ходе делителя происходит изменение фазы импульсов на выходе элемента, при котором появляется дополнительный узкий импульс на входе делителя. Временные диаграммы его работы (К=8) при результирующем коэффициенте деления 7 показаны на рис. 7, б. Следует отметить, что скважность импульсов при этом будет равной 2 («меандр»), что недостижимо другими способами.

Применив два элемента ИСКЛЮЧАЮЩЕЕ ИЛИ и включив их по схеме на рис. 8, можно собрать делитель с изменяемым коэффициентом деления от 1 до 16, кроме 9. Для получения коэффициента деления 9 необходимо добавить на входе устройства еще один элемент ИСКЛЮЧАЮЩЕЕ ИЛИ и группу контактов переключателя SA1, включенные так же, как и два показанных элемента. Неподвижный контакт 9 нового переключателя SA1.4 подключают к выходу 8 счетчика DD2 (вывод 11), с него же снимают формируемые импульсы на выход устройства. Остальные неподвижные контакты переключателя SA1.4 соединяют с общим проводом. Контакты 9 переключателей SA1.1 и SA1.2 подключают соответственно к выходам 4 (вывод 8) и 2 (вывод 9) счетчика.

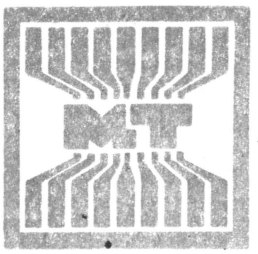
В предыдущих вариантах применения использовалось от одного до трех элементов ИСКЛЮЧАЮЩЕЕ ИЛИ. Однако один корпус микросхем серий К155 и К176 содержит четыре таких элемента. Неиспользованные элементы можно применить в качестве инверторов (рис. 9). Для этого на один из входов подают сигнал, на другой — напряжение питания через резистор сопротивлением 1 кОм для серии К155 или 300 кОм для серии К176.

Если элемент ИСКЛЮЧАЮЩЕЕ ИЛИ включить так, как изображено на рис. 10, а, то на выходе его возникнут узкие импульсы в моменты, совпадающие с фронтом и спадом входных (рис. 10, *б*). Частота их следования вдвое больше частоты входного сигнала. Сопротивление резистора R1 равно 1 кОм для элементов микросхемы К155ЛП5 и 300 кОм для К176ЛП2. Емкость конденсатора С1 определяет длительность импульсов на выходе. Минимальное значение — 40 пФ.

При необходимости задержать импульсы на некоторое время, не изменяя их формы, можно включить элементы по схеме на рис. 11, а. Временные диаграммы работы устройства приведены на рис. 11, б. Номиналы резистора R1 и конденсатора С1 выбирают так же, как и в предыдущем случае.

А. ИВАНОВ

г. Москва



Бейсик для «Микро-80»

Программа на Бейсике состоит из последовательности пронумерованных строк. Им принято присваивать номера с интервалом равным 10. В дальнейшем это может оказаться полезным, если понадобится вставить несколько дополнительных строк в программу. Номера могут быть любыми от 0 до 65529. Каждая строка программы может состоять из одного или нескольких операторов, предписывающих интерпретатору определенные действия. Если операторов несколько, то их отделяют друг от друга символом «:» (двоеточие). В качестве операндов выступают выражения, составленные из констант и переменных. В Бейсике существуют два типа констант: числовые и символьные. Числовые константы — это любые десятичные числа в интервале от $-1{,}7 \cdot 10^{38}$ до $+1{,}7 \cdot 10^{38}$, символьные — последовательность любых отображаемых символов, заключенная в кавычки. Например, «МОСКВА», «КП350», «РЕЗУЛЬТАТ=» — символьные константы, 220, —380, 3.14,— — 3.003E—03, 8E12 — числовые. Две последние константы заданы в экспоненциальной форме, на что указывает буква E, за ней следуют знак (у положительных чисел знак «+» можно опускать) и величина порядка. Точность задания констант — 6 значащих цифр.

