Глава VII. Дополнительные возможности языка MSX BASIC

VII.1. Работа с клавиатурой

VII.1.1. Нестандартный ввод с клавиатуры

Как правило, ввод данных с клавиатуры пользователь осуществляет с помощью оператора `INPUT`. При этом нужно нажать как минимум две клавиши: одну для ввода символа и вторую для помещения его в память. При этом знак вопроса («подсказка») и вводимый символ («эхо») отображаются на экране.

Строковая функция `INKEY$` подавляет «подсказку» и «эхо» символа на экране и не требует нажатия клавиши ввода Ввод ⏎.

Функция

```
INKEY$
```

,

где `INKEY` («KEY IN» — «вводить с клавиатуры») — служебное слово, в момент нажатия на какую-либо клавишу возвращает значение, равное вводимому по ней символу. Если клавиша не нажата, то значением функции `INKEY$` является пустая строка ("").

Пример 1.
0711-01.bas
0711-01.bas

```
10 W$=INKEY$:IF W$="" THEN 10 'Если клавиша не нажата, то компьютер переходит к
                выполнению строки с номером 10.
20 PRINT W$:GOTO 10 'Вывод на экран символа, соответствующего нажатой клавише,
           и переход к выполнению строки 10.
run
a      ◄— нажата клавиша "a"
p     |
r     |
i     |◄— нажата клавиша "F1" (для компьютера MSX 1)
n     |
t     |
      |
▉   ◄— нажата клавиша "STOP"
Break in 10 ◄— нажаты клавиши "CTRL"+"STOP"
Ok
```

При выполнении программы клавиатура постоянно опрашивается (сканируется), и полученные коды записываются в область памяти, называемую *буфером* клавиатуры. Функция `INKEY$`, как Вы только что увидели, извлекает из буфера клавиатуры код набранного символа. Если буфер «пуст» (никакая клавиша не нажата!), то функция `INKEY$` возвращает пустую строку.



Важной особенностью функции `INKEY$` является то, что она не выводит на экран символ, набранный на клавиатуре (если, конечно, в программе отсутствует оператор `PRINT` !), и не «ожидает» ввода символа, если клавиша не нажата.

С помощью функции `INKEY$` можно «прочитать» нажатие только тех клавиш, которым соответствует код какого-либо символа. Это делается так:

```
IF INKEY$="с и м в о л" THEN …
```

Для обнаружения нажатия какой-либо из клавиш: ESC, TAB, Ввод ⏎, SELECT, CLS/HOME, INS, DEL, ←, ↑, →, ↓, CTRL+буква используйте оператор:

```
IF INKEY$=CHR$(K) THEN …
```

где K — код клавиши, нажатия которой ожидается.

К сожалению, клавиши CAPS, SHIFT, GRAPH, CTRL, РУС, STOP «нечитаемы» операторами `INPUT$` или `INKEY$`. Тем не менее для чтения этих клавиш достаточно прочесть буфер NEWKEY (11 байтов) по адресу &HFBE5 (см. раздел X.9.1.).

Пример 2. Пример программы, останавливаемой нажатием клавиши SHIFT:
0711-02.bas
0711-02.bas

```
10 Z=PEEK(&HFBEB):IF Z<>254 THEN 10
```

Пример 3. Вспомним, что функция `ASC(α)`, где α — строковое выражение, возвращает десятичный код ASCII первого символа значения выражения α. Используя функцию `INKEY$`, можно получать десятичные коды ASCII «печатаемых» символов и некоторых управляющих клавиш. Интересны результаты нажатия функциональных клавиш.
0711-03.bas
0711-03.bas

```
10 W$=INKEY$:IF W$="" THEN 10 ELSE PRINT ASC(W$):GOTO 10
run
27 ◄— нажата клавиша "ESC"         112 │
112 ◄— нажата клавиша "p"           114 │
114 ◄— нажата клавиша "r"           105 │ ◄— нажата клавиша "F1"
105 ◄— нажата клавиша "i"           110 │
110 ◄— нажата клавиша "n"           116 │
116 ◄— нажата клавиша "t"           32  │
32 ◄— нажата клавиша "пробел"     Break in 10◄— нажаты клавиши"CTRL"
 ◄— нажата клавиша "STOP"          Ok              +"STOP"
49 ◄— нажата клавиша "SHIFT"+"1"
```

Таким образом, функциональные клавиши имеют ту особенность, что нажатие любой из них вызывает вывод кодов нескольких символов, составляющих её текст.

Приведём ещё один пример применения функции `INKEY$`.

```
10 PRINT "Нажмите любую клавишу для продолжения"
20 IF INKEY$="" THEN 20    'Задержка до нажатия клавиши
```

Функция `INPUT$(n)` служит для получения n последовательно вводимых с клавиатуры символов. Вводимые символы «накапливаются» в буфере клавиатуры без изображения их на экране дисплея. Когда ввод окончен (время ввода не ограничено!), то функция `INPUT$()` возвращает на экран дисплея строку введённых символов и курсор.


Запомните, что только нажатие клавиш CTRL+STOP прерывает действие данной функции!

Пример 4.
0711-04.bas
0711-04.bas

```
10 W$=INPUT$(2):?W$:GOTO 10
run
```

```
▉
▉ ◄— нажата клавиша "a"
ar  ◄— нажата клавиша "r"
▉
Break in 10 ◄— нажаты клавиши "CTRL"+"STOP"
Ok
```

Этот оператор неудобно использовать для ввода *длинных* символьных строк с клавиатуры, ибо Вы можете потерять контроль над вводом (введённые символы не видны на экране до тех пор, пока не введён последний из них).

Однако можно использовать функцию `INPUT$(1)` для ввода с клавиатуры по одному символу (аналогично функции `INKEY$`).

Пример 5.

0711-051.bas
0711-051.bas

```
10 W$=INPUT$(1):?W$:GOTO10
run
▉
a   ◄— нажата клавиша "a"
p   ◄— нажата клавиша "p"
p  │
r  │
i  │◄— нажата клавиша "F1"
n  │    на компьютере
t  │    серии MSX 1
   │
▉  и так далее...
```

0711-052.bas
0711-052.bas

```
10 ? ASC(INPUT$(1)):GOTO 10
run
▉
 112  │
 114  │
105 │◄— нажата клавиша "F1"
110 │  на компьютере
116 │  серии MSX 1
 32   │
▉
97 ◄— нажата клавиша "a"
▉    и так далее...
```

Таким образом, функцию `INPUT$()` можно также использовать для получения десятичных кодов ASCII символов, а также десятичных кодов ASCII некоторых управляющих и функциональных клавиш.

Пример 6. *Пароль !*
0711-06.bas
0711-06.bas

```
40 CODE$="Секрет":COUNT=0:PRINT"Имя":INPUT N$
70 PRINT "Пароль:"
110 P$=INPUT$(6) 'Чтение ш е с т и символов
120 IF P$=CODE$ THEN GOTO 150
130 COUNT=COUNT+1
140 IF COUNT<3 THEN GOTO 70 ELSE PRINT"Проход закрыт":END
150 PRINT:PRINT N$:PRINT"Проходи, приятель":END
```

Результат работы программной строки

```
10 A$=INKEY$:IF A$="" THEN 10
```

может показаться эквивалентным результату работы команды

```
A$=INPUT$(1)
```
!

Однако между ними есть различие: нажатие клавиши STOP задерживает программу в цикле с оператором `INKEY$`, тогда как во время ожидания нажатия клавиши, «организованного» оператором `INPUT$`, клавиша STOP временно отключается.



Это единственный способ временно отключить клавишу STOP в MSX BASIC!

VII.1.2. Функциональные клавиши

При нажатии клавиши F1 на дисплее компьютера серии MSX 1 появляется слово `print` (как если бы оно набиралось на клавиатуре посимвольно). Аналогичным образом действует любая из остальных девяти функциональных клавиш: F2, F3, F4, F5, F6, F7, F8, F9, F10, — генерируя на экране соответствующее слово. Подобные клавиши называются *функциональными* (программируемыми), поскольку они не связаны жёстко с каким-либо определённым символом. Генерируемые ими слова называются *значениями* функциональных клавиш.

MSX BASIC располагает средством, позволяющим изменять эти значения так, что при нажатии соответствующей клавиши генерируется заранее заданная цепочка символов. Этим средством является оператор

```
KEY n,T
```

,

где:

- KEY («key» — «клавиша») — служебное слово;
- n — арифметическое выражение, целая часть значения которого задаёт номер функциональной клавиши (n=1, 2, ..., 10);
- T — символьное выражение.

При выполнении данного оператора значение клавиши Fn становится равным значению символьного выражения T. Длина последнего не должна превышать 15.

В противном случае будут использованы лишь первые 15 символов этого значения. Отметим, что оператор

```
KEY n,""
```

«сбрасывает» значение функциональной клавиши с номером n.

Оператор

```
KEY LIST
```

выводит на экран дисплея значения 10 функциональных клавиш по порядку, по одной на дисплейную строку. Приведём ряд *примеров*:

1. ```
   10 KEY 1,CHR$(1)+CHR$(78)
   20 KEY LIST
   run
   ♫           |
   auto        | для компьютеров
   goto        |  серии MSX 1
   …           |
   _comterm    |
   Ok
   ```

2. ```
   10 KEY 5,"print x;y"
   ```

3. ```
   10 FOR S=1 TO 5:KEY S,"":NEXT: KEY 1,"GOSUB"
   20 KEY S,"YES":KEY LIST
   ```

4. ```
   10 X$="for x=1 to 15'заголовок":KEY5,X$:KEY LIST
   ```

5. `KEY 2,"LIST"+CHR$(13) 'При нажатии клавиши F2 выполняется команда LIST.`

Заметим, что символ `CHR$(13)` вызывает ввод команды (оператора) так же как и нажатие клавиши Ввод ⏎!

Отметим, что при включении компьютера серии MSX 1 значения функциональных клавиш устанавливаются по умолчанию следующим образом:

```
print          auto          goto          list          run
run"COM:"      save"COM:"    load"COM:"    merge"COM:"   _comterm
```

Однако, если одновременно нажать на кнопку сброса RESET и клавишу DEL, то ключи будут установлены следующим образом:

```
color          auto          goto          list          run
color 15,4,4   cload"        cont          list.         run
```

Кстати, этого состояния функциональных клавиш можно достичь программным путём. Для этого используйте команды:

```
DEFUSR=&H003E:X=USR(X)
```

При помощи функциональных клавиш можно значительно ускорить ввод программ в память компьютера, если «занести в них» часто встречающиеся в программе служебные слова.

Использование функциональных клавиш очень полезно на этапе *редактирования* программы. Каждая из клавиш может быть специально определена для ускорения набора текста программы.

Предположим, что Вы хотите ввести большое количество двоичных чисел. Это часто бывает необходимо при определении образа спрайта. Чтобы облегчить себе работу, определите одну из функциональных клавиш так:

```
KEY1,"DATA 00000000"+CHR$(13)
```

а затем введите команду `AUTO`. Нажмите клавишу F1 несколько раз для получения начальных «шаблонов» операторов `DATA` с последовательными номерами строк. Вам остаётся только отредактировать те строки, в которых необходимо заменить нули на единицы в нужных местах!

Пример.
Установка новых значений функциональных клавиш[1)]
fk.bas
fk.bas

Описание структур хранения значения функциональных клавиш

Таблица содержит значения по умолчанию для десяти функциональных клавиш (от F1 до F10)[2)].

Место в ОЗУ: &HF87F–&HF91E = 160 байт.

MSX 1

Место в ПЗУ: &H13A9–&H1448 = 160 байт.

Каждой клавише назначается область памяти из 16 байт.

Клавиша	Диапазон адресов	Значение	Содержимое
F1	&H13A9–&H1388	63 6F 6C 6F 72 20 00 00 00 00 00 00 00 00 00 00	color
F2	&H1389–&H13C8	61 75 74 6F 20 00 00 00 00 00 00 00 00 00 00 00	auto
F3	&H13C9–&H13D8	67 6F 74 6F 20 00 00 00 00 00 00 00 00 00 00 00	goto

Клавиша	Диапазон адресов	Значение	Содержимое
F4	&H13D9–&H13E8	6C 69 73 74 20 00 00 00 00 00 00 00 00 00 00 00	list
F5	&H13E9–&H13F8	72 75 6E 0D 00 00 00 00 00 00 00 00 00 00 00 00	runВвод ⏎
F6	&H13F9–&H1408	63 6F 6C 6F 72 20 31 35 2C 34 2C 34 0D 00 00 00	color 15,4,4Ввод ⏎
F7	&H1409–&H1418	63 6C 6F 61 64 22 00 00 00 00 00 00 00 00 00 00	cload"
F8	&H1419–&H1428	63 6F 6E 74 0D 00 00 00 00 00 00 00 00 00 00 00	contВвод ⏎
F9	&H1429–&H1438	6C 6F 6E 74 2E 0D 1E 1E 00 00 00 00 00 00 00 00	list.Ввод ⏎⇑⇑
F10	&H1439–&H1448	0C 72 75 6E 0D 00 00 00 00 00 00 00 00 00 00 00	CLSrunВвод ⏎

- Clefs pour MSX (Tome 1) p.154
- http://www.angelfire.com/art2/unicorndreams/msx/RR-BIOS.html

MSX 2

Место в ПЗУ &H13C3–&H140B = 72 байта.

