SINCLAIR RECEARCH LTD.

TANNAKY

1

ZX SPECTRUM

АННОТАЦИЯ

Микрокомпьютер ZX Spectrum фирмы Sinclair Research благодаря низкой стоимости, относительно развитой графике, простоте обслуживания и программирования, ясности и доступности подробностей архитектуры и системного матобеспечения занял на черном советском рынке доминирующую позицию. Его наследуют стыкующиеся с ним программно компьютеры Timex 2048, Timex 2068 (Unipolbrit 2068), ZX Spectrum 2+ (новейшая модель Amstrad - Sinclair) и Elwro 800 Junior, способные в одном из своих режимов работать, как ZX Spectrum.

Эта брошюра задумана как дополнение к фирменной инструкции по эксплуатации. Для пользователей, не имеющих доступа к оригиналу документации или не способных ею воспользоваться из-за языковых трудностей, приводим вкратце наиболее важную информацию, изложенную в инструкции. С сожалением пришлось отложить до следующего раза описание дополнительных возможностей ZX Spectrum, оборудованного ZX Interface-1 (обслуживание ZX Microdrive, RS 232 для локальной сети).

Лаконичность примеров заставит многих читателей экспериментально проверять возникающие сомнения. Это наилучшая дорога для изучения собственного компьютера (его нельзя испортить, нажимая на клавиши).

Эта брошюра не предназначена для системного изучения. Подбор и состав материала должны обеспечить поиск необходимой для работы с компьютером информации и разрешать сомнения, как начинающих, так и опытных пользователей ZX Spectrum, позволит им лучше понять его работу.

СОДЕРЖАНИЕ

1. КЛАВИАТУРА

Клавиатура ZX Spectrum состоит из 40, а ZX Spectrum+ - из 58 клавиш. Ключевые слова бейсика вводятся одним нажатием клавиши. Вводимые знаки и слова высвечиваются в нижней части экрана в месте, обозначенном мигающим курсором (прямоугольником). Курсор не виден сразу после включения компьютера, либо после ввода сообщения. Он появляется только после нажатия клавиши. Буква внутри курсора указывает, что ожидает от вас компьютер, или в каком режиме он находится:

K - ключевые слова (Keyboards);
L - прописные буквы (Letters);
C - заглавные буквы и некоторые символы (Capitals);
E - ключевые слова и знаки, размещенные над и под клавишей (Extended mode);
G - графические символы (Graphics).

Значение клавиши зависит, кроме заданного режима, также от нажатия (или нет) одновременно с ней (чуть раньше) одной из двух специальных клавиш:

Caps Shift - заглавная буква (CS);
Symbol Shift - символ (SS).

Пользуясь ними, необходимо вначале нажать <Caps Shift> или <Symbol Shift>, а затем, не отпуская, функциональную клавишу, например, цифру 2 (подобные действия будут обозначаться символом "/", например SS/2).

Одна из клавиш в ZX Spectrum может иметь до 8 различных значений. Вот что будет высвечено на экране при нажатии, например, клавиши "R":

режим **E** : INT
режим **L** : r
режим **C** : R
режим **K** : RUN
режим **L,C,K** + SS/ :
режим **E** + SS/ : VERIFY
режим **G**: знак в форме буквы R, либо другой, определенный пользователем (для клавиш A - U); клавиши V, W, X, Y, Z в этом режиме не используются. Режим **C** для этой группы клавиш равнозначен режиму **L** с постоянным нажатием CS и обозначает высвечивание заглавных литер.

Над некоторыми клавишами верхнего ряда, например, над клавишей с цифрой 3, размещено по две надписи:

режим **E** + CS/ : MAGENTA (цвет знака)
режим **E** : MAGENTA (цвет фона)

режим **K, L, C** + CS/ : TRUE VIDEO
режим **E** + SS/ : LINE и т.д.

В режиме G после нажатия этой клавиши на экране появится графический символ, а в случае одновременного нажатия CS или SS - тот же символ с измененным цветом. Надпись над символом обозначает контрольный символ. Они не высвечиваются как знаки на экране, а только вызывают определенные эффекты. Например, в рассмотренном режиме такая клавиша определяет цвет фона и самого знака:

	Режим **E** + CS/ (цвет символа)		Режим **E** (цвет фона)
CS/0	Черный	(BLACK)	0
CS/1	Темно-голубой	(BLUE)	1
CS/2	Красный	(RED)	2
CS/3	Фиолетовый	(MAGENTA)	3
CS/4	Зеленый	(GREEN)	4
CS/5	Светло-голубой	(CYAN)	5
CS/6	Желтый	(YELLOW)	6
CS/7	Белый	(WHITE)	7
CS/8	Одинаково с FLASH1 - одинаково с BRIGHT1		8
CS/9	Одинаково с FLASH1 - одинаково с BRIGHT1		9

Обратим внимание, что клавиши <8> и <9> тоже имеют в этом режиме значения, несмотря на то, что на клавиатуре значений нет. При курсоре **K, L** или **C** только комбинация CS/3 и CS/4 вызывает световые эффекты:

CS/3 - смена цветов фона и символов (Inverse Video);
CS/4 - возврат их на первоначальный цвет (True Video).

В Spectrum+ этим комбинациям соответствуют специальные клавиши.

Обратим внимание, что использование вышеописанных комбинаций клавиш - единственный способ замены цветов текста программы. Остальные комбинации будут рассмотрены далее.

Теперь займемся способами изменения состояния курсора. На экране слово "System" сигнализирует о том, что решение об изменении режима принимает сам компьютер. Видно, что прямого перехода от режима **G** к **E** нет. Пользователь также не может непосредственно влиять на переход компьютера из режима **K** в режимы **L** или **C** и обратно. Эта система является сложной лишь на первый взгляд. Уже после нескольких проб все оказывается просто, и мы даже начинаем не замечать определенные достоинства такой организации клавиатуры.

Клавиша <Break Space> во время ввода данных служит для внесения пропусков (пробелов) между знаками.

В модели ZX Spectrum+ клавиатура расширена и более удобна в использовании. Смена режимов достигается однократным нажатием. Все контрольные символы, получаемые в классической модели с помощью CS/цифра, в нем представлены отдельными клавишами, выделены также наиболее употребительные знаки препинания, информация, записанная ранее над клавишей и под ней, здесь находится в верхней части клавиши. Остальные принципы использования смены режимов идентичны.

2. РЕДАКТОР

Редактор (Editor) - это заключенная в ПЗУ программа, способствующая взаимодействию с компьютером: вводу данных, их модификации и корректировке. Все, что мы набираем на клавиатуре, обычно появляется в нижней части экрана Текст в любой области можно модифицировать, или редактировать. Клавиши CS/5 и CS/8 (стрелки влево и вправо в ZX Spectrum+) позволяют передвигать курсор вдоль редактируемой строки без стирания информации. Для стирания символов используется клавиша CS/0 (в Spectrum+ <Delete>). Стирается знак, расположенный непосредственно перед курсором, при этом ключевые слова бейсика трактуются как отдельные знаки.

Во время набора программы или отдельных директив редактор во многих случаях (после ввода ключевого слова, двоеточия, THEN и т.п.) сам изменяет режим ввода (между **K**, **L**, или **C**), оберегая пользователя от случайных ошибок. Но если кому-то надо такую ошибку сделать специально, то можно его "обмануть", вставляя, к примеру, ":" для получения режима **K**, а затем удаляя его. Такой обман, конечно, не избавляет от последующих неприятностей с вводом строки в программу.

Конец редактирования, ввода строки команд или очередной программной строки сигнализируется клавишей <Enter>. Компьютер молниеносно проверяет синтаксис. В случае необходимости в непонятном ему месте появляется мигающий знак "?". Он может появляться значительно дальше, чем ошибка. Например, при отсутствии одной из замыкающих скобок в сложном выражении Spectrum не в состоянии определить место, куда надо было ее вставить и подаст сигнал об ошибке только в конце выражения.

Но восприятие строки компьютером не гарантирует ни правильности ее логики, ни согласия с желаниями автора. Нет возможности принудить компьютер к запоминанию ошибочной (в смысле синтаксиса бейсика, а не логики программы) последовательности команд. Между прочим, это означает то, что без присоединения к Spectrum "Interface-1" невозможно даже подготовить управляющую программу для обмена с памятью типа "Microdrive", так как нельзя ввести потребных для этого специальных команд.

Если в корректируемом месте размещены контрольные символы, перемещаемый курсор ведет себя иногда странно и необычно (исчезает, отступает). Не стоит из-за этого расстраиваться.

В процессе удаления таких знаков могут появиться вопросительные знаки, которые надо убирать отдельно. Контрольные символы обычно занимают в памяти 2 байта, и Spectrum, вынужденный высвечивать знак с кодом менее 32, выводит в его месте "?", если не сможет интерпретировать его иначе.

Чтобы исправить уже введенную в память строку программы, необходимо ее вначале скопировать в нижнюю часть экрана. С этой целью указатель текущей строки перемещаем вверх или вниз клавишами CS/6 и CS/7 (в Spectrum+ стрелки). При более длинных программах быстрее пользоваться инструкцией LIST N. можно вводить число чуть меньшее, чем номер нужной строки, если мы уверены, что в программе нет строки с таким номером, чтобы случайно не затереть нужную строку. После нажатия <Enter> указатель будет установлен на первой строке с номером, больше заданного, либо последней, если заданной нет в программе. Указатель ">" после такого предприятия может быть временно невидимым, но после нажатия CS/1 (<Edit> в Spectrum+) желаемая строка будет скопирована в рабочее поле редактора.

Нажатие одной <Enter> при пустой нижней части экрана приводит к высвечиванию фрагмента программы, включающего текущую строку (указываемую ">"). Однократно можно ввести оператор любой длины. Поначалу нижняя часть экрана будет автоматически расширяться, но постепенно компьютер будет реагировать все медленней, сигнализируя растущее неодобрение действиями человека. После запоминания 22 строк протест компьютера станет значительно резче, он перестанет высвечивать последующие символы, вводимые с клавиатуры, и на каждое нажатие клавиши отреагирует неприятным ворчанием. Но, несмотря на это он и далее примет и запомнит все то, что мы наберем. Его сопротивление можно смягчить, модифицируя соответствующие системные переменные. Клавиша CS/1 (Edit) позволяет также одноразовое удаление всей нижней части экрана.

3. БЕЙСИК

В этом разделе кратко представлен алфавитный перечень функций и команд языка ZX-бейсик. Его стоит просмотреть даже тем, кто хорошо знает этот диалект, так как мы приводим информацию, пропущенную в оригинале инструкции по эксплуатации.

В перечне используются следующие обозначения:

@ - единичная литера;
V - переменная любого типа;
X, Y, Z - численные выражения с действительными значениями;
K, M, N – численные выражения со значениями, автоматически округляемыми до ближайших целых чисел;
E - численное выражение или строка;
F - строка;

S - последовательность команд, отделенных друг от друга двоеточием;

C - последовательность инструкций, определяющих цвета и способ высвечивания (INK, PAPER, BRIGHT, FLASH, OVER, INVERSE), отделяемых запятыми и точками с запятой.

Символы, помещенные в [ ] можно не употреблять. Например, LOAD F CODE [M[,N]] означает, что верными являются формы:

```
LOAD F CODE
LOAD F CODE M
LOAD F CODE M,N
```

В этом случае Spectrum обычно дополняет пропущенные параметры значениями, вытекающими из контекста (в нашем случае считанными с магнитной ленты).

Запись @[Z] означает или @Z, или @.

Spectrum почти во всех случаях допускает использование в качестве параметров выражения, требующие предварительных расчетов. Ограничения затрагивают лишь допустимые значения расчитываемых результатов и их типы. Выражения типа K, M, N могут давать в результате любые действительные числа меньше 65535, которые будут округлены ближайшими целыми числами. Появление большего числа сигнализируется как ошибка сообщением 8 (Integer out of range).

3.1. Структура программы

Программа на бейсике - это последовательность инструкций, разделенных на строки. Каждая строка начинается со своего номера (натурального числа в диапазоне 1-9999). Пронумерованные операторы могут вводиться в любой последовательности. Они будут автоматически упорядочены по возрастанию номеров. Необязательно нумеровать подряд (1,2,3...). Такой стиль не рекомендуется, удобнее пользоваться номерами черев 10 (10,20,30...), что облегчит вставку новых строк между старыми.

Ввод новой строки с уже существующим номером вызывает замену старой версии этой строки. Допускается размещение в тексте программы строк, содержащих лишь номер строки с целью оптического выделения фрагментов программы.

Единичный оператор программы может быть значительно длиннее экранной строки (32 знака), но не может содержать более 127 инструкций. Команды в операторе должны быть отделены двоеточием, между инструкциями в строке можно размещать любое число пробелов с целью улучшения читаемости программы.

Нажатие <Enter> завершает ввод строки. Компьютер проверяет ее синтаксис и, если она начинается с номера, включает ее в существующую программу, вставляя на новое место. Если номер опущен, компью-

тер приступает к немедленному выполнению введенной последовательности команд.

3.2. Типы данных

ZX-бейсик позволяет непосредственно оперировать числами и текстами. Числа могут записываться в одной из трех форм:

- целочисленной: 0; -17; 13; +21,
- дробной: 33.1968; 1422.77; -.91,
- показательной: 23.53E-7 = 23.53*10**-7.

Максимальное значение действительного числа не может превышать 1.7E38. Числа хранятся с точностью до 9-10 значащих цифр. Пробелы в середине записи числа не допускаются.

Spectrum для высвечивания числа требует максимально 14 знаков. Значения, не требующие более 8 цифр, выводятся точно, остальные - в показательной форме.

Другим типом данных бейсика являются текстовые константы Это любая последовательность знаков (символов с кодами от 0 до 256), размещенная в кавычках. В текстовую константу можно включать также контрольные (управляющие) символы, коды ключевых слов и даже коды символов, не задействованных в Spectrum. Длина текстовой константы - это число содержащихся в ней символов, а не число знаков, необходимых для ее высвечивания на экране. Это число ограничено лишь размером допустимой памяти. Возможны тексты длиной 0 (пустые).

Во время конструирования текста большинство символов можно вводить с клавиатуры, но некоторые символы могут потребовать дополнительных усилий. Чтобы включить в текст кавычку, нужно ввести ее с клавиатуры дважды. Универсальным способом вставки в текстовую строку символов, не задействованных в ZX, является использование выражений и функций CHR$ Например, "это" + CHR$13 + "текст" даст в результате:

```
это
текст
```

ZX-бейсик не знает логических переменных. Эту роль исполняют числа. Значения, отличающиеся от 0, интерпретируются как истина, равные 0 - ложь. Результатом расчета логических выражений являются числа 0 и 1.

Кроме констант бейсик допускает использование числовых и текстовых переменных. Они служат для сохранения в памяти компьютера констант определенного типа. Переменные модифицируются именами, которые присваивает им программист.

Переменные, хранящие число или цепочку знаков, называются простыми. Кроме них мы можем применять так называемые табличные переменные (массивы), но они должны быть объявлены инструкцией DIM. Допускаются как числовые массивы, так и знаковые. Элементом числовой таблицы является число, а сим вольной - единичный символ с ко-

дом от 0 до 255. Операторы знаковых массивов могут использоваться как алфавитно-цифровые переменные постоянной длины, определенной последним индексом (размерностью) декларации данного массива. Число размерностей ограничено 255. Индексы не могут превышать значение 65535. На практике эти значения должны быть значительно меньше из-за допустимых значений памяти. В объявлении массива дается только верхняя граница индекса. За нижнюю границу индекса всегда принимается значение 1.

3.3. Имена

Одно из неудобств бейсика - ограничения в определении имен используемых переменных. Только простые переменные могут быть любой последовательности букв, цифр и пробелов. Первым символом имени должна быть литера. Все остальные переменные должны иметь однобуквенные имена, в том числе и управляющие циклами (FOR...NEXT).