В программах на Бейсике используются переменные двух типов: числовые и символьные. Обращаются к переменным по имени, которое состоит из одного или двух символов. Первый из них обязательно должен быть буквой латинского алфавита, второй — буквой этого же алфавита или цифрой. Символьные переменные после имени содержат знак «$». Например: A, A8, K — допустимые имена числовых переменных; A$, AT$, C3$ — символьных.

Группе переменных одного типа может быть присвоено общее имя, и их в этом случае называют переменными с индексами или массивами. Мы будем пользоваться в дальнейшем термином «массив». Для обращения к каждой отдельной переменной в массиве используют один или несколько индексов. Наименьшее значение индекса равно 0, а наибольшее определяется размером массива. Если индекс один, то говорят, что массив одномерный, два — двумерный и т. д. Имена массивов подчиняются тем же правилам, что и имена переменных. Индексы необходимо указывать в круглых скобках после имени массива. Разрешено использование массивов как числовых, так и символьных переменных.

Например:

AB (4) — четвертый элемент одномерного массива AB;

LS (3,8) — элемент, стоящий на пересечении 3-й строки и 8-го столбца двумерного массива LS;

K$ (5) — пятый элемент одномерного массива K$.

При работе программы для каждого массива в памяти ЭВМ резервируется соответствующая область. Перед использованием массив должен быть описан при помощи оператора DIM. Более подробно об этом мы поговорим позже.

Переменные и константы образуют выражения языка Бейсик. Кроме них, в выражения входят знаки операций, скобки и имена функций. Обращаются к функциям по имени, аргумент при этом указывают в круглых скобках после него. Все выражения можно разделить на 4 типа: арифметические, символьные, выражения отношения и логические. В табл. 3 приведены знаки операций, принятые в Бейсике. Числовые переменные и константы могут принимать участие в выражениях любого типа. Для символьных переменных и констант разрешены только операции отношения и конкатенации (слияния), обозначаемые знаком «+»:

Вместо символа $ далее по тексту использован символ $.

Продолжение. Начало см. в «Радио», 1985, № 1.

```
ПРИМЕР №2 !
==========
ПУСТЬ Ax="ТЕЛЕ", Bx="ФОН", Cx="ВИЗОР",
ТОГДА:  Mx=Ax+Bx = "ТЕЛЕФОН"
        Lx=Ax+Cx = "ТЕЛЕВИЗОР"
             Bx<>Cx
```

При вычислении результата выражения все операции имеют определенный приоритет. Табл. 4 показывает приоритет операций, определяющий порядок вычислений. Чем выше в таблице находится знак той или иной операции, тем выше ее приоритет.

Во всех алгоритмических языках одним из основных операторов является оператор присваивания. В Бейсике это — оператор LET. Так, запись LET B1=12 означает, что переменной

Таблица 3

ЗНАК ОПЕРАЦИИ	ОПЕРАЦИЯ
АРИФМЕТИЧЕСКИЕ ОПЕРАЦИИ	
+	СЛОЖЕНИЕ
-	ВЫЧИТАНИЕ
*	УМНОЖЕНИЕ
/	ДЕЛЕНИЕ
^	ВОЗВЕДЕНИЕ В СТЕПЕНЬ
ОПЕРАЦИИ ОТНОШЕНИЯ	
>	БОЛЬШЕ
<	МЕНЬШЕ
=	РАВНО
<>	НЕ РАВНО
>=	БОЛЬШЕ ИЛИ РАВНО
<=	МЕНЬШЕ ИЛИ РАВНО
ЛОГИЧЕСКИЕ ОПЕРАЦИИ	
NOT	ОТРИЦАНИЕ ("НЕ")
AND	ЛОГИЧЕСКОЕ УМНОЖЕНИЕ ("И")
OR	ЛОГИЧЕСКОЕ СЛОЖЕНИЕ ("ИЛИ")

Таблица 4

^,

*, /,

+, -,

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

NOT

AND

OR

(или константе) B1 присваивается значение 12. В описываемом интерпретаторе слово LET не используется и поэтому нужно просто писать B1=12.

Теперь мы можем вернуться к примеру 1*. В строке 20 константе PI (число π) присваивается значение 3.14156. В строках 30, 50 и 80 записаны операторы печати. Поэтому на экран будут выведены соответствующие сообщения. В строках 40, 60 и 90 записаны операторы ввода, в результате выполнения которых интерпретатор запрашивает ввод с клавиатуры дисплея в первых двух случаях числовых значений, в последнем — символьного значения. Введенное с клавиатуры слово присваивается символьной переменной X$.

В строке 100 использован условный оператор, который в случае выполнения условия передает управление на начало программы (строка 30). В противном случае будет выполнен оператор STOP, прекращающий выполнение программы и переводящий интерпретатор в непосредственный режим работы.

Мы уже говорили, что интерпретатор реализует также функции редактора текстов и отладчика. Перед вводом текста новой программы необходимо сначала стереть в памяти микро-ЭВМ старую программу, воспользовавшись директивой NEW, а затем с клавиатуры набрать программу (не забывая набирать в начале каждой строки ее номер). Для исправления неверно набранных символов служит клавиша «←» (так же, как и в Мониторе), а клавиша «→» — для стирания всей неверно набранной строки. Ввод каждой строки заканчивают нажатием на клавишу «ВК». Если случайно будет нажата клавиша «СТР», то управление передается программе Монитор и для повторного запуска интерпретатора в работу необходимо воспользоваться директивой Монитора «J0». Перезапуск интерпретатора не приводит к потере текста ранее набранной программы.

Просмотреть текст программы можно при помощи директивы LIST. Если необходимо удалить какую-либо строку программы, достаточно набрать ее номер и нажать на клавишу «ВК», а чтобы вставить в текст новую, набрать номер любой строки, попадающий в интервал между двумя соседними, затем саму строку и нажать на клавишу «ВК».

Интерпретатор игнорирует любые пробелы, если только они не стоят внутри символьных констант. Это создает определенные удобства, так как при использовании пробелов для выделения операторов улучшается читаемость программы (хотя несколько и увеличивается ее объем). Для исправления строки, содержащей ошибки, набирают ее заново (с тем же номером). После нажатия на клавишу «ВК» вновь набранная строка встанет на место старой.

Отладка программы на Бейсике производится также при помощи интерпретатора. Для этой цели использована возможность перевода интерпретатора из программного режима в непосредственный при выполнении оператора STOP или одновременном нажатии на клавиши «УС» и «С». В непосредственном режиме можно просмотреть и при необходимости модифицировать (изменить) значения переменных, снова просмотреть текст программы по директиве LIST и затем продолжить выполнение программы с того места, где она была прервана.

Перейдем теперь к подробному описанию нашей версии интерпретатора с языка Бейсик и прежде всего рассмотрим его системные директивы. Отметим только, что далее по тексту фигурные скобки будут означать, что данные параметры в конкретной директиве могут отсутствовать. В этом случае используют так называемые соглашения по умолчанию, которые будут оговорены особо.

Директивы языка Бейсик

Директива NEW подготавливает интерпретатор для ввода новой программы с клавиатуры дисплея. Текст программы, набранный ранее, стирается.

Директива RUN{N} служит для запуска программы со строки с номером N. Если номер строки отсутствует, то работа программы начинается со строки с наименьшим номером. Всем переменным присваивается значение 0 или «пустая строка».

Директива LIST {N} инициирует распечатку текста программы, находящейся в ОЗУ, начиная со строки с номером N. Если номер отсутствует, то распечатка начинается с начала программы.

Директива CONT позволяет продолжить выполнение программы с того места, где она была прервана нажатием на клавиши «УС» и «С», или при выполнении оператора STOP. Эта директива — одно из основных средств отладки программ на Бейсике.

В случае невозможности дальнейшего выполнения программы на экран дисплея выдается соответствующее сообщение об ошибке.

Директива MSAVE {имя} позволяет переписать программу на Бейсике из ОЗУ на магнитную ленту. Имя программы должно состоять из одного символа латинского алфавита. Впро-

* См. «Радио», 1985, № 1, с. 36.

чем, его можно и не указывать. В этом случае и при последующем вводе программы ее имя также указывать не следует.

Директива MLOAD {имя} предназначена для загрузки программ на Бейсике с магнитной ленты в ОЗУ. Если имя программы указано, то происходит поиск данной программы, если нет — в ОЗУ загружается первая же встретившаяся программа на Бейсике, записанная без указания имени.

Операторы языка Бейсик

В Бейсике имеется ряд операторов, которые можно разделить на две группы: выполняемые и невыполняемые (операторы описания). Рассмотрим сначала последние.

Оператор REM служит для вставки в текст программы комментариев. Он не влияет на выполнение программы, так как все, что стоит в строке за этим оператором, интерпретатором игнорируется.

Оператор DIM предназначен для описания массивов, используемых в программе. Массив можно не описывать, если его размерность не превышает 10. Одним оператором DIM можно описать сразу несколько массивов.

```
ПРИМЕР №3 !
============
 10 DIM A(30), B(15,15), C2(3,3,3)
 20 DIM P%(5), R%(7,4)
```

Массив должен быть описан в программе до его использования, иначе вступает в силу описание по умолчанию.

Оператором DATA можно описывать данные непосредственно в программе. Значения данных присваиваются переменным программы при помощи оператора READ. Программа может содержать любое число операторов DATA, и располагаться они могут в любом ее месте, независимо от положения операторов READ. Оператором DATA могут быть описаны любые данные, как числовые, так и символьные:

```
ПРИМЕР №4 !
============
 10 DATA 12,865,"КП103М","КР580ВК28"
 20 DATA "ТРАНЗИСТОР", "МИКРОСХЕМА"
```

Все данные, описанные при помощи оператора DATA, образуют так называемый блок данных. Данные из блока можно считать, воспользовавшись оператором READ. Они будут выведены последовательно, начиная с первого. После каждого обращения к блоку происходит перемещение на одну позицию так называемого внутреннего (для интерпретатора) указателя данных.

Существует еще один оператор описания, но с ним мы познакомимся далее, в разделе, посвященном встроенным функциям Бейсика.

Выполняемые операторы, в свою очередь, также могуть быть разделены на ряд групп: ввода-вывода, управления ходом выполнения программы, организации циклов, графические и связи с машинными ресурсами. Особое место занимает условный оператор.

Оператор READ предназначен для чтения данных из блока и присвоения конкретных значений переменным программы. Запись этого оператора в программе выглядит так: READ X1, X2, ..., XN, где XN — имена числовых или текстовых переменных. При описании данных программист обязан строго следить за соответствием типов данных и переменных. При каждом выполнении оператора READ указатель данных смещается на одну позицию.

Оператор RESTORE. Служит для перемещения указателя в первую позицию, обеспечивая тем самым возможность повторного считывания данных из блока.

Оператор INPUT позволяет вводить данные с клавиатуры дисплея непосредственно при выполнении программы на Бейсике. Значения введенных данных присваиваются переменным, имена которых указывают вслед за оператором INPUT. Это могут быть как числовые, так и символьные переменные. Использование оператора иллюстрирует следующий пример:

```
ПРИМЕР №5 !
============
 10 INPUT A1
 20 INPUT A2,A3,A4
 30 INPUT B%,E2
```

При выполнении оператора INPUT на экране дисплея возникает символ «?». В ответ на этот вопрос с клавиатуры вводят данные, которые «распечатываются» на экране в строку сразу после этого символа.

Если оператором INPUT необходимо ввести несколько переменных, то после ввода очередного значения необходимо нажать на клавишу «,». Ввод данных заканчивается нажатием на клавишу «ВК». Если после появления знака «?» сразу нажать на клавишу «ВК», то интерпретатор переходит из программного режима в непосредственный.

После оператора INPUT может стоять строка символов, заключенная в кавычки. В этом случае при выполнении оператора на экран дисплея будет выведено сначала это сообщение, а затем знак «?»:

```
ПРИМЕР №6 !
============
 40 INPUT "ВАШЕ ИМЯ ";I%
```

При вводе данных, пока не нажата клавиша конца ввода «ВК», для внесения исправлений можно пользоваться клавишей «←».

Оператор PRINT предназначен для вывода на экран дисплея значений переменных, сообщений и результатов вычислений. Если оператор использован без операнда, то это приводит к печатанию одной пустой строки. При вводе программы и в непосредственном режиме вместо слова PRINT можно набрать символ «?». В этом случае при последующем просмотре текста программы по директиве LIST, вы увидите, что в тексте знак «?» автоматически заменен на слово PRINT.

Операндов, стоящих вслед за оператором PRINT, может быть несколько, и тогда их отделяют друг от друга разделителями «,» или «;», причем при использовании первого под каждое выводимое значение отводится 14 позиций в строке, второго — столько, сколько необходимо.

Числа при печати могут быть представлены в виде целого числа, числа с десятичной точкой и в экспоненциальном виде. В любой форме печатается не более 6 цифр.

Если после последнего операнда в операторе PRINT стоит разделитель, то при выполнении следующего оператора PRINT печать будет продолжена в той же строке. Если же разделителя нет, то печать начнется с новой строки. Для печати данных в определенном месте экрана при помощи оператора PRINT в языке Бейсик предусмотрены специальные функции, рассматриваемые ниже. Следующий пример иллюстрирует использование оператора PRINT:

```
ПРИМЕР №7 !
============
10 PRINT "ПРОГРАММА РАСЧЕТА УСИЛИТЕЛЯ"
20 PRINT "КОЛИЧЕСТВО =";A6
30 PRINT A,B,S4
40 PRINT 45*A2+S3
50 PRINT A1%,A2%;N7
```

Оператор CUR X, Y служит для перемещения курсора в позицию с координатой X по горизонтали и координатой Y по вертикали. Начало отсчета — левый нижний угол экрана. Диапазон изменения координат курсора по горизонтали — 0—63, по вертикали — 0—31. Если после оператора CUR сразу выполняется оператор печати PRINT, то вывод информации на экран начнется с позиции с координатами X и Y:

```
ПРИМЕР №8 !
============
10 CUR25,15
20 PRINT "ГРАФИК №5"
```

После выполнения этой программы в центре экрана появится текст «ГРАФИК 5».

Оператор CUR позволяет создавать программы, реализующие так называемый экранный режим работы и полностью использующие возможности нашего дисплея. К ним можно отнести разнообразные игровые программы, экранные редакторы текстов, программы обработки информации, представленной в табличной форме, и многие другие.

Программа на Бейсике выполняется строка за строкой, в соответствии с их номерами. Однако имеется ряд операторов, изменяющих естественный ход выполнения программы.

Оператор GOTO N — это оператор безусловной передачи управления на строку с номером N.

Операторы GOSUB N и RETURN служат для организации подпрограмм. Что такое подпрограммы и необходимость их применения мы уже обсуждали ранее применительно к языку ассемблера. Оператор GOSUB N передает управление на строку с номером N. Заканчиваться подпрограмма должна обязательно оператором RETURN. После его выполнения происходит возврат в то место основной программы, откуда произошел вызов подпрограммы. Допускается многократная вложенность подпрограмм, степень вложенности ограничена только объемом свободной памяти.

Операторы ON X GOTO N1, N2, ...NM и ON X GOSUB N1, N2 ... NM в зависимости от результата вычисления выражения X реализуют условную передачу управления на одну из строк программы, номер которой указан в списке, следующем за оператором.

При выполнении оператора сначала вычисляется значение выражения X, от результата берется целая часть, которая и указывает в списке на номер строки. Если результат выражения равен 1, то управление будет передано на строку N1, двум — на строку N2 и т. д. Если же результат выражения меньше единицы или больше, чем количество номеров строк в списке, то выполняется оператор непосредственно следующий за оператором ON—GOTO или ON—GOSUB. Рассмотренные операторы в литературе часто называют «переключателями». Действительно, их работа похожа на работу многопозиционного переключателя, коммутирующего прохождение сигнала по одному из возможных направлений. «Переключатель» позволяет сократить текст программы в тех случаях, когда необходим анализ множества вариантов, программисту же необходимо так «подобрать» выражение X, чтобы при всех возможных значениях входящих в него переменных происходило «переключение» на строку с нужным номером.

Оператор STOP предназначен для прекращения выполнения программы и перевода интерпретатора в непосредственный режим. Этот оператор очень удобен при отладке программ, так как позволяет создать в них контрольный останов. Выполнение прерванной программы можно продолжить, выдав интерпретатору директиву CONT. После выполнения оператора STOP на экран дисплея выводится сообщение: «СТОП в XX», где XX — номер строки, в которой произошел останов.

Программы, написанные на любом языке программирования, обычно содержат многократно повторяющиеся фрагменты — циклы. Для организации циклов в языке Бейсик имеются специальные операторы.

Оператор FOR X TO Y STEP Z — это оператор инициализации цикла, а **оператор** NEXT X — оператор конца цикла. Все, что находится между ними, называют телом цикла. В операторе инициализации цикла X — выражение, задающее имя переменной цикла и присваивающее ей начальное значение; Y — выражение, определяющее конечное значение переменной цикла, Z — значение, на которое должна измениться переменная цикла (шаг цикла) после каждого выполнения оператора NEXT. Если шаг цикла равен +1, то выражение STEP Z можно опустить. Проиллюстрируем сказанное на примере:

```
ПРИМЕР №9 !
============
10 FOR I=0 TO 10 STEP 2
20 PRINT I;
30 NEXT I
```

После выполнения этой программы на экране дисплея появится 6 значений: 0 2 4 6 8 10. В данном примере оператор, стоящий в строке 20, является телом цикла. Таким образом, сущность работы оператора цикла заключается в следующем:

— задается начальное значение переменной цикла;

— выполняются операторы, входящие в тело цикла;

— проводится проверка достижения переменной цикла конечного значения;

— изменяется значение переменной цикла на величину, равную шагу цикла;

— если конечное значение не достигнуто, то все перечисленные выше операции повторяются вновь;

— если конечное значение достигнуто, управление передается оператору, следующему непосредственно за оператором NEXT.

Шаг цикла может принимать и отрицательное значение:

```
ПРИМЕР №10!
============
10 FOR I=10 TO 0 STEP -2
20 PRINT I;
30 NEXT I
```

В этом случае последовательность выведенных на экран значений будет обратной: 10 8 6 4 2 0.

Заметьте: операторы, входящие в тело цикла, в любом случае выполняются хотя бы один раз, так как проверка условия окончания производится в конце цикла.

Организовать циклическую работу программы можно и без специального оператора цикла (например, при помощи оператора GOTO и условного оператора, описываемого ниже), однако его использование значительно упрощает разработку программ, освобождая программиста от необходимости проведения изменения переменной цикла и проверки его окончания.

(Окончание следует)

Г. ЗЕЛЕНКО,
В. ПАНОВ,
С. ПОПОВ

г. Москва