Для MSX 2 и MSX 2 + структуру буфера в ПЗУ попытались оптимизировать (по сравниванию с MSX 1) с целью уменьшить его объём, учитывая что длина строковых заготовок чаще всего меньше 16 байт. Задаются только значимые коды символов и команд, а строки разделяются байтом с значением 0.

При этом естественно увеличился объём кода процедуры, которая формирует буфер FNKSTR.

Всего 10 записей длиной до 15 символов (с обязательным 0 в конце строки).

Значение	Содержимое
63 6F 6C 6F 72 20 00 61 75 74 6F 20	color .auto
00 67 6F 74 6F 20 00 6C 69 73 74 20	.goto .list
00 72 75 6E 0D 00 63 6F 6C 6F 72 20	.run..color
31 35 2C 34 2C 34 0D 00 63 6C 6F 61	15,4,4..cloa
64 22 00 63 6F 6E 74 0D 00 6C 69 73	d".cont..lis
74 2E 0D 1E 1E 00 0C 72 75 6E 0D 00	t......run..

— Камиль Каримов (caro)

VII.1.3. Джойстик

Fix Me!

> Человеческая изобретательность ни в чем не проявляется так, как в играх.
>
> *—Из письма Г.Лейбница Б.Паскалю*

В комплекте MSX–компьютера имеется два периферийных устройства для ввода информации, которые называются *джойстиками* («joy stick» — «палочка радости»).

Подробнее джойстиках здесь[3)].

Каждый джойстик состоит из двух блоков: *рукоятки и кнопки*. Обычно у джойстика имеется одна кнопка. Кроме того, к каждому джойстику можно присоединить дополнительно ещё по одной кнопке (отметим, что у некоторых моделей

их две изначально!).

Рукоятку джойстика можно перемещать вверх, вниз, направо, налево, а также в комбинациях: вверх+направо, вверх+налево, вниз+направо, вниз+налево. Таким образом, получается *восемь* допустимых направлений движения.



Важно отметить, что MSX–компьютер имеет встроенный «джойстик» (с номером 0). Роль *рукоятки* этого «джойстика» играют клавиши управления курсором, а роль *кнопки* играет клавиша «пробел»!

В MSX BASIC имеется возможность нестандартного ввода информации при помощи джойстика. Делается это при помощи функций `STICK()`, `STRIG()`, `PDL()` и `PAD()`.

Формат функции `STICK()` следующий:

```
STICK(α)
```

,

где:

- STICK («рукоятка») — служебное слово;
- α — арифметическое выражение, целая часть значения которого принадлежит отрезку [0,2] и задаёт номер джойстика.

```
  8   1   2
   ◤  ▲  ◥
    \ | /
7 ◀── 0 ──▶ 3
    / | \
   ◣  ▼  ◢
  6   5   4
```

Эта функция возвращает целое число, принадлежащее отрезку [0,8] и отмечающее направление перемещения рукоятки джойстика, номер которого определяется значением выражения α. Рисунок демонстрирует различные значения функции `STICK()` (когда джойстик находится в нейтральном положении, функция возвращает значение 0).

Говорят, что функция `STICK(0)` «читает» клавиши управления курсором, функция `STICK(1)` «читает джойстик» в первом разъёме, функция `STICK(2)` «читает джойстик»во втором разъёме.

Приведём возможные значения функции `STICK(0)`:

- X=1, если нажата клавиша ↑;
- X=2, если нажаты одновременно клавиши → и ↑;
- X=3, если нажата клавиша →;
- X=4, если нажаты одновременно клавиши → и ↓;
- X=5, если нажата клавиша ↓;
- X=6, если нажаты одновременно клавиши ← и ↓;
- X=7, если нажата клавиша ←;
- X=8, если нажаты одновременно клавиши ↑ и ←;
- X=0, если не нажата ни одна из этих клавиш.

Например, если Вы запустите программу
0713-01.bas
0713-01.bas

```
10 D=STICK(0):LOCATE 18,11:PRINT D:GOTO 10
```

и будете нажимать клавиши управления курсором, то будете получать значения переменной D от 0 до 8.

Если же у Вас есть джойстик, установите его в порт джойстика номер 1 (он находится на правой боковой панели

компьютера), замените в строке 10 оператор D=STICK(0) на оператор D=STICK(1) и вновь запустите программу (итог будет аналогичным).

Функция STRIG() определяет, нажата или нет нужная вам *кнопка*.

Формат функции STRIG() следующий:

```
STRIG(β)
```
,

где:

- STRIG («Stick TRIGer» — «кнопка джойстика») — служебное слово:
- β — арифметическое выражение, целая часть которого принадлежит отрезку [0,4] и указывает на соответствующую кнопку джойстика.

Функции STRIG(1) и STRIG(2) читают состояние кнопок джойстиков из основного комплекта MSX. Функции STRIG(3) и STRIG(4) читают состояние дополнительных кнопок джойстиков 1 и 2, причём:

1. функция STRIG(0) читает состояние клавиши «пробел»;
2. функция STRIG(1) читает состояние кнопки 1 джойстика в разъёме 1;
3. функция STRIG(2) читает состояние кнопки 1 джойстика в разъёме 2;
4. функция STRIG(3) читает состояние кнопки 2 джойстика в разъёме 1;
5. функция STRIG(4) читает состояние кнопки 2 джойстика в разъёме 2.

Функция STRIG() возвращает значение либо ноль, если кнопка не нажата, либо -1, если кнопка нажата.

Пример 1.

```
IF STRIG(0) THEN BEEP
```

Оператор вызывает звуковой сигнал (BEEP), если кнопка джойстика нажата (т.е. STRIG(0) имеет значение -1).

При использовании функции STRIG(0), например, в процессе компьютерной игры (скоростная стрельба по цели), она ежесекундно задействуется много раз. Для этих частых проверок значений существует специальный термин «опрос».

Метод опроса иллюстрируется примером 2. Программа начинается с вопроса пользователю о том, располагает ли он джойстиком или работает за клавиатурой. Далее запрашиваются состояния пусковых кнопок, и в зависимости от того, какая из них нажата, выбирается джойстик.

Пример 2 [76].
0713-02.bas
0713-02.bas

```
40 CLS:WIDTH 40
60 PRINT "Нажмите пусковую кнопку":PRINT "выбранного джойстика или"
80 PRINT "клавишу ПРОБЕЛ - для клавиатуры"
90 PRINT:PRINT:PRINT "Нажимайте":PRINT:PRINT
150 F=-1
160 FOR I=0 TO 4:IF STRIG(I) THEN F=I:NEXT I
190 IF F=-1 THEN GOTO 160
200 BEEP
240 IF F=0 THEN PRINT "Клавиатура ";
250 IF F=1 OR F=3 THEN PRINT "Джойстик A";
260 IF F=2 OR F=4 THEN PRINT "Джойстик B";
270 PRINT " в действии":END
```

В ожидании нажатия клавиши «пробел» опрос будет выполняться постоянно, что делает возможной *немедленную* реакцию программы. Напомним Вам, что нажатие клавиши «пробел» может также проверяться с помощью функции INKEY$, но в этом случае компьютер будет проверять текущее состояние буфера клавиатуры.

Замечание. Функция `STRIG(0)` возвращает -1 только, если клавиша «пробел» нажата во время *вычисления* значения этой функции.

Пример 3.
0713-03.bas
0713-03.bas

```
10 FOR I=0 TO 500:PRINT I:NEXT
20 IF STRIG(0) THEN 10 ELSE 20
```

Обратите внимание, что если клавиша «пробел» нажата во время вывода информации на экран, это событие не будет зафиксировано функцией `STRIG(0)`, как произошло бы в случае замены программной строки 20 на строку

```
20 IF INKEY$=CHR$(32) THEN 10 ELSE 20
```

При применении функции `STRIG(0)` компьютер «реагирует» на нажатие клавиши «пробел» *только* при выполнении 20 строки. Функция `STRIG(0)` не перехватывает предварительного нажатия клавиши «пробел», тогда как функция `INKEY$` реагирует на *все* предварительные нажатия. Сколько раз была нажата клавиша «пробел», столько раз и выполнится соответствующая команда.

Пример 4. Ввод рисунков с клавиатуры при помощи «рукоятки» (клавиши управления курсором) и «кнопки» (клавиша «пробел») встроенного «джойстика» с номером 0.

1. 0713-041.bas
   0713-041.bas

   ```
   10 CLS:X=16:Y=11
   30 D=STICK(0)
   40 IF D=1 OR D=2 OR D=8 THEN Y=Y-1
   50 IF D=4 OR D=5 OR D=6 THEN Y=Y+1
   60 IF D=2 OR D=3 OR D=4 THEN X=X+1
   70 IF D=6 OR D=7 OR D=8 THEN X=X-1
   80 IF X<0 THEN X=0 ELSE IF X>36 THEN X=36
   90 IF Y<0 THEN Y=0 ELSE IF Y>20 THEN Y=20
   95 LOCATE X,Y
   100 IF STRIG(0) THEN PRINT "∗":GOTO 30 ELSE PRINT "+":GOTO 30
   ```

   А теперь исправьте «∗» на « ». Что изменилось в работе программы?

2. 0713-042.bas
   0713-042.bas

   ```
   10 COLOR 1,15:SCREEN 2
   20 GOSUB 60:LINE (X,Y)-(X+10,Y+10)
   30 X1=X:Y1=Y:A$=INKEY$:IF A$="n" OR A$="N" THEN ELSE 20
   40 GOSUB 60:LINE (X1,Y1)-(X1+10,Y1+10),15:LINE (X,Y)-(X+10,Y+10)
   50 X1=X:Y1=Y:A$=INKEY$:IF A$="r" OR A$="R" THEN 20 ELSE 40
   60 D=STICK(0)
   70 IF D=1 OR D=2 OR D=8 THEN Y=Y-1
   80 IF D=4 OR D=5 OR D=6 THEN Y=Y+1
   90 IF D=2 OR D=3 OR D=4 THEN X=X+1
   100 IF D=6 OR D=7 OR D=8 THEN X=X-1
   110 RETURN'⟶
   ```

Пример 5. А теперь настала очередь «настоящего» джойстика!

1. 0713-051.bas
   0713-051.bas

   ```
   10 CLS:X=16:Y=11
   30 D=STICK(1)
   40 IF D=1 OR D=2 OR D=8 THEN Y=Y-1
   50 IF D=4 OR D=5 OR D=6 THEN Y=Y+1
   60 IF D=2 OR D=3 OR D=4 THEN X=X+1
   ```

```
70 IF D=6 OR D=7 OR D=8 THEN X=X-1
80 IF X<0 THEN X=0 ELSE IF X>36 THEN X=36
90 IF Y<0 THEN Y=0 ELSE IF Y>20 THEN Y=20
95 LOCATE X,Y
100 IF STRIG(1) THEN PRINT "*":GOTO 30 ELSE PRINT "+":GOTO 30
```

2. 0713-052.bas
   0713-052.bas

```
10 COLOR 1,15:SCREEN 2
20 GOSUB 60:LINE (X,Y)-(X+10,Y+10)
30 X1=X:Y1=Y:A$=INKEY$:IF A$="n" OR A$="N" THEN ELSE 20
40 GOSUB 60:LINE (X1,Y1)-(X1+10,Y1+10),15:LINE (X,Y)-(X+10,Y+10)
50 X1=X:Y1=Y:A$=INKEY$:IF A$="r" OR A$="R" THEN 20 ELSE 40
60 D=STICK(2)
70 IF D=1 OR D=2 OR D=8 THEN Y=Y-1
80 IF D=4 OR D=5 OR D=6 THEN Y=Y+1
90 IF D=2 OR D=3 OR D=4 THEN X=X+1
100 IF D=6 OR D=7 OR D=8 THEN X=X-1
110 RETURN'⟶
```

Замечание.

С помощью регистров Программируемого Звукового Генератора (PSG) можно определить состояние джойстиков (состояние клавиш управления курсором с помощью регистров PSG определить нельзя!) и изменять состояние индикатора «РУС».

Для Вас, конечно же, будет неожиданным сообщение о том, что PSG имеет 16 регистров, причём регистры с номерами 0÷13 отвечают за генерацию звука, а регистры 14 и 15 — за состояние джойстиков и индикатора «РУС».

Для того, чтобы определить состояние джойстиков нужно в регистр с номером 15 занести следующие значения:

Для джойстика 1:

7	6	5	4	3	2	1	0
–	0	–	0	–	–	1	1

Для джойстика 2:

7	6	5	4	3	2	1	0
–	1	0	–	1	1	–	–

...и прочитать регистр PSG с номером 14. Четыре его младших бита покажут состояние выбранного с помощью регистра 15 джойстика:

7	6	5	4	3	2	1	0
–	–	–	–	→	←	↓	↑

Если бит в данной позиции «сброшен», то джойстик направлен в показанном на таблице направлении.

Пример 6. Автор программы: Г.Беленький. 26.05.90.
0713-06.bas
0713-06.bas

```
10 DATA 3E,0F     :' LD   A,0Fh      ; В 15-й регистр PSG поместим
20 DATA 1E,03     :' LD   E,03h      ; 3 (для джойстика 1)
30 DATA CD,93,00  :' CALL 0093h      ; BIOS записи (оператор SOUND не
35 '                ; действует!)
```

```
40 DATA 3E,0E     :' LD   A,0Eh       ; Читаем содержимое регистра 14
50 DATA CD,96,00  :' CALL 0096h       ; BIOS  чтения
60 DATA 32,00,E0  :' LD   (E000h),A   ; Полученное число поместим в
65'               ; ячейку с адресом  &HE000
70 DATA C9        :' RET
80 FOR I=0 TO 15:READ A$:POKE &HD000+I,VAL("&H"+A$):NEXT I
90 DEFUSR=&HD000
100 A=USR(0)                          ' Прочитаем состояние джойстика 1
110 PRINT RIGHT$("0"+BIN$(NOT(PEEK(&HE000) AND 15)),4)
```

Пример 7. Автор программы: Г.Беленький. 26.05.90.
0713-07.bas
0713-07.bas

```
10 DATA 3E,0F      :' LD   A,0Fh        ; Подпрограмма, позволяющая
20 DATA 1E,83      :' ld   E,83h        ; прочитать содержимое
30 DATA CD,93,00   :' CALL 0093h        ; регистра  PSG  с номером 14,
40 DATA 3E,0E      :' LD   A,0Eh        ; для определения
50 DATA CD,96,00   :' CALL 0096h        ; состояния джойстика 1.
60 DATA 32,00,E0   :' LD   (E000h),A    ;
70 DATA C9         :' RET
80 FOR I=0 TO 15:READ A$:POKE &HD000+I,VAL("&h"+A$):NEXT I
90 DEFUSR=&HD000
100 SCREEN 2,0
110 SPRITE$(0)="ЧЧЧЧЧЧЧЧ"       ' Образ спрайта
120 X=128:Y=96                          ' Начальные координаты спрайта
140 PUT SPRITE 0,(X,Y),15,0
150 A=USR(0)                            ' Определяем состояние
160 A=PEEK(&HE000) AND 15               ' джойстика 1
170 IF A=10 OR A=14 OR A=6 THEN Y=Y-2   ' Изменение координат
180 IF A=9  OR A=13 OR A=5 THEN Y=Y+2
190 IF A=10 OR A=11 OR A=9 THEN X=X-2
200 IF A=6  OR A=7  OR A=5 THEN X=X+2
210 GOTO 140
```

Для того, чтобы изменить состояние индикатора «РУС», нужно в регистре PSG с номером 15 изменить старший бит. Если этот бит равен 0, то индикатор включается, а если 1, то выключается.

Пример 8. Автор программы: Г.Беленький. 26.05.90.
0713-08.bas
0713-08.bas

```
10 DATA 3E,0F     :' LD   A,0Fh
20 DATA 1E,00     :' LD   E,00h
30 DATA CD,93,00  :' CALL 0093h
40 DATA C9        :' RET
50 FOR I=0 TO 7:READ A$:POKE &HD000+I,VAL("&H"+A$):NEXT I
60 DEFUSR=&HD000
70 A=USR(0)                 ' Включаем индикатор "РУС"
80 FOR I=0 TO 50:NEXT I
90 POKE &HD003,128:A=USR(0) ' Выключаем индикатор "РУС"
100 FOR I=0 TO 50:NEXT I
110 POKE &HD003,0:GOTO 70
```

Кроме только что рассмотренных функций STICK и STRIG, для работы с джойстиком можно использовать функции PDL и PAD.

Формат функции PDL следующий:

```
PDL(N)
```
,

где:

- PDL — служебное слово;
- N — арифметическое выражение, целая часть которого должна принадлежать отрезку [1,12] (в противном случае последует сообщение об ошибке

  «Illegal function call».

Функция PDL(N) возвращает значение кода положения *игрового рычажка*, подключённого к разъёму для джойстика.

Функция проверяет, происходило ли действие и возвращает 255 («истина»), или 0 («ложь») — в противном случае.

Для чётных значений N считывается значение в разъёме 1, для нечётных — в разъёме 2.

Если же у Вас нет игрового рычажка, но есть *джойстик*, не огорчайтесь и читайте дальше!

Аргумент функции PDL(N)	**Действие**	**Разъём джойстика**
1	↑ (вперёд)	1
2		2
3	↓ (назад)	1
4		2
5	← (влево)	1
6		2
7	→ (вправо)	1
8		2
9	Кнопка 1 (A)	1
10		2
11	Кнопка 2 (B)	1
12		2

Основное назначение функции PAD — работа с графическим планшетом.

Формат функции PAD следующий:

```
PAD(N)
```
,

где:

- PAD («pad» — «графический планшет») — служебное слово;
- N — арифметическое выражение, целая часть которого должна принадлежать отрезку [0,7] (в противном случае последует сообщение об ошибке

  «Illegal function call».

Функция PAD(N) считывает данные с графического планшета, подключённого к разъёму для джойстика. Значение разъёма 1 считываются при N, равном 1, 2 или 3, а с разъёма 2 при N, равном 4, 5, 6 или 7.

Смысл возвращаемой величины для каждого значения N:

- 0 или 4 : возвращается 0 или -1 в зависимости от того, есть прикосновение пера к планшету или нет;
- 1 или 5 : возвращается координата X точки касания;
- 2 или 6 : возвращается координата Y точки касания;
- 3 или 7 : возвращается состояние кнопки–переключателя (-1, если кнопка нажата, иначе 0).

Однако функцию PAD() можно применять и в том случае, если у Вас нет графического планшета, но есть *джойстик*.

Функция PAD(N) проверяет, происходило ли перемещение рукоятки джойстика в соответствующем разъёме и возвращает 0 («ложь»), если рукоятка не перемещалась, или -1 («истина») — в противном случае.

Аргумент функции PAD(N)	Действие	Разъём джойстика
0	↑ (вперёд)	1
3	→ (вправо)	1
4	↑ (вперёд)	2
7	→ (вправо)	2

Функции PAD(1), PAD(2), PAD(5) и PAD(6) всегда возвращают значение 255.

VII.2. Переход к подпрограмме по событию

Переход к подпрограмме по событию (*прерывание* в программе пользователя) представляет собой особый тип условного перехода.

Для выполнения *«обычного»* условного перехода где–то в программе располагается оператор, проверяющий истинность условия и передающий управление на указанный номер строки в соответствии с результатами проверки. Переход происходит только в том случае, если при выполнении программы в последовательности операторов был *обнаружен* оператор перехода.

Что же касается перехода к подпрограмме по *событию*, то компьютер непрерывно «проверяет» условие его возникновения в то время, как программа пользователя выполняет другую работу.

Обработка прерываний при наступлении события выполняется с помощью операторов двух типов:

```
ON событие GOSUB номер строки
```
(1)

Этот оператор определяет событие, вызывающее проверку, а также место в программе, куда будет передано управление при наступлении события.

```
событие ON/OFF/STOP
```
(2)

Если событие *включено* (ON), то происходит переход, определённый оператором (1). Разумеется, что этот оператор должен быть предварительно выполнен.

Если событие *выключено* (OFF) (по умолчанию), то перехода на подпрограмму обработки не происходит.

Если событие *остановлено* (STOP), то переход при наступлении события не происходит, но информация о том, что событие произошло, запоминается, а переход происходит только после обнаружения в программе оператора

```
событие ON
```
.

Отметим, что операторы (1) или (2) не выполняют «сами по себе» переход на подпрограмму обработки события.

Когда выполнен оператор типа (1), относящийся к указанному событию, компьютер начинает проверять, не произошло ли данное событие во время выполнения программы.

Если событие произошло, текущий оператор все равно выполняется до конца. Затем компьютер проверяет, выполнен ли оператор типа (2).

Возможны *три* варианта:

1. оператор

   ```
   событие ON/OFF/STOP
   ```

   не был выполнен.

   Программа выполняется, как если бы событие не произошло;

2. оператор

   ```
   событие ON
   ```

   был выполнен и является последним оператором этого типа, относящимся к данному событию.

   Происходит переход к подпрограмме, указанной в операторе

   ```
   ON событие GOSUB номер строки
   ```
   .

   При входе в эту подпрограмму неявно выполняется оператор

   ```
   событие STOP
   ```
   .

   При выполнении оператора `RETURN` автоматически выполняется оператор

   ```
   событие ON
   ```
   ,

   и управление передаётся оператору, следующему за тем, при выполнении которого было обнаружено наступление события;

3. был выполнен оператор

   ```
   событие STOP
   ```
   ,

   и он является последним оператором обработки данного события.

   Программа продолжает выполняться так, как будто бы событие не произошло. Это событие запоминается, и если в дальнейшем в программе встречается оператор

   ```
   событие ON
   ```
   ,

   относящийся к тому же событию, то происходит переход к подпрограмме обработки события.

Имеется возможность организовать переход на подпрограммы по совершению таких событий, как истечение определённого интервала времени, нажатие какой–либо клавиши F1÷F10, нажатие клавиш CTRL+STOP и т.п. Об этом и пойдёт речь в следующих разделах.



Перед тем, как выполнить программу, содержащую переход к подпрограммам по событию, не забудьте (если работаете на компьютере MSX 2) выполнить команду `CALL NETEND`.

VII.2.1. Переход по ключу

Стоять я не могу… Мои колени слабеют…
душно… Где ключи? Ключи, ключи мои…

—А.Пушкин

Интерпретатор MSX BASIC включает в себя средство, позволяющее использовать функциональные клавиши не только для ввода некоторого слова, но и для прерывания выполнения программы и вызова определённой подпрограммы.

Оператор

```
ON KEY GOSUB n1, n2, …, nk
```
,

где:

- ON («по»), KEY («ключ»,«клавиша»), GOSUB — служебные слова;
- n1, n2, ..., nk — номера строк программы (1≤k≤10);

осуществляет передачи управления подпрограммам, расположенным со строк n1, n2, ..., nk, причём нажатие функциональной клавиши Fk (k=1, 2, ..., 10) приводит к передаче управления подпрограмме, начинающейся со строки с номером nk (k=1, 2, ..., 10). Если Вы хотите использовать только некоторые функциональные клавиши, то пропустите ненужные номера строк (оставив запятые!).

Предварительно функциональную клавишу Fk необходимо «активизировать» оператором

```
KEY(k)ON
```
.

Записанное в скобках значение k определяет,какая именно функциональная клавиша будет «активизирована». После выполнения приведённого выше оператора нажатие клавиши Fk не будет приводить к генерации какой–либо строки символов.

Теперь перед выполнением каждого нового оператора программы интерпретатор будет проверять, не была ли нажата какая–нибудь «активизированная» функциональная клавиша, и если такое нажатие произошло, то интерпретатор вызывает соответствующую подпрограмму.

Для блокирования (запрещения) такого вызова необходимо выполнить оператор

```
KEY(k)OFF
```
.

Как и в предыдущих случаях, значение в скобках k определяет номер функциональной клавиши, для которой выполняется данный оператор.

Иногда возникает необходимость временно оградить программу от прерываний, вызываемых функциональными клавишами, и в то же время запомнить каждое нажатие такой клавиши, с тем чтобы впоследствии можно было выполнить все соответствующие вызовы подпрограммы. Это достигается с помощью оператора

```
KEY(k)STOP
```
,

который временно «дезактивизирует» функциональную клавишу Fk.

Если после выполнения указанного выше оператора производились нажатия функциональной клавиши Fk, то все соответствующие вызовы подпрограмм всё–таки будут реализованы, но не сразу, а только после выполнения оператора

```
KEY(k)ON
```
,

вновь «активизирующего» эту клавишу.

Примеры.

1. 0721-01.bas
0721-01.bas

```
10 ON KEY GOSUB 100
20 KEY(1)ON:SCREEN 2
30 LINE(50,50)-(200,150),,B
40 GOTO 30
100 ' Подпрограмма
105 :::
110 BEEP:CLS
120 FOR I=10 TO 90 STEP 10
130 CIRCLE(120,100),I:NEXT I
140 CLS:RETURN 30
```

2. 0721-02.bas
0721-02.bas

```
10 ON KEY GOSUB 100
20 KEY(1)ON:SCREEN 2
30 LINE(50,50)-(200,150),,B
40 GOTO 30
100 ' Подпрограмма
105 KEY(1)ON
110 BEEP:CLS
120 FOR I=10 TO 90 STEP 10
130 CIRCLE(120,100),I:NEXT I
140 CLS:RETURN 30
```

3. Наберите пример 2, вставив строку:

```
135 IF I=50 THEN KEY(1)STOP
```

Оператор `ON KEY GOSUB` позволяет легко организовывать «листание» законченных программных фрагментов во время выполнения программы; например, страницы справочной информации 1÷3 вызываются нажатием на клавиши F1÷F3, демонстрационный пример выполняется при нажатии клавиши F4, выход из программы — при нажатии клавиши F5 и т.д.

VII.2.2. Переход по временному интервалу

Оператор

```
ON INTERVAL=α GOSUB n
```

,

где:

- `ON`, `INTERVAL` («интервал»), `GOSUB` — служебные слова;
- α — арифметическое выражение, целая часть значения которого принадлежит отрезку [1,65535];
- n — номер программной сроки, назначает интервалы времени, равные INT(α)/60 секунд, через которые компьютер периодически передаёт управление подпрограмме, расположенной с программной строки с номером n. Отсчёт временных интервалов реализуется специальным счётчиком, принимающим значения

```
1, 2, …, INT(α), 1, 2, …, INT(α), 1, 2, …, INT(α), …
```

По аналогии с пунктом VII.2.1 счётчик «активизируется» оператором

`INTERVAL ON`,

«дезактивизируется» оператором

`INTERVAL OFF`,

и временно «дезактивизируется» оператором

`INTERVAL STOP`.

Примечание. Прежде чем запустить составленную Вами программу, убедитесь в том, что интервал INT(α)/60 не короче, чем время выполнения подпрограммы обработки, иначе она будет бесконечно вызывать саму себя, ибо в конце её неявно выполняется оператор `INTERVAL ON`.

Пример 1. Проследите за выполнением следующего фрагмента при K=1, 2, 6.
0722-01.bas
0722-01.bas

```
10 INPUT K:ON INTERVAL=K*60 GOSUB 80
30 INTERVAL ON:GOSUB 100:INTERVAL STOP:GOSUB 100
50 INTERVAL ON:GOSUB 100:INTERVAL OFF:GOSUB 100
70 END
80 PRINT "Проверка":RETURN
100 FOR K=1 TO 7000:NEXT:RETURN
```

Пример 2. «Секундомер»!
0722-02.bas
0722-02.bas

```
5 INPUT "Сколько секунд отсчитать";K:ON INTERVAL=60 GOSUB 80:S=1:
  PRINT 0;TAB(4)"мин"0TAB(11)"сек":LOCATE POS(0),CSRLIN-1
10 INTERVAL ON:FOR I=1 TO K*1000/2.395:NEXT:LOCATE POS(0),CSRLIN+1:END
80 PRINT M;TAB(4) "мин";S;TAB(11) "сек":LOCATE POS(0),CSRLIN -1:S=S+1:
   IF S MOD 60=0 THEN J=0:LOCATE POS(0),CSRLIN:M=M+1:S=0
90 RETURN
```

Обработка интервала оказывается очень удобной в игровых программах, где нужно повторять какое–либо действие через одинаковые интервалы времени.

Пример 3 [76]. Данная программа позволяет наряду с вводом текста организовать отсчёт времени («часы») в правом верхнем углу экрана. Заметим, что «опрос» клавиатуры выполняется с помощью функции `INKEY$`. В этом случае функция `INPUT$()` неприменима, так как она ожидает нажатия клавиши, поэтому ввод с клавиатуры будет мешать нормальному изменению состояния счётчика времени.
0722-03.bas
0722-03.bas

```
40 CLS:WIDTH 40:ON INTERVAL=50 GOSUB 230
60 TIME=0:INTERVAL ON
80 LOCATE 16,0,0:PRINT"Время:":LOCATE 24,0:PRINT H;":";M;":";S
100 INTERVAL ON
110 A$=INKEY$:IF A$=""THEN GOTO 110 ELSE PRINT A$;
140 'Очистка заполненного экрана
160 IF CSRLIN=23 AND POS(0)=37 THEN PRINT A$:CLS:LOCATE 0,1,0:GOTO 80
170 GOTO 110
190 'Подпрограмма обработки прерывания
230 X=POS(0):Y=CSRLIN 'Запоминание координат курсора
240 S=S+1
```

```
250 IF S=60 THEN M=M+1:S=0
260 IF M=60 THEN H=H-1:M=0
270 IF H=24 THEN H=0:LOCATE 24,0,0:PRINT
280 LOCATE 24,0,0:PRINT H;":";M;":";S:LOCATE X,Y,1:RETURN
```

VII.2.3. Переход по нажатию клавиш "CTRL"+"STOP"

По оператору

```
ON STOP GOSUB n
```
,

где:

- ON, STOP, GOSUB — служебные слова;
- n — номер программной строки;

при нажатии на клавиши CTRL+STOP может осуществиться передача управления подпрограмме, расположенной со строки n. При этом действие CTRL+STOP должно быть подтверждено (ON), отменено (OFF) или приостановлено (STOP) соответственно операторами:

```
STOP ON
STOP OFF
STOP STOP
```
.

Пример.
0723-011.bas
0723-011.bas

```
10 CLS:STOP ON:ON STOP GOSUB 80
20 PRINT"Нажмите клавишу RESET !!!":GOTO 20
80 PRINT"Вам уже ничего не поможет! Делайте, как подсказывают."
90 RETURN
```

«На закуску» отметим интересный факт:
0723-012.bas
0723-012.bas

```
10 STOP ON:ON STOP GOSUB 50:FOR I=1 TO 5000:NEXT:END
50 POKE &HFBB0,1:RETURN '⟶
```

Нажатие клавиш CTRL+STOP игнорируется. «Доступ» к тексту программы становится возможным, если одновременно нажать пять клавиш: CTRL+GRAPH+SHIFT+1+3 или четыре клавиши CTRL+GRAPH+РУС+SHIFT !

Если же оператор

```
POKE &HFBB0,1
```

заменить на оператор

```
POKE &HFBB0,0
```
,

то Вас уже ничто не спасёт!

Обратите внимание, что обрабатывается только одновременное нажатие двух клавиш: CTRL+STOP. Нажатие одной клавиши STOP просто «задержит» программу. Средствами языка MSX BASIC нажатие клавиши STOP «перехватываться» не может. Однако её можно «отключить» во время выполнения функции INPUT$.

Обработка нажатия клавиш CTRL+STOP позволит избежать любого вмешательства в программу, выполняющуюся в Ваше отсутствие!

VII.2.4. Переход по джойстику

Поговорим теперь об операторе перехода к подпрограмме по нажатию кнопки джойстика:

```
ON STRIG GOSUB n[,n1][,n2][,n3][,n4]
```
,

где:

- ON, STRIG, GOSUB — служебные слова;
- n — номер первой строки подпрограммы, которой передаётся управление при нажатии клавиши пробел (кнопка «джойстика» с номером 0);
- n1 — номер первой строки подпрограммы, которой передаётся управление при нажатии кнопки 1 джойстика 1;
- n2 — номер первой строки подпрограммы, которой передаётся управление при нажатии кнопки 1 джойстика 2;
- n3 — номер первой строки подпрограммы, которой передаётся управление при нажатии кнопки 2 джойстика 1;
- n4 — номер первой строки подпрограммы, которой передаётся управление при нажатии кнопки 2 джойстика 2.

Для того чтобы переход к подпрограмме по джойстику был возможен, его можно подтвердить, отменить или приостановить соответственно операторами

```
STRIG(k)ON
STRIG(k)OFF
STRIG(k)STOP
```
,

где k — арифметическое выражение, целая часть значения которого определяет номер кнопки джойстика, INT(k)∈[0,4] (k=0 соответствует клавише пробел).

Оператор ON STRIG GOSUB приостанавливает нормальный ход выполнения программы и передаёт управление соответствующей подпрограмме, если нажата соответствующая кнопка джойстика. Переход к первому номеру строки из списка осуществляется в том случае, если нажата клавиша пробел, ко второму — если нажата кнопка 1 джойстика 1 и т.д.

Пример 1.
0724-01.bas
0724-01.bas

```
5 STRIG(0) ON:STRIG(1) ON:STRIG(3) ON
7 ON STRIG GOSUB 100,200,,300
10 FOR I=1 TO 2000:NEXT:END
100 PRINT "Нажата клавиша 'ПРОБЕЛ'"     :END
200 PRINT "Нажата кнопка 1 джойстика 1":END
300 PRINT "Нажата кнопка 2 джойстика 1":END
```

Пример 2 [76]. Моделирование работы простого реле времени.
0724-02.bas
0724-02.bas

```
50 ON STRIG GOSUB 270
60 CLS:R=RND(-TIME):PRINT "Реле времени"
100 PRINT "Когда раздастся звонок, нажмите клавишу 'пробел'"
130 T=INT(RND(1)*500)+10
140 ON INTERVAL=T GOSUB 200
```

```
150 INTERVAL ON
160 GOTO 160
190 'Подпрограмма отсчета времени
200 INTERVAL OFF:BEEP:TIME=0:STRIG(0)ON
230 GOTO 230
260 'Анализ прерывания от джойстика
270 STRIG(0) OFF
280 PRINT "Ваша реакция":PRINT"=";TIME/60;"секунд":END
```

Пока выполняется подпрограмма, всякий новый сигнал от кнопки джойстика откладывается до окончания текущей обработки.

После возврата управления из подпрограммы интерпретатор определит наличие ожидающих сигналов, после чего будет выполнен очередной переход на подпрограмму обработки.

Оператор

```
STRIG(k) ON
```

включает оператор `ON STRIG GOSUB` для соответствующего джойстика с номером k, после чего перед выполнением каждого оператора интерпретатор проверяет, нажата ли кнопка джойстика.

Оператор

```
STRIG(k) STOP
```

приостанавливает разрешение обработки прерываний по нажатию кнопки джойстика, но сохраняет информацию о нажатии.

Пример 3. Использование «встроенного» джойстика !
0724-03.bas
0724-03.bas

```
10 STRIG(0) ON:ON STRIG GOSUB 130:DEFINT A-Z
20 DATA 3,7,15,15,31,31,31,31,15,0,49,1,1,0,0,0,192,240,248,252,180,182,182,
    182,252,2,244,240,236,216,112,32 : '"Череп"
30 DATA 16,56,56,56,16,16,16,16          : 'Вид оружия
40 COLOR 15,1,1:SCREEN 2,2:X=125:Y=96
50 FOR T=1 TO 32:READA:B$=B$+CHR$(A):NEXT:SPRITE$(1)=B$:FOR I=1 TO 8:READ W:C$=C$+CHR$(W):NEXT:
   SPRITE$(2)=C$   'Формирование спрайтов
60 GOSUB 70:GOTO 60   'Основная программа.
70 D=STICK(0):IF D=1 OR D=2 OR D=8 THEN Y=Y-3   'Движение "черепа" при
80 IF D=2 OR D=3 OR D=4 THEN X=X+3            'помощи клавиш управления
90 IF D=4 OR D=5 OR D=6 THEN Y=Y+3                    'курсором
100 IF D=6 OR D=7 OR D=8 THEN X=X-3
110 IF Y<60 THEN Y=60 ELSE IF Y>145 THEN Y=145 ' Ограничение движения
120 PUT SPRITE 1,(X,Y):RETURN'⟶
130 A=X:B=Y:C=A+8   'Подпрограмма полета стрелы
140 B=B-16:PUT SPRITE 2,(A,B),8,2:PUT SPRITE 3,(C,B),8,2:GOSUB 70
150 E=STRIG(0):IF B<10 THEN IF E=0 THEN RETURN ELSE 130 ELSE 140
```

Пример 4. А теперь — использование «настоящего» джойстика !
0724-04.bas
0724-04.bas

```
10 STRIG(1) ON:ON STRIG GOSUB ,130:DEFINT A-Z
20 DATA 3,7,15,15,31,31,31,31,15,0,49,1,1,0,0,0,192,240,248,252,180,182,182,182,252,
    2,244,240,236,216,112,32 : '"Череп"
30 DATA 16,56,56,56,16,16,16,16          : 'Вид оружия
40 COLOR 15,1,1:SCREEN 2,2:X=125:Y=96
50 FOR T=1 TO 32:READA:B$=B$+CHR$(A):NEXT:SPRITE$(1)=B$:FOR I=1 TO 8:READ W:C$=C$+CHR$(W):NEXT:
   SPRITE$(2)=C$ 'Формирование спрайтов
```

```
60 GOSUB 70:GOTO 60  'Основная программа.
70 D=STICK(1):IF D=1 OR D=2 OR D=8 THEN Y=Y-3  'Движение "черепа" при
80 IF D=2 OR D=3 OR D=4 THEN X=X+3           'помощи клавиш управления
90 IF D=4 OR D=5 OR D=6 THEN Y=Y+3                  'курсором
100 IF D=6 OR D=7 OR D=8 THEN X=X-3
110 IF Y<60 THEN Y=60 ELSE IF Y>145 THEN Y=145 ' Ограничение движения
120 PUT SPRITE 1,(X,Y):RETURN'⟶
130 A=X:B=Y:C=A+8  'Подпрограмма полета стрелы
140 B=B-16:PUT SPRITE 2,(A,B),8,2:PUT SPRITE 3,(C,B),8,2:GOSUB 70
150 E=STRIG(1):IF B<10 THEN IF E=0 THEN RETURN ELSE 130 ELSE 140
```

Fix Me!

А теперь настало время описать этот оператор:

```
ON STICK(α) GOSUB n1,n2,n3,n4,n5,n6,n7,n8
```
,

где:

- `ON`, `STICK`, `GOSUB` — служебные слова;
- α — арифметическое выражение, целая часть значения которого принадлежит отрезку [0,2] и задаёт номер джойстика.
- n — номер первой строки подпрограммы, которой передаётся управление при нажатии соответствующей клавиши–стрелки на клавиатуре или джойстика. Направление соответствует определению, данному в `STICK()`

Пример 5 [101]. Тестирование стрелок клавиатуры и джойстиков
0724-05.bas
0724-05.bas

```
10 CLS:SCREEN 1:KEY OFF
20 INPUT"Stick";S
30 LOCATE 0,2
40 ON STICK(S) GOSUB 70,80,90,100,110,120,130,140
50 GOTO 30
60 PRINT "0   (0)":RETURN
70 PRINT "▲   (1)":RETURN
80 PRINT "▲ ► (2)":RETURN
90 PRINT "  ► (3)":RETURN
100 PRINT "▼ ► (4)":RETURN
110 PRINT "▼   (5)":RETURN
120 PRINT "◄ ▼ (6)":RETURN
130 PRINT "◄   (7)":RETURN
140 PRINT "◄ ▲ (8)":RETURN
```

VII.2.5. Особенности одновременной обработки событий различных типов

Обрадуем Вас: в одной программе допустима обработка нескольких событий различных типов! Однако для того, чтобы структура программы оставалась ясной, необходимо учитывать следующее:

1. Перед входом в подпрограмму обработки события всегда неявно выполняется оператор

   ```
   событие STOP
   ```
   .

   Если в процессе выполнения этой подпрограммы происходит это же событие, то подпрограмма по завершении будет сразу же *вызвана* снова. Чтобы избежать этого, помещайте в начале подпрограммы оператор

   ```
   событие STOP
   ```

или оператор

```
событие OFF
```

.

2. Применение операторов

```
событие OFF
```

и

```
событие STOP
```

«очищает» от обработки прерываний определённые участки Вашей программы.

3. Если событие остановлено (STOP), то факт возникновения этого события запоминается.

   Если же событие того же типа возникает ещё раз, то компьютер запоминает наступление только последнего события!

4. Приоритет (в порядке убывания) при одновременном возникновении событий:
   - обработка ошибок в программе пользователя;
   - ON KEY GOSUB;
   - ON STOP GOSUB;
   - ON SPRITE GOSUB;
   - ON STICK GOSUB;
   - ON STRIG GOSUB;
   - ON INTERVAL GOSUB;
   - прерывания в локальной сети;
   - другие прерывания.

В заключение запомните, что использование в Вашей программе полдюжины (≈шесть!) операторов перехода к подпрограммам по событию может мгновенно завести в тупик любого, пытающегося разобраться в ней программиста!

Пример [76]. Предположим, что Вы управляете космическим кораблём, который может перемещаться вверх и вниз. Вы можете обстреливать появившийся вражеский корабль, который также стреляет по Вашему кораблю. Программа структурирована таким образом, что возможны лишь следующие столкновения спрайтов:

1. вражеский корабль с Вашим снарядом;
2. Ваш корабль с вражеским снарядом;
3. вражеский снаряд с Вашим снарядом.

При попадании в цель вражеского снаряда Ваш корабль гибнет, и игра прекращается; если же вам удаётся одним из своих снарядов поразить противника — Ваш счёт увеличивается.

Игры такого типа безжалостно обнажают серьёзную слабость языка BASIC: это весьма *«медленный»* язык. Спрайты позволяют несколько увеличить скорость выполнения программы, но ясно, что без машинных кодов обойтись не удастся!
0725-01.bas
0725-01.bas

```
70 DEFINT A-Z:OPEN"GRP:"AS#1:DIM M(100):DIM T(7)
110 R=RND(-TIME)
120 ON STRIG GOSUB 1000
130 ON SPRITE GOSUB 1250
140 ON INTERVAL=10 GOSUB 1110
150 ON SPRITE GOSUB 1250
160 COLOR 15,1,1:SCREEN 2,0,0
190 'Начальная установка спрайтов
210 FOR I=0 TO 3
```

```
220    S$=""
230    FOR J=1 TO 8:READ S:S$=S$+CHR$(S):NEXT J
260    SPRITE$(I)=S$
270 NEXT I
280 'И з о б р а ж е н и е  ф о н а
310 DRAW"bm0,176":DRAW"s4a0"
330 LINE(0,0)-(255,191),15,B:LINE(16,2)-(239,2),15
350 LINE(16,2)-(239,189),15,B:LINE(16,16)-(239,16),15
370 DRAW"bm136,5":COLOR 15
380 T$="Счет:":GOSUB 920
390 SC=0
400 PRINT#1,USING"######";SC
420 'З в е з д н о е  н е б о
440 FOR I=1 TO 100
450    E=INT(RND(1)*224)+16:F=INT(RND(1)*160)+16:PSET(E,F)
480 NEXT I
500 'Случайное движение корабля противника
530 FOR I=1 TO 100:M(I)=INT(RND(1)*130)+16:NEXT I
560 CIRCLE(160,60),40,11:PAINT(160,60),11
570 CIRCLE(175,45),10,9,,,13/10:PAINT(175,45),9
590 'У с т а н о в к а  о т м е т о к
610 DRAW"bm10,4"
620 FOR I=10 TO 13:PUT SPRITE I,STEP(10,0),15,1:NEXT
650 X=24:Y=96:LI=14
660 F$="t180v12o2s11m100e16r16d#2"
670 M$="t180v12o6l24bgbgab"
680 W$="t180s1m3000o3c"
690 MX=200:MY=96:IX=IX+1:T=-2:F=0:J=0
700 STRIG(0)ON:SPRITE ON:INTERVAL ON
730 IF IX>100THEN IX=0
740 L=SGN(M(IX)-MY)
760 'О с н о в н о й  ц и к л
780 PUT SPRITE 0,(X,Y),15,1
790 PUT SPRITE 1,(MX,MY),15,0
800 IF F THEN GOSUB 1040
810 IF J THEN GOSUB 1160
820 IF STICK(0)=1 THEN Y=Y-2
830 IF STICK(0)=5 THEN Y=Y+2
840 IF Y<=16 THEN Y=16
850 IF Y>=176 THEN Y=176
860 MY=MY+L:IF MY=M(IX)THEN IX=IX+1:GOTO 730
870 MX=MX-T:IF MX<40 OR MX>228 THEN T=-T
880 GOTO 780
890 'Подпрограмма вывода на экран
920 FOR I=1 TO LEN(T$):PRINT#1,T$:DRAW"bm-2,0":NEXT
960 RETURN
970 'Свой  с н а р я д
1000 IF F THEN RETURN
1010 PLAY"l8s8m300o6c","l8s8m300o7c"
1020 A=Y:B=X+16:F=1
1030 RETURN
1040 B=B+7
1050 IF B>232 THEN PUT SPRITE 3,(0,209):F=0:RETURN
1060 PUT SPRITE 3,(B,A),15,2:RETURN
1070 RETURN
1080 'С н а р я д  противника
1110 IF J THEN RETURN
1120 PLAY"l16s14m800o2a"
1130 BX=MX-8:BY=MY+4
1140 PUT SPRITE 4,(BX,BY),2,3
1150 J=-1:STRIG(0)ON:RETURN
1160 BX=BX-5
1170 IF BX<16 THEN BY=209:J=0
1180 PUT SPRITE 4,(BX,BY),2,3
```

```
1190 RETURN
1200 'Подпрограмма обработки прерываний.
1210 'Определение спрайтов по отношению к их расположению
1250 SPRITE OFF:INTERVAL OFF
1260 IF A+2>MY AND A+5<=MY+7 AND B+8>=MX THEN SI=30:PLAY M$:GOTO 1300
1270 IF BY+2>Y AND BY-7<Y AND BX<34 THEN LI=LI-1:PLAY F$:GOTO 1300
1280 IF A+2>=BY AND A+5<=BY+7 AND BY+8>=36 THEN SI=10:PLAY W$:GOTO 1400
1290 SPRITE ON:INTERVAL ON:STRIG(0)ON:RETURN
1300 'В с п ы ш к а  на экране
1330 FOR I=0 TO 50
1340 COLOR,,RND(1)*16
1350 NEXT
1360 MY=96:MX=220
1370 'Удаление снаряда
1400 PUT SPRITE 3,(0,209):F=0:A=0:B=209
1410 PUT SPRITE 4,(0,209):J=0:BX=0:BY=209
1420 IX=IX+1
1430 'Увеличение счета
1460 DRAW"bm178,5":COLOR 1
1470 PRINT#1,USING"######";SC
1480 SC=SC+SI
1490 DRAW"bm178,5":COLOR 15
1500 PRINT#1,USING"######";SC
1510 COLOR,,1
1520 IF LI=9 THEN GOTO 1590
1530 PUT SPRITE LI,(0,209)
1540 SPRITE ON:INTERVAL ON
1550 RETURN 730
1560 'К о н е ц  и г р ы
1590 STRIG(0)OFF:INTERVAL OFF:SPRITE OFF
1600 PUT SPRITE 0,(0,209)
1610 PUT SPRITE 1,(0,209)
1620 PUT SPRITE 3,(0,209)
1630 PUT SPRITE 4,(0,209)
1640 LINE(24,60)-(232,76),15,BF
1650 DRAW"bm26,64":COLOR 1
1660 T$="     ИГРА ОКОНЧЕНА"
1670 GOSUB 920
1680 FOR I=1 TO 5000:NEXT
1690 S=STICK(0):IF S=0 THEN 1690
1700 IF S<>1 THEN 1720
1710 CLS:GOTO 310
1720 COLOR 15,4,4
1730 END
1740 'Д а н н ы е  для  с п р а й т о в
1770 DATA 0, 60, 66,255, 66,255, 66, 60
1780 DATA 0,224, 68,255,255, 68,224,  0
1790 DATA 0, 64, 34, 63, 34, 64,  0,  0
1800 DATA 0,  2, 68,252, 68,  2,  0,  0
```

VII.3. Инициализация в языке MSX BASIC

При включении компьютера он приводится автоматически в некоторое исходное состояние (происходит его *инициализация*); так, например, экран устанавливается в режим `SCREEN 0` и окрашивается в определённый цвет.

В языке MSX BASIC для компьютеров MSX 2 существуют операторы для дополнительной инициализации. Действие этих операторов в своём роде уникально:



будучи *один* раз выполненными, они воздействуют на компьютер при *каждом* включении (установленные ими значения запоминаются в небольшой по объёму памяти, сохраняющей своё содержимое при



отключённом основном питании) (при наличии батарей питания RTC)!

1. Оператор

```
SET ADJUST(X,Y)
```

, где:

   - SET («установить»), ADJUST («корректировка») — служебное слово;
   - X и Y — арифметические выражения, целые части значений которых принадлежат отрезку [-7,8] (по умолчанию X=0,Y=0), регулирует положение используемой части экрана.

2. Оператор

```
SET BEEP [звук][,громкость]
```

, где:

   - SET, BEEP — служебные слова;
   - *звук* — арифметическое выражение, целая часть значения которого принадлежит отрезку [1,4]:
     - 1 — громкий (как на MSX1);
     - 2 — тихий;
     - 3 — 2-тональный;
     - 4 — 3-тональный;
   - *громкость* — арифметическое выражение, целая часть которого принадлежит отрезку [1,4];

   устанавливает звук и громкость при выполнении оператора BEEP.

   Любой из параметров может быть опущен (его значение совпадёт со значением, установленным ранее), причём, если опущен первый параметр, то запятая должна быть сохранена.

3. Оператор

```
SET TITLE ["заголовок"][,цвет]
```

, где:

   - SET, TITLE («заголовок») — служебные слова;
   - *заголовок* — значение строковой константы, длина которого не должна превышать 6 ;
   - *цвет* — арифметическое выражение, целая часть значения которого принадлежит отрезку [1,4]:
     - 1 — синий;
     - 2 — зелёный;
     - 3 — красный;
     - 4 — оранжевый;

   позволяет переопределять заголовок, возникающий на экране при включении компьютера. Если *заголовок* имеет длину 6, то система будет «ожидать» нажатия любой клавиши после вывода заголовка на экран.

4. Оператор

```
SET PASSWORD "пароль"
```

, где:

   - SET, PASSWORD — служебные слова;
   - *пароль* — значение строковой константы, длина которого не должна превышать 255 ;

   позволяет ввести пароль, проверяемый всякий раз при включении компьютера.

Если Вы забыли пароль, нажмите клавиши STOP и GRAPH при включении.

5.

Оператор

```
SET PROMPT "подсказка"
```
,

где:

- SET, PROMPT — служебные слова;
- подсказка — значение строковой константы, длина которого не более 6 символов;

позволяет переопределить стандартную системную подсказку Ok.

Примечание. Можно сохранить результат действия только одного из операторов SET TITLE, SET PASSWORD или SET PROMPT: использованный последним «замещает» предыдущий.

Пользуясь этой возможностью, можно отменить пароль, вернуться к исходному заголовку, к подсказке «Ok». Достаточно ввести команду

```
SET PROMPT "Ok"
```

6.

Оператор

```
SET SCREEN
```
,

где: SET, SCREEN — служебные слова; позволяет *сохранить* значения параметров, относящихся к состоянию экрана, которые были ранее установлены операторами SCREEN, COLOR, KEY ON/OFF, WIDTH.

Сохраняются только те параметры, которые могут вводиться в командном режиме:

- режим экрана (0 или 1);
- ширина экрана (от 1 до 80, если режим экрана =0; от 1 до 32, если режим экрана =1);
- цвет текста (от 0 до 15);
- цвет фона (от 0 до 15);
- цвет бордюра (от 0 до 15);
- вкл./выкл. отображение текста функциональных клавиш (ON или OFF);
- размер спрайтов (от 0 до 3);
- звук при нажатии клавиш (0 или 1);
- скорость передачи информации (1200 или 2400 бод);
- тип используемого принтера (MSX или другой тип);
- режим вывода (от 0 до 3).

При включении компьютера неявно выполняются операторы SCREEN, COLOR, KEY ON/OFF и WIDTH, параметры которых были сохранены.

Этот оператор задаёт режим экрана, размер и масштаб увеличения спрайтов; управляет эхо-сигналом при нажатии клавиш, скоростью передачи данных на «CAS:» и объявляет тип подключённого принтера; определяет режим вывода на экран.

Мартобря 86 числа.

—Н.Гоголь. Записки сумасшедшего

7.

Оператор

```
SET DATE "ММ/ДД/ГГ",[A]
```
,

где:

- SET, DATE («дата») — служебные слова;

- ММ/ДД/ГГ — значение строковой константы, определяющей дату;
- A — необязательный параметр;

определяет дату в формате:

```
ММ/ДД/ГГ
```
,

где:

- ММ — месяц (от 00 до 12);
- ДД — день (от 00 до 31);
- ГГ — две последние цифры года, диапазон: 1980–2079 [4)]
  - 00–79 (2000–2079)
  - 80–99 (1980–1999)

Отметим, что наличие нулей обязательно!

Например:

```
SET DATE "01/05/89"
```

Если присутствует параметр A, то дата будет рассматриваться как *дата* подачи сигнала («будильник с календарём») (при этом игнорируются месяц и год).

8.

Оператор

```
SET TIME "ЧЧ:ММ:СС",[A]
```
,

где:

- SET, TIME — служебные слова;
- ЧЧ:ММ:СС — значение строковой константы, определяющей время;
- A — необязательный параметр;

задаёт время в следующем формате:

```
ЧЧ:ММ:СС
```
,

где:

- ЧЧ — часы (от 00 до 23);
- ММ — минуты (от 00 до 59);
- СС — секунды (от 00 до 59).

Отметим, что наличие нулей обязательно!

Присутствие параметра A задаёт *время* подачи сигнала («будильник»), причём секунды игнорируются. Учтите, что используя параметр A, всегда необходимо вводить вначале *время*, а лишь затем *дату* !

> Всё врут календари.
>
> *—А.С.Грибоедов. Горе от ума*

Для получения даты и времени MSX BASIC предоставляет также ещё два оператора:

- 1)

```
GET DATE Z$,[A]
```
,

где:

- GET («получать»), DATE — служебные слова;
- Z$ — строковая переменная;
- A — необязательный параметр.

Этот оператор присваивает значение даты переменной Z$ в формате ММ/ДД/ГГ .

Команда `GET DATE Z$,A` возвращает значение, определяющее дату подачи сигнала.

- 2)

```
GET TIME Z$,[A]
```

,

где:

- GET, TIME — служебные слова;
- Z$ — строковая переменная;
- A — необязательный параметр.

Этот оператор присваивает значение времени переменной Z$ в формате ЧЧ:ММ:СС .

Команда `GET TIME Z$,A` возвращает значение, определяющее время подачи подачи сигнала.

Отметим, что подача сигнала не является автоматической. Для её осуществления необходимо использовать подпрограмму, сравнивающую текущие дату и время с заданными параметром A и в случае совпадения выдающую сигнал BEEP или определённое сообщение.

Примеры:

- 1)

```
set date "01/17/89"
Ok
get date d$:?d$
01/17/89
Ok
```

- 2)

```
set date "01/17/89",a
Ok
get date d$,a:?d$
00/17/00
Ok
```

- 3)

```
set time "20:45:30" │ Начался отсчёт времени
Ok
get time t$:?t$
20:45:40 ◄─── 10 с прошло с момента выполнения предыдущей команды
Ok
```

- 4)

```
set time "20:45:30",a
Ok
get time t$,a:?t$
20:45:00
Ok
```

- 5)

```
set time "20:45:30":set time"20:50:30",a
Ok
get time t$:get time tm$,a:?t$,tm$
20:45:43      20:50:00
Ok
Более того:
get time m$,a:?m$ ◄─── !!!
20:50:00
```

```
Ok
```

- 6) Ваш компьютер можно использовать и в бытовых целях. Если не хотите опаздывать на свидание, пользуйтесь возможностями компьютеров MSX 2 !
  073-06.bas
  073-06.bas

```
10 SET BEEP 3,4
20 SET TIME "22:30:45":SET DATE "01/17/89"
30 SET TIME "22:31:45",A:SET DATE "01/17/89",A
40 GET TIME TM$,A:GET DATE DT$,A
50 GET TIME T$:GET DATE D$
60 IF MID$(T$,1,6)+"00"=TM$ AND "00"+MID$(D$,3,4)+"00"=DT$ THEN BEEP:GOTO 50 ELSE GOTO 50
```

- 7) Если Вы хотите сохранить несколько версий программы, с которой Вы работаете, то Вас наверняка заинтересует следующий пример:
  073-07.bas
  073-07.bas

```
10 GET TIME Y$
20 SAVE"A"+LEFT$(Y$,2)+MID$(Y$,4,2)+RIGHT$(Y$,2)
30 'Основная программа
```

  Запись программы в 14 ч 53 мин 33 с приведёт к созданию файла с именем A145333. Записанные ранее версии Вашей программы остаются в неприкосновенности!

- Настройка часов реального времени.[5)]
  dt.bas
  dt.bas

Также средством инициализации является файл AUTOEXEC.BAS.

Fix Me!

Оператор CALL служит для вызова дополнительных инструкций [101]. Он имеет следующий синтаксис:

```
CALL <инструкция> [(<параметр>[,<параметр>])]
```
,

где

- CALL («вызов»)
- инструкция
- параметр — один или несколько паратетров

Слово CALL можно заменить символом «_» (подчёркивание) после которого не требуется пробелов.

Используется в следующих случаях:

- Использование RAM–диска
- MSX Disk BASIC
- MSX Network BASIC
- MSX Music

VII.4. Работа с принтером



Примеры программ этой главы были проверенны только в эмуляторе openMSX [6)].

Вначале напомним известные вам факты.

С помощью оператора PRINT можно «печатать» на *экране* дисплея текст в любом месте экрана (ему «активно помогает» в этом оператор LOCATE).

При печати на *бумаге* оператора, аналогичного LOCATE, нет. Более того, при печати нельзя возвращаться в предыдущие строки (прогон бумаги осуществляется только *вперёд*). Возврат в начало текущей строки из её конца *возможен*.

Установка печатающей головки в заданное место строки производится с помощью так называемого *табулятора* — команды TAB (его использование в «обычном» операторе PRINT, конечно же, вам известно).

Пример 1.
074-01.bas

```
10 FOR I=1 TO 50
20    LPRINT TAB(25-20*SIN(I/3));"Принтер!"
30 NEXT I
```

Арифметическое выражение, задающее номер позиции печатающей головки, указывается после служебного слова TAB в скобках и значение его целой части не должно превосходить 255.

Использование в команде LPRINT специальных управляющих кодов и функции CHR$ позволяет управлять принтером, например, менять вид шрифта, его плотность и высоту.

И наконец,

```
1 дюйм ≈ 2,54 см
```

VII.4.1. Команды для управления типом печати

Эти команды используются для управления типом шрифта, шагом печати и «создания» специальных эффектов.

VII.4.1.1. Управление типом шрифта

Взгляните на следующую таблицу:

Функция	Управляющий код
Установка стандартных символов	`CHR$(27);"5"`
Установка наклонной печати («италик»)	`CHR$(27);"4"`
Печать графического символа MSX	`CHR$(1);CHR$(Код символа)`

По умолчанию (при включении питания) осуществляется печать *стандартными* символами. Отметим, что первые два кода *отменяют* друг друга.

Пример 2.
074-02.bas

```
10 LPRINT CHR$(27);"4":LPRINT "Данная строка напечатана италиком!"
20 LPRINT CHR$(27);"5":LPRINT "Эта строка напечатана стандартным шрифтом"
```

Строка 10 включает тип шрифта *«италик»* с помощью кода

```
CHR$(27);"4"
```
,

а строка 20 выключает его кодом

`CHR$(27);"5"` .

```
^4Буквы текста, напечатанного "италиком", наклонены вправо^5
^4"Италик" может использоваться для выделения некоторых букв и слов^5
```

Заметим, что *каждый* тип шрифта может комбинироваться с любым из трёх шагов шрифта, а также с расширенной и выделенной печатью, о которых речь пойдёт ниже.

VII.4.1.2. Управление шагом шрифта

Для печати используются три плотности шага:

- *цицеро (нормальная)*,
- *элите*,
- *уплотнённая*.

Каждая из них может быть изменена с помощью команды расширения режима печати, при этом шаг увеличивается в два раза (см. раздел VII.4.1.3.).

Предлагаем Вашему вниманию таблицу:

Плотность шага	Знаки/дюйм	Управляющий код
Цицеро	10	`CHR$(27);"P"`
Цицеро	10	`CHR$(27);"N"`
Элите	12	`CHR$(27);"E"`
Уплотнённая	17	`CHR$(27);"Q"`

Поговорим о шаге *«цицеро»*. Команда, его устанавливающая, определяет максимальное число печатаемых колонок до 80 в печатающем устройстве десятидюймового типа (например, «Gemini–10XR») и до 136 — в печатающем устройстве пятнадцати–дюймового типа. При этом плотность шага устанавливается равной 10 знакам на дюйм в режиме печати текста и 160 точкам на дюйм в режиме *растровой* (точечной) графики (о ней мы расскажем ниже).

Немного об *«элите»*. Выбор этого шага устанавливает максимальное число печатаемых колонок до 96 в печатающем устройстве десятидюймового типа и до 163 в печатающем устройстве пятнадцати–дюймового типа. В режиме растровой печати плотность шага устанавливается равной 90 точкам на дюйм.

Установка *уплотнённой* печати приводит к тому,что вся последующая печать выполняется с плотностью 17 знакам на дюйм. Данная команда также устанавливает максимальное число печатаемых колонок на 136 в печатающем устройстве десятидюймового типа и на 233 в печатающем устройстве пятнадцати–дюймового типа.

Пример 3.
074-03.bas

```
10 LPRINT CHR$(27);"N":LPRINT "WHAT IS YOUR NAME?"
30 LPRINT CHR$(27);"E":LPRINT "YOUR PRINTER PRINTS IN ELITE PITCH"
50 LPRINT CHR$(27);"Q":LPRINT "AS WELL AS CONDENSED, THE NARROOWEST PITCH"
70 LPRINT CHR$(27);"P"
80 LPRINT "PICA PITCH PRINTING IS PRINTER'S THIRD PITCH"

run
^N
  WHAT IS YOUR NAME?
^E
  YOUR PRINTER PRINTS IN ELITE PITCH
^Q
  AS WELL AS CONDENSED, THE NARROOWEST PITCH
^P
  PICA PITCH PRINTING IS PRINTER'S THIRD PITCH
^E
```

```
  Ok
```

Пример 4.
074-04.bas

```
10 LPRINT CHR$(27);"P":GOSUB 100
20 LPRINT CHR$(27);"E":GOSUB 100
30 LPRINT CHR$(27);"Q":GOSUB 100
40 LPRINT CHR$(27);"N":GOSUB 100:END
100 LPRINT "Я Вас любил...":RETURN

run
^P
 Я Вас любил...
^E
 Я Вас любил...
^Q
 Я Вас любил...
^N
 Я Вас любил...
^E
  Ok
```

Различные типы плотности можно комбинировать с «италиком» и стандартным шрифтом.

VII.4.1.3. Специальные режимы печати

«М*** сказал молодому человеку, который не замечал,что его любит женщина:«Вы ещё так юны, что, видно, разбираете только крупный шрифт».

—С.Шамфор. Характеры и анекдоты

Коды для *расширенной* печати

Функция	Управляющий код
Установка режима расширенной печати	`CHR$(14)`
Отмена режима расширенной печати	`CHR$(15)`

Режим расширенной печати можно комбинировать с различной плотностью шага. При этом каждый символ будет напечатан в два раза шире, чем это предусмотрено выбором плотности шага.

Пример 5.
074-05.bas

```
10 LPRINT CHR$(14);"EXPANDED PICA PITCH"
20 LPRINT CHR$(27);"E":LPRINT "EXPANDED ELITE PITCH"
40 LPRINT CHR$(27);"Q":LPRINT "EXPANDED CONDENSED PITCH"
60 LPRINT CHR$(27);"P";CHR$(15)
70 LPRINT "NOW WE ARE BACK TO UNEXPANDED PICA PRINTING"
run
^N
  EXPANDED PICA PITCH
^E
  EXPANDED ELITE PITCH
^Q
  EXPANDED CONDENSED PITCH
^P
```

```
  NOW WE ARE BACK TO UNEXPANDED PICA PRINTING
^E
  Ok
```

Коды *выделения* печати

Функция	Управляющий код
Выбор печати двойным ударом	`CHR$(27);"G"`
Отмена печати двойным ударом	`CHR$(27);"H"`
Выбор выделенной печати	`CHR$(27);"!"`
Отмена выделенной печати	`CHR$(27);CHR$(34)`

Выделить текст можно *двумя* способами:

1. используя двойную печать со смещением вниз (печать двойным ударом);
2. используя двойную печать со смещением вправо (выделенная печать).

В обоих этих случаях символы пропечатываются *дважды*.

Выделенную печать рекомендуется использовать с шагом «цицеро». Для другой плотности печати используйте печать *двойным* ударом.

Заметим, что отмена печати двойным ударом возвращает печатающее устройство к предыдущему стилю печатания, а отмена выделенной печати возвращает печатающее устройство к стандартной печати.

Пример 6.
074-06.bas

```
10 LPRINT CHR$(14):GOSUB 100      'Расширенная печать включена
20 LPRINT CHR$(27);"4":GOSUB 100  '"Италик" включен
30 LPRINT CHR$(15):GOSUB 100      'Расширенная печать выключена
35 LPRINT CHR$(27);"5":GOSUB 100:END
100 LPRINT CHR$(27);"G":GOSUB 1000:LPRINT CHR$(27);"H"
110 LPRINT CHR$(27);"!":GOSUB 1000:LPRINT CHR$(27);CHR$(34)
120 END
1000 LPRINT "БУДЬТЕ СЧАСТЛИВЫ !!! Живите долго!":RETURN

run
^N
 БУДЬТЕ СЧАСТЛИВЫ !!! Живите долго!
^4
  БУДЬТЕ СЧАСТЛИВЫ !!! Живите долго!

  БУДЬТЕ СЧАСТЛИВЫ !!! Живите долго!
^5
  БУДЬТЕ СЧАСТЛИВЫ !!! Живите долго!
^G
  БУДЬТЕ СЧАСТЛИВЫ !!! Живите долго!
^H^!
  БУДЬТЕ СЧАСТЛИВЫ !!! Живите долго!
^"^E    Ok
```

Пример 7.
074-07.bas

```
10  LPRINT  CHR$(27);"Q";"Миша";CHR$(27);"P";"  +  ";CHR$(27);"4";"Ирочка  ";CHR$(27);"5";"=
";CHR$(27);"!";"ЛЮБОВЬ  !!!";CHR$(15);CHR$(27);"Q";" Караул";CHR$(27);CHR$(34)
run
^QМиша^P + ^4Ирочка ^5= ^!ЛЮБОВЬ  !!!^Q Караул^"^c1^E
Ok
```

Коды для *однонаправленной* печати

Функция	Управляющий код
Выбор однонаправленной печати	`CHR$(27);">"`
Отмена однонаправленной печати	`CHR$(27);"]"`

Однонаправленная печать используется в таблицах и схемах, так как обеспечивает *строго* вертикальное расположение символов и выравнивает печать вертикальных столбцов. После включения принтера устанавливается режим двунаправленной печати, которая несколько быстрее однонаправленной.

Пример 8.
074-08.bas

```
20 FOR I=1 TO 10:LPRINT "++++++++++++++++++++":NEXT I
50 LPRINT:LPRINT:LPRINT CHR$(27);">"
70 FOR I=1 TO 10:LPRINT "++++++++++++++++++++":NEXT I
```

Коды для *подчёркивания*

Функция	Управляющий код
Выбор подчёркивания	`CHR$(27);"X"`
Отмена подчёркивания	`CHR$(27);"Y"`

Пример 9.
074-09.bas

```
10 LPRINT CHR$(27);"Q"                'Уплотненный шаг включен
20 LPRINT "Текст ";
30 LPRINT CHR$(14);"можно ";CHR$(15);"выделить";
40 LPRINT CHR$(27);"X";" ПОДЧЕРКИВАНИЕМ";CHR$(27);"Y":END
run

^QТекст можно выделить^X ПОДЧЕРКИВАНИЕМ^Y^c1^E
 Ok
```

Использование кода

```
CHR$(27);"X"
```

приводит к автоматическому подчёркиванию *всех* последующих печатаемых символов (в том числе и пробелов).

Отмена подчёркивания осуществляется командой:

```
LPRINT CHR$(27);"Y"
```

Коды печати *верхнего и нижнего* индексов

Функция	Управляющий код
Выбор верхнего индекса	`CHR$(27);"s1"`
Выбор нижнего индекса	`CHR$(27);"s2"`
Отмена верхнего и нижнего индексов	`CHR$(27);"s0"`

Верхние и нижние индексы по высоте составляет половину *строчных* символов. *Верхние* индексы печатаются наравне с верхней частью обычного символа, а *нижние* индексы — наравне с нижней частью обычного символа. Индексы часто применяются для ссылок, примечаний и в математических формулах.

Пока печатающее устройство находится в режиме печати верхних или нижних индексов, обычная двунаправленная печать отменяется и заменяется однонаправленной. Более того, при печати верхних и нижних индексов устанавливается режим *двойного удара*.

Выбор верхних и нижних индексов можно совмещать с выбором шрифта «италик», однако индексы нельзя использовать с подчёркиванием и расширенной печатью.

Заметим, что команда *отмены* верхних и нижних индексов отключает и направленную печать, и двойной удар.

Пример 10.
074-10.bas

```
10                                                                                  LPRINT
"очень";CHR$(27);"s1";"полезная";CHR$(27);"s0";"ВОЗМОЖНОСТЬ";CHR$(27);"s2";"ПРИНТЕРА";CHR$(27);
"s0"
run
очень^s1 полезная ^s0ВОЗМОЖНОСТЬ^s2 ПРИНТЕРА^s0
Ok
```

VII.4.2. Управление вертикальной позицией печатающей головки

1. *Изменение высоты строки*

Функция	Управляющий код
Установка высоты строки на n/144 дюйма	`CHR$(27);"T";"n1";"n2"`
Установка высоты строки на 1/6 дюйма	`CHR$(27);"A"`
Установка высоты строки на 1/8 дюйма	`CHR$(27);"B"` (переключатель 1–2 выключен)
Установка высоты строки на 1/9 дюйма	`CHR$(27);"B"` (переключатель 1–2 включён)

При включении печатающего устройства плотность строк устанавливается равной 6 строкам в дюйме. Однако для данного печатающего устройства можно установить высоту строки на любое требуемое значение. Это делается командой:

```
LPRINT CHR$(27);"T";"n1";"n2"
```
.

где n1 и n2 — цифры от 0 до 9, определяющие высоту строки в n/144 дюйма, величины n, n1 и n2 связаны при помощи формулы:

```
n=10×n1+n2
```
.

А теперь посмотрите, как легко и просто изменяется плотность строк.

Пример 11.
074-11.bas

```
10 FOR I=48 TO 50
20   FOR J=48 TO 57
30      LPRINT CHR$(27);"T";CHR$(I);CHR$(J)
40      LPRINT "VARIABLE LINE SPACING WITH PRINTER"
50   NEXT J
60 NEXT I
70 LPRINT CHR$(27);"A"
```

Приведём программу, «настраивающую» принтер на «очень мелкую» печать (она пригодится вам для изготовления шпаргалок!)

Пример 12.
074-12.bas

```
10 LPRINT CHR$(27);"T07";CHR$(27);"s1";CHR$(27);"Q"
```

Имеется и другой управляющий код для задания высоты строки.

Пример 13.
074-13.bas

```
10 FOR I=1 TO 20
20    LPRINT CHR$(27);"B":LPRINT "LINE NUMBER";I
40 NEXT I
```

В зависимости от положения двухпозиционного переключателя 1–2 (включён или выключен), который находится внутри принтера, задаются две различные высоты. Попробуйте!

2. *Управление переводом строк*

Функция	Управляющий код
Возврат печатающей головки на левый край	`CHR$(13)`
Перемещение бумаги вперёд на одну строку	`CHR$(10)`
Подача бумаги вперёд на n строк	`CHR$(31);CHR$(n)`
Подача бумаги на n строк без возврата каретки	`CHR$(27);"a";CHR$(n)`

Эти коды применяются для перемещения бумаги относительно печатающей головки.

Возврат печатающей головки в исходную позицию (возврат каретки) производится командой

```
LPRINT CHR$(13)
```

При этом бумага не продвигается.

«Получив» код `CHR$(10)`, печатающее устройство продвигает бумагу *вперёд* на одну строку.

Для перемещения печатающей головки на левый край *следующей* строки примените команду:

```
LPRINT CHR$(13);CHR$(10)
```

Пример 14. Сравните результаты работы командных строк:
074-14.bas

```
1 LPRINT "PROGRAM";CHR$(10);"BASIC"
2 LPRINT "PROGRAM";CHR$(13);"BASIC"
```

Команда

```
LPRINT CHR$(31);CHR$(n)
```
,

где n — целое число (1≤n≤15), позволяет продвинуть бумагу *вперёд* на n строк.

Команда

```
LPRINT CHR$(27);"a";CHR$(n)
```
,

где n — целое число (1≤n≤255), перемещает бумагу на n строк (используя высоту текущей строки) без перемещения печатающей головки. Однако это не изменяет текущее значение вертикального положения табуляции. Отметим, что данную команду можно записать и в таком виде:

```
LPRINT CHR$(27);CHR$(97);CHR$(n)
```
,

где 97 — код символа «a».

Пример 15.
074-15.bas

```
10 LPRINT "LINE NUMBER 1.":LPRINT "LINE NUMBER 2.";
30 LPRINT CHR$(27);"a";CHR$(3);
```

```
40 LPRINT "LINE NUMBER 3.":LPRINT "LINE NUMBER 4."
```

Программная строка 30 позволяет переместить страницу вверх на 3 строки, печатающая головка при этом не перемещается. Следовательно, строка 40 печатает сообщение начиная с колонки, в которой печатающая головка осталась в конце строки 20.

3. *Управление страницами*

Функция	Управляющий код
Перемещение бумаги на верхнюю часть следующей страницы	CHR$(12)
Перемещение бумаги в позицию следующей вертикальной метки табуляции. Позиции вертикальной табуляции устанавливаются на строках с номерами, кратными 6 (6, 12, 18, ... 60)	CHR$(11)

Команда

```
LPRINT CHR$(11)
```

приводит к перемещению печатающей головки *вниз* в позицию следующей *вертикальной* метки табуляции или на верхнюю часть следующей страницы в порядке первоочерёдности нахождения.

Пример 16. Перед тем как запустить программу, следует выключить печатающее устройство и отрегулировать бумагу таким образом, чтобы лист был на одном уровне с верхом направляющей ленты на печатающей головке, а затем включить печатающее устройство.
074-16.bas

```
10 LPRINT "LINE NUMBER 1.":LPRINT "LINE NUMBER 2.";
30 LPRINT CHR$(12);
40 LPRINT "LINE NUMBER 3.":LPRINT "LINE NUMBER 4."
50 LPRINT CHR$(11);
60 LPRINT "LINE NUMBER 5.":LPRINT "LINE NUMBER 6."
```

VII.4.3. Управление горизонтальной позицией печатающей головки

Как вам известно, любой текст имеет правую и левую границу, называемые *полями*.

1. Поговорим об *установке полей*

Функция	Управляющий код
Установка левого поля в колонке n	`CHR$(27);"L";"n1n2n3"`
Установка правого поля в колонке n	`CHR$(27);"/";"n1n2n3"`

Команда

```
LPRINT CHR$(27);"L";"n1n2n3"
```

устанавливает позицию *начала* строки, определённую трёхзначным числом. Например, при выполнении команды

```
LPRINT CHR$(27);"L";"126"
```

левая граница будет сдвинута на 126 символов.

Команда

```
LPRINT CHR$(27);"/";"n1n2n3"
```

устанавливает позицию *конца* строки, определённую трёхзначным числом.

Например, при выполнении команды

```
LPRINT CHR$(27);"/";"065"
```

правая граница будет установлена на 65 символов.

Для печатающего устройства *десяти–дюймового* типа (например, Gemini–10XR) максимальная печатная позиция равна:

для шага *«цицеро»*	80
для шага *«элите»*	96
для *уплотнённого* шага	136

Для печатающего устройства *пятнадцати–дюймового* типа максимальная печатная позиция равна:

для шага *«цицеро»*	136
для шага *«элите»*	163
для *уплотненного* шага	233

Пример 17. Установка полей.
074-17.bas

```
10 GOSUB 40
20 LPRINT CHR$(27);"L";"010";:LPRINT CHR$(27);"/";"070"
30 GOSUB 40:END
40 FOR I=1 TO 80:LPRINT "=";:NEXT I:RETURN
```

Подпрограмма осуществляет печать 80 символов «=» в одной строке. При первом вызове подпрограммы все символы «=» печатаются в одной строке. Затем строка 20 устанавливает левое и правое поля. Теперь, после обращения к подпрограмме в строке 30, на одной строке уместятся лишь 60 символов, остальные будут напечатаны со следующей строки.

Для того, чтобы установить поля в исходное положение *по умолчанию*, имеются две возможности:

1. Вы можете выключить и затем снова включить печатающее устройство;
2. установить поля равными значениям по умолчанию. Это означает, что Вы должны установить левое поле на 0–й и правое поле на 80–й позиции на печатающем устройстве *десяти–дюймового* типа или 136 — на печатающем устройстве *пятнадцати–дюймового* типа.

Если изменять шаг печати после установки полей, поля не будут изменяться (количество символов в строке останется тем же)!

2. *Установка горизонтальных меток*

Функция	Управляющий код
Перемещение на следующую горизонтальную метку	`CHR$(9)`
Установка положений горизонтальных меток	`CHR$(27);"(";"n11n12n13,...,nn1nn2nn3."`
Стирание установленных горизонтальных меток (селективное стирание меток)	`CHR$(27);")";"n11n12n13, …,nn1nn2nn3."`
Стирание всех горизонтальных меток	`CHR$(27);"2"`

Версия MSX BASIC предоставляет возможность использования двух режимов работы с горизонтальными метками. Дело в том, что у компьютера и у принтера *разный* набор меток.

У компьютера метки располагаются строго через *восемь* позиций. У принтера же сразу после его включения горизонтальные метки автоматически устанавливаются через *десять* печатных позиций. Кроме того, набор меток печатающего устройства можно изменять программным путём.

Информация о том, из какого набора будут взяты горизонтальные метки табуляции, хранится в ячейке памяти с адресом &HF418. Если содержимое этой ячейки равно нулю, то принтер будет работать с метками из набора компьютера. Если же содержимое ячейки памяти отлично от нуля,то печатающее устройство будет работать с горизонтальными метками из своего набора.

Для перемещения печатающей головки в позицию следующей горизонтальной метки табуляции применяется команда

```
LPRINT CHR$(9);
```

Пример 18. Демонстрация использования горизонтальных меток.

- 1)
  074-181.bas

```
10 LPRINT "ONE";CHR$(9);"TWO";CHR$(9);"THREE";CHR$(9);"FOUR"
run
  ONE     TWO     THREE   FOUR
          ▲       ▲       ▲
          |       |       |
8-я позиция  16-я позиция 24-я позиция
Ok
```

- 2)
  074-182.bas

```
1 POKE &HF418,1
10 LPRINT "ONE";CHR$(9);"TWO";CHR$(9);"THREE";CHR$(9);"FOUR"
run
ONE      TWO       THREE     FOUR
         ▲         ▲         ▲
         |         |         |
 10-я позиция 20-я позиция 30-я позиция
Ok
```

- 3) Установка позиции горизонтальной табуляции (установка положений меток) осуществляется следующей командой:

```
LPRINT CHR$(27);"(";"n11n12n13,n21n22n23,...nn1nn2nn3."
```

При этом аннулируются все позиции текущей горизонтальной табуляции и навливаются новые, определённые трёхзначными числами.
074-183.bas

```
1 POKE &HF418,1
10 LPRINT CHR$(27);"(";"008,016,024."
20 LPRINT "ONE";CHR$(9);"TWO";CHR$(9);"THREE";CHR$(9);"FOUR"
run
  ONE    TWO     THREE   FOUR
         ▲       ▲       ▲
         |       |       |
8-я позиция 16-я позиция 24-я позиция
Ok
```

Метки будут установлены в колонках 8, 16 и 24. Запятые в 10-й строке используются для разделения цепочки меток, точка — для окончания определения.

- 4) Для того чтобы стереть часть горизонтальных меток из набора печатающего устройства, существует следующая команда:

```
LPRINT CHR$(27);")";"n11n12n13,n21n22n23,...nn1nn2nn3."
```

По коду селективного стирания табуляции стираются позиции табуляции, определённой трёхзначными номерами.
074-184.bas

```
1 POKE &HF418,1
10 LPRINT CHR$(27);"(";"008,016,024."
20 LPRINT "ONE";CHR$(9);"TWO";CHR$(9);"THREE";CHR$(9);"FOUR"
30 LPRINT CHR$(27);")";"016."
40 LPRINT "ONE";CHR$(9);"TWO";CHR$(9);"FOUR"
run
ONE    TWO     THREE   FOUR
ONE    TWO             FOUR
Ok
```

В настоящей программе метки стираются в колонке 16.

Если командой

```
LPRINT CHR$(27);"2"
```

Вы уберёте все метки, то при использовании меток из набора принтера команда `LPRINT CHR$(9)` будет игнорироваться.

3. *Прочие команды для управления горизонтальной позицией печатающей головки*

Перемещение на n пробелов	`CHR$(27);"b";CHR$(n)`
Перемещение на n точек	`CHR$(27);"F";"n1n2n3n4"`
Смещение на один символ влево	`CHR$(8)`

Команда

```
LPRINT CHR$(27);"b";CHR$(n)
```
,

где n — целое число из отрезка [1,255], позволяет пропустить n печатных позиций. Эта команда не изменяет текущего значения позиции горизонтальной табуляции.

Команда

```
LPRINT CHR$(27);"F";"n1n2n3n4"
```
,

где n1, n2, n3, n4 — цифры от 0 до 9, перемещает печатающую головку на n точечных позиций *вправо*. При этом величина n должна принадлежать отрезку [0,479] и вычисляется по следующей формуле:

```
n = 1000×n1+100×n2+10×n3+n4
```

Команда

```
LPRINT CHR$(8)
```

смещает печатающую головку на одну колонку *влево*. Если печатная головка находится в исходной позиции, то команда игнорируется. Эта команда используется для печати дополнительных знаков.

VII.4.4. Команды управления графикой

Принтер можно использовать в качестве *графопостроителя*, т.е. устройства, которое позволяет вывести на печать рисунки, графики и т.д. Дело в том, что принтер печатает восемью штырьками печатающей головки. Каждый штырёк печатает *одну* точку. Существует *режим растровой графики*, позволяющий непосредственно управлять ударом *каждого* штырька.

Взгляните на следующую таблицу:

Функция	Управляющий код
Печать растровой графики	`CHR$(27);"S";"n1n2n3n4"`
Повторение растровой графики	`CHR$(27);"V";"n1n2n3n4";CHR$(m)`

Режим *растровой* графики устанавливается командой

```
LPRINT CHR$(27);"S";"n1n2n3n4";
```
,

где n1, n2, n3, n4 — цифры от 0 до 9.

Если обозначить n — количество положений печатающей головки в строке при растровой печати, то

```
n=1000×n1+100×n2+10×n3+n4
```
.

На *каждое* такое положение нужно подать команду

```
LPRINT CHR$(Двоичный код положения штырька);
```
,

с указанием того, какие штырьки печатающей головки должны печатать.

— Боюсь, что от объяснений все проигрывает,— сказал он.
— Одни только результаты впечатляют гораздо больше.

—А.Конан Дойль. Приключения клерка

Пример 19.
074-19.bas

```
Код штырька
&b00000001      ⟶  ·
&b00000010      ⟶  ·           ·
&b00000100      ⟶  ·           ·           ·
&b00001000      ⟶  ·           ·           ·           ·
&b00010000      ⟶  ·           ·           ·           ·
&b00100000      ⟶  ·           ·           ·
&b01000000      ⟶  ·           ·
&b10000000      ⟶  ·
                    ▲           ▲           ▲           ▲
                    │           │           │           │
                 &b11111111  &b01111110  &b00111100  &b00011000

10 LPRINT CHR$(27);"S";"0008";
20 FOR T=1 TO 2:LPRINT CHR$(&B11111111);:NEXT
30 FOR T=1 TO 2:LPRINT CHR$(&B01111110);:NEXT
40 FOR T=1 TO 2:LPRINT CHR$(&B00111100);:NEXT
50 FOR T=1 TO 2:LPRINT CHR$(&B00011000);:NEXT
run
..
....
......
........
........
......
....
..
Ok
```

Пример 20. Программа печатает прямоугольник размерами 200 точек × 16 точек.
074-20.bas

```
10 DATA &B00000001,&B10000000
30 LPRINT CHR$(27);"T16";  'Команда устранения пробелов между строками
40 FOR I=1 TO 2
50    LPRINT CHR$(27);"S0200";:LPRINT CHR$(255);:READ A
80    LPRINT STRING$(198,CHR$(A));:LPRINT CHR$(255)
100 NEXT I
```

Сделаешь раз по сто — вот и будет просто.

—П.Федотов

Команда

```
LPRINT CHR$(27);"V";"n1n2n3n4";CHR$(m)
```

,

где

- n1, n2, n3, n4 — цифры от 0 до 9;
- m — целое число, принадлежащее отрезку [0,255];

позволяет повторить N раз изображение вертикальной линии (8 точек), шаблон которой устанавливается двоичным представлением числа m.

Величина N определяется по формуле:

```
N=1000×n1+100×n2+10×n3+n4
```
.

Отметим, что для выполнения этой команды *не* требуется предварительной установки растровой печати.

Пример 21. Результат действия следующих двух фрагментов одинаков!

- 1)
  074-211.bas

```
10 LPRINT CHR$(27);"S";"0008";
20 FOR T=1 TO 8:LPRINT CHR$(&B11111111);:NEXT
```

- 2)
  074-212.bas

```
10 LPRINT CHR$(27);"V";"0008";CHR$(&B11111111)
```

VII.4.5. Использование макрокоманд

Функция	Управляющий код
Определение макрокоманды (макроопределение)	`CHR$(27);"+";… ;CHR$(0)` ▲ | Совокупность команд
Выполнение макрокоманды	`CHR$(27);"%"`

Команда

```
LPRINT CHR$(27);"+";...;CHR$(0)
```

отменяет существующую макрокоманду и заменяет её определяемой макрокомандой. Максимальное число символов в макрокоманде (не считая символов "+" и `CHR$(0)`) равно 16, в противном случае печатающее устройство будет реагировать только на первые 16 символов макроопределения.

Команда

```
LPRINT CHR$(27);"%"
```

выполняет макрокоманду, которая предварительно была определена при помощи макроопределения.

После описания макроопределения Вы можете использовать макрокоманду в любой требуемый момент времени и печатающее устройство будет «делать» все, что Вы включили в макроопределение.

Пример 22. Макроопределение для установки принтера в исходное положение всех стилей печати.
074-22.bas

```
10  LPRINT CHR$(27);"+";       ' Начало макроопределения
20  LPRINT CHR$(27);"P";       '    Шаг "цицеро"
30  LPRINT CHR$(15);           '    Отмена расширенной печати
40  LPRINT CHR$(27);"H";       '    Отмена двойного удара
```

```
50  LPRINT CHR$(27);CHR$(34); '    Отмена выделенной печати
70  LPRINT CHR$(27);"s";"0";  '    Отмена индексов
80  LPRINT CHR$(27);"5";      '    Отмена шрифта "италик"
90  LPRINT CHR$(27);"Y";      '    Отмена подчеркивания
100 LPRINT CHR$(27);"A"       '    Высота строки =24/144 дюйма
110 LPRINT CHR$(0)            ' Конец макроопределения
```

Печатающее устройство будет «помнить» об этом макроопределении до тех пор, пока не будет введено новое макроопределение!

Теперь посмотрите, как можно использовать описанное макроопределение.

Пример 23. Программа печатает всего одну строку, используя несколько различных стилей печати. Затем программная строка 50 позволяет вызвать макрокоманду, определённую в примере 21.
074-23.bas

```
10 LPRINT CHR$(27);"4";              ' "Италик"
20 LPRINT CHR$(27);"G";              ' Двойной удар
30 LPRINT CHR$(14);                  ' Расширенная печать
40 LPRINT "TESTING ONE, TWO, THREE"
50 LPRINT CHR$(27);"%";              ' Вызов макрокоманды
60 LPRINT "TESTING FOUR, FIVE, SIX"
```

VII.4.6. Другие полезные команды

> Все правила достойного поведения давным-давно известны, остановка за малым — за умением ими пользоваться.
>
> *—Б.Паскаль. Мысли*

Предлагаем Вашему вниманию следующую таблицу:

Функция	Управляющий код
Повторение символа	`CHR$(27);"R";"n1n2n3";CHR$(m)`
Отмена последней строки («стирание» последней строки в буфере)	`CHR$(24)`
Установка печатающего устройства в автономный режим («OFF LINE»)	`CHR$(19)`
Установка печатающего устройства в неавтономный режим («ON LINE»)	`CHR$(17)`
Звучание звонка печатающего устройства	`CHR$(7)''`
Выключение из работы детектора отсутствия бумаги	`CHR$(27);"p";"0"`
Включение в работу детектора отсутствия бумаги	`CHR$(27);"p";"1"`
Инициализация печатающего устройства	`CHR$(27);"c";"1"`

Выполнение команды повторения символа

```
LPRINT CHR$(27);"R";"n1n2n3";CHR$(m)
```
,

где n1, n2, n3 — цифры от 0 до 9, приводит к повторению n раз (n=100×n1+10×n2+n3) символа с кодом m.

Пример 24.

- 1)
  074-241.bas

```
1 LPRINT CHR$(27);"R120";"+"
```

- 2)
  074-242.bas

```
1 LPRINT CHR$(27);"R120";"+";CHR$(24)
```

Команда

```
LPRINT CHR$(19)
```

приводит к установке печатающего устройства в автономный режим. При этом игнорируются все последующие символы и функциональные коды, кроме кода `CHR$(17)`, который служит для возвращения печатающего устройства в режим «ON LINE».

Действие команды

```
LPRINT CHR$(17)
```

отличается от нажатия на кнопку «ON LINE», расположенную на принтере. Когда горит лампочка «ON LINE», то печатающее устройство «игнорирует» код `CHR$(17)`.

Команда

```
LPRINT CHR$(7)
```

«заставляет» печатающее устройство издавать звук в течение приблизительно 1/4 секунды.

Командой

```
LPRINT CHR$(27);"p";"0"
```

отменяется сигнал, подаваемый детектором отсутствия бумаги. Сигнал отсутствия бумаги обычно заставляет звучать звонок печатающего устройства и останавливает печать до тех пор, пока не будет вставлена бумага.

Команда

```
LPRINT CHR$(27);"p";"1"
```

восстанавливает функции детектора отсутствия бумаги.

Команда *инициализации* печатающего устройства

```
LPRINT CHR$(27);"c";"1"
```

повторно инициализирует печатающее устройство. Буфер печати при этом очищается.

Диск с примерами

Загрузить образ диска

Открыть диск в WebMSX

[1)]

Примечание редактора: добавлено 2019-10-20

[2)]

Примечание редактора: добавлено 2019-11-01

[3)] , [4)] , [6)]

Примечание редактора

5)

Примечание редактора: добавлено 2019-10-18

https://sysadminmosaic.ru/msx/basic_dialogue_programming_language/007

2023-02-19 16:20