Текстовые переменные отличаются от числовых знаком $, размещенным после литеры. Это также касается названий функций, определяемых пользователем с помощью директивы DEF FN и ее аргументов.

Для задания имен в программе можно использовать как малые, так и большие литеры. Spectrum автоматически заменяет большие литеры на малые; также игнорируются все пробелы.

Ограничение использования имен частично компенсируется возможностью использования одной и той же литеры как имени переменных различных типов. Единственное требование при этом - текстовая переменная не должна помечаться тем же символом, что и знаковый массив. Литера A в программе может обозначать:

```
A                  - числовая или управляющая переменная;
A$ или A$(i,j)     - текстовая переменная или знаковый массив;
A(i)               - числовой массив;
FN A...            - функция с числовым значением;
FN A$...           - функция с текстовым значением;
DEF FN A$(A,A$...) - параметры в определении функций.
```

3.4. Выражения

Выражения в бейсике строятся из постоянных, переменных функций, а также скобок. Последовательность выполнения действий в выражении определяют круглые скобки и приоритеты отдельных операций. Ниже рассмотрены все допустимые в бейсике операции в последовательности их приоритетов.

3.4.1. Выделение фрагмента текста

Операция может быть выполнена над цепочками, текстовыми переменными, а также знаковыми массивами. Для цепочек и переменных параметры, определяющие извлекаемый фрагмент, должны записываться в

в круглых скобках. Для массивов эти данные размещают в поле последнего индекса или, при его отсутствии, в отдельных скобках. Аналогично простым переменным эти параметры могут быть представлены:

```
         - отсутствовать, изменению не подвергать;
(K)      - арифметическое выражение, результат - единичный символ,
           находившийся на K-ой позиции текста;
(K to M) - арифметическое выражение с ключевым словом "to". Результат - фрагмент текста от K-го до M-го символов включительно. Если K > M > 0, результат - пустая цепочка. Если текст короче M, сигнализируется ошибка.
```

Например при выполнении программы

```
10 LET A$="текст"
20 DIM B$(2,3)
30 LET B$(1)="ABS"
40 LET B$(2)="XYZ"
```

получаем:

```
A$=A$()=A$(TO)=A$(1 TO 5)="текст"
A$(1)=A$(TO 1)=A$(1 TO 1)="т"
A$(3 TO 3)=A$(3)=A$(5-2)="к"
A$(TO 2)=A$(1 TO 2)="те"
A$(3 TO)=A$(3 TO 5)="кст"
A$(2 TO 4)="екс"
B$(1)=B$(1,)=B$(1)()=B$(1,TO)= B$(1,1 TO 3)=B$(1)(1 TO 3)="ABS"
B$(2,2)=B$(2)(2)=B$(2)(2 TO 2)=B$(2)(2 TO 2)="Y"
```

В других диалектах бейсика подобную роль выполняют функции:

```
LEFT (A$,N) = A$(TO N)
RIGHT (A$,N) = A$(N TO)
MID (A$,K,M) = A$(K TO M)
TL (A$) = A$(2 TO).
```

3.4.2. Возведение в степень

Аргументами операции возведения в степень являются вещественные числа. Основание всегда должно быть положительным числом. Следует помнить, что A^B расчитывается как EXP(B*LN(A)), поэтому A^2 будет вычисляться дольше, чем A*A.

3.4.3. Минус, как знак числа (-)

Аргументом этой операции всегда является число, результатом - то же число с противоположным знаком. Минус как знак следует отличать от обозначенной тем же символом операции вычитания. Высокий

приоритет этой операции позволяет написать A*-B, что интерпретируется как A*(-B). В иных диалектах Бейсика употребление двух операций подряд не допускается.

3.4.4. Умножение и деление (;/)*

Умножение и деление арифметически выполняются в той же последовательности, в какой встречаются в выражении. Знаки умножения нельзя опускать. Умножение выполняется компьютером быстрее, чем деление, следовательно лучше писать A*0.5 вместо A/2.

3.4.5. Сложение и вычитание (+;-)

Обе операции имеют одинаковый приоритет и выполняются в очередности появления.

Сложение в применении к текстовым константам и переменным дает текст, который складывается из символов первого текста и добавленных к нему символов второго. В отличие от сложения чисел свойство перестановки здесь не выполняется.

3.4.6. Знаки отношения (=;<;>;<=;>=;< >)

Эти двухаргументные операции имеют одинаковый приоритет и означают отношения: "равно", "меньше", "больше", "меньше или равно", "больше или равно", "не равно". Результатом является чило 1, если отношение соблюдается, или 0 - если нет. Результаты проверки отношения могут быть использованы в арифметических выражениях как обычные числа.

Как пример разберем следующую программу. Имеем числовой массив B, объявленный инструкцией DIM B(100). Необходимо определить, сколько элементов этого массива больше 7. Проще всего добиться этого командами:

```
LET X=0: FOR I=1 TO 100: LET X=X+(B(I)>7): NEXT I
```

После выполнения этих команд переменная X будет содержать искомое число.

Эти же операторы применяются к текстовым переменным и константам. Результат операции сравнения последовательности символов определяется лексикографическим порядком - наименьший код у пустого текста. Сравнивая поочередно символ за символом, ищется первое место, в котором обе последовательности отличаются. За меньший принимается тот текст, в котором первый отличающийся символ имеет меньший код. Например:

```
" " < "текст"; "text2" > "text"
```

3.4.7. Логическое отрицание (NOT)

Это операция логического отрицания значения числового выражения, являющегося ее аргументом, которая рассматривается как логическая величина.

```
NOT X=0, если X<>0          NOT X=1, если X=0.
```

3.4.8. Логическое "и" (AND)

Это весьма полезная двухаргументная операция, применение которой выходит далеко за расчет логических выражений. Первым ее аргументом может быть как числовое, так и алфавитно-числовое выражение, вторым - только числовое.

```
X AND Y=X, если Y<>0,  X AND Y=0, если Y=0,
F AND Y=F, если Y<>0,  F AND Y=0, если Y=0.
```

Это определение может показаться странным и не связанным с тем, что мы привыкли считать логическим "и". Обратим внимание, что если оператор AND будет находиться между двумя логическими выражениями, то его результатом будет 1 тогда, и только тогда, когда оба эти выражения будут истинны. Достоинства такого определения AND проявляются ниже в примерах. Допустим, переменной Max мы хотим присвоить значение максимального из чисел A и B. Для этого достаточно одной инструкции:

```
LET Max=(A AND A>B)+(B AND A<=B)
```

Скобки в этом выражении обязательны ввиду высокого приоритета операции "+".

Еще более наглядные результаты можно получить, работая с текстами. Последующая программа распознает пол пользователя по его имени:

```
10 INPUT "Сообщи свое имя"; A$
20 PRINT "Привет пан"+("и" AND A$(LEN A$)="A")
```

Не располагая операцией AND, необходимо было бы многократно использовать инструкции IF...THEN, а также GO TO...

3.4.9. Логическое сложение (OR)

Это двухаргументная операция. Оба аргумента должны быть числовыми выражениями. Действие этой операции следующее:

```
X OR Y=1, если Y<>0,  X OR Y=X, если Y=0.
```

Применение OR вне логических выражений значительно реже, чем AND. Например, инструкцию, расчитывающую максимум из 2 чисел можно записать в следующей форме:

```
LET Max=A*(0 OR A>B)+B*(0 OR A<=B).
```

3.5. Функции

ZX-Бейсик содержит набор стандартных функций для математических расчетов, операций над знаковыми последовательностями, а также для

опроса экрана и клавиатуры. За исключением функций ATTR, POINT и SCREEN$ аргументы не требуется заключать в скобки.

ABS X - абсолютная величина числа,

ACS X - арккосинус. Если X не относится к диапазону (-1,1), сигнализируется ошибка A (ошибочный аргумент). Значение функции выражается в радианах.

ASN X - арксинус.

ATN X - арктангенс.

ATTR(K,M) - задание атрибутов в K-ом операторе и M-ой колонке. Способ кодирования атрибутов описан в разделе 5. Если параметры не находятся в диапазонах 0<=K<=23 и 0<=M<=31, сигнализируется ошибка B (превышение допустимого значения).

BIN - преобразование двоичного числа без знака (максимально 16 цифр) в десятичное (на экране) либо в форму, принятую в Spectrum (в памяти). ZX-бейсик не имеет функции, осуществляющей обратное преобразование десятичного числа в двоичное.

CHR$ K - замена числа на соответствующие символы ASCII или управляющий символ. Если K не относится к диапазону (0,255), сигнализируется ошибка B.

CODE F - замена символов ASCII на число. Аргументом является любое алфавитно-цифровое выражение, а значением функции является код левого символа этого выражения или 0, если аргумент - пустая цепочка.

COS X - косинус. Аргумент в радианах.

EXP X - показательная функция E**X.

IN K - опрос состояния буфера ввода-вывода процессора. Соответствует двум инструкциям процессора Z80: LD BC,K и IN A,(C). Значением функции IN является целое число в диапазоне (0,255). Это единственная инструкция, применение которой может привести к тому, что ваша программа не будет выполняться на других экземплярах Spectrum. Применение IN к чтению с клавиатуры: каждый ряд клавиш разделен на 2 группы по 5, и каждая пятерка относится к определенному буферу:

Клавиша (десятичный)		Номер буфера (шеснадцатиричный)
CS...U	65276	#FEFE H
A...G	65022	#FDFE H
Q...T	64510	#FBFE H
1...5	63486	#F7FE H
0...6	61438	#EFFE H
P...Y	57342	#DFFE H
Enter...H	49150	#BFFE H
Space...B	32766	#7FFE H

Пять младших битов в этом буфере сигнализируют, какая клавиша нажата (0 - нажата, 1 - отжата). Трудности возникают с тремя стар-

шими битами. В версии 1 (самой старой) Spectrum они всегда равны 1, следовательно IN 64510=255 означает освобождение клавиш между Q и T. В версии 2 шестой бит равен 0 и в описанной ситуации IN 64510=191. В версии 3 состояние шестого бита и необходимы дополнительные операции, например:

```
LET P=IN K: LET P=P-32*INT(P/32)
```

Чтение с клавиатуры с помощью IN делает возможным распознавание нескольких одновременно нажатых клавиш.

INKEY$ - чтение с клавиатуры. Значением функции является знак нажатой клавиши. Клавиатура читается в режимах **L** и **C**. Если нажато несколько клавиш или ни одной, то значением INKEY$ будет пустая цепочка. Эта функция не ждет нажатия клавиши.

INT X - целая часть числа. Действительное число округляется до ближайшего целого, не превышающего X. Например: INT 3.1=INT 3.9=3, INT -5.1=-6.

LEN X - длинная цепочка знаков.

LN X - натуральный логарифм. Если X <= 0, сигнализируется ошибка A.

PEEK K - содержимое ячейки памяти с адресом K

PI - значение постоянной 3.14159265

POINT (K,M) - проверяет состояние точки экрана. Эта функция имеет значение 1, если точка на экране с координатами K,M имеет цвет чернил (выведена) и 0 - если имеет цвет фона. Функция обнаруживает выведенные точки даже при одинаковом цвете фона и чернил, когда они не видимы. Координаты должны находиться в диапазоне 0<=K<=255 и 0<=M<=175.

RND - генератор псевдослучайных чисел. Очередное значение равняется SEED/65536, где SEED - двухбайтовая системная переменная с адресом 23561=#5C76. Каждый раз с вызовом RND она модифицируется в соответствии с формулой: SEED=((SEED+1)*75M0D65537)-1. Таким образом генерируется последовательность в пределах 65536. Переменной SEED можно задавать первоначальное значение инструкцией RANDOMIZE K

SCREEN$(K,M) - распознает знак на экране. Эта функция на основании формы определяет символ, выведенный на экран в пересечении K-ой и M-ой колонки. Распознаются только символы с кодами от 32 до 127. В случае нераспознания знака значением является пустой текст. Координаты должны лежать в диапазонах 0<=K<=23 и 0<-M<-31.

SGN X - знак числа. Значением функции являются -1,0,1 в зависимости от знака аргумента.

SIN X - синус. Аргумент - в радианах.

SQR X - квадратный корень. При X<0 сигнализируется ошибка A.

STR$ X - замена числа на цепочку знаков. Эта функция заменяет

двоичное представление числа на символьное, которое может быть представлено на экране.

TAN X - тангенс. Аргумент должен выражаться в радианах.

USR - работа этой функции зависит от аргумента. Ее вызов с числовым аргументом (USR K) приводит в действие машинный код с адреса K. После его выполнения и возврата управление передается бейсику. Значением функции является содержимое пары регистров BC. Вызов функции с текстовым выражением (USR F), где F содержит литеру (малую или большую) от "A" до "U" включительно. Значением функции является адрес в области UDG, с которого хранятся 8 байт, определяющих вид графического символа, определенного пользователем и получаемого в режиме **G** путем нажатия клавиши с соответствующей литерой.

VAL F - расчитывает значение числового выражения, представленного в символьной форме. Это весьма важная функция с разнообразными приложениями. Если выражение F не представляет верного (для бейсика) числового выражения, сигнализируется ошибка C. В ходе расчета могут возникать и др. ошибки, зависящие от вида выражения.

**VAL$** F - определяет значение текстового выражения, представленного в символьной форме. Область ее применения меньше, чем у VAL. Трудно найти ситуации, в которых ее применение необходимо, кроме вывода текстовых переменных с именами, заранее не известными и определяемыми в процессе выполнения программы. Допустим, что значением переменной A$ является литера, определяющая имя алфавитно-цифровой переменной. Не располагая этой функцией трудно было бы распечатать значения этой переменной.

К сожалению не все функции выполняются в полном соответствии с замыслами авторов интерпретатора ZX-бейсик. В разделе 9 мы приводим известные неточности в выполнении функций USR, STR$ и SCREEN$.

3.6. Инструкции ZX-бейсика

ZX-бейсик представляет в наше распоряжение 50 инструкций. Все они могут быть помещены в программе или выполнены сразу, как приказы, отданные с клавиатуры.

BEEP X,Y

Позволяет генерировать звуки. X - время звучания в секундах (0.5<X<10.5). Y - высота тона (-60<=Y< =69.8). Значение Y = 0 соответствует "си" малой октавы. Диапазон изменения Y соответствует примерно трем октавам. Выход за допустимые пределы сигнализируется сообщением B. Во время генерации тона не контролируется клавиша <Break>. В связи с этим прерывание программы возможно только при завершении инструкции.

BORDER К

Задает цвет рамки (бордюра). К - номер выбранного цвета (от 0 до 7). Выход за пределы К - ошибка B. Командой устанавливают также цвет фона в нижней части экрана, предназначенной для системных сообщений и ввода данных.

BRIGHT К

Оттенок фона. К=0 - нормальный, К=1 - яркий, К=8 - ранее определенный оттенок. BRIGHT К может выступать как самостоятельная инструкция или в составе PRINT или INPUT, действуя в этом случае только на время выполнения последних. Проверим это на печати CHR$ 19 + CHR$ К. Эти символы можно ввести как в текст программы, так и в текстовые константы, нажимая в расширенном режиме (курсор **E**) клавишу 8 - как BRIGHT 0 и клавишу 9 - как BRIGHT 1.

CAT...

Эта инструкция может использоваться лишь с подключенным ZX-интерфейсом.

CIRCLE [C,]K,M,N

Вывод окружности радиусом N с центром в точке (К,М) в цвете С. Если С опущен, то принимается INK 8, PAPER 8; BRIGHT 8, FLASH 8. Допустимые значения параметров: 0<=К<=255, 0<=М<=175. Выход за пределы экрана - ошибка B.

CLEAR [К]

Очищает область переменных, освобождая занятую ими память, стирает экран и стек адресов возврата из подпрограмм, выполняет инструкцию RESTORE, обнуляет указатели, определяющие координаты рисующих и пишущих инструкций. Если задан параметр К, его значение будет принято как новый RAMTOP. К должно быть в пределах 23821...65535, иначе появится сообщение M (RAMTOP NO GOOD). Если в памяти компьютера уже есть программа, то К должно соотваетственно возрасти. Слишком малое значение RAMTOP может сделать невозможным выполнение программ и команд с клавиатуры.

CLOSE #К

Отключает К-ый поток от приписанного ему канала. Хотя К может принимать значения от 0 до 15, на практике попытка отключения потоков от 0 до 3 системой игнорируется (см. разделы 7 и 8).

CLS

Очищает экран и обнуляет экранные указатели инструкций печати и графики. После стирания экран приобретает цвет, определенный последними инструкциями PAPER и BRIGHT.

CONTINUE

Эта инструкция позволяет продолжить программу после остановки. Если прерывание поступило в результате ошибки, сигнализируемой от 9 до L, то после CONTINUE будет повторена последняя выполненная инструкция. В остальных случаях будет выполнена следующая инструкция. Это позволяет исправить с клавиатуры найденные ошибки и продолжать работу. Не применяется к инструкциям, задаваемым с клавиатуры. Локализация CONTINUE в программе не сделана специально, так как ее работа зависит от текущего значения системных переменнных OLDPPC и OSPPC, которые, в свою очередь, модифицируются системой, и отслеживание и управление их состоянием может быть затруднено, особенно во время тестирования программы.

COPY

Если к Spectrum подключен принтер, то по этой команде содержимое 22 верхних строк экрана будет скопировано на бумагу (копируется не только текст, но и рисунки). При отсутствии принтера зта инструкция просто игнорируется. Spectrum сам определяет, подключен ли принтер. Во время вывода внутренний таймер останавливается, системные, переменные FRAMES в это время не модифицируются.

DATA E1,E2

Позволяет размещать в тексте программы списки данных. В списке можно размещать любые выражения:

```
    100 DATA SIN X+COS Y+1,"значение"+("полож." AND Z>2)+("отр."
AND Z<0)
    110 READ Z,Z$
```

Переменная Z примет значение SIN X + COS Y + 1, расчитан-ное для текущих значений X и Y, переменная Z$ будет содержать цепочку "значение полож." или "значение отр." в зависимости от расчитанного Z. Бессмысленно задавать команду DATA с клавиатуры, так как Spectrum после проверки синтаксиса тут же забудет о ней. В программе может присутствовать любое число инструкций DATA, и их можно размещать в любом месте.

DEF FN @[$]([@1[$],@2[$]...@I[$]])=E

Определение функции пользователем. Именем функции может быть только одна литера, возможно со знаком $. Его использование определяет тип функции как текстовой. После имени в осязательных скобках можно задавать список аргументов, разделенных запятыми. Допускается определение безаргументных функций. Имена аргументов должны быть однолитерными (возможно со знаком $). Ограничение на число аргументов: 52 (26 числовых и 26 алфавитно-цифровых). После закрытия скобкой списка параметров и знака "=" следует определение функции. Им может быть любое выражение соответствующего типа, состоящее из кон-

стант, переменных, аргументов, стандартных функций и функций, определенных программистом. Следует избегать рекурсивного вызова процедур, как косвенно, так и непосредственно. Spectrum не распознает эту ситуацию. После заполнения памяти появится сообщение 1 или 6, или дойдет даже до краха системы. Инструкцию DEF FN не имеет смысла задавать с клавиатуры. Зато с клавиатуры можно вызвать функции, имеющиеся в программе. Размещение определений функций и их вызовов не имеет значения. В случае многократного определения в тексте программы функций с одинаковыми именами и типами, действующим определением будет версия, размещенная в строке с меньшим номером. Обратим внимание, что в определении функции могут выступать имена переменных, используемых в других местах программы. Переменные, имеющие имена, идентичные именам параметров, становятся для функции невидимыми, а вместо них выступают значения параметров функции.

DIM @[$](K1...KJ)

Объявление массива. Эта команда резервирует в памяти место для числового или текстового массива. Размерность массива ограничена числом 255. Определенные индексы задают верхние границы, нижней границей всегда является 1. Приказ стирает в памяти существующий массив с таким же именем того же типа. Значения, определяющие размерности, должны быть известны в момент выполнения инструкции DIM, но не обязательно в момент запуска программы.

Числовые массивы в момент определения автоматически заполняются нулями, а текстовые - пробелами. Элементами знакового массива являются единичные символы. Такие массивы могут трактоваться как массивы простых текстовых переменных с одинаковой, заранее заданной длиной. Длина эта устанавливается последним параметром, определяющим массив. Обращаться к массиву знаков как к списку простых текстовых переменных следует при опускании последнего индекса. Если в программе объявлено DIM A$(10): DIM B$(5,12), то A$=A$() является алфавитно-цифровой переменной с длиной 10, а B$(1)=B$(1,) является пятью переменными с длиной 20. Если LET A$="ARA", то длина A$ тоже 20 символов. Нехватающие символы автоматически добавляются пробелами.

DRAW [C,]K,M[,X]

Эта инструкция позволяет рисовать отрезки прямых (X опускается или =0) и фрагментов дуги окружности с центральным углом X, выраженным в радианах. Началом рисунка является последняя точка графической инструкции (PLOT, DRAW, CIRCLE)-(XOLD, YOLD), а концом - точка (XOLD+K, YOLD+M). Если X>0, то дуга рисуется против часовой стрелки, а дляX< 0 - по часовой, любопытным советуем проверить выполнение команд: PLOT 55,27: DRAW OVER 1,120,120,59^3*PI.

ERASE

Команда действует только при наличии ZX-интерфейса.

FLASH K

Включение мерцания поля. K=0 - нет мерцания, K=1 - есть, K=8 - статус данного поля должен оставаться неизменным.

FLASH, как самостоятельная инструкция, определяет способ высвечивания в пределах всей программы, помещенный же в список PRINT или INPUT - только во время выполнения последних.

FOR @=X **TO** Y [**STEP** Z]

Цикл: выполнить для @ от X до Y с шагом Z (Z>0 - увеличение, Z<0 - уменьшение). Конец цикла указывает команда NEXT @. Заданием инструкции FOR является создание управляющей переменной (после уничтожения в памяти числовых или управляющих переменных с тем же именем). Эта переменная имеет в памяти представление, отличное от числовых переменных. Кроме ее значения хранятся также Y, Z и номер строки инструкции FOR. После формирования переменной контролируется: может ли цикл быть выполнен хотя бы раз. Если да, то управление передается следующей за FOR инструкции, если нет - следующей за NEXT @ команде, а если NEXT @ нет, то выводится сообщение I. Если STEP опущен, то Z=1. Несмотря на то, что бейсик формально не отличает чисел с плавающей запятой от целых, для целых X, Y, Z обработка цикла ускоряется на 28%.

FORMAT

Инструкция выполняется только при наличии ZX-интерфейса.

GO SUB K

Вызов подпрограммы. Эта команда размещает в специальном стеке свой собственный адрес, а затем передает управление строке под номером K (или первой большей). Далее выполняются команды до встречи с RETURN, после которой со стека снимается адрес последнего GO SUB и управление передается адресу, следующему за ним. Подпрограммы позволяют выполнять одну и ту же последовательность команд в разных местах программы без их многократного повторения. На глубину вызовов подпрограмм ограничений нет. Допускается рекурентность, т.е. вызов процедурой самой себя.

GO TO K

Переход к строке с номером K или следующей. Если нет такой строки, то программа заканчивается сообщением 0. Эта команда, набранная с клавиатуры, запускает программу с заданной строки. Запуск программы с помощью GO TO (по сравнению с RUN) имеет то преимущество, что не вызывает потерь значений переменных.

IF X **THEN** S

Условный оператор. Если значение выражения X отличается от 0, выполняется последовательность команд S. В противном случае все

команды до конца строки пропускаются и управление передается следующей строке.

INK K

Определение цвета очередных знаков на экране. Эта команда определяет цвет чернил: значения K от 0 до 7 определяют номер выбранного цвета, 8 - сигнализирует согласие на цвет, ранее заданный для данного поля экрана, 9 - задаёт использование белого или черного в зависимости от контрастности установленного в данном поле цвета фона. В нижних строках экрана система всегда использует INK 9. Эта инструкция может применяться как самостоятельно, так и в списке инструкций PRINT, INPUT, CIRCLE и т.д.

INPUT...

Позволяет во время выполнения программы вводить с клавиатуры данные для указанных в списке переменных. Эта команда также позволяет ввод в нижнюю часть экрана, надпись сохраняется лишь на время ввода. После ключевого слова INPUT следует список элементов, определяющих, что и как должно быть выполнено. Элементы списка должны быть разделены запятыми, точкой с запятой или апострофами. Запятая указывает, что надо сдвинуть курсор в следующую половину экрана в той же самой строке, а если это невозможно, то в начало следующей строки. Апостроф обозначает переход к новой строке, а ";" не изменяет положения курсора. Элементы списка, предусмотренные к отображению на экране, должны быть константами, либо выражениями, заключенными в круглые скобки. Спецификации цветов INK, PAPER, BRIGHT, FLASH, INVERSE и OVER позволяют получить любые цветовые эффекты. Переменные, значения которых должны быть заданы с клавиатуры, должны заноситься в список по одной между разделителями. Если мы хотим, чтобы программа считала с клавиатуры элемент массива A(i,j) и проинформировала пользователя, что она от него ожидает, то воспользуемся:

```
INPUT "Введете значение элемента A(i,j)"; A(i,j)
```

INPUT ожидает ввода информации и завершается только после ввода всех переменных, указанных в списке. Каждое значение переменных при вводе необходимо завершать нажатием клавиши <Enter>. Если вводится текстовая переменная, то <Enter> вместо ввода ее значения вызовет дополнение области переменной пробелами. Запросы на числовые переменные нельзя игнорировать этим способом и нужно ввести какое-либо число, используя любые выражения, существующие переменные и константы. При считывании значений числовых переменных внизу экрана мигает курсор **L** или **C**, но при вводе знаковой последовательности курсор появится вместе с окружающими его кавычками. В случае необходимости его можно удалить. Существует возможность считывания текста без этих кавычек. В этом случае текстовую переменную необходимо пометить в списке с ключевым словом LINE. Одно LINE будет отно-

ситься к одной переменной, недопустимо использование этого слова вместе с числовыми переменными. Во время выполнения инструкций INPUT две нижние строки экрана, в случае необходимости, будут передвигаться вверх, создавая свободное место для вводимых данных и вывода. Область эта может максимально расширяться до 22 строк. Ввод данных базируется на редакторе. Благодаря этому перед нажатием <Enter> возможны любые исправления введенных величин. В списке INPUT, кроме этого, можно размещать TAB N, а также AT K,N. Действие первого из них вызывает перемещение курсора до N-ой колонки в данной либо следующей строке. В свою очередь, AT разместит курсор в K-ой строке и N-ой колонке. Некоторые трудности может вызвать факт, что положение начала координат - поля (0,0) - изменяется вместе с расширением нижней части экрана. Поле всегда указывает левый верхний угол рабочей области редактора, и значение поля (K,N) определяется этим. Попытки заставить расширить эту область выше 22 строк кончаются сигналом ошибки 5.

INVERSE K

Очередной определитель цветности. K=0 обозначает высвечивание цветов фона и чернил, а K - 1 меняет их местами. Может применяться как самостоятельно, так и в списке инструкций PRINT, INPUT, CIRCLE и т.д.

LET V=E

Присвоение значения переменной. Выполнение опирается на вычисление выражения E и присвоение полученного значения переменной V, если такая переменная уже существует. Если V еще нет, то она организуется в соответствующем поле памяти. Это не относится к массивам, которые обязательно должны быть объявлены перед использованием.

LIST [K]

Высвечивание на экране текста программы на бейсике с K-ой строки. Если K опущен, то принимается K=0. Данная команда устанавливает указатель текущей строки в K, делая ее доступной для редактирования. Высвечивание всегда производится до конца текста программы. Прервать его можно после появления в нижней части экрана вопроса "SCROLL ?" нажатием одной из клавиш: Break, Stop или N.

LLIST [K]

Инструкция, аналогичная LIST, но печать производится на подключенном принтере. Когда принтера нет LLIST только устанавливает указатель текущей строки на строке K. Во время печати программы останавливается таймер (системные переменные FRAMES не модифицируются).

LOAD...

Spectrum может считывать с магнитофона различные типы данных.

LOAD F вызывает считывание кассеты программы на языке бейсик, записанной на ленте, как набор с названием F. Все другие наборы данных на кассете игнорируются. Вместо полного имени отыскиваемого набора можно дать пустую цепочку слов: LOAD "". Тогда будет считан первый найденный набор данных на бейсике, независимо от его имени. После отыскивания на ленте необходимого блока, находящегося в памяти, программа и ее переменные удаляются (модифицируются соответствующие системные переменные и лишь незначительное число ячеек памяти меняет свое значение) и подготавливается место для новой программы. Доступ к удаленным данным для системы невозможен.

LOAD F **DATA** @[S]

Считывает числовой или текстовый массив и размещает его в области переменных под именем @, удаляя перед этим массив с тем же названием и тем же типом данных, @ должна иметь имя, под которым выступал массив данных перед записью на кассету. Размерность массивов не задается, но скобки обязательны. Как текстовые массивы можно считывать и записывать простые переменные этого типа.

LOAD F **CODE** [K[,N]]

Форма оператора, позволяющая считывать наборы байтов без определения их структуры. Параметр K говорит, по какому адресу должен быть записан в память считываемый блок, а N определяет его длину. Если эти параметры опустить, то набор будет считан на то же самое место, с которого был сброшен на ленту. Эта инструкция не проверяет, может ли считываемый блок уничтожить жизненно важную для системы информацию, и беззаботно затирает соответствующие области памяти новой информацией. Задание параметра N меньше, чем фактическая длина набора на ленте, сигнализируется сообщением R, следовательно, этой инструкцией нельзя читать только начальные фрагменты блоков.

LPRINT...

Инструкция работает как PRINT, но данные посылаются на принтер. Когда принтера нет, инструкция игнорируется. Помещенные в список LPRINT определители цветов PAPER, INK, FLASH и BRIGHT будут игнорироваться. Разделители, а также TAB действуют обычным способом. В AT певый параметр (номер строки) опускается. Печать осуществляется с помощью буфера длиной 32 знака (256 байт). Печать фактически наступает после заполнения буфера, команды перехода к новой строке, а также в момент обычного завершения программы. После прерывания программы в случае ошибки или вмешательства пользователя печать неполного буфера осуществляется нажатием <Enter>.

MERGE F

Считывание программы на бейсике с ленты и присоединение ее к уже существующей в памяти. Строки, номера которых в обе их програм-

мах перекрываются, заменяются новыми, так же как и переменные с одинаковыми именами. Оставшиеся строки и переменные заносятся в соответствующую область памяти. Это позволяет создавать библиотеки программ для последующего их использования. Действие инструкции основывается на считывании всего блока в рабочую область, и только после этого старые и новые программы объединяются. При длинных программах этот процесс может длиться несколько минут. Инструкция может быть применена даже тогда, когда в памяти ничего нет. Отличается от LOAD F значительно долгим временем выполнения и тем, что не запускает программу, даже если она была записана SAVE F LINE K (что очень ценно для пиратов). Вместо имени можно использовать пустой текст.

MOVE

Инструкция выполняется только при наличии Interface-1.

NEW

Действует практически как отключение питания. Инициализирует систему бейсика, сохраняя только системные переменные RAMTOP, PRAM, RASP, PIR И UDG. Память ниже RAMTOP очищена, а область выше - остается нетронутой.

NEXT @

Замыкает цикл FOR. После встречи NEXT @ интерпретатор отыскивает переменную @, увеличивает ее значение на величину шага и сравнивает с предельным значением. Если оно еще не превзойдено, то управление передается инструкции, следующей за FOR, в противном случае - следующей за NEXT @. Если в области переменных не найдется такая числовая переменная @, то это сигнализируется сообщением 2; если же эта переменная не является управляющей - сообщением 1. Одновременно может быть организовано максимум 26 циклов. Управляющие переменные могут быть использованы как внутри, так и вне цикла как обычные переменные. После открытия цикла нельзя модифицировать ни предельного значения, ни шага. Не существует ограничений на взаимное расположение циклов. Например, инструкция, приведенная ниже, недопустимая в других весиях бейсика, воспринимается ZX-бейсиком как правильная конструкция:

```
10 FOR I=1 TO 10
20 FOR J=1 TO 20
30 PRINT I,J
40 NEXT I
50 NEXT J
```

OPEN #K

Спецификация канала директив, позволяющая подключить К-й поток к указанному каналу (см. раздел 7). "Голый" Spectrum (без внешних устройств, а конкретно - Interface-1) распознает только каналы S, K, P, s, k, p. Нельзя использовать потоки 0...3, остальные

(4...15) - в распоряжении пользователя.

OUT M,N

Позволяет с уровня бейсика посылать информацию в порты вывода процессора и может применяться для работы с внешними устройствами. M должно быть числом из диапазона 0...65535, N - из диапазона -255...255, при этом отрицательные числа будут увеличиваться на 256, следовательно, OUT M,-3 это то же самое, что и OUT M,253. Эта инструкция - залог команд ассемблера:

```
LD A,N
LD BC,M
OUT (C),A
```

OVER K

Спецификатор наложения точек или символов. K=0 означает замену соответствующей точки или знака на новый, K=1 - наложение новых знаков на уже существующие. В местах, где старый и новый знак или точка имеют одинаковые цвета, остается цвет фона. Там же, где цвета различные - появляется цвет чернил. Новый рисунск вызывает, следовательно, инверсию предыдущего цвета Это позволяет удалять единичные точки, линии и т.д. на экране. Как и для спецификаторов цветов, область действия заданного режима рисования может быть глобальной во всей программе или только на время действия одной инструкции.

PAPER K

Спецификатор цвета фона. Значения K от 0 до 7 определяют номер выбранного цвета, 8 - цвет, ранее заданный для данного поля экрана, 9 - задает использование белого или черного цветов в зависимости от контрастности установленного в данном поле цвета фона. В нижних строках экрана система всегда использует PAPER 9. Эта инструкция может применяться как самостоятельно, так и в списке инструкций PRINT, INPUT и т.д. Цвет фона может появляться в 2 разных оттенках (см. BRIGHT).

PAUSE K

Временная остановка программы. Параметр K определяет задержку в пятидесятых долях секунды. Нажатие любой клавиши прерывает паузу независимо от пройденного времени. Значение K=0 означает останов, продолжающийся до нажатия любой из клавиш (см. р.9).

PLOT [E,]K,M

Рисование единичной точки с координатами 0<=ABS K<=255 и 0<=M<=175 на экране. Начало координат помещено в левом нижнем углу выше двух строк, зарезервированных для системных сообщений. Эта директива не может применяться в двух нижних операторах.

POKE K,M

Эта инструкция в ячейку с адресом 0<=K<=65535 загружает число -255<=M<=255 (увеличение на 256 для M<0).

PRINT...

Инструкция вывода на экран последовательности знаков и чисел. PRINT печатает элементы своего списка в верхних 22 строках экрана. В списке можно размещать то же самое, что и в INPUT. Нет необходимости заключать выражения в скобки, которые перед вызовом должны быть расчитаны. Лучше применять AT K,N, если положение начала координат постоянно (левый верхний угол). Параметр 0<=K<=21 определяет строку, 0<=M<=5 колонку. Превышение допустимых значений параметров - ошибка 5. После заполнения 22 строк экрана, перед дальнейшим выводом, компьютер предлагает продолжить вывод (SCROLL ?). Клавиши Break, Stop и N прерывают выполнение программы с сообщением D, остальные - согласие на сдвиг экрана.

RANDOMIZE [K]

Инструкция инициализации генератора случайных чисел (RND). Если параметр K задан и не равен 0, то он размещается б системной переменной SEED (описано выше). Пропуск параметра или гадание K=0 вызовет размещение в SEED двух младших байтов системных переменных FRAMES. Побочным применением инструкции RANDOMIZE может быть назначение значений младшего и старшего байтов данного целого числа K, размещенного в диапазоне 1...65535. Меньший байт обозначим как PEEK 23670, а старший - PEEK 23671.

READ V1,V2...

Задает очередным переменным из списка значения соответствующих выражений, помещенных в списках DATA. Каждая из инструкций READ модифицирует соответствующую переменную DATADD, помнящую, какое из выражений списка DATA было считано последним. Попытка чтения уже не существующих данных - ошибка E.

REM

Вставка комментариев. После слова REM можно разместить любую последовательность знаков (за исключением символа с кодом 13) которая при выполнении будет игнорироваться.

RESTORE [K]

Устанавливает системный указатель на первое выражение в списке DATA в строке с номером K или первым, большим K. Если K опущен, то по умолчанию принимается K=0. Перед новым запуском программы инструкцией GO TO обязательно использовать директиву RESTORE, если в этой программе есть операторы READ.

RETURN

Возврат из подпрограммы.

RUN [K]

Запуск программы на бейсике. Выполнение программы начинается со сроки K или, если K опущен, с начала программы. Перед запуском программы RUN автоматически выполняет программу CLEAR, что соответствует уничтожению всех переменных.

SAVE...

Инструкция, обратная LOAD. С ее помощью на кассету записывают программы на бейсике, числозые или знаковые массивы, а также последовательности байтов для уничтожения их структуры. Форма команды такая же, как у LOAD, с тем, что нельзя опускать ни одного параметра. Имя, под которым данный блок должен быть записан на ленту, должно быть непустой последовательностью знаков, не превышающей длиной 10. При записи программы на бейсике после имени можно поместить LINE K. Записанная таким образом программа после считывания будет автоматически запускаться со строки с номером K.

STOP

Останов программы. Программа может быть продолжена нажатием клавиши <Continue>.

VERIFY...

Инструкция, по форме аналогичная LOAD, однако, ничего в компьютере не считывается, а только проверяется, совпадает ли содержимое соответствующих областей памяти с тем, что записано на кассете. Зсе выявленные несоответствия сигнализируются сообщением R.

3.7. Управляющие символы

Среди символов, используемых, в Spectrum, есть так называемые управляющие символы. Они имеют коды меньше 32 (причем не все коды используются). Применение их на уровне бейсика ограничено, т.к. те же самые эффекты можно получить с помощью других инструкций. Однако, они незаменимы для программирования в машинных кодах. Ниже приведены те из них, ввод которых дает эффект:

CHR$ 6 - Этот символ может использоваться вместо запятой, как разделитель в списках INPUT и PRINT;
CHR$ 8 - Вызывает сдвиг курсора на одно поле влево;
CHR$ 9 - см. p.9
CHR$ 15 - символ клавиши Enter. В списках INPUT и PRINT может применяться вместо единичного апострофа. В программах ZX-бейсика он помещается в конце каждой строки;
CHR$ 16 - Ключевое слово INK;
CHR$ 17 - Ключевое слово PAPER;
CHR$ 18 - Ключевое слово FLASH;
CHR$ 19 - Ключевое слово BRIGHT;
CHR$ 20 - Ключевое слово INVERSE;
CHR$ 21 - Ключевое слово OVER;
CHR$ 22 - Ключевое слово AT;
CHR$ 23 - Ключевое слово TAB.

Символы от CHR$ 16 до CHR$ 21 должны набираться в форме CHR$ K+CHR$ M (или CHR$ K; CHR$ M), где M - значение соответствующего данной инструкции параметра.

После символов CHR$ 22 и CHR$ 23 должны следовать 2 параметра. Для AT это понятно, а для TAB единственный аргумент представляется двумя байтами. На практике значение второго байта (старшего) не имеет значения, т.к. это берется по модулю 32.

Оставшиеся из управляющих символов используются редактором в процессе считывания с клавиатуры и не имеют значения для программиста. Во время вывода последовательности знаков символы, не описанные выше, с кодами менее 32, заменяются вопросительными знаками.

3.8. Системные сообщения

Каждая ошибка, обнаруженная Spectrum в программе, вызывает останов и вывод соответствующего сообщения в нижнюю часть экрана Эти сообщения имеют одинаковую форму:

НОМЕР ТЕКСТ СООБЩЕНИЯ K:M

НОМЕР - цифра от 0 до 8 или литера от A до R, ТЕКСТ сообщает по-английски причину передачи управления в редактор, числа K:M ин-

формируют, в какой строке какой инструкции вызвано да иное сообщение. К=0 относится обычно к командам, задаваемым непосредственно с клавиатуры.

0 OK

Нормальное завершение команд, выданных с клавиатуры, конец программы либо пераход с помощью GO TO на строку с наибольшим номером. При этом не модифицируются системные переменные OLDPPC и OSPPC, в результате чего команда CONTINUE, выданная после такого сообщения, вызовет выполнение последней инструкции в программе (это не относится к директивам с клавиатуры).

1 NEXT WITHOUT FOR

Интерпретатор встретил инструкцию NEXT @, а в области переменных нет управляющей переменной с именем @, но есть простая числовая переменная @. Если бы не было и такой, то в аналогичной ситуации было бы выведено сообщение 2. Наиболее частой причиной ошибок является отсутствие соответствующей инструкции FOR или переход внутрь цикла.

2 VARIABLE NOT FOUND

Попытка использования переменной, не существующей в области переменных, следовательно, такой, которой еще не придано никакого значения с помощью LET, READ, INPUT, FOR или DIM.

3 SUBSCRIPT WRONG

Индексы массивов вышли за предельные значения (но поместились в диапазон 0... 65535), или задано их неправильное число, а также попытка присвоения символов простой алфаевитно-числовой переменной с размерностью больше, чем длина переменной. Если индексы превзойдут значение 65535, то в аналогичной ситуации появится сообщение 8.

4 OUT OF MEMORY

Не хватает места в памяти для выполнения требуемой акции. Чаще всего бывает при выполнении инструкций LET, LOAD, MERGE. попытке рекурсивного обращения к функции. Вообще, причиной является слишком низкое положение RAMTOP. Для выхода из данной ситуации может оказаться полезным удаление какой-либо строки из программы, чтобы освободить место для выполнения директив, задаваемых с клавиатуры.

5 OUT OF SCREEN

INPUT пытается занять более 22 строк, или параметры AT указывают поле за 22 верхними строками экрана.

6 NUMBER TOO BIG

Попытка вычисления выйти за 1.7E38. Этим сообщением заканчиваются попытки деления на 0 или вычисления TAN (PI/2).

7 RETURN WITHOUT GO SUB

Попытка выполнить RETURN без предшествующего GO SUB, а также, когда число выполняемых GO SUB меньше, чем число RETURN.

8 END OF FILE

Сообщение, появляющееся в Spectrum в связи с присоединенным ZX-интерфейсом.

9 STOP STATEMENT

Выполнена инструкция STOP.

A INVALID ARGUMENT

Задан неверный аргумент стандартной функции, например, попытка вычислить SQR или LN отрицательного числа, или алфавитно-цифровой аргумент функции USR не является одной верной литерой. Эта ошибка не относится к аргументам функций, определенных программистом.

B INTEGER OUT OF RANGE

Целочисленный параметр инструкции превысил предельное значение.

C NONSENSE IN BASIC

Анализируемый текст некорректен с точки зрения правил языка. Чаще всего появляется, когда аргументы функций VAL или VAL$ не представляют верной формы выражения. Это сообщение появится также при попытке выполнения нераспознанных системой команд. Например, считана и запущена программа, работающая с Microdrive, а ZX-интерфейс не подключен.

D BREAK-CONT REPEATS

Работа программы прервана во время выполнения инструкций, связанных с внешними устройствами (принтер, магнитофон, телевизор), когда после вопроса SCROLL ? нажата клавиша N, Break или Stop. Введенная с клавиатуры директива CONTINUE повторит еще раз прерванную команду.

E OUT OF DATA

Попытка считать данные из списка DATA, если он исчерпан.

G NO ROOM FOR LINE

Нехватка места в памяти для очередной сборки программы, если она слишком длинная, или низко установлен RAMTOP.

H STOP IN INPUT

Останов во время инструкции INPUT.

I FOR WITHOUT NEXT

Инициализируемый цикл не может быть выполнен ни разу (например, FOR I=1 TO 0) или в программе нет инструкции NEXT.

J INVALID I/O DEVICE

Ошибка возможна при подключении Interface-1.

K INVALID COLOUR

Аргумент инструкции INK, PAPER и т.д. вышел за пределы, определенные для данной инструкции. Это сообщение также может появиться после ввода символа, управляющего цветом, если следующий вводимый символ имеет несоответствие с введенным управляющим символом.

L BREAK INTO PROGRAMM

Прерывание выполнения программы пользователем.

M RAMTOP NO GOOD

Попытка присвоения системной переменной RAMTOP слишком малой или слишком большой величины.

N STATEMENT LOST

Попытка выполнения с помощью RETURN, NEXT или CONTINUE перехода к несуществующей инструкции.

O INVALID STREAM

Попытка пересылки информации потоком до подключения к какому-нибудь каналу с номером, большим 15.

P FN WITHOUT DEF

Вызвана инструкция, которая не определена.

Q PARAMETER ERROR

Рассогласованность количества или типов аргументов при вызове функции пользователем.

R TAPE LOADING ERROR

Требуемый набор был найден на кассете, но по какой-либо причине не может быть считан в компьютер.

4. КОМПЬЮТЕРНАЯ АРИФМЕТИКА

Единичная ячейка памяти компьютеров, сделанных на основе процессора Z80, размещает в себе 1 байт, состоящий из 8 цифр, каждая из которых может быть или 0 или 1. Переход от двоичного представления к десятичному очень прост. Выпишем в ряд по очереди числа 2^7, 2^6, ..., 2^1, 2^0. Под каждым из них запишем одну из цифр двоичного представления. Складывая между собой только те числа из верхнего ряда, под которыми оказались единицы, получим десятичное значение. Например:

```
120  64  32  16  8  4  2  1
0    1   0   0   1  1  0  1
```

64+8+4+1=77.

Содержимым ячейки памяти может быть положительное число в диапазоне 0...255.

Иногда выгоднее рассматривать эти числа, как значение со знаком. Самый старший разряд (левый бит) определяет знак, если 0 - плюс, если 1 - минус. При такой интерпретации мы имеем числа в диапазоне -128...127. Однако, компьютер всегда выполняет действия лишь над натуральными числами без знака, поэтому в Spectrum для представления чисел применяется код дополнения до двух. Числа <0 записываются в нем как 256-7=249. В этой системе самый старший бит интерпретируется как знак, поэтому проблем, связанных с арифметикой, не существует. Основные арифметические операции ведут к верным операциям независимо от выбора способа записи чисел. Например: 246+1=247 является очевидным равенством для чисел без знака. Оно также является верным, если считать 246 как -7.

Использование таких маленьких чисел недостаточно, поэтому большие числа приходится хранить в нескольких байтах, следующих один за другим. В Spectrum важную роль играют 16-значные числа (2 байта). Первый (левый) байт называется старшим, другой - младшим. Во всех компьютерах, организованных на Z80, принято правило, что если 2 очередных байта с адресами x, x+1 содержат какое-либо число, то по адресу x размещен младший байт, а по адресу x+1 - старший. Чтобы определить, какое число содержат эти ячейки, необходимо старший байт умножить на 256 и прибавить младший. Если это отрицательное число, представленное в коде с дополнением до 2, то нужно еще отнять 2^16=65536. Эти двухбайтовые числа позволяют хранить значения в диапазоне 0...65535 без знака или -32768...32767 со знаком.

При программировании в машинных кодах часто возникает необходимость выполнения разных операций над битами. Использование десятичного представления при этом затруднительно, а двоичного - слишком громоздко. Поэтому наиболее удобной является 16-ричная запись. Для этой записи используются цифры 0...9 и буквы от A до F (A=10, B=11, C=12, D=13, E=14, F=15). Пересчет 16-ричного представления в десятичное нетруден: умножать очередную цифру на очередную степень числа 16 и складывать произведения. Например, 16-ричное число FFFF представляется в виде: 15*16^3+15*16^2+15*16^1+15*16^0=15*4096+15*256+15*16+15.

Переход от двоичной формы к 16-ричной (и обратно) особенно прост. Делим цифры двоичного числа на группы по 4 бита и каждую четверку представляем 16-ричной цифрой. Например:

1100001001111110=1100+0010+0111+1110=C27E.

Как видно, главным преимуществом 16-ричных чисел является простота перехода к двоичным и наоборот при одновременной высокой скорости записи. Далее 16-ричным числам будет предшествовать знак #.

Способы кодирования больших чисел и чисел с плавающей запятой для различных компьютеров различны. В ZX-бейсике все числа хранятся в пяти последовательных байтах. Целые числа из диапазона -65536...65535 кодируются иначе, чем остальные. Первый байт в их представлении всегда =0, последний байт положительного числа =0, а отрицательного =-256. В третьем и четвертом разместятся соответственно младший и старший байты данного числа, причем отрицательные значения хранятся в коде дополнения до 2. Например, число 38 выглядит так: 0 0 0 38 0, а число 743=2*256+231 имеет вид: 0 0 231 2 0. В свою очередь, число -1231 запишется как 0 255 251 49 0.

С остальными числами все запутано еще больше. Используется факт, что каждое число можно однозначно представить в виде 2^N*M, где M - число в диапазоне (1/2,1). В первом байте Spectrum хранит значение N+128, в остальных четырех размещено число M в двоичной форме, а точнее M*2^32, с тем, что самый старший бит, котрый всегда должен быть равен 1, служит для хранения знака числа (1 означает минус, а 0 - плюс).

Достоинством языков высокого уровня (в т.ч. ZX-бейсика) является то, что программист освобожден от записи представления различных чисел в память.

5. ИСПОЛЬЗОВАНИЕ ПАМЯТИ

После включения компьютера в сеть автоматически запускается программа, размещенная в ROM по адресу 0. Она проверяет объем допустимой памяти, разделяет ее на различные блоки, а также присваивает системным переменным начальные значения.

Диаграмма, приведенная низке, описывает раздел памяти. Объем от 0 до 16383 размещается в ROM, остальное - в RAM. Только часть блоков имеет свое постоянное положение, другие могут перемещаться в памяти, и их актуальные адреса хранятся в соответствующих системных переменных. Числа в конце некоторых областей обозначают указатели конца области и всегда присутствуют там.

АДРЕС	ОБЛАСТЬ	КОНЕЦ
#0000(0)	Системные процедуры	
#3000(15616)	Прообразы символов в кодах 32...127	
#4000(16384)	Экран	
#5800(22528)	Атрибуты	
#5B00(23296)	Буфер принтера	
#5C00(23552)	Системные переменные	
#5CB6(23734)	Карта микродрайвера	
CHANS	Информация о каналах	#80(128)
PROG	Бейсик	

VARS	Переменные ZX-бейсика	#80(128)
E LINE	Буфер редактора	#80(128)
W0RKSP	Буфер инструкции INPUT	#80(13)
	Рабочая область ZX-бейсика	
STKBOT STKEND	Стек калькулятора	
SP	Свободная область памяти	
UDG P RAM	Прообразы символов, определенных пользователем ZX-бейсика	
ERR SP	Машинный стек	
RAMTOP	Стек адресов возврата GO SUB	#3E(62)
	Свободная область памяти	
UDG P RAMTOP	Прообразы символов, определенных пользователем	

5.1. Прообразы символов

Формы печатных символов с кодами от 32 до 127 (8 байт на каждый знак) заполняют 768 последних ячеек ROM. Они закодированы в той же форме, как и символы, определяемые пользователем. Знаки мозаичной графики (коды от 128 до 143) конструируются каждый раз. Адрес начала этой области (256) хранится в системной переменной CHARS.

5.2. Экран

Эта область предназначена для хранения информации о каждой точке экрана, точнее о том, должна ли она быть в цвете чернил или фона. 6144 байта этого блока позволяют адресовать 49152 различных точек (1 байт описывает 8 точек). Они организованы в таблицу из 192 строк и 256 столбцов. Для графических инструкций бейсика доступны только 176 верхних строк. Каждый графический знак выводится в поле 8*8 точек. Это позволяет выводить тексты в формате 24 строки по 32 позиции. Две нижние текстовые строки зарезервированы для системных сообщений и для рабочей области редактора.

Интересен способ, которым эта область переносится на экран. К сожалению, последовательные 32 байта не описывают последовательно графическую строку на экране, а 256 последовательных байт не описывают оператор текста. Размещение графической строки, с точки зрения ZX-бейсика, похоже на изобретение сумасшедшего. Легче всего это можно увидеть при считывании экрана с ленты. Здесь мы только ограничимся примерами, позволяющими пересчитывать текстовые и графические координаты на графические адреса.

Байт, содержащий графическую точку с координатами (К,М), (точка (0,0) лежит в верхнем левом углу экрана выше двух низших текстовых операторов) на экране имеет адрес:

```
16384+32(INT((l75-M)/8)-INT((175-M)/64)*8+
+8*(175-M-INT((175-M)/8)*8+64*INT((175-M)/64)*INT(K/8)).
```

Положение бита в байте расчитывается как 8-K+INT(K/8)*8. В свою очередь адреса текстовых знаков расчитываем по образцу:

```
16384+2048*INT(K/8)+32*(K-8*INT(K/8))+256*M+N,
```

где К обозначает номер оператора (0...23), N является номером колонки (0...31), М - номер графической строки (линии), формирующей этот знак (0...7), за нулевую принимается лежащая поверху блока знаков линия. Начало координат для инструкций, пишущих знаками, помещено в левом верхнем углу экрана.

5.3. Атрибуты (22528...23295)

В этой области хранится информация о размещении цветов на экране. Наименьшая единица поверхности, цвет которой можно модифицировать - поле единичного знака размером 8*8 точек. Для такого поля можно определить цвет фона и цвет чернил, а также мерцание. На выбор имеется 8 цветов:

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

Каждый цвет может выступать в двух оттенках: обычном и интенсивном (BRIGHT 0,1).

Все сведения, касающиеся единичного знакового поля закодированы в одном байте:

БИТЫ	ЗНАЧЕНИЕ
7	Мерцание
6	Интенсивность фона
5,4,3	Цвет фона
2,1,0	Цвет чернил

Чтобы установить желаемые атрибуты поля и определить, какое значение следует поместить в соответствующий байт, лучше всего пользоваться образцом:

128*F+64*B+32*P+I,

где F определяет мерцание (FLASH 0,1), B - оттенок тона (BRIGHT 0,1), P - номер цвета фона, I - цвет чернил.

Определение адреса байта, описываемого атрибутами знакового поля с координатами (K,M) осуществляется по формуле:

22528+32*K+M.

5.4. Буфер принтера

В этой области собираются данные, которые будут посланы на принтер. Фактически, печать следует после заполнения этого буфера (256 байт или 32 знака) или после символа "конец строки" (CHR$13). Если принтер не подключен, то область не будет использоваться системой и может использоваться для др. целей.

5.5. Карта микродрайва

Эта область используется только в случае подключения к Spectrum Interface-1. В "голом" компьютере эта карта не существует физически и CHANS = 23734.

5.6. Программа ZX-бейсик

После инициализации системы, без подключенных внешних устройств, этот блок начинается с адреса #5CCB (23755). В нем хранится текст программы бейсик. Отдельные строчки программ в памяти компьютера имеют следующую форму:

2 байта	2 байта	Длина текста	1 байт
Номер	Длина	ТЕКСТ	Код <Enter>
Строки	текста		

В случае номера строки сделано отступление от правил, и первым здесь выступает старший байт, а затем младший. Все ключевые слова, появляющиеся в тексте, кодируются единичными символами.

Интересным является способ хранения чисел в тексте программы. За последовательностью знаков, представляющих CHR$14, а за ним 5 байтов, содержащих двоичное представление этого числа. Во время вывода программы на экран эти 6 добавочных байтов опускаются. Это объясняет, почему некоторые программисты предпочитают запись VAL "17" вместо просто 17. Первая занимает, фактически, 5 байтов, а вторая - 8. Речь идет об экономии памяти. Во время выполнения программы, в свою очередь, пропукаются символы, которыми записано чис-

ло, и используется 5-байтовое двоичное представление. Это позволяет прятать правильные значения определенных чисел (например, стартовых адресов процедур в машинном коде) от "любопытных". В этом случае необходимо определить адрес внутреннего представления данного числа и инструкцией POKE присвоить ей желаемое значение. При выводе текста на экран и в дальнейшем будет фигурировать старое представление числа, уже не связанное с записанным за ним двоичным числом. Применяя эту технику, необходимо помнить, что каждое извлечение такой строки в нижнюю часть экрана и возврат ее на место, даже без модификации, изменит внутреннее представление каждого числа в тексте, присваивая им значения, видимые на экране.

Из других любопытных трюков приведем еще два. Если в первой строке программы, в ячейках, содержащих ее номер, поместим число 0, то такой оператор невозможно будет скопировать в нижнюю часть экрана в область редактора, а затем модифицировать.

В другом конце программы возможна другая штучка. Допишем там строку 9999 REM и после установления ее адреса в обоих байтах, содержащих ее номер, разместим значение 255. Первый результат этого проявится при выводе программы на экран. Модифицированная нами строка не появится. Второй эффект, более ценный, проявится при попытке считывания такой программы с кассеты инструкцией MERGE. Spectrum обижается на пользователя и перестает реагировать на любые клавиши.

5.7. Переменные бейсика

В этой области система хранит все переменные, инициализированные как программой, так и пользователем с клавиатуры. Появляющиеся переменные последовательно дописываются в конце этой области (все последующие блоки автоматически сдвигаются) и остаются там, пока область не будет очищена. Исключением являются простые текстовые переменные: когда какая-нибудь из них модифицируется, ее предыдущая версия убирается (что сопровождается необходимым перемещением других блоков памяти), а новая дозаписывается в конце этой области. То же при новом объявлении массивов.

Способ хранения переменных зависит от их типов. В ZX-бейсике существует 6 различных видов переменных, обозначенных числами от 2 до 7:

2(010) - простые текстовые переменные;
3(011) - простые числовые переменные с однолитерными номерами;
4(100) - числовые массивы;
5(101) - простые числовые переменные с многолитерными номерами;
6(110) - знаковые массивы;
7(111) - переменные, управляющие циклами FOR...NEXT.

Во всех случаях первый байт описания переменной содержит ее тип, а также номер первой литеры имени. Метод размещения этих сведений в одном слове следующий:

биты 7,6,5 - тип,
биты 4,3,2,1,0 - номер литеры.

Обратим внимание, что биты 4...0 содержат порядковый номер литеры в алфавите, а не ее код. Храня имена переменных, Spectrum автоматически заменяет большие литеры малыми и пропускает все пробелы. Содержимое последующих байт зависит от типа переменной.

Простая текстовая переменная.

Длина описания N+3 байта: тип и литера; N=длина текста (2 байта); текст.

Простая числовая переменная с однолитерным именем.

Длина описания 6 байт: тип и литера; содержимое (5 байт).

Числовой массив.

Длина описания N+3 байта: тип и литера; N - длина описания (2 байта); К=число индексов; 1-й...К-й индексы (по 2 байта); элементы (по 5 байт).

Простая числовая переменная с многолитерным именем.

Длина описания К+5 байт: тип и литера; 2-й...К-й знаки имени; значение (5 байт). Второй и последующий символы имени хранятся как коды соответствующих знаков. Последний байт имени имеет самый старший бит, всегда установленный в 1, что позволяет распознать его.

Знаковый массив.

Длина описания N+3 байта: тип и литера; N=длина описания (2 байта); К-число индексов: 1-й...К-й индексы (по 2 байта); элементы (по 1 байту).

Последний байт области переменных, как указатель, всегда содержит значение #80 (128).

5.8. Буфер редактора (E LINE - WORKSP-1)

В этой области размещается строка программы или последовательность директив для непосредственного выполнения во время их ввода с клавиатуры. Вводимые символы копируются в нижнюю часть экрана. После нажатия <Enter> компьютер проверяет верность вводимого текста и предпринимает соответствующие действия.

5.9. Буфер инструкции INPUT (WORKSP - STKBOT-1)

В этот буфер первоначально вводятся данные, считываемые командой INPUT, и только после нажатия <Enter> следует их преобразование. В этой области Spectrum также выполняет некоторые операции, требующие рабочей памяти (например, сцепление символьных переменных).

5.10. Стек калькулятора (STKBOT-STKEND-1)

Рабочая область системных программ, выполняющих большинство арифметических операций.

5.11. Свободная область системы бейсик (STKEND+1-SP)

Этот блок памяти свободен, но предназначен для потребностей системы бейсик. Рабочие области с обоих концов этого блока пополняются за его счет. Данные, размещенные здесь, могут подвергнуться уничтожению без предупреждения. Перед началом действий, требующих увеличения какого-либо из рабочих блоков, Spectrum проверяет, останется ли в этом блоке хотя бы 80 свободных байт на возможные потребности машинного стека. Адрес конца этой области хранится в регистре процессора Z80, называемом указателем стека (SP) и недоступном из системы бейсика.

5.12. Стек машинный (SP-ERR SP)

Стек, используемый процессором Z80 для текущего хранения различных данных, адресов и т.п. Эта область абсолютно бесполезна для программирования на бейсике.

5.13. Стек адресов возврата GO SUB (ERR SP+1-RAMTOP)

Стек, предназначенный для хранения номера строки и положения в ней инструкции GO SUB. Эти данные необходимы для команды RETURN при аварийном выходе из подпрограммы. Многоразовое выполнение GO SUB или RETURN вызывает сдвиг машинного стека, адресуемого через ERR_SP на 3 байта вниз или вверх. Поэтому при модификации блока рекомендуется особая осторожность.

5.14. Область свободной памяти (RAMTOP+1-P_RAMT)

Блок действительно свободной памяти и безопасной в том смысле, что его содержимое недоступно для системы бейсик и может быть модифицировано только по требованию пользователя с помощью инструкции POKE. Даже директива NEW не нарушает этой области. В принципе, она используется для хранения программ в машинном коде или специфицированных данных, для которых переменные бейсика не проходят.

5.15. Образцы символов пользователя (UDG-UDG+167)

После инициализации системы этот блок начинается с ячейки с адресом RAMTOP+1=#FF58-65368 и кончается в ячейке с адресом P RAMT-#FFFF=65535. Адрес этого блока в памяти можно получить, выполняя инструкцию USR "A". Эта область служит для хранения очертаний графических символов, определенных пользователем. Определение единичного символа состоит из 8 байт, описывающих состояние каждой точки в знаковом поле размером 8*8.

Для примера определим графический символ и разместим его в области UDG так, чтобы он высвечивался при нажатии клавиши "O" в режиме **G**. Первым этапом является проектирование формы нового знака в квадрате 8*8. В поле, предназначенном для замазывания, размещается 1, а в пустом - 0. Каждый заполненный таким образом оператор рассматривается как восьмицифровое бинарное число. Полученный набор 8 чисел описывает форму проектируемого символа:

```
00000100       4
00001000       8
00000000       0
00111100      60
01000010      66
00111100      60
00000000       0
```

Введем теперь его в компьютер. Проще всего это сделать с помощью программы:

```
10 DATA 4,8,0,60,66,60,0
20 FOR I=USR "O" TO USR "O"+7
30 READ A: POKE I,A
40 NEXT I
```

После ее выполнения клавиша "O" в режиме **G** выводит закодированный символ. В цепочке символов он будет иметь код 158. Данные в строке 10 можно подать также в двоичной форме вместо десятичной, используя функцию BIN. Это требует больше писанины, но значительно облегчает все модификации определенного нами образца. 168 байт области UDG позволяют хранить до 21 различных новых символов. Их коды размешаются в диапазоне от 144 (A) до 164 (U). После инициализации системы этот блок содержит образцы форм больших литер латинского алфавита.

6. СИСТЕМНЫЕ ПЕРЕМЕННЫЕ

Системные переменные хранятся с адреса #5C00 (23552) до области, занимаемой программой бейсик. Программы из памяти ROM используют ее для запоминания информации, описывающей состояние компьютера. Ниже будут рассмотрены те переменные, которые могут модифицировать-

ся и использоваться программистом. Остальные лучше не трогать, т.к. это может привести к порче операционной системы. Описываемые переменные разделим (согласно специфических функций, а не положения в памяти) на 5 групп:

- хранящие важные для системы адреса,
- обслуживающие клавиатуру,
- описывающие состояние системы,
- связанные с каналами и потоками,
- обслуживающие экран телевизора.

6.1. Важные для системы адреса

ERR_SP	#5C3D (23613)	STKEND	#5C65 (23653)
VARS	#5C4B (23627)	RAMTOP	#5CB2 (23730)
PROG	#5C53 (23635)	P_RAMT	#5CB4 (23732)
E_LINE	#5C59 (23641)	CHARS	#5C36 (23606)
WORKS	#5C61 (23649)	UDG	#5C7B (23657)
STKBOT	#5C63 (23651)		

Применение этих переменных описано в предыдущем разделе. Естественным использованием этих переменных может быть перераспределение свободной памяти (либо занятой программой) переменными и т.п. Только RAMTOP может быть легко модифицирована из бейсика посредством CLEAR K. К сожалению, эта команда имеет и другие побочные действия, что временами делает ее применение невозможным.

P_RAMT. Содержит адрес последней, физически присутствующей в компьютере, ячейки памяти. Следовательно, может быть использована для распознавания программой, в какой модели ZX Spectrum она использована (16K или 48K).

CHARS. Благодаря этой переменной можно определить собственные формы вывода. После размещения их в память RAM под адресом K достаточно задать переменной CHARS значение K=256. Начиная с этого момента все тексты, включая программный листинг, будут печататься нашими новыми знаками. Этим способом можно изменить очертания всех символов с кодами от 32 до 127. Следует подчеркнуть, что модифицируются только очертания знаков, но не их значения, которые зависят от кодов символов. Можно использовать эту переменную для трюков, делающих текст программы абсолютно нечитаемым. Достаточно уменьшить значение CHARS на 8, тогда в распечатке программы трудно будет найти какой-либо смысл. Зато надписи, вводимые запущенной на выполнение программой, останутся разборчивыми, если будут заблаговременно сдвинуты на одно значение кода (т.е. в исходном тексте нужно поместить B вместо A, O вместо N и т.д.).

UDG. Польза от размещения этого адреса в списке системных пере-

менных состоит в том, что этот блок мы можем перемещать в памяти. Столь же ценной является возможность использования нескольких наборов нетиповых символов одновременно. В этом случае, перед выводом данного знака, требуется модифицировать UDG, присваивая переменной значение набора, необходимого в данный момент. Модификация остальных переменных этой группы очень опасна и может легко привести к сбою и зависанию системы.

ERR_SP. Эта переменная - основание машинного стека. Там, в свою очередь, помещен адрес процедуры з памяти ROM, отвечающей за обслуживание всех ошибочных ситуаций, требующих прерывания программы и выдачи соответствующего сообщения. Под ошибкой здесь понимается также и нормальное завершение программы, сигнализируемое сообщением 0 OK. Страстные программисты, знающие ассемблер Z80, могут благодаря этому писать собственные программы обработки ошибок, что позволяет, между прочим, добавление новых команд к языку. Модификация ERR_SP с уровня бейсика всегда ведет к краху системы в момент возникновения ситуаций, требующих вывода любого сообщения. Это часто используется для защиты программы от несанкционированного просмотра и кодирования. Этот метод в соединении с записью программ при помощи SAVE F LINE..., а также с исключением возможности считывания программы инструкцией MERGE, обычно полностью позволяет обезопасить программу от менее грамотных пиратов. К сожалению, спасения от многочисленных копирующих программ нет.

6.2. Переменные, обслуживающие клавиатуру

KSTATE #5C00 (23552).

Эта переменная построена на 8 байт и используется для чтения клавиатуры и обслуживания самопроверяемости клавиши. Для программиста лишь ячейка с адресом #5C04 (23556) может иметь значение. Она содержит значение 255. Если ни одна клавиша не нажата или код основного значения нажатой клавиши в режиме **C** (большая литера алфавита или цифра), PEEK 23556 дает поэтому тот же самый эффект, что и CODE INKEY$ в режиме **C**. Достоинством является факт, что значение не зависит от состояния курсора (**L** и **C**). В случае нажатия нескольких клавиш одновременно, всегда будет распознана первая (INKEY$ в такой ситуации может игнорировать все). Клавиши CS и SS, нажимаемые по отдельности, не воздействуют на нашу ячейку, но нажатые вместе дают код 14. Кроме того, комбинации CS/9, CS/3, CS/4, CS/2 выдают соответственно коды 15, 4, 5, 6.

LAST_K #5C08 (23560).

Эта байтовая переменная хранит код нажатой клавиши, независимо от того, нажимают на нее или нет. В случае нажатия нескольких - будет запомнен первый. В отличие от предыдущей переменной LAST_K следит за состоянием курсора. С этой переменной частично связана следующая переменная.

FLAGS #5C3B (23611)

Сколько раз переменная LAST_K принимает новое значение, столько раз пятый бит переменной FLAGS устанавливается в 1. Это важно, если мы пытаемся распознать многократное нажатие на одну и ту же клавишу. Третий бит этой переменной вместе с содержимым MODE позволяет распознать, в какой ситуации относительно чтения клавиатуры должен быть использован курсор **K**. Однако, эту информацию можно использовать лишь с уровня ассемблера. Седьмой бит переменной FLAGS сигнализирует системе, с клавиатуры или из программы пришла выполняемая команда. Размещенная в программе инструкция POKE 23611,0 прервет ее и выполнит сообщение 0 OK.

REPDEL #5C09 (23561) и REPER #5C0A (23562).

Две эти переменные необходимо рассматривать одновременно. Первая из них определяет, как долго надо нажимать клавишу, чтобы заработал механизм самоповторения. Вторая определяет время между считыванием с клавиатуры. В обеих переменных время дано в 1/50 сек. Поначалу эти переменные инициализированы значениями 35 и 5, что обозначает, что нажатая клавиша через 0,7 сек начинает саморазмножаться с темпом 10 знаков в секунду. В процессе считывания с клавиатуры эти значения можно уменьшить до 0. Придавая им значение 0 добиваются, наибольшей остановки, продолжающейся столько же, сколько PAUSE 256. Занесенные в эти ячейки значения 1 приводят к тому, что работа со Spectrum становится трудной, но возможной (ввод любой директивы будет требовать отличной реакции).

RASP #5C38 (23608)

Эти переменные определяют звук, сопровождающий ввод данных. RASP задает (в 1/50 с) длительность предупредительного ворчания, когда Spectrum не желает больше считывать данные. Начальное значение RASP=64 (1,26 с).

PIP определяет длительность звука, подтверждающего нажатие клавиши (начальное значение=0). Стоит помнить, что PIP и RASP, в отличие от большинства других системных переменных, не инициализируются заново инструкцией NEW.

6.3. Переменные, описывающие состояние системы

Эти переменные позволяют системе контролировать ситуацию и интерпретировать очередные программные строки.

NEWPPC #5C42 (23618) и #5C44 (23620).

Первая из них (2 байта) содержит номер строки. Вторая (1 байт) осуществляет переход. Следующие команды вызывают непосредственный переход к M-й инструкции в N-й строке, следовательно, действуют значительно точнее команды GO TO. POKE 23618,L-256*INT(L/256):

POKE 23619,INT(L/256): POKE 23620,M

PPC #5C45 (23621) и SUBPPC #5C47 (23623).

Эти переменные содержат соответственно номер строки и номер выполняемой команды. Вместе с переменной ERR_NR они могут служить для завершения программы с ранее заданным сообщением без провоцирования фактической ошибки. С этой целью программу следует завершить инструкцией GO TO 9999, а в этой строке поместить команды:

9999 POKE 23621,L-256*INT(L/256): POKE 23622,INT(L/256): POKE 23610,M-1: POKE 23623,K

В этом случае программа завершается сообщением: M TEXT L;K.

ERR_NR #5C3A (23510).

В этой ячейка появляется номер ошибки, которая произошла и будет сигнализироваться сообщением. Содержимое этого байта может быть ценным при написании собственных процедур обработки ошибок на ассемблере, т.к. сразу определяет причину прерывания программы.

E_PPC #5C49 (2325).

Эта переменная (2 байта) содержит номер текущей строки и доступна для редактирования. Ее модификация вызовет тот же эффект, что и LIST N, но без вывода программы на экран. Это делает возможным выбор с клавиатуры строки программы для модификации, а затем копирование ее в нижнюю часть экрана и корректировку без уничтожения содержимого экрана.

MODE #5C41 (23517).

Эта переменная информирует редактор, какой курсор должен использоваться в данный момент. Система использует следующие значения: 0 - **L**, **C** или **K**; 1 - **E**; 2 - **G**. Эти числа затем используются в различных выражениях для назначения кода символу, который должен высвечивать курсор. Программируя на бейсике можно склонить Spectrum, чтобы в ближайшей инструкции INPUT вместо стандартного курсора **L** или **C** использовался курсор **G** или **E**, причем **E** будет действовать только для первого вводимого символа Режим **E** включается POKE 23617,1. Интересные эффекты дает размещение в этой переменной других значений, отличных от 1, 2. POKE 23617,K, для K=3...127 вызывает курсор **G**, хотя на экране вообще не будет печататься литера **G**, а нечто другое. POKE 23617,K для K=128...255 устанавливает режим **L** или **C**, но с выводом на экран другого мерцающего курсора.

FLAGS #5C6A (23658).

Эта переменная определяет, какой должен быть режим: **L** или **C**. За это отвечает третий бит FLAGS. POKE 23658,0 устанавливает режим **L**, а POKE 23658,8 - режим **C**. Подобные изменения получают с клавиатуры

с помощью CS/2.

DF SZ #5C6B (23659).

Эта переменная содержит число строк нижней части экрана, зарезервированной для системных сообщений и данных, вводимых с помощью INPUT. Инициализируется значением 2. Практически не применяется, хотя иногда может быть использована для защиты программ от прерываний. Помещение в нее 0 вызывает зависание системы, как только возникнет необходимость вывода чего-нибудь в нижнюю часть экрана (например, сообщения о нажатии <Break>). Однако, это невыгодное оружие, т.к. одновременно делает невозможным применение инструкций INPUT, CLS, а также не позволяет задать вопрос "Scroll ?".

OLDPPC #5C6E (23622) и OCPPC #5C70 (23664).

Они содержат номер строки и команды в строке, к которым будет осуществляться переход в случае выдачи команды CONTINUE. Модификация этих переменных в программе позволяет найти точнейшую альтернативу инструкции GO TO. В некоторых ситуациях это лучше, чем модификация NEWPPC и NSRRC, т.к. переход наступает лишь при встрече с CONTINUE, а не сразу же после модификации переменных. Эти переменные автоматически корректируются системой всякий раз, когда наступает прерывание с сообщением, отличным от 0 OK.

SEED #5C76 (23670).

Это основа генератора псевдослучайных чисел. Использование SEED рассматривалось при описании функций RND.

FRAMES #5C78 (23672).

Три байта составляют внутренние часы Spectrum. Они представляют число PEEK 23672+256*PEEK 23673+65536*PEEK 23574 и определяют, сколько 1/50 долей секунды прошло с момента инициализации системы. Максимальным значением является 2^24-1 = = 16777215, что отвечает 3 суткам 12 минутам 24,3 секундам. Точность таймера составляет 0,01% или менее 9 секунд в сутки. Этот таймер выключается на время обслуживания внешних устройств (принтера, магнитофона, динамика), а также выполнения программ в машинном коде, которые включают или обходят контроль замаскированными прерываниями.

DATADD #5C57 (23639).

Эта переменная хранит адрес элемента в списке DATA, который будет считан очередной инструкцией READ. Запоминая содержимое этих 2 ячеек и используя его потом, можно получить точнейшую альтернативу команды RESTORE K.

SCR_CT #5C8C (23692).

Этот байт определяет, после вывода скольких строк + 1 на экране должен появиться вопрос "Scroll ?". Если надо, чтобы вывод не прерывался, то, по крайней мере один раз на 250 выведенных строк, не-

обходимо в эту ячейку занести значение 255. Также, если нам надо прервать вывод раньше, можно присвоить этой переменной значение, меньшее 23.

6.4 Переменные, обслуживающие экран телевизора

Эта группа переменных отвечает за правильность взаимодействия компьютера с телевизором. Одна часть из них управляет цветом, другая - определяет место вывода очередного знака или графического символа.

BORDER #5C48 (23624).

Эта переменная содержит атрибуты, описывающие нижнюю часть экрана, а также цвет рамки. В обычных условиях Spectrum не позволяет устанавливать одинаковый цвет фона и чернил в нижних строках (пользователь всегда должен видеть вводимые им символы). Спецификаторы цветов в списке INPUT будут действовать только для текстов, печатаемых в нижней части экрана, но цвет чернил для вводимых данных всегда устанавливается 9 (белый или черный в зависимости от цвета фона). Обойти это можно (например, если компьютер должен считывать с клавиатуры тайный пароль) используя команду POKE 23624,128+F+64*B+B*P+I и присваивая литерам F, B, P, I значения параметров команд FLASH, BRIGHT, PAPER и INK, необходимых для получения желаемого эффекта. Тем, кто не убежден, рекомендуем получить этот эффект иным способом: BORDER 3: POKE 23624,222: CLS

ATTR_P #5C8D (23693) и ATTR_T #5C8F (23695).

Обе переменные однобайтовые и хранят значения атрибутов FLASH, BRIGHT, PAPER и INK. Литера P обозначает величины, установленные постоянно для всей программы соответствующими командами, литера T - текущие значения, устанавливаемые теми же самыми командами, размещенными в списках соответствующих пишущих и рисующих команд. При отсутствии спецификаторов цвета в списках пишущих инструкций, переменная ATTR_P копируется в ATTR_T. Способ хранения информации о цветах в одном байте описан в р.5. Переменная ATTR_P может быть использована в программе бейсика для установления всех атрибутов одной инструкцией POKE. ATTR T, в свою очередь, может быть важна для программирующих в машинном коде, т.к. обычно используется процедурами памяти ROM для установления заранее заданных цветов.

MASK P #5C8E (23694) и MASK T #5C90 (23696).

Эти переменные используются при реализации команд FLASH, BRIGHT, PAPER и INK с параметром 8. Значение литер P и T в именах переменных такое же, как и для переменных ATTR. Установление какого-либо бита этих переменных в 1 означает, что бит с тем же самым номером в соответствующем байте атрибутов должен оставаться неизменным. Обратим внимание, что команда INK 8 устанавливает в 1 все три младших бита переменной MASK_P, модифицируя эту переменную, ин-

струкцией POKE. Можем, например, установить в 1 только один младший бит. Тогда бу-дет получен эффект фильтра. Неизменной в цвете чернил останется только основная синяя компонента, в то время, как остальные могут подвергаться изменениям. Обратим внимание на то, что нумерация цветов совсем случайна: синий - 1, красный - 2, зеленый - 3. Это основные цвета, смесь которых позволяет получить все остальные. В Spectrum роль смешивания выполняет добавление номеров соответствующих цветов.

COORDS #5C7D (23677).

Два последовательных байта этой переменной содержат координаты (X,Y) точки на экране, в которой завершили рисова-нир инструкции PLOT, DRAW или CIRCLE. Модификация этой переменной дает тот же эффект, что и PLOT OVER I,K,N: PLOT OVER I,K,N, то есть смещение указателя экрана без рисования какой-либо точки или линии.

S_POST #5C88 (23688).

Эта переменная (2 байта) содержит значение 33-K, 24-M, где K, M - координаты последнего выведенного на экран знака. Непосредственная модификация этих переменных затруднительна, т.к. необходимо одновременно модифицировать DF CC.

DF_CC #5C84 (23684).

Эта переменная содержит адрес байта на экране, с которого начинается вывод очередного символа инструкцией PRINT. Модифицируется только с Z_POSN. Значительно проще применить команду PRINT AT K,N.

SPOSNL #5C8A (23690) и DFSSL #5C86 (23686).

Переменные, аналогичные S_POSN и DF_CC, описывающие нижнюю часть экрана.

P_FLAG #5C91 (23697).

Переменная содержит информацию о режиме печати и рисования на экране. Описывает режимы, установленные бейсиком с помощью инструкций INVERSE, OVER, INK 9 и PAPER 9. Нечетные биты этих режимов соответствуют постоянному заданию режимов, четные - временному.

	временно	постоянно
OVER 1	0	1
INVERSE 1	2	3
INK 9	4	5
PAPER 9	6	7

Если необходимо установить постоянно OVER 1: INVERSE 1: INK 9: PAPER 9, то вместо четырех команд достаточно одной: POKE 23697,2+8+32+128.

Внимание ! Названия системных переменных не распознаются бейсиком. Они введены авторами операционной системы и приняты во всей литературе, относящейся к Spectrum.

7. Каналы и потоки

Перемещение информации между программой и периферийными устройствами управляется с помощью каналов и потоков. Желательно представлять канал как физическое устройство, получающее информацию (телевизор, принтер) или выдающее ее (клавиатура), а поток - как дорожку, по которой данные передаются к каналу или от него.

В Spectrum без ZX-Interface-1 различают 4 канала:

S - Выходной канал. Пересылаемые к этому каналу данные высвечиваются в верхней части экрана;
K - Канал I/O (ввода/вывода). Обслуживает клавиатуру и нижнюю часть экрана;
P - Выходной канал. Данные посылаются на принтер.
R - Выходной сигнал, используемый только редактором для вывода в буфер редактора данных, считанных с клавиатуры.

Данные в каналы пересылаются подключением к ним потоков. Их мы имеем в распоряжении 16 (0...15). После инициализации системы следует подключение потоков 0 и 1 к каналу "K", потока 2 - к "S", потока 3 - к "P". Канал "R" недоступен из ZX-бейсйка.

Инструкции бейсика PRINT, LPRINT, INPUT, LIST, LLIST делают возможным доступ ко всем этим каналам, полностью скрывая их существование. В действительности, необходимы только 2 из них: INPUT и LIST, а также существующая система каналов и потоков. Каждая из этих инструкций может быть использована для пересылки информации любым потоком. Для этой цели необходимо лишь символом #K сигнализировать, каким каналом мы хотели бы воспользоваться.

Инструкция LLIST равнозначна LIST #3, LIST - то же, что и LLIST #2. Можно также писать LIST #0 или LLIST #0, направляя печать в нижнюю часть экрана, но это не очень практично, т.к. система не позволяет вводить туда более 22 экранных операторов и часто сама очищает эту область. Команда PRINT #1 вызовет печать в нижней части экрана, как и команда INPUT, зато PRINT #3 перешлет данный текст на принтер. Сам PRINT равнозначен PRINT #2. Иногда может быть выгодно использование инструкции INPUT #2;#0;A$, что вызовет вывод директивы не внижней части экрана, а в верхней. #0 перед A$ необходим, т.к. канал "S" является только выходным каналом и из него нельзя читать.

В распоряжении пользователя остаются потоки с номерами от 1 до 15. Перед их использованием необходимо соответствующие потоки подключить к каналам, которые должны их обслуживать. Для этого служит инструкция OPEN #K;"L", где #K является выражением, дающим в результате одну литеру, идентифицирующую канал. В голом Spectrum должны быть K, S или P. Например, после команды OPEN #6;"P" инструкция LIST #6 будет действовать как LLIST, также и PRINT #6 может быть использован как LPRINT.

После использования данного потока, перед подключением его к другому каналу, обязательно его отключение. Для этого служит команда CLOSE #K. Хорошее правило рекомендует закрывать каналы (отключать от них потоки) сразу после их использования. Это может быть важно для периферийных устройств (позволит им, к примеру, отключать контролируемые устройства). Потоки от 0 до 3 автоматически подключаются к своим каналам, и их закрытие нецелесообразно, т.к. система подключит их опять. Попытки подключения их к другим каналам могут иметь неопределенные последствия, включая крах системы. Также нельзя закрывать потоки, которые не подключены ни к какому каналу (см. р.9). "Голый" Spectrum не дает возможности оценить достоинства такой системы перемещения информации (особенно, когда не возникает необходимости совместной работы с уровня ZX-бейсика с нетиповыми периферийными устройствами). Выявляются они только после подключения Interface-1. при работе с Microdrive или в компьютерной сети.

Важнейшей пользой, приносимой системой каналов и потоков, является ее большая гибкость, определяемая легкостью управления новыми каналами. На практике это позволяет значительно упростить конструкции интерфейсов путем размещения в памяти необходимых программ, обслуживающих данные устройства (что также снижает расходы). Включение собственных каналов в систему бейсика требует обычно двух простых и коротких процедур в машинном коде. Теоретически, для этих целей хватило бы двух инструкций - IN и OUT, но в случае устройств, требующих сигналов от единиц до десятков микросекунд, это так и остается теорией.

Для пользователей данного устройства также немаловажным является удобство написания программ. Значительно проще использовать PRINT #K или INPUT #K, чем каждый раз дописывать в ZX-бейсик процедуры, готовящие данные, а затем высылающие их по одному знаку.

Перед тем, как будут объяснены способы организации собственных каналов, мы должны знать, как Spectrum хранит необходимую информацию о них и о потоках, а также как он ими пользуется.

В области памяти, начинающейся с адреса, хранимого системной переменной CHANS #5C4F(23631), до PROG-1, размещены базовые данные о каналах. Они имеют стандартный формат. Описание канала занимает 5 байт и имеет форму:

X - Адрес выводной процедуры канала (2 байта), X+2 - Адрес вводной процедуры канала (2 байта), X+4 - Код литера канала (1 байт).

Процедура вывода будет вызываться с кодом очередного символа в регистре A. Процедура ввода, чтобы беспрепятственно взаимодействовать с ZX-бейсиком, должна поставлять коды очередных знаков, распознаваемых системой, и сигнализировать распознавание данных установкой указателя C. Отсутствие выходных данных должно сигнализироваться обнулением указателей C и Z.

Часто случается, что данное устройство является односторонним, тогда как адрес обработки некорректной операции выдает положение процедуры (обычно в ROM):

```
RST 0008
DEFB #0A
```

Код соответствующего сообщения об ошибке (содержимое ячейки, следующей за RST 8) выбирается как параметр.

После инициализации системы область информации о каналах занимает 20 байт+1 (содержащий указатель конца области (#80 128)). Они содержат:

```
CHANS  - Адрес процедуры, пишущей в нижней части экрана
+ 2    - Адрес процедуры, считывающей данные с клавиатуры
+ 4    - "K" - идентификатор канала
+ 5    - Адрес процедуры, пишущей в верхней части экрана
+ 7    - Адрес процедуры, сигнализирующей ошибку
+ 9    - "S" - идентификатор канала
+10    - Адрес процедуры, вводящей считанные данные в буфер ре-
         дактора
+12    - Адрес процедуры, сигнализирующей ошибку
+14    - "R" - идентификатор канала
+15    - Адрес процедуры, обслуживающей принтер
+17    - Адрес процедуры, сигнализирующей ошибку
+ 19   - "P" - идентификатор канала
+ 20   - #80 (128) - указатель конца области
+ 21   - Начато области PROG
```

Как видно, в этой области нет места для размещения данных о новых каналах. Из существующих для модификации может подойти только "P", т.к. остальные автоматически открываются системой. Это не является наилучшим способом, т.к. позволяет определить только один дополнительный канал, а также делает невозможным одновременное использование принтера Выгоднее сдвинуть весь блок PROG-1 до STEND на соответствующее число байт вместе с модификацией системных переменных. Проще всего это делается системной процедурой MFRT ROM. Менее элегантно, но столь же действенно, размещение информации о новых каналах где-нибудь в области системных переменных на 38 байтах, начиная с STRMS #5C10 (23568). Для каждого блока предназначается 2 байта. Адрес пяти байт, описывающих данный канал, имеет вид CHANS+X-1, где X - содержимое двух байт, связанных с данным потоком. Описание потока с номером X размещается с адреса STRMS+6+2*K.

В момент подключения потока к каналу этим ячейкам присваиваются соответствующие значения. Сколько раз в программе появится инструк-

ция INPUT #K или PRINT #K, столько раз, на основе данных из таблицы, STRMS назначается адрес соответствующей процедуры и помечается в системной переменной CURCHL #5C51 (23633). Далее, в случае пишущей инструкции, очередные символы загружаются в накопитель, и вызывается эта процедура. В случае чтения, из накопителя выбираются очередные знаки. Если они доступны (выставлен указатель C), неактивные потоки помечаются нулем в соответствующем месте таблицы STRMS.

Как видно, эта система действительно очень эластична, и подключение новых каналов не трудоемка. Проблема только в том, что инструкции OPEN и CLOSE работают лишь со стандартными идентификаторами каналов "K", "S" и "P". Подключение и отключение потоков от новых каналов (модификация данных в STRMS, а также переменной CURCHL) должны быть выполнены программой. Это требует пересылки дополнительных сигналов, инициализирующих данное устройство или информирующих его о завершении сеанса работы.

В конце - неприятная новость для обладателей ZX-Interface-1. После подключения данного устройства к ZX-Spectrum изменяются фрматы хранения данных о каналах и потоках, и приведенные выше соображения не удается применить.

8. СИСТЕМНЫЕ ПРОЦЕДУРЫ

Для программирования на ассемблере Z80 ПЗУ (ROM) Spectrum является ценным хранилищем готовых и отлаженных процедур в машинном коде. Некоторые из них могут с успехом использоваться с уровня бейсика, создавая дополнительные возможности, недоступные при другом способе. Мы ограничимся описанием важнейших системных процедур.

8.1. Работа со звуком

В ZX-Spectrum динамик можно возбудить двумя способами:

BEEPER #03B5 (949)

Эта процедура требует двух параметров: в регистре DE помещается время продолжительности звука, а в регистре HL -частота. Эти значения, перед помещением их в регистры, требуют предварительных преобразований. Допустим, мы хотим получить тон частотой F и продолжительностью T, тогда в DE загружается F * T, а в HL - 437500 / F - 30, 125. Избежать этих расчетов можно только вызывая другую процедуру.

BEEP #03FB (1016)

Требует задания времени и частоты звука в нормальных единицах измерения. Эти параметры передаются BEEP путем помещения их в стек калькулятора. BEEP снимает их оттуда самостоятельно, выполняет необходимые преобразования и вызывает BEEPER.

BEEPER немного лучше инструкции ZX-бейсика BEEP с той точки зрения, что на него не распространяются ограничения на значение параметров. Данные контролируются ео время преобразований, выполняемых BEEP. Во время генерации звука все прерывания запрещены.

8.2. Работа с магнитофоном

Как заголовки, так и блоки данных записываются на кассету одной и той же процедурой:

SAVE BYTES #04C2 (1218)

В регистрах DE должна находиться длина записываемого блока, в X - адрес первого байта, а в буфере A - тип записываемого блока: 0 - заголовок, 255 - блок данных. В принципе, они отличаются тем, что заголовку предшествует вступительный сигнал около 5 сек, а блоку данных - около 2 сек. Пользователь может использовать и промежуточные значения (1...254) на обозначение типа блока. Это полезно при обозначении разных типов данных в специальных приложениях. Такие блоки игнорируются инструкцией LOAD при чтении (на экран не выводятся о них никакие данные), а также некоторыми программами копирования.

Стандартный заголовок состоит всегда из 17 байт:

1-й байт	Тип блока данных: 0 - программа бейсика, 1 - числовой массив, 2 - знаковый массив, 3 - набор байтов.
2-11 байты	Название длиной 10 байт. Допустимы все коды от 0 до 255. Если название короче 10 байт, то оно дополняется пробелами.
12-13 байты	Длина блока, следующего за заголовком.
14-17 байты	Зависит от типа данных. Для типа 0: 14-15 - номер строки запуска. Отказ от запуска сигнализируется значением >32767 (#7FFF). 16-17 - длина самой программы без области переменных. Для типов 1 и 2 используется только 15 байт для задания однолитерного имени записываемого массива (под которым он хранился в области переменных). Для типа 3 14-15 байты - начальный адрес памяти, в которой находился массив в момент записи.

Приведенный выше формат заголовка должен сохраняться лишь для тех случаев, когда записанные блоки должны быть считаны инструкцией LOAD.

Считывание записанных наборов производится инструкцией:

```
LOAD BYTES #0556 (1566)
```

Так же, как и для SAVE BYTES, перед вызовом, в пару DE заносим длину считываемого блока, в IX - начальный адрес памяти для записи блока, а в A - тип считываемого набора. Дополнительно необходимо установить указатель с инструкцией SCF. Вызов этой процедуры с нулевым указателем позволяет отказаться от верификации данного блока (аналог команды VERIFY). Возможная ошибка считывания или верификации сигнализируется обнулением указателя C после выхода из подпрограммы. Нормальное выполнение процедуры сигнализируется установкой C = 1.

8.3. Вывод на экран и печать

Вывод на экран единичного символа, код которого находится в буфере, в верхнюю или нижнюю часть экрана, а также на принтер выполняется одной и той же процедурой. Перед ее вызовом необходимо открыть соответствующий канал с помощью процедуры:

```
CHAN OPEN #1601 (5633)
```

В буфере должен находиться признак необходимого канала: 1 - для "K", 2 - для "S", 3 - для "P". Этот канал будет открыт до тех пор, пока мы его сами не закроем. Если после этого выдать команду ассемблера RST #10, то она будет печатать единичный символ, находящийся в буфере A, по выбранному каналу.

Программа, приведенная ниже, иллюстрирует применение RST #10. Ее выполнение равнозначно команде: PRINT FLASH 1; AT 5,3;"X";#3;A.

```
LD    A,2                ;открыть
CALL  CHAN_OPEN          ;канал "S"
LD    A,#12              ;код символа FLASH
RST   #10
LD    A,1                ;аргумент FLASH
RST   #10
LD    A,#16              ;код символа AT
RST   #10
LD    A,3
RST   #10
LD    A,#58              ;код X
RST   #10
LD    A,3                ;открыть
```

```
CALL   CHAN_OPEN          ;канал "P"
LD     A,#41              ;код символа "A"
RST    #10
RET                       ;выход из подпрограммы
```

Обратим внимание, что знак "A" будет послан в буфер принтера. Фактически он отпечатается на бумаге только после заполнения буфера, пересылки в буфер конца строки или вызова процедуры:

```
COPY BUF #0ECD (3789)
```

Этот пример показывает, что пересылка цепочки знаков по одному символу может быть крайне утомительной. Проще применить процедуру:

```
PR STRING #2036 (8252)
```

Она печатает цепочку знков, адрес которой задан в регистрах DE, а длина - BC. Перед ее вызовом необходимо открыть соответствующий канал. Если печатаемые символы не уточняют цвета, то он устанавливается на основании переменных ATTR_T, MASK_T, а также нечетных битов P FLAG.

Печать чисел более затруднительна, т.к. необходимо осуществлять перевод двоичного числа в последовательность символов его десятичного представления. Все необходимые вычисления и печать выполняет процедура:

```
PRINT FP #2DE3 (11747)
```

Она снимает со стека калькулятора 5 байтов, считая их числами в формате, принятом в системе ZX-бейсика. Затем печатает их с учетом системных переменных, определяющих цвет, положение и т.д. Способы размещения чисел на стеке калькулятора мы оговорим далее.

В случае натуральных чисел, от 0 до 9999, можно использовать более быструю процедуру:

```
OUT_NUM1 #1A1B (6683)
```

Она печатает число, содержащееся в BC, на четырех полях, выводя в начале необходимое число пробелов. Spectrum использует эту процедуру для печати номеров строк в листингах программ.

8.4. Экранная графика

Для рисования на экране у нас есть аналоги процедур PLOT, DRAW и CIRCLE:

```
PL0T_SUB #22E5 (8933)
```

Позволяет вывести на экран единичную точку с координатами (X,Y). Перед вызовом необходимо поместить X в C, Y - в B.

PIXEL_ADD #22AA (8874)

После занесения в BC значений (Y,X) и вызова этой функции мы получаем в регистрах HL адрес байта, описывающего данную точку экрана. К тому же, в буфер заносится значение X MOD 8, уточняющее о каком бите данного блока идет речь.

DRAW_1 #2477 (9335)

Снимает со стека калькулятора число X и Y, после чего рисует соответствующий отрезок. Координаты PLOT выбираются из системных переменных.

DRAW_3 #24BA (9402)

Для рисования аналогичного отрезка этой процедурой необходимо задать ABS Y-> B, ABS Y-> C, SGN Y-> D, SGN X-> E.

DRAW_ARC #2394 (9108)

Рисует фрагменты дуг. Параметры X, Y, Z для этой процедуры должны передаваться через стек калькулятора. Z размещается наверху стека.

CIRCLE_1 #2320 (9005)

Рисует полную окружность с центром в (X,Y) и радиусом Z. Все параметры должны быть помещены в стек калькулятора.

ВНИМАНИЕ ! Приведенные выше процедуры модифицируют регистры HL (см. p.9).

После вывода нужного рисунка на экран, его можно вывести на принтер. Аналогом инструкции COPY является процедура:

COPY #0EAC (3756).

Она копирует на принтер 22 верхние строки. Параметры всех вышеприведенных процедур подчинены тем же ограничениям, что и соответствующие команды в бейсике.

8.5. Очистка и перемещение экрана

CLS #0D6B (3435)

Основная процедура очистки экрана. Ее действие аналогично директиве CLS.

CLS_LOWER #0D6E (3438)

Очистка нижней части экрана. Она действует на всей нижней части экрана, независимо от ее текущих размеров, и одновременно устанавливает ее высоту в две строки. Инициализируются также системные переменные DF_CL и SPOSNL, определяющие положение курсора.

CL_LINE #0E44 (3652)

Очистка, начиная с нижнего края экрана, заданного числа строк, которое содержится в регистре E.

Перед вызовом этих процедур нужно убедиться в том, что необходимые каналы открыты. Две первые из них оставляют после выхода канал "K" открытым. При стирании экрана цвета устанавливаются на основании системных переменных: BORDER для нижней части экрана, а также ATTR P и MASK P - для верхней, после чего их содержимое копируется в ATTR_T и MASK_T.

CL_SC_ALL #ODFE (3582)

Перемещает содержимое всего экрана (24 полные строки) на одну строку вверх (верхняя строка исчезает). Ее можно вызвать непосредственно из бейсика, т.к. она не требует параметров.

CL_SCROLL #0E00 (3584)

После занесения в регистр B числа строк -1 для сдвига (не менее 2) эта процедура продвинет на одну строку столько строк, сколько хотим, не трогая расположенных выше. Следовательно, можно иметь вверху экрана рисунок или текст, которые не уничтожаются при сдвиге на одну строку.

Применяя эти процедуры надо помнить, что вверх будут подниматься и обязательные атрибуты в нижней части экрана.

8.6. Считывание с клавиатуры

Информацию с клавиатуры выгоднее всего снимать с системной переменной LAST_K. Проверяя состояние пятого бита переменной FLAGS, можно определить, нажата или не нажата очередная клавиша (1 - нажата новая, 0 - еще не нажата ни одна с момента обнуления этого бита). После считывания кода клавиши, обнуляя этот бит, мы обеспечиваем себе возможность знать, модифицировалось ли еще раз или нет содержимое LAST_K. Действует это только при включенных замаскированных прерываниях в режиме 1. Именно в этом случае Spectrum автоматически вызывает эту процедуру 50 раз в секунду:

KEYBOARD #02BF (703).

Она опрашивает всю клавиатуру, идентифицирует верную комбинацию клавиш, заносит считанный код в буфер и в переменную LAST_K и устанавливает пятый бит FLAGS. Интерпретация нажатия будет зависеть от трех системных переменных. Сначала проверяется содержимое MODE, рассматриваемое как число со знаком в коде дополнения до двух:

MODE-1 <0 - курсор **L** или **C**;
MODE-1 =0 - курсор **E**;
MODE-1 >0 - курсор **G**.

Курсор **K** отличается от **L** и **C** с помощью третьего бита переменной FLAGS: 0 означает **K**, 1 - **L** или **C**. Третий бит FLAGS окончательно определяет, является курсор **L** (0) или **C** (1). Эта процедура не ожидает нажатия клавиши.

Чтобы независимо от состояния курсора уточнить, была ли нажата конкретная клавиша, более продуктивным может быть непосредственный опрос клавиатуры с помощью директивы IN. Здесь мы поступаем так же, как и в случае функции с тем же самым названием в бейсике.

8.7. Калькулятор

Различные действия над числами с плавающей запятой выполняются с помощью большого набора процедур, занимающих пространство ROM от #2F9B (12187) до #386D (14445). Калькулятор инструкцией RST #28, которая выполняет переход на адрес #355B (13659).

Основой рабсты калькулятора являются 66 различных процедур, выполняющих набор базовых функций на стеке калькулятора. Очередность их выполнения задается цепочкой байт, размещенных непосредственно за командой RST #28. Конец такой цепочки всегда помечается байтом со значением #38 (56).

Мы ограничимся операциями, позволяющими выполнять различные численные расчеты. Допустим, что наверху стека находятся числа... X, Y, Z.

значение байта 10-ное	16-ное	Операция	Состояние стека после операции
1	#01	Замена элементов	... Z Y X
3	#03	Вычитание	... Z X-Y
4	#04	Умножение	... Z X*Y
5	#05	Деление	... Z X/Y
6	#06	Степень	... Z X**Y
15	#0F	Сложение	... Z X+Y
27	#1B	Изменение знака	... Z X -Y
31	#1F	Синус	... Z X SIN Y
32	#20	Косинус	... Z X COS Y
33	#21	Тангенс	... Z X TG Y
34	#22	Арксинус	... Z X ASN Y
35	#23	Арккосинус	... Z X ACS Y
36	#24	Арктангенс	... Z X ATG Y
37	#25	Логарифм натур.	... Z X LN Y
38	#26	Экспонента	... Z X EXP Y
39	#27	Целая часть числа	... Z X INT Y
40	#28	Корень квадратный	... Z X SQR Y
41	#29	Знак числа	... Z X SGN Y
42	#2A	Абсол. величина	... Z X ABS Y

49	#31	Копирование стека	...	Z	X	Y	Y
50	#32	N MOD M	...	Z	остаток	частн.	
52	#34	Дописать в стек	...	Z	X	Y	D
56	#38	Конец расчетов	...	Z	X	Y	
58	#3A	INT (Y+0.5)	...	Z	X	INT(Y+0.5)	
160	#A0	Дозапись	...	Z	X	Y	0
161	#A1	Дозапись	...	Z	X	Y	1
162	#A2	Дозапись	...	Z	X	Y	0.5
163	#A3	Дозапись	...	Z	X	Y	PI/2
164	#A4	Дозапись	...	Z	X	Y	10

Как пример использования калькулятора, расчитаем значение 1.5*SIN (X)+X**2*COS (X*PI/2). Допустим, что значение X находится наверху стека. Размещение за инструкцией RST #28 следующих байт первой колонки вызовет:

Байт	Состояние стека
#31	X X
#31	X X X
#31	X X X X
#A3	X X X X PI/2
#04	X X X X*PI/2
#20	X X X COS (X*PI/2)
#04	X X X*COS (X*PI/2)
#20	X X*X*COS (X*PI/2)
#01	X*X*COS (X*PI/2) X
#1F	X*X*COS (X*PI/2) SIN X
#34	Дозапись 1.3 (5 байт)
#F1	
#26	
#66	
#66	
#66	X*X*COS (X*PI/2) SIN X 1.3
#04	X*X*COS (X*PI/2) 1.3*SIN X
#0F	X*X*COS (X*PI/2)+1.3*SIN X
#38	Конец вычислений

Объяснения требует запись в стек калькулятора данных, как последовательности байт, следующих непосредственно за #34. Байты #F1, #26, #66, #66, #66 представляют число 2**113*0.13, а не 0.13. Это связано с интерпретацией их как числа в укороченной записи. Схема действий следующая:

- Первый байт делим на #40 (64), и как значение показателя степени, принимаем остаток от этого деления плюс #50
- Целая часть от деления (0, 1, 2, 3) плюс 1 определяет, сколько затребовано байт для мантиссы, недостающие до 5 байт дополняются нулями.

В нашем примере #F1 (241), деленное на #40 (64), дает остаток

#31 (49), а также целую часть 3. Это означает, что показателем степени нашего числа является #31+#50=#81, и что затребованы все четыре байта мантиссы. В этой системе число 0 имеет представление #40,#80,#00, т.к. 0, деленный на #40, дает остаток и целую часть числа, равную 0. Затем становится второй байт +#50 или #80+#50=#00 (все эти расчеты на отдельных байтах ведутся по модулю 256) и задается лишь первый байт мантиссы. Остальные байты (до 5) дополняются нулями. В свою очередь, число 10 имеет представление #40, #80, #0A Символ #34 позволяет размешать числа в тексте программ на ассемблере. Однако часто требуется поместить в стек параметры инструкции или значения, расчитанные в программе. Для этого предназначено много вспомогательных процедур. Они позволяют как извлечение из стека, так и запись в него различных чисел:

STK_TO_BC #2307 (8967)

Два следующих за собой числа (в 5-байтовом представлении) извлекаются из стека и заносятся в регистры B и C. Их значения должны лежать в диапазоне от -255 до 255, иначе осцшествляется возврат в бейсик с сообщением B (будет выполнена инструкция RST #8). Эта процедура может применяться к отрицательным числам. Их знаки возвращаются в регистрах D и E. Числа, извлекаемые из стека, округляются до ближайшего целого числа.

STK_TOA #2514 (8980)

Процедура аналогична предыдущей, с тем отличием, что из стека извлекается только одно число и, после округления до целого, помещается в буфер. Знак числа возвращается в регистре C.

STACK_FETCH #2BF1 (11249)

Эта процедура извлекает из стека все число (5 байтов) и размещает его в регистрах A, E, D, C, B.

FP_TO_BC #2DA2 (11682)

Число из стека округляется до ближайшего целого и размещается в регистрах BC. Знак числа определяется указателем Z (0 - для отрицательных). Если значение в стеке превысило максимальное (65535), то указатель устанавливается в 1, и это единственная реакция на ошибку (возврата в ZX-бейсик не происходит).

STK_STORE #2AB6 (10934)

Эта процедура размещает наверху стека 5 байтов из регистров A, E, D, C, B. Число 1.3 (#81 #26 #66 #66 #66) можно занести в стек двумя способами (обратим внимание на то, что второй способ позволяет сэкономить 3 байта):

```
LD   A,#81        или     RST  #28
```

```
LD   DE,#6626          DEFB #34
LD   BC,#6666          DEFB #F1
CALL #2AB6             DEFB #26
                       DEFB #66
                       DEFB #66
                       DEFB #66
                       DEFB #3B
```

STACK_A #2D28 (11560)

Значение регистра A периодически заносится в стек в пятибайтовом представлении.

STACK_BC #2D2B (11563)

Эта процедура аналогична предыдущей, но в стек заносится число с регистров BC.

После завершения расчетов калькулятор в регистрах HL размещает адрес первого из пяти байтов, находящихся в верхушке стека. При работе с калькулятором необходимо заботиться о соответствующей работе со стеком. Следует также помнить, что процедура PRINT_FP снимает со стека печатаемое число.

В конце приводим вызов из программы в машинном коде генератора псевдослучайных чисел:

```
LD   A,#A5
CALL STACK_A
RST  #28
DEFB #2F
DEFB #1D
DEFB #38
RET
```

Эта программа помещает очередное псевдослучайное число в вершину стека и модифицирует системную переменную SEED.

8.8. Дополнительные системные процедуры

SET_MIN #16B0 (5808)

Производит глобальную очистку рабочих областей системы бейсик, а также стека калькулятора. Она модифицирует системные переменные, указывающие на соответствующие области памяти. Физически затираются только те байты, в которые заносятся указатели конца области.

MAKE_ROOM #1655 (5717)

Вызывая ее, следует в регистрах HL дать адрес байта начала добавочного блока, а в BC - размер блока. Эта процедура сама устанавливает, какие системные переменные должны быть модифицированы, и при необходимости модифицирует их. Следовательно, она допускает вставку в существующую программу на бейсике или в область переменных новых операторов, переменных и т.д.

RECLAIM_2 #19E8 (8168)

Обратная предыдущей функция. Здесь в HL заносится адрес первого байта, который необходимо убрать, а в BC - размер стираемого блока.

CLEAR_BUFF #0EEF (8815)

Процедура очищает буфер принтера и модифицирует связанные с ним системные переменные.

LINE_ADDR #196E (6510)

Отыскание строки с заданным номером в программе на бейсике. Перед вызовом в HL заносится номер разыскиваемой строки. На выходе в HL имеем адрес этой строки или первой с большим номером. Если строка с данным номером есть, то это сигнализируется установкой указателя Z.

FREE_MEM #1F1A (7962)

Определение свободной памяти в области бейсика, т.е. между STKEND и RAMTOP. Регистры HL и BC содержат то же самое отрицательное число, представленное в коде дополнения до 2, являющееся разницей между STKEND +80 и SP (указатель стека процессора Z80).

BREAK_KEY #1F54 (8020)

Ее вызывают для проверки одновременного нажатия клавиши CS и BREAK. Если они нажаты, то указатель C обнулен.

9. ОШИБКИ В СИСТЕМЕ

Авторы операционной системы и интерпретатора бейсика в ZX-Spectrum выполнили превосходную работу, но не смогли уберечься от нескольких ошибок.

9.1. Ошибка деления

Под адресом Д3200 помечено значение #E1 вместо #DA. В результате, иногда теряется последний бит, что приводит к ошибочным округлениям. Последствия ошибки демонстрирует программа:

```
10 LET A=A/B
20 IF A THEN GO TO 10
30 PRINT "ПОЛУЧЕН 0"
```

Запуская ее со значениями A=1 и B=3, через секунду получим печать "ПОЛУЧЕН 0". Запуская ее заново со значениями A=1 и B=2, зацикливаемся, так как Spectrum считает, что 2^-128 = 2^-128/2.

9.2. Ошибка "-65536"

Авторы допустили неточность в представлении этого числа Один раз оно хранится в представлении с плавающей запятой, другой - как целое число в коде представления до 2. Последствия неоднократно появляются, к примеру, при выполнении директивы PRINT -65536, на экране появится число -1.

9.3. Ошибка CHR$ 8

Этот управляющий символ должен перемещать курсор на одну позицию влево или на конец предыдущей строки. Именно так и происходит в строках 1...23, но с начала первой строки на конец нулевой перейти невозможно. К любопытным эффектам приводит просьба сдвинуть курсор влево от поля (0,0).

9.4. Ошибка CHR$ 9

Этот символ должен сдвигать курсор вправо на одну позицию. Здесь, однако, допущена серьезная ошибка: все необходимые расчеты выполняются, но авторы забыли модифицировать системные переменные.

9.5. Ошибка "Press any key..."

В некоторых ситуациях Spectrum прерывает работу и вдет толчка нажатием любой клавиши пользователем. Ошибка ