к первому элементу первого оператора DATA программы. Если номер

строки задан, то следующий оператор READ имеет доступ к первому элементу в заданном операторе DATA.

*Пример:*

```
10 READ A. B. C
20 RESTORE
30 READ D, E, F
40 DATA 57, 68, 49
50 PRINT A, B, C, D, E, F
RUN
57        68        49        57        68
49
Ок
```

### 3.72. Оператор RESUME

RESUME [0]
RESUME NEXT
RESUME [номер строки]

Оператор продолжает выполнение программы после выполнения процедуры устранения ошибок. В зависимости от того, где выполнение должно быть восстановлено, используют один из следующих форматов:

RESUME [0]

Выполнение восстанавливается с оператора, который вызвал ошибку.

RESUME NEXT

Выполнение восстанавливается с оператора, следующего непосредственно за оператором, который вызвал ошибку.

RESUME [номер строки]

Выполнение восстанавливается с оператора с заданным номером строки. Оператор RESUME, для которого нет подпрограммы обработки ошибки, вызывает сообщение об ошибке RESUME БЕЗ ОШИБКИ.

*Пример:*

```
10 ON ERROR GOTO 900
:
900 IF (ERR = 230) AND (ERL = 90) THEN PRINT «ПОПРОБУЙТЕ ЕЩЕ РАЗ»: RESUME 80
```

### 3.73. Оператор RETURN

RETURN

Оператор возвращает управление обратно в программу из подпрограммы и осуществляет возврат к оператору, следующему за тем, из которого проводилось обращение к подпрограмме.

*Пример:*

```
50 GOSUB 400
:
```

```
400 REM ПОДПРОГРАММА
:
500 RETURN
```

### 3.74. Функция RIGHT¤

Y¤ = RIGHT¤ (X¤, n)

Функция возвращает n самых правых символов строки X¤. Если n > LEN (X¤), то возвращается строка X¤. Если n = 0, то возвращается пустая строка.

*Пример:*

```
10 A¤ = «АБВГДЕЖЗ ABCDEFG»
20 PRINT RIGHT ¤ (A ¤, 7)
RUN
ABCDEFG
Ok
```

### 3.75. Функция RND

Y = RND (n)

Функция возвращает случайное число между 0 и 1. RND (0) повторяет последнее сгенерированное число.

*Пример:*

```
10 FOR I = 1 TO 3
20 PRINT RND (I),
30 NEXT I
RUN
.6291626        .1948297        .6305799
```

### 3.76. Команда RUN

RUN [номер строки]

Команда начинает выполнение программы, находящейся в памяти в текущий момент. Если задан номер строки, то выполнение программы начинается с этой строки. А если номер строки не задан, то выполнение начинается с первой строки программы.

### 3.77. Команда SAVE

SAVE «CAS : имя файла»

Команда сохраняет программный файл на НКМЛ. Имя файла не должно быть больше 6 символов. Если имя не указывается, то по умолчанию на ленту будет записываться имя ЪЪЪЪЪ.

*Пример:*

```
SAVE «CAS : PROBA»
```

### 3.78. Оператор SCREEN

SCREEN [N] [, M]

Оператор переключает графические страницы памяти,

где N — номер страницы отображения;
M — номер страницы чтения/записи;
N, M — целые числа в диапазоне 0...3.

Если параметры N и M не заданы, то выдается сообщение ОШИБКА СИНТАКСИСА

*Пример:*

```
10 SCREEN 0, 0
20 COLOR 4, 3
30 LINE — (100, 200), 5     ; рисует линию в 0 странице
40 SCREEN 0, 1
50 CIRCLE (100, 200), 100 ; рисует окружность в 1 странице
60 SCREEN 1, 1   ; отображается 1 страница
```

### 3.79. Функция SGN

Y = SGN (X)

Функция возвращает математическую функцию знака.
Если X положительно, то SGN (X) возвращает 1.
Если X равно 0, то SGN (X) возвращает 0.
Если X отрицательно, то SGN (X) возвращает —1.

*Пример:*

```
50 ON SGN (X) + 2 GOTO 100, 200, 300
```

### 3.80. Функция SIN

Y = SIN (X)

Функция SIN вычисляет тригонометрическую функцию синус. X — угол, заданный в радианах. Если надо перевести градусы в радианы, то результат вычисления умножается на PI/180, где PI = 3.141593. Функция SIN (X) вычисляется с одинарной точностью.

*Пример:*

```
PRINT SIN (1. 5)
.997495
Ok
```

### 3.81. Функция SPACE¤

Y¤ = SPACE¤(n)

где n — число в диапазоне 0—255.

Функция возвращает строку, состоящую из n пробелов.

*Пример:*

```
10 FOR I = 1 TO 5
20 X¤ = SPACE¤(I)
30 PRINT X¤; I
40 NEXT I
RUN
```

```
1
 2
  3
   4
    5
Ок
```

### 3.82. Функция SPC

PRINT SPC (n)

где n — число в диапазоне 0—255.

Функция используется для печати n пробелов. Если n больше, чем заданная ширина устройства, то используется значение (n MOD ширина). Функция SPC может быть использована только с оператором PRINT.

*Пример:*

```
PRINT «БОЛЬШЕ», SPC (15) «ЧЕМ»
БОЛЬШЕ          ЧЕМ
Ок
```

### 3.83. Функция SQR

Y = SQR (X)

где X — число, должно быть больше или равно 0.

Функция возвращает квадратный корень числа X.

*Пример:*

```
10 FOR X = 10 TO 25 STEP 5
20 PRINT X, SQR (X)
30 NEXT
RUN
10          3.16228
15          3.87298
20          4.47214
25          5
Ок
```

### 3.84. Оператор STOP

STOP

Оператор заканчивает выполнение программы и возвращает управление на уровень команд. Чтобы закончить выполнение, оператор STOP может быть использован в любом месте программы. Когда встречается оператор STOP, на экране появляется сообщение

СТОП В n

где n — номер строки, на которой встретился оператор STOP.

Выполнение программы восстанавливается командой CONT.

*Пример:*

```
10 INPUT A, B, C
20 K = A ↑ 2 * 5.3 : L = B ↑ 3/.26
```

```
30 STOP
40 M = C*K + 100 : PRINT M
RUN
? 1, 2, 3
СТОП В 30
PRINT L
30.7692
Ок
CONT
115.9
Ок
```

### 3.85. Функция STR¤

X¤ = STR¤(X)

Функция возвращает строковое представление значения X. Функция VAL является инверсией функции STR¤

*Пример:*

```
5 REM АРИФМЕТИКА
10 INPUT «ВВЕДИТЕ ЧИСЛО»; N
20 REM ВЫБОР ЗАВИСИТ ОТ КОЛИЧЕСТВА ЦИФР
30 ON LEN (STR¤(N) ) GOSUB 60, 100, 200
```

### 3.86. Функция STRING¤

Y¤ = STRING¤(n, m)
Y¤ = STRING¤(n, X¤)

где n и m — числа в диапазоне 0—255.

Функция возвращает строку длиной n, все символы которой имеют заданный код КОИ-8 или являются первым символом строки X¤. При n = 0 возвращается пустая строка.

Если значения n и m лежат вне указанного диапазона, то выдается сообщение об ошибке НЕВЕРЕН ВЫЗОВ ФУНКЦИИ.

*Пример:*

```
10 X¤ = STRING¤ (10, 45)
20 PRINT X¤; «ОТЧЕТ»; X¤
RUN
---------- ОТЧЕТ ----------
Ок
10 X¤ = «ABCD»
20 Y¤ = STRING ¤ (10, X¤)
30 PRINT Y¤
RUN
AAAAAAAAAA
Ок
```

### 3.87. Оператор SWAP

SWAP переменная 1, переменная 2

Оператор обменивает значения двух переменных. Могут быть обменены переменные любого типа, но обе переменные должны

быть одного типа, иначе появится сообщение об ошибке ОШИБОЧНЫЙ ТИП. Эти переменные могут быть элементами массива.

*Пример:*

```
10 A¤ = «ONE␣»: B¤ = «ALL␣»: C¤ = «FOR␣»: PRINT A¤; C¤; B¤
20 SWAP A¤, B¤: PRINT A¤; C¤; B¤
RUN
ONE FOR ALL
ALL FOR ONE
Ok
```

### 3.88. Функция TAB

PRINT TAB (n)

где n — число в диапазоне 1—255.

Эта функция осуществляет табуляцию в n-ю позицию. Если позиция текущей печати больше, то TAB перемещает курсор в позицию на следующей строке. Позиция 1 — это самая левая позиция, заданная ширина — самая правая позиция. Функция TAB может использоваться только с оператором PRINT.

Если n — нецелое число, то происходит округление до целого числа.

Если n лежит вне указанного диапазона, то выдается сообщение об ошибке НЕВЕРЕН ВЫЗОВ ФУНКЦИИ.

*Пример:*

```
10 PRINT «ИМЯ» TAB (25) «КОЛИЧЕСТВО»: PRINT
20 READ A¤, B¤
30 PRINT A¤, TAB (25) B¤
40 DATA Л. М. ЯНШИН, 25. 68
RUN
ИМЯ                       КОЛИЧЕСТВО
Л. М. ЯНШИН               25. 68
Ok
```

### 3.89. Функция TAN

Y = TAN (X)

Функция возвращает тригонометрическую функцию тангенса угла X. Угол X должен быть задан в радианах. Для перевода градусов в радианы результат вычисления умножается на PI/180, где PI = 3.141593. Функция TAN (X) вычисляется с одинарной точностью.

*Пример:*

```
10 Y = Q*TAN (X)/2
```

### 3.90. Команды TRON и TROFF

```
TRON
TROFF
```

Команда TRON включает режим трассировки. По этой команде будет печататься каждый номер строки программы, которая выполняется. Номера заключаются в квадратные скобки. Команда TROFF (или команда NEW) отключает режим трассировки.

*Пример:*

```
10 K = 10
20 FOR J = 1 TO 2
30 L = K + 10
40 PRINT J; K; L
50 K = K + 10
60 NEXT
70 END
TRON
Ok
RUN
[10] [20] [30] [40] 1          10          20
[50] [60] [30] [40] 2          20          30
[50] [60] [70]
Ok
TROFF
Ok
```

### 3.91. Функция USR

Y = USR [цифра] (X)

где цифра — 0—9. Соответствует цифре в операторе DEFUSR для заданной подпрограммы.

Функция USR вызывает указанную подпрограмму на машинном языке с фиктивным аргументом X. Если цифра опущена, то принимается USR 0.

*Пример:*

```
50 C = USR (0)
```

### 3.92. Функция VAL

X = VAL (X$)

Функция возвращает числовое значение строки X$. Функция VAL убирает начальные пробелы, табуляцию и перевод строки из строки аргумента, для того чтобы вычислить результат. Например, VAL («—3») возвращает —3. Если X — не число, то VAL (X$) возвращает 0.

## ИНСТРУКЦИИ ЯЗЫКА БЕЙСИК

### 1. Команды

Таблица 1

| Команда | Формат | Действие |
|---|---|---|
| AUTO | AUTO [номер строки] [,[приращение]] | Автоматически нумерует строки программы |
| CLEAR | CLEAR [числовое выражение] | Очищает переменные, устанавливает размер строкового пространства |
| COM | COM «текст» | Заполняет почтовый ящик и устанавливает признак заполнения |
| CONT | CONT | Продолжает выполнение после останова |
| DELETE | DELETE [номер строки] [— номер строки] | Стирает программные строки |
| EDIT | EDIT номер строки EDIT. | Обеспечивает редактирование строки |
| LIST | LIST [номер строки] [—[номер строки]] | Распечатывает программные строки на экране |
| LOAD | LOAD имя программы на кассете | Загружает программу с кассеты |
| NEW | NEW | Очищает память |
| RENUM | RENUM [новый номер строки] [,[старый номер строки] [,приращение]] | Перенумеровывает программные строки |
| RUN | RUN [номер строки] | Выполняет программу |
| SAVE | SAVE имя программы на кассете | Сохраняет программу на кассете |
| TRON | TRON | Включает трассировку программы |
| TROFF | TROFF | Выключает трассировку программы |

Таблица 2

## 2. Операторы

| Оператор | Формат | Действие |
|---|---|---|
| BEEP | BEEP | Воспроизводит короткий звуковой сигнал |
| CIRCLE | CIRCLE [STEP] (X, Y), радиус [, [цвет] [, [начало] [, [конец] [, аспект]]]] | Рисует эллипс или дугу любого размера и цвета |
| CLS | CLS | Очищает текстовый экран |
| COLOR | COLOR [цвет переднего плана] [, цвет фона] | Устанавливает цвета переднего и заднего фона |
| DATA | DATA константа [, константы]... | Задает последовательность данных для операторов READ |
| DEFINT<br>DEFSNG<br>DEFDBL | DEF тип буква [— буква] [, буква [— буква]]... (где тип — это INT, SNG, DBL или STR) | Определяет по умолчанию типы переменных |
| DEFFN | DEFFN имя [(переменная [, переменная]...)] = выражение | Определяет функцию пользователя |
| DEFUSR | DEFUSR [цифра] = адрес | Указывает адрес программы на машинном языке |
| DIM | DIM имя массива (индекс [, индекс]...) [, имя массива (индекс...)]... | Резервирует место для массива и определяет максимальное значение индекса |
| END | END | Останавливает выполнение программы |
| ERASE | ERASE имя массива [, имя массива]... | Удаляет массивы |
| ERROR | ERROR код ошибки | Моделирует ошибку |
| FOR | FOR переменная = начальное значение TO конечное значение [STEP приращение] | Определяет начало цикла |
| GOSUB | GOSUB номер строки | Передает управление подпрограмме |
| GOTO | GOTO номер строки | Передает управление строке с определенным номером |
| IF | IF условное выражение THEN оператор [: оператор]... [ELSE оператор [: оператор]...] | Выполняет часть THEN или ELSE в зависимости от значения условного выражения |

| Оператор | Формат | Действие |
|---|---|---|
| INPUT | INPUT [«подсказка»;] переменная [, переменная] | Считывает данные с клавиатуры |
| LET | [LET] переменная = выражение | Присваивает значение выражения переменной |
| LINE | LINE [[STEP] (X1, Y1)] —[STEP] (X2, Y2) [, [цвет] [, B [F]]] | Рисует линию или прямоугольник на графическом экране |
| LINE INPUT | LINE INPUT [«подсказка»;] строковая переменная | Считывает строку с клавиатуры |
| LOCATE | LOCATE [позиция в строке], [номер строки], [переключатель] | Устанавливает курсор в заданной позиции на текстовом экране |
| LUT | LUT X% (I) | Программирует таблицу цветов |
| MID$ | X$ = MID$ (строковое выражение, начальная позиция [, количество символов]) | Присваивает часть значений строкового выражения строковой переменной |
| MOTOR | MOTOR [ON] | Включает мотор кассетного магнитофона |
| | MOTOR [OFF] | Выключает мотор кассетного магнитофона |
| NEXT | NEXT [переменная] [, переменная] ... | Оканчивает циклы |
| ON ERROR | ON ERROR GOTO номер строки | Задает номер строки для передачи управления при появлении ошибки |
| ON ... GOSUB | ON числовое выражение GOSUB номер строки [, номер строки] ... | Переходит к подпрограмме |
| ON ... GOTO | ON числовое выражение GOTO номер строки [, номер строки] ... | Переходит на определенную строку |
| PAINT | PAINT [STEP] (X, Y) [, цвет [, цвет границы]] | Заполняет область графического экрана определенным цветом |
| PCLS | PCLS | Очищает графический экран |
| POKE | POKE адрес, байт | Записывает байт данных по адресу |
| PRESET | PRESET [STEP] (X, Y) [, цвет] | Рисует точку определенного цвета |
| PRINT | PRINT [USING формат;] список значений [;] или [,] | Выводит данные на экран |

| Оператор | Формат | Действие |
|---|---|---|
| PSET | PSET [STEP] (X, Y) [, цвет] | Рисует точку определенного цвета |
| READ | READ переменная [, переменная]... | Присваивает переменным значения из последовательности данных, заданных операторами DATA |
| RELOC | RELOC (X, Y) | Устанавливает начало координат при работе с графическими операторами |
| REM | REM любой текст | Помещает комментарий в текст программы |
| RESTORE | RESTORE [номер строки] | Снова устанавливает указатель DATA для считывания по READ |
| RESUME | RESUME [0]<br>RESUME NEXT<br>RESUME номер строки | Возвращает управление из программы обработки ошибок |
| RETURN | RETURN | Возвращает управление программе |
| SCREEN | SCREEN [номер страницы отображения] [, номер рабочей страницы] | Задает рабочие графические страницы |
| STOP | STOP | Прерывает выполнение программы |
| SWAP | SWAP переменная 1, переменная 2 | Обменивает значения двух переменных |

## 3. Функции

Таблица 3

| Функция | Формат | Действие |
|---|---|---|
| ABS | X = ABS (числовое выражение) | Возвращает абсолютное значение аргумента |
| ASC | X = ASC (строковое выражение) | Возвращает код КОИ-8 первого символа аргумента |
| ATN | X = ATN (числовое выражение) | Вычисляет арктангенс угла, заданного в радианах |
| BIN□ | X□ = BIN□ (целочисленное выражение) | Переводит целое число в двоичную строку |
| CDBL | X = CDBL (числовое выражение) | Осуществляет перевод аргумента в число двойной точности |
| CHR□ | X□ = CHR□ (числовое выражение) | Возвращает символ, код которого равен значению аргумента |

| Функция | Формат | Действие |
|---|---|---|
| CINT | X = CINT (числовое выражение) | Переводит аргумент в целое число |
| COS | X = COS (числовое выражение) | Вычисляет косинус угла, заданного в радианах |
| CSNG | X = CSNG (числовое выражение) | Переводит аргумент в число одинарной точности |
| CSRLIN | X = CSRLIN | Возвращает номер строки экрана, в которой находится курсор |
| ERL | X = ERL | Возвращает номер строки, в которой была ошибка |
| ERR | X = ERR | Возвращает код встретившейся ошибки |
| EXP | X = EXP (числовое выражение) | Экспонента числа |
| FIX | X = FIX (числовое выражение) | Возвращает целую часть аргумента |
| FRE | X = FRE (фиктивный аргумент) | Выдает сообщение о наличии свободной памяти |
| HEX$ | X$ = HEX$ (числовое выражение) | Переводит аргумент в строку шестнадцатиричных символов |
| INKEY$ | X$ = INKEY$ | Получает один символ с клавиатуры |
| INSTR | X = INSTR ([начало,] строка, подстрока) | Находит расположение подстроки в строке |
| INT | X = INT (числовое выражение) | Возвращает наибольшее целое значение, не превосходящее аргумент |
| LEFT$ | X$ = LEFT$ (строковое выражение, длина) | Возвращает левые символы из строки |
| LEN | Y = LEN (строковое выражение) | Возвращает длину заданной строки |
| LOG | X = LOG (числовое выражение) | Возвращает натуральный логарифм аргумента |
| MID$ | MID$ (строковое значение, начальный символ [, длина]) | Возвращает часть заданной строки |
| OCT$ | X$ = OCT$ (целочисленное выражение) | Переводит аргумент в строку восьмиричных цифр |
| PEEK | X = PEEK (адрес) | Возвращает байт данных, считанный из указанного адреса памяти |
| POINT | X = POINT (X, Y) | Возвращает цвет точки на графическом экране |

| Функция | Формат | Действие |
|---|---|---|
| POS | X = POS (фиктивный аргумент) | Возвращает позицию курсора в строке экрана |
| RIGHT$ | X = RIGHT$ (строковое выражение, длина) | Возвращает правые символы из строки |
| RND | X = RND (числовое выражение) | Возвращает случайное число между нулем и единицей |
| SGN | X = SGN (числовое выражение) | Возвращает —1, 0 или 1 в зависимости от знака аргумента |
| SIN | X = SIN (числовое выражение) | Вычисляет синус угла, заданного в радианах |
| SPACE$ | X$ = SPACE$ (целочисленное выражение) | Возвращает строку из определенного количества пробелов |
| SPC | PRINT SPC (число) | Возвращает последовательность пробелов |
| SQR | X = SQR (числовое выражение) | Возвращает квадратный корень аргумента |
| STR$ | X$ = STR$ (числовое выражение) | Переводит аргумент в строку символов |
| STRING$ | X$ = STRING$ (длина, числовое выражение)<br>X$ = STRING$ (длина, строковое выражение) | Повторяет символы |
| TAB | PRINT TAB (позиция) | Осуществляет табуляцию в указанную позицию |
| TAN | X = TAN (числовое выражение) | Вычисляет тангенс угла, заданного в радианах |
| USR | USR [цифра] (выражение) | Передает управление программе, находящейся по адресу, задаваемому оператором DEFUSR |
| VAL | X = VAL (строковое выражение) | Преобразует аргумент в число |

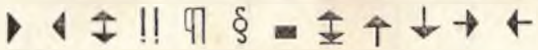
## СООБЩЕНИЯ ОБ ОШИБКАХ ИНТЕРПРЕТАТОРА БЕЙСИК

| Код ошибки | Сообщение | Комментарий |
|---|---|---|
| 1 | NEXT БЕЗ FOR | Переменная в операторе NEXT не соответствует ни одной переменной в уже выполненных операторах FOR |
| 2 | ОШИБКА СИНТАКСИСА | Обнаружена строка, содержащая некорректную последовательность символов (непарные скобки, ошибка в названии команды или оператора, неправильная пунктуация и т. д.) |
| 3 | RETURN БЕЗ GOSUB | Встречается оператор RETURN, для которого нет соответствующего оператора GOSUB |
| 4 | ВНЕ DATA | Выполняется оператор READ, когда отсутствуют операторы DATA с оставшимися несчитанными данными |
| 5 | НЕВЕРЕН ВЫЗОВ ФУНКЦИИ | В числовой или строковой функции указан неправильный параметр или используется отрицательный или слишком большой индекс массива |
| 6 | ПЕРЕПОЛНЕНИЕ | Численный результат вышел за допустимые пределы |
| 7 | НЕТ ПАМЯТИ | Программа слишком большая или имеет много циклов FOR, GOSUB или переменных |
| 8 | НЕОПРЕДЕЛЕННЫЙ НОМЕР СТРОКИ | В операторе GOTO, GOSUB или IF идет обращение к несуществующей строке |
| 9 | ИНДЕКС ВНЕ ДИАПАЗОНА | Идет обращение к элементу массива с помощью индекса, который находится вне границ массива, или с помощью неверного числа индексов |
| 10 | ПЕРЕОПРЕДЕЛЕНИЕ МАССИВА | Даны два оператора DIM для одного массива или оператор DIM дан после того, как размерность 10 по умолчанию была установлена для этого массива |
| 11 | ДЕЛЕНИЕ НА НОЛЬ | Обнаружено деление на 0 или возведение 0 в отрицательную степень |
| 12 | ОШИБОЧНАЯ ДИРЕКТИВА | Обнаружена попытка непосредственно выполнить оператор, имеющий смысл только в рамках программы |

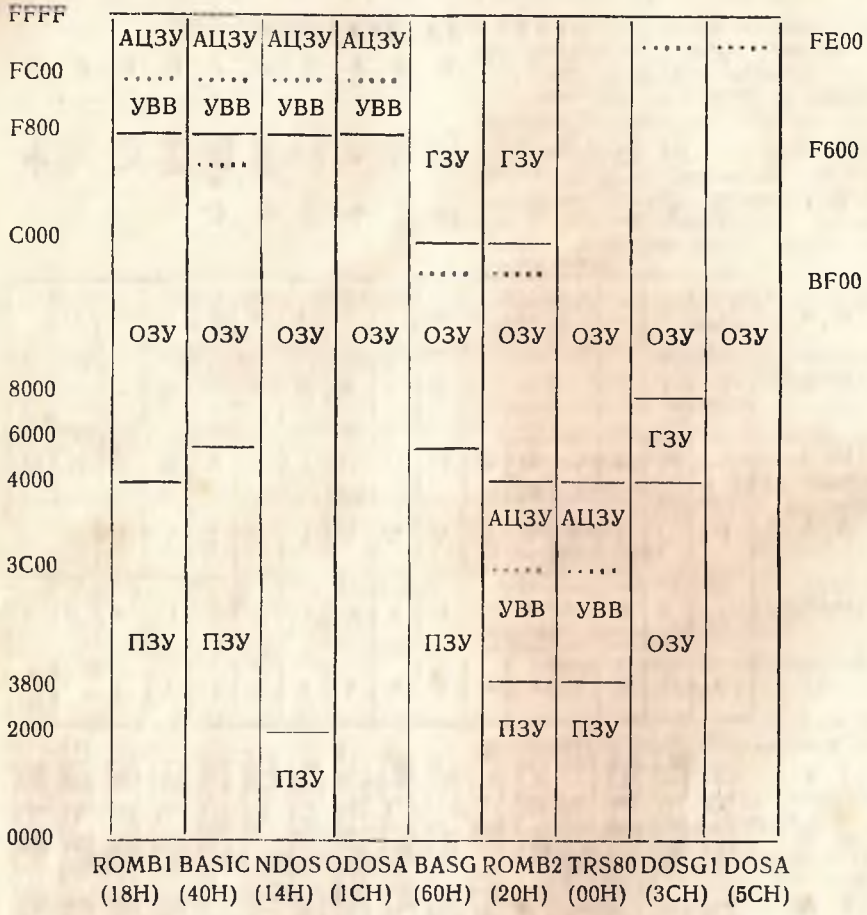
| Код ошибки | Сообщение | Комментарий |
|---|---|---|
| 13 | ОШИБОЧНЫЙ ТИП | Задается строковое значение там, где ожидалось числовое, или наоборот, или обнаружена попытка обменять значения одинарной и двойной точности (в операторе SWAP) |
| 14 | НЕТ ПАМЯТИ ДЛЯ СТРОК | Недостаточно памяти для размещения значений строковых переменных. Рекомендуется выполнить оператор CLEAR с параметром, указывающим большую длину строкового пространства |
| 15 | ДЛИННАЯ СТРОКА | Сделана попытка создать строку длиной более 255 символов |
| 16 | ОЧЕНЬ СЛОЖНАЯ СТРОКА | Строковое выражение слишком сложное или длинное. Выражение следует разбить на части |
| 17 | ОШИБКА CONT | Сделана попытка продолжить выполнение программы, которая:<br>1) была остановлена из-за ошибки;<br>2) была модифицирована во время останова;<br>3) не существует |
| 18 | НЕТ ФУНКЦИИ ПОЛЬЗОВАТЕЛЯ | Сделано обращение к функции пользователя до ее определения с помощью оператора DEFFN |
| 19 | НЕТ RESUME | В подпрограмме обработки ошибок отсутствует оператор RESUME |
| 20 | RESUME БЕЗ ОШИБКИ | Оператор RESUME встретился перед выполнением подпрограммы обработки ошибок |
| 21 | НЕОПРЕДЕЛЕННАЯ ОШИБКА | Обычно это результат ERROR с неопределенным кодом ошибки |
| 22 | НЕТ ОПЕРАНДА | В выражении за оператором не следует операнд, в команде или операторе не заданы параметры |

## КОДЫ И НАЧЕРТАНИЯ СИМВОЛОВ, ЗАЛОЖЕННЫХ В ПЗУ ЗНАКОГЕНЕРАТОРА

Младшая тетрада

| Старшая тетрада | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| C 0 |  | | | | | | | | | | | | | | | |
| T 1 |  | | | | | | | | | | | | | | | |
| A 2 | пробел | ! | " | # | ¤ | % | & | ' | ( | ) | * | + | , | — | . | / |
| P 3 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | : | ; | < | = | > | ? |
| Ш 4 | C | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O |
| A 5 | P | Q | R | S | T | U | V | W | X | Y | Z | [ | \ | ] | ^ | — |
| Я 6 | ' | a | b | c | d | e | f | g | h | i | j | k | l | m | n | o |
| 7 | p | q | r | s | t | u | v | w | x | y | z | { | : | } | — | ⌂ |
| T 8 |  | | | | | | | | | | | | | | | |
| E 9 |  | | | | | | | | | | | | | | | |
| T A |  | | | | | | | | | | | | | | | |
| P B |  | | | | | | | | | | | | | | | |
| A D C | ю | а | б | ц | д | е | ф | г | х | и | й | к | л | м | н | о |
| A D | п | я | р | с | т | у | ж | в | ь | ы | з | ш | э | щ | ч | ъ |
| E | Ю | А | Б | Ц | Д | Е | Ф | Г | Х | И | Й | К | Л | М | Н | О |
| F | П | Я | Р | С | Т | У | Ж | В | Ь | Ы | З | Ш | Э | Щ | Ч | Ъ |

## РАСПРЕДЕЛЕНИЕ ПАМЯТИ



## ПРОГРАММНО-ДОСТУПНЫЕ ЭЛЕМЕНТЫ ПЭВМ ПК 8015

### 1. Перечень программно-доступных элементов

ПЭВМ представляется совокупностью программно-доступных узлов, в число которых входят:

центральный процессор КР580ВМ80А;
узел дешифрации адресов и задания конфигураций памяти (системный регистр);
клавиатура;
контроллер отображения алфавитно-цифровой информации;
контроллер отображения графической информации;
системный таймер;
контроллер прерываний;
адаптер локальной сети;
адаптер магнитофона;
программируемый генератор звука;
адаптер печатающего устройства;
адаптер последовательного интерфейса ИРПС;
контроллер внешнего ЗУ на гибких магнитных дисках.

### 2. Системный регистр и конфигурация памяти

Все узлы ПЭВМ адресуются как ячейки памяти. Различные типы памяти (ПЗУ, ОЗУ и т. д.) могут логически подключаться, отключаться и перемещаться, образуя различные конфигурации для разных системных и прикладных программ. Область ПЗУ всегда начинается с 0 адреса. Клавиатура, регистры адаптеров и контроллеров внешних устройств, а также отображаемое на экране алфавитно-цифровое ЗУ (АЦЗУ) сведены в единую область адресов размером 3 кбайта (область УВВ), подключаемую, отключаемую и перемещаемую как единое целое. В некоторых конфигурациях используются сокращенные варианты области УВВ, состоящие только из адресов регистров и портов периферийных БИС. Поскольку на дешифратор адресов поступает только старший байт адреса, конфигурации могут задаваться с точностью до одной страницы, при этом относительное расположение адресов внутри каждой страницы постоянно.

Ниже определяются относительные адреса различных элементов ПЭВМ в соответствующих страницах и начальные смещения этих страниц для различных конфигураций памяти. Адрес каждого элемента в программе должен задаваться в виде суммы относительного адреса и начального смещения.

Выбор одной из 9 имеющихся конфигураций осуществляется путем задания содержимого специального пятиразрядного ре-

гистра, называемого системным регистром. Адрес этого регистра располагается в текущей области УВВ.

Условные названия конфигураций памяти и соответствующие им константы для записи в системный регистр:

| | | |
|---|---|---|
| TRS80 | 00H | ПЗУ 14К, область УВВ с 3800 по 3FFF, ОЗУ с 4000 по FFFF |
| ROMB1 | 18H | ПЗУ 16К, область УВВ с FB00 по FFFF, ОЗУ с 4000 по F7FF |
| ROMB2 | 20H | ПЗУ 14К, область УВВ с 3800 по 3FFF, ОЗУ с 4000 по BFFF, графическое ЗУ (ГЗУ) с С000 по FFFF |
| ODOSA | 1CH | Конфигурация для ОС МикроDOC, ОЗУ с 0 по F7FF, область УВВ с F800 по FFFF |
| NDOS | 14H | ПЗУ 8К, ОЗУ с 2000 по F7FF, область УВВ с F800 по FFFF |
| BASIC | 40H | ПЗУ 24К, ОЗУ с 6000 по F7FF, область УВВ с F800 по FFFF |
| BASG | 60H | ПЗУ 24К, ОЗУ с 6000 по BEFF, регистры ВВ с BF00 по BFFF, ГЗУ с С000 по FFFF |
| DOSA | 5CH | ОЗУ с 0 по FDFF, регистры с FE00 по FFFF (конфигурация для ОС CP/M) |
| DOSG1 | 3CH | ОЗУ с 0 по 3FFF и с 8000 по FDFF, ГЗУ с 4000 по 7FFF, регистры с FE00 по FFFF |

Системный регистр вместе с регистром цвета и цветовой таблицей располагается в странице, начальное смещение которой определяется параметром RGBASE, способным принимать одно из 4 значений:

| | | |
|---|---|---|
| RGBASE1 | 3A00H | в TRS80 и ROMB2 |
| RGBASE2 | 0FA00H | в ROMB1, ODOSA, NDOS и BASIC |
| RGBASE3 | 0FF00H | в DOSA и DOSG1 |
| RGBASE4 | 0BF00H | в BASG |
| SYSREG | 7FH | Относительный адрес системного регистра |

Адреса остальных регистров, представляющих собой порты периферийных БИС, сведены в одну страницу, начальный адрес которой PBASE может принимать одно из 3 значений:

| | | |
|---|---|---|
| PBASE 1 | 3B00H | в TRS80 и ROMB2 |
| PBASE 2 | FB00H | в ROMB1, ODOSA, NDOS и BASIC |
| PBASE 3 | FE00H | в DOSA и DOSG1 |

### 3. Клавиатура

Состояние клавишных переключателей клавиатуры доступно программе при опросе групп клавиш или всей клавиатуры целиком как ячеек памяти. Ниже приводятся адреса ячеек, соответствующие группам по 8 клавиш. Нажатое состояние клавиши дает «1» в соответствующем разряде считанного байта данных. Опрос одновременно нескольких групп с целью выявления хотя бы одной нажатой клавиши можно производить путем объединения их адресов логическим ИЛИ.

| | | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|---|---|
| Основное поле | | | | | | | | | |
| KB00 | 01H | Г | Ф | Е | Д | Ц | Б | А | Ю |
| KB01 | 02H | О | Н | М | Л | К | Й | И | Ф |
| KB02 | 04H | В | Ж | У | Т | С | Р | Я | П |
| KB03 | 08H | ъ | Ч | Щ | Э | Ш | З | Ы | Ь |
| KB04 | 10H | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| KB05 | 20H | ? | > | = | < | + | * | 9 | 8 |
| KB06 | 40H | ПРОБЕЛ | ТАБ | ВШ | ВЗ | ИЗ | СТП | СТР | ВК |
| KB07 | 80H | РПГ | ФКС | УПР | СЕЛ | ПРФ | ГРФ | АЛФ | РГЛ |
| Дополнительные поля | | | | | | | | | |
| KB08 | 101H | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| KB09 | 102H | | | | | | | 9 | 8 |
| KB10 | 104H | | | | F5 | F4 | F3 | F2 | F1 |

Начальное смещение области адресов клавиатуры KBBASE может принимать одно из двух значений:

| | | |
|---|---|---|
| KBBASE1 | 3800H | в TRS80 и ROMB2 |
| KBBASE2 | F800H | в ROMB1, ODOSA, NDOS и BASIC |

### 4. Контроллер отображения алфавитно-цифровой информации

Контроллер состоит из 9-разрядного статического ОЗУ (АЦЗУ) объемом 1 кбайт, регистра управления и регистра атрибута. АЦЗУ располагается в конце текущей области адресов УВВ (XC00 — XFFF), каждая ячейка отображается на экране в виде одного символа, начиная с левого верхнего угла, всего 16 строк по 64 символа. Коды и начертания символов определяются знакогенератором, причем за основу взят код КОИ-8, дополненный элементами псевдографики. Регистр управления, реализованный в виде порта одной из микросхем КР580ВВ55А, устанавливает режим отображения и позволяет управлять состоянием атрибута инверсии изображения в 9-м разряде АЦЗУ,

в частности, формировать на экране курсор. Этот регистр работает на вывод, но доступен и для чтения. Одноразрядный регистр атрибута позволяет программно анализировать состояние атрибута в 9-м разряде АЦЗУ.

| | | |
|---|---|---|
| VIBASE1 | 3C00H | Начальный адрес АЦЗУ в конфигурациях TRS80 и ROMB2 |
| VIBASE2 | FC00H | Начальный адрес АЦЗУ в конфигурациях ROMB1, ODOSA, NDOS и BASIC |
| VIREG | 3AH | Относительный адрес регистра управления отображением |
| VISTS | 38H | Относительный адрес регистра состояния атрибута видеоинверсии |

Константы для работы с регистрами:

| | | |
|---|---|---|
| ATRMSK | 8 | Маска атрибута видеоинверсии |
| VBLMSK | 2 | Маска кадрового гасящего импульса |
| FONT1 | 4 | Выбор альтернативного набора символов в знакогенераторе |
| LARGE | 8 | Режим отображения расширенных символов (32 символа в строке) |
| ATRSET | 10H | Установка бита атрибута |
| ATRRES | 20H | Сброс бита атрибута |
| ATRFRE | 30H | Сохранение состояния атрибута и условие чтения атрибута |

### 5. Контроллер отображения графической информации

Контроллер состоит из двух или трех слоев (банков) динамического ОЗУ, содержимое которых отображается на экране в виде карты бит, регистра цвета, регистра выбора страниц, таблицы присвоения цветов и логической матрицы для анализа содержимого ГЗУ. Объем каждого слоя может быть 16К или 64К, но отображается всегда только 16К из каждого слоя.

Четырехразрядный регистр выбора страницы при объеме 64К определяет, какая часть (страница) каждого слоя отображается в текущий момент на экране и какая открыта для доступа процессору.

Байты информации одновременно из всех слоев выводятся на экран, начиная с верхнего левого угла, таким образом, что одноименные биты совмещаются и определяют логический цвет соответствующей точки на экране.

Старший бит каждого байта отображается слева, младший — справа. Окончательный (физический) цвет точки определяется содержимым программируемой таблицы присвоения цветов, причем учитывается также информация из АЦЗУ. Таблица присвоения цветов расположена в области УВВ и адресуется как

одна ячейка памяти, работающая только на запись, при этом выбор строки таблицы определяется содержимым записываемого байта.

| | | |
|---|---|---|
| PAGREG | VIREG | Адрес регистра выбора страниц |
| VPAGE0 | 00000000B | Выбор 0 страницы отображения |
| VPAGE1 | 00000001B | Выбор 1 страницы отображения |
| VPAGE2 | 00000010B | Выбор 2 страницы отображения |
| VPAGE3 | 00000011B | Выбор 3 страницы отображения |
| RWPAG0 | 00000000B | Выбор 0 страницы чтения/записи |
| RWPAG1 | 01000000B | Выбор 1 страницы чтения/записи |
| RWPAG2 | 10000000B | Выбор 2 страницы чтения/записи |
| RWPAG3 | 11000000B | Выбор 3 страницы чтения/записи |

Если объем слоя ограничен 16К, то состояние битов безразлично.

LUT 0FBH Адрес таблицы присвоения цветов (относительный).

При записи в LUT биты D0...D3 задают номер одной из 16 строк таблицы, а D4...D7 — содержимое этой строки. D3 соответствует АЦЗУ, D2 — слою ГЗУ 2, D1 — слою 1, D0 — слою 0, D7 задает интенсивность (яркость), D6 — наличие красного цвета, D5 — зеленого цвета, D4 — синего цвета.

Регистр цвета задает маски логических цветов и режимы записи и считывания ГЗУ. Имеются два основных режима — послойный и цветовой. Этот регистр работает только на запись.

NCREG EQU OBFH Относительный адрес регистра цвета

Константы для программирования регистра цвета:

| | | |
|---|---|---|
| BANKMD | 0 | Послойный режим чтения/записи |
| WSEL0 | 11111101B | Выбор слоя 0 при записи |
| WSEL1 | 11111011B | Выбор слоя 1 при записи |
| WSEL2 | 11110111B | Выбор слоя 2 при записи |
| RSEL0 | 00010000B | Выбор слоя 0 при чтении |
| RSEL1 | 00100000B | Выбор слоя 1 при чтении |
| RSEL2 | 01000000B | Выбор слоя 2 при чтении |
| WBIT | 00000001B | Маска значения бита, записываемого в биты ГЗУ в послойном режиме |
| COLORMD | 80H | Цветовой режим чтения/записи (выбраны все три слоя) |
| WRMSK | 00001110B | Маска битов, задающих логический цвет при записи |
| RDMSK | 01110000B | Маска битов, задающих логический цвет при чтении |

При записи в ГЗУ записываемый байт данных определяет, какие биты ГЗУ изменяют (примут значение WBIT или WRMSK), а какие сохраняют свое значение. Признаком изменения является соответствие битов записываемого байта значению BITWR, признаком сохранения — противоположное значение.

| | | |
|---|---|---|
| BITWR | 1 | Изменяются биты, заданные единицами |

При чтении ГЗУ в послойном режиме в считываемом байте получается содержимое ячейки выбранного слоя.

При чтении в цветном режиме в считанном байте содержатся признаки совпадения цвета в каждой точке с заданной маской. В случае совпадения в соответствующих битах читаются нули.

### 6. Системный таймер

Функции системного таймера выполняет канал 2 микросхемы КР580ВИ53. Таймер формирует запросы прерывания однократно или многократно через заданные интервалы времени. На вход таймера поступает частота 50 Гц, которую он делит в заданное число раз. Режим работы канала микросхемы задается в регистре режима, а коэффициент пересчета — в регистре счета.

| | | |
|---|---|---|
| STMRM | 03H | Относительный адрес регистра режима |
| STMRD | 02H | Относительный адрес регистра счета |

Константы для программирования режимов таймера:

| | | |
|---|---|---|
| CTR0S | 00000000B | Выбор счетчика 0 |
| CTR1S | 01000000B | Выбор счетчика 1 |
| CTR2S | 10000000B | Выбор счетчика 2 |
| LCTR | 00000000B | Стробирование счетчика |
| RLMB | 00100000B | Чтение/загрузка старшего байта |
| RLLB | 00010000B | Чтение/загрузка младшего байта |
| RLLM | 00110000B | Чтение/загрузка младшего и старшего байта |
| MODE0 | 00000000B | Режим 0 (датчик прерывания) |
| MODE1 | 00000010B | Режим 1 (одновибратор) |
| MODE2 | 00000100B | Режим 2 (делитель частоты) |
| MODE3 | 00000110B | Режим 3 (делитель частоты) |
| MODE4 | 00001000B | Режим 4 (программный формирователь строба) |
| MODE5 | 00001010B | Режим 5 (аппаратный формирователь строба) |
| BCDC | 00000001B | Двоично-десятичный счет |

Работа 2 канала в режиме периодического прерывания с заданием коэффициента пересчета в десятичной форме:

STMD = (CTR2S OR RLLM OR MODE2 OR BCDC)

Номер запроса прерывания от системного таймера:

| | |
|---|---|
| TMPINT | 5 |

### 7. Контроллер прерываний

Контроллер реализован на основе микросхемы КР580ВН59 и обслуживает 8 запросов (уровней) прерывания от контроллеров и адаптеров внешних устройств. В системе контроллер имеет два адреса для обращения со стороны процессора, а его внутренние регистры выбираются дополнительно разрядами данных при записи или их выбор определяется порядком обращения. Регистры делятся на регистры инициализации, регистры управления и регистры ввода. После инициализации контроллер готов к работе с фиксированными приоритетами запросов (0 — высший, 7 — низший). Контроллер при возникновении прерывания генерирует код команды CALL с адресом, соответствующим обрабатываемому запросу. Адреса всех 8 команд CALL попадают в одну область памяти, заданную программистом, где с шагом 4 или 8 ячеек должны располагаться команды переходов к подпрограммам обслуживания запросов.

Параметры для инициализации:

| | | |
|---|---|---|
| SICP0 | 28H | Относительный адрес первого регистра инициализации |
| ICW1 | 00010010B | Основная часть первого слова инициализации |
| LADR | 11100000B | Маска разрядов A5...A7 адреса таблицы переходов |
| SHORT | 00000100B | Интервал 4 байта |
| SICP1 | 29H | Относительный адрес второго регистра инициализации |
| ICW2 | 00000000B | Второе слово инициализации (Старший байт адреса таблицы переходов) |

Любой запрос прерывания можно запретить путем установки разрядов специального регистра маски (первого регистра управления). Этот регистр доступен для чтения.

| | | |
|---|---|---|
| MSKREG | 29H | Относительный адрес регистра маски |

Маски для уровней прерывания:

| | | |
|---|---|---|
| INT0 | 00000001B | Запрет прерывания уровня 0 |
| INT1 | 00000010B | Запрет прерываний уровня 1 |
| INT2 | 00000100B | Запрет прерывания уровня 2 |
| INT3 | 00001000B | Запрет прерываний уровня 3 |
| INT4 | 00010000B | Запрет прерываний уровня 4 |
| INT5 | 00100000B | Запрет прерываний уровня 5 |

| | | |
|---|---|---|
| INT6 | 01000000B | Запрет прерываний уровня 6 |
| INT7 | 10000000B | Запрет прерываний уровня 7 |
| INTA | 00000000B | Разрешение всех прерываний |

Второй регистр управления позволяет задавать различные условия окончания обработки прерываний.

| | | |
|---|---|---|
| EIOREG | 28H | Адрес регистра конца прерывания |

Константы условий окончания обработки прерываний:

| | | |
|---|---|---|
| PRSHF | 10000000B | Циклический сдвиг приоритетов |
| SEOI | 01000000B | Признак указания сбрасываемого уровня прерывания |
| EOI | 00100000B | Конец прерывания |
| LEVEL | 00000111B | Маска битов, указывающих уровень, которому присваивается низший приоритет |

Третий регистр управления позволяет задавать особые условия маскирования и осуществлять выбор одного из регистров ввода.

| | | |
|---|---|---|
| SELREG | 28H | Адрес регистра выбора |
| SELRID | 00001000B | Признак адресации (идентификатор) регистра выбора |
| SMEN | 01000000B | Разрешение управления спецмаскированием |
| SMSET | 00100000B | Установка спецмаскирования |
| POLL | 00000100B | Режим опроса |
| SELEN | 00000010B | Разрешение выбора регистра |
| SELIRR | 00000000B | Выбор регистра запросов |
| SELISR | 00000001B | Выбор регистра обрабатываемых прерываний |
| IRISRG | 28H | Адрес регистров опроса, входных запросов и обрабатываемых прерываний |

## 8. Адаптер магнитофона

В состав этого узла входят регистр записи и регистр чтения, функции которых выполняют отдельные цепи портов микросхемы КР580ВВ55А. В ПЭВМ «Орбита» ПК8015 имеется возможность прямого управления битами.

Разряды регистров чтения и записи отражают только состояние входных и выходных сигналов, а кодирование и декодирование информации должны осуществляться программно.

| | | |
|---|---|---|
| CASOUT | 32H | Адрес регистра записи |
| CASIN | 38H | Адрес регистра чтения |

| | | |
|---|---|---|
| CASMSK | 01H | Маска входного сигнала |
| SETLOW | 00B | Посылка сигнала низкого уровня |
| SETMID | 01B | Посылка сигнала среднего уровня |
| SETHIGH | 11B | Посылка сигнала высокого уровня |
| STOP | 4 | Временный останов ленты (пауза) |

### 9. Программируемый генератор звука

Генератор звука построен на основе канала f0 микросхемы КР580ВИ53. На вход счетчика подана частота 2000 кГц. Генератор дополнительно управляется одноразрядным регистром разрешения, доступным также для чтения.

| | | |
|---|---|---|
| SOUNDD | 00H | Адрес регистра счета |
| SOUNDM | 03H | Адрес регистра режима |
| SOUNDC | 32H | Адрес регистра разрешения |
| SNDEN | 8 | Разрешение формирования звука |

Константа для задания режима работы счетчика:

SNDMD = (CTROS OR RLLM OR MODE3)

### 10. Адаптер локальной сети

Этот адаптер построен на основе микросхемы КР580ВВ51А, работающей в асинхронном режиме на общую сигнальную линию локальной сети класса. В указанной микросхеме используются только цепи принимаемых и передаваемых данных. На вход синхронизации подана частота 312 кГц, что позволяет вести прием и передачу со скоростью около 19500 бит/с. Адаптер способен генерировать запрос прерывания при приеме байта данных из канала. Адрес (номер) рабочего места в локальной сети задается 4 перемычками на разъеме локальной сети. Адрес может быть прочитан через регистр LANADR (порт А первой микросхемы типа КР580ВВ55А).

Адреса регистров адаптера локальной сети:

| | | |
|---|---|---|
| LAND | 20H | Регистр данных |
| LANC | 21H | Регистр управления |
| LANS | 21H | Регистр состояния |
| LANADR | 38H | Регистр адреса рабочего места |
| ADRMSK | 0F0H | Маска адреса рабочего места |

Ниже приведены константы для программирования режимов.

| | | Коэффициент деления частоты синхронизации: |
|---|---|---|
| R64X | 00000011B | 64 |
| R16X | 00000010B | 16 |

| | | |
|---|---|---|
| R1X | 00000001B | 1 |
| CL8 | 00001100B | Длина символа = 8 бит |
| CL7 | 00001000B | Длина символа = 7 бит |
| CL6 | 00000100B | Длина символа = 6 бит |
| CL5 | 00000000B | Длина символа = 5 бит |
| PEHB | 00010000B | Разрешение контроля |
| PEVEH | 00100000B | Контроль на четность |
| | | Пауза при передаче: |
| ST2 | 11000000B | 2 стоповых бита |
| ST15 | 10000000B | 1.5 стоповых бита |
| ST1 | 01000000B | 1 стоповый бит |

Устанавливаемый режим работы:

LANMD = (R16X OR CLB OR PEHB OR ST1)

Отдельные биты команды:

| | | |
|---|---|---|
| TXEN | 00000001B | Разрешение работы передатчика |
| RXEN | 00000100B | Разрешение работы приемника |
| SBCH | 00001000B | Посылка признака обрыва |
| CLERR | 00010000B | Сброс признаков ошибок |
| USRST | 01000000B | Программный сброс |

Маски отдельных признаков в регистре состояния:

| | | |
|---|---|---|
| TRDY | 00000001B | Признак готовности передатчика |
| RRDY | 00000010B | Признак готовности приемника |
| TXBE | 00000100B | Передающий буфер пуст |
| RPAR | 00001000B | Признак ошибки по четности |
| ROV | 00010000B | Признак ошибки переполнения |
| RFR | 00100000B | Признак ошибки формата байта |
| LANINT | 3 | Используемый уровень прерывания |

## 11. Адаптер интерфейса ИРПС

Данный узел построен на основе микросхемы КР580ВВ51А и одного из каналов КР580ВИ53 и реализует последовательный интерфейс типа «Токовая петля» с дополнительными цепями для сигнализации о состоянии устройств, участвующих в обмене. Скорость передачи задается путем установки соответствующего коэффициента деления счетчика микросхемы КР580ВИ53, на вход которого подана частота 2000 кГц. Выделены отдельные уровни прерывания для приемника и передатчика.

Адреса регистров микросхемы КР580ВВ51А:

| | | |
|---|---|---|
| SERD | 10H | Регистр данных |
| SERC | 11H | Регистр управления |
| SERS | 11H | Регистр состояния |

Предполагаемый режим работы:

SERMD = (RI6X OR CL8 OR STI)

Дополнительный бит команды:

| | | |
|---|---|---|
| DTR | 00000010B | Сигнал «Готовность ПК» |

Дополнительный признак в регистре состояния:

| | | |
|---|---|---|
| DSR | 10000000B | Сигнал «Готовность УВВ» |

Адреса регистров каналы БИС КР580ВИ53:

| | | |
|---|---|---|
| BCNTC | 03H | Регистр управления |
| BCNTD | 01H | Регистр счета |

Константы для программирования счетчика:

BCMD = (CTR1S OR RLLM OR MODE3) Режим работы

Коэффициенты деления для скорости обмена:

| | | |
|---|---|---|
| B9600 | 13—9600 бод | |
| B4800 | 26—4800 бод | |
| B2400 | 52—2400 бод | |
| B1200 | 104—1200 бод | |
| SERINT1 | 1 | прерывание от приемника |
| SERINT2 | 2 | прерывание от передатчика |

### 12. Адаптер печатающего устройства

В состав этого узла входят регистр данных, регистр управления и регистр состояния, реализованные в виде портов параллельного адаптера КР580ВВ55А; причем регистр управления обеспечивает непосредственное управление битами (сброс и установку). Передача байта данных осуществляется при наличии признака готовности принтера в регистре состояния путем записи кода символа в регистре данных с последующей установкой и сбросом бита стробирующего сигнала в регистре управления. В драйверы печатающего устройства в некоторых случаях необходимо вводить инвертирование передаваемого байта и байта состояния (условное ассемблирование в зависимости от значения параметра LSTINV).

| | | |
|---|---|---|
| LSTDAT | 30H | Регистр данных принтера |
| LSTSTS | 38H | Регистр состояния |
| LSTCTL | 33H | Регистр управления |
| LSTINV | 1 | Признак инверсии данных и состояния |

| | | |
|---|---|---|
| LSTRDY | 4 | Маска признака готовности принтера |
| STBSET | 0BH | Установка строба данных |
| STBRST | 0AH | Сброс строба данных |
| LSTINT | 6 | Уровень прерывания по готовности принтера |

Программируемые параллельные адаптеры КР580АА55А, упомянутые выше в составе адаптеров внешних устройств, имеют следующие адреса портов и регистров для установки режимов:

| | | |
|---|---|---|
| ППРf1: | | |
| PIA1 | 38H | Порт А (LANADR, VISTS, LSTSTS, CASIN) |
| PIB1 | 39H | Порт В (DRVREG) |
| PIC1 | 3AH | Порт С (VIREG) |
| PIMC1 | 3BH | Регистр режима и управления |
| PI1MD | 90H | Байт задания режима |
| ППРf2: | | |
| PIA2 | 30H | Порт А (LSTDAT) |
| PIB2 | 31H | Порт В (резерв) |
| PIC2 | 32H | Порт С (CASOUT, LSTCTL, SOUNDC) |
| PIMC2 | 33H | Регистр режима и управления |
| PI2MD | 82H | Байт задания режима |
| ППРf3 (порт расширения системы): | | |
| PIA3 | 08H | Порт А |
| PIB3 | 09H | Порт В |
| PIC3 | 0AH | Порт С |
| PIMC3 | 0BH | Регистр управления |

## 13. Узел сопряжения и контроллер НГМД

### 13.1. Узел сопряжения с НГМД

В состав этого узла входят БИС контроллера НГМД КР1818ВГ93, регистр выбора и управления НГМД и одновибратор включения двигателя НГМД.

Регистр выбора и управления НГМД:

DRVREG 39H Порт В КР580ВВ55 f1

Этот регистр работает на вывод, но доступен программно и для чтения, поэтому состояние его разрядов можно изменять с помощью логических операций И, ИЛИ, ИСКЛЮЧАЮЩЕЕ ИЛИ.

Разряды регистра выбора и управления:

| | | |
|---|---|---|
| DS0 | 00000001B | Выбор НГМД f0 |
| DS1 | 00000010B | Выбор НГМД f1 |
| DS2 | 00000100B | Выбор НГМД f2 |
| DS3 | 00001000B | Выбор НГМД f3 |
| SIDE1 | 00010000B | Выбор стороны диска f1 |
| MOTOR | 00100000B | Запуск одновибратора включения двигателя |
| SDEN | 01000000B | Режим работы с одинарной плотностью записи |
| D200 | 10000000B | Работа с диском 203 мм |

Одновибратор включения двигателя запускается на время 3 с при переходе сигнала (бита) MOTOR из 0 в 1. Для разгона двигателя требуется время около 0.5 с, после этого можно инициировать операции контроллера НГМД. Если двигатель уже был включен, время на разгон не требуется. Опрос состояния одновибратора включения двигателя производится через регистр запросов контроллера прерываний.

| | | |
|---|---|---|
| MOTINT | 7 | Прерывание от одновибратора управления двигателем |

### 13.2. Контроллер НГМД

Контроллер состоит из следующих регистров:

| | | |
|---|---|---|
| CMDREG | 18H | Регистр команд |
| STSREG | 18H | Регистр состояния |
| TRKREG | 19H | Регистр текущей дорожки |
| SECREG | 1AH | Регистр сектора |
| DATREG | 1BH | Регистр данных |

Команды контроллера НГМД записываются в регистр команд при наличии разрешения, т. е. при отсутствии признака BUSY в регистре состояния. Опрос регистра состояния можно производить не ранее чем через 30 мкс после записи команды в регистр команд.

Команды позиционирования (команды 1 типа):

| | | |
|---|---|---|
| RESTORE | 00000000B | Позиционирование на 0 дорожку |
| SEEK | 00010000B | Позиционирование на дорожку, заданную в регистре данных |
| STEP | 00100000B | Позиционирование на одну дорожку в направлении, заданном предыдущей командой |

| | | |
|---|---|---|
| STEPIN | 01000000B | Позиционирование к центру на одну дорожку |
| STEPOUT | 01100000B | Позиционирование от центра на одну дорожку |

Модификаторы команд позиционирования:

| | | |
|---|---|---|
| UPDATE | 00010000B | Отслеживание перемещения головки в регистре дорожки |
| HLD | 00001000B | Прижим головки при движении |
| VERIFY | 00000100B | Проверка установки головки после перемещения |
| RATE | 00000011B | Коэффициент скорости позиционирования (для мини-дисков время перехода с дорожки на дорожку при 00 будет 6 мс, 01 — 12 мс, 10 — 20 мс, 11 — 30 мс). |

Модификаторы HLD, UPDATE и VERIFY могут использоваться по усмотрению системного программиста.

Использование модификатора RATE является обязательным, причем для дисководов типа FD55FV его значение может быть установлено равным 00.

Команды чтения/записи (команды 11 типа):

| | | |
|---|---|---|
| RDSEC | 10000000B | Чтение сектора |
| WRSEC | 10100000B | Запись сектора |

Модификаторы:

| | | |
|---|---|---|
| MULTI | 00010000B | Операция с несколькими секторами |
| DELAY | 00000100B | Включение задержки операции на время подвода головки |

Использование модификатора DELAY для используемых НГМД является обязательным.

Команды форматирования (команды 111 типа):

| | | |
|---|---|---|
| RDADR | 11000000B | Чтение идентификатора |
| RDTRK | 11100000B | Чтение дорожки |
| WRTRK | 11110000B | Запись дорожки |

Модификатор — DELAY (см. выше).

Команды прерывания (команды IV типа):

| | | |
|---|---|---|
| FORCE | 11010000B | Прекращение операции и/или генерация запроса прерывания на ЦП |

Модификаторы (используются по усмотрению программиста):

| | | |
|---|---|---|
| ONRDY | 00000001B | Прерывание по готовности НГМД |
| ONNRDY | 00000010B | Прерывание по неготовности НГМД |
| ONINDX | 00000100B | Прерывание по индексному импульсу |
| IMMID | 00001000B | Немедленное прерывание |

Признаки состояния контроллера НГМД могут быть получены путем чтения регистра состояния контроллера НГМД с последующим выделением их путем логического умножения на соответствующую маску.

Маски признаков (флагов) состояния контроллера НГМД:

| | | |
|---|---|---|
| BUSY | 00000001B | Признак неготовности контроллера к приему команд |
| DRQ | 00000010B | Запрос на передачу очередного байта данных (кроме команд 1 типа) |
| INDX | 00000010B | Индексный маркер, т. е. начальная позиция диска (только для команд 1 типа) |
| LOST | 00000100B | Признак потери данных (кроме команд 1 типа) |
| TRK00 | 00000100B | Признак 0 дорожки (для команд 1 типа) |
| CRCERR | 00001000B | Признак ошибки по циклическому контрольному коду |
| RECNF | 00010000B | Признак ненахождения заданного сектора (кроме команд 1 типа) |
| SEEKER | 00010000B | Ошибка позиционирования (для команд 1 типа) |
| FAULT | 00100000B | Ошибка при записи (для команд 11 и 111 типа) |
| HLDD | 00100000B | Признак прижима головки (для команд 1 типа) |
| WPRTD | 01000000B | Признак диска, защищенного от записи и стирания |
| NOTRDY | 10000000B | Признак отсутствия готовности НГМД |

## ДОПОЛНИТЕЛЬНЫЕ ВОЗМОЖНОСТИ ИНТЕРПРЕТАТОРА БЕЙСИК

### 1. Команда LOAD.

LOAD «CAS : [имя файла] [, R]
LOAD «CAS : [имя файла], A [, R]
LOAD «CAS : [имя файла], B [, R [, смещение]]

Команда загружает программный файл с кассеты:
во внутреннем сжатом формате, если не указаны параметры A или B;
в кодах КОИ-8, если указан параметр A;
в машинных кодах, если указан параметр B.
Параметр R определяет, что программа будет автоматически запущена после загрузки.
Загрузка программ в машинных кодах осуществляется с адреса, указанного в команде SAVE с параметром B при записи программы на МЛ. Если указано смещение, то все адреса, заданные в команде SAVE с параметром B, получат это смещение.
Если имя файла опущено, то будет загружена первая программа, хранящаяся на МЛ в данном формате.

### 2. Команда RUN.

RUN «CAS : [имя файла]
RUN «CAS : [имя файла], A
RUN «CAS : [имя файла], B

Выполнение данных форматов команды RUN аналогично соответствующим форматам команды LOAD.

LOAD «CAS : [имя файла], R
LOAD «CAS : [имя файла], A, R
LOAD «CAS : [имя файла], B, R

### 3. Команда SAVE.

SAVE «CAS : [имя файла]
SAVE «CAS : [имя файла], A
SAVE «CAS : [имя файла], B, начальный адрес, конечный адрес,
[, стартовый адрес]

Команда сохраняет на НКМЛ программный файл:
во внутреннем сжатом формате, если не указаны параметры A или B;
в кодах КОИ-8, если указан параметр A;
в машинных кодах, если указан параметр B.
При записи файла в машинных кодах необходимо указать начальный и конечный адрес области памяти, сохраняемой на

ПКМЛ. Можно указать стартовый адрес — адрес запуска программы, на который передается управление при загрузке ее с помощью команды RUN или LOAD с параметром R.

**4. Оператор** SCREEN.

SCREEN [N] [,[M] [,L]]

Оператор переключает графические страницы памяти:
N — номер страницы отображения;
M — номер страницы чтения/записи;
L — режим отображения символов на экране:
0 — 64 символа в строке;
1 — 32 символа в строке.

**5. Оператор** WIDTH.

WIDTH ширина строки

Устанавливает ширину строки экрана от 15 до 64 символов в строке.

## ЛИТЕРАТУРА

1. Либеров А. Б., Башмаков Е. С. Язык программирования Бейсик.— М., МЦНТИ, 1987.
2. Пул Л. Работа на персональном компьютере.— М., Мир», 1986.
3. Моррил Г. Бейсик для ПК ИБМ.— М., «Финансы и статистика», 1987.

---

**Перечень ошибок и исправлений**

| № страницы | № строки | Напечатано | Должно быть |
|---|---|---|---|
| стр. 39 | 8 сверху | заполняется этим цветом | заполняется этим цветом (одновременное использование параметров STEP и F не допускается). |
| стр. 41 | 8 снизу | 30LINE-STEP (60, 30), I, BF : NEXT | 30LINE-STEP (60, 30), I, B : NEXT |
| стр. 71 | 18 | KB09 9B | KB09 9H |

Зак. 213 24.08.89 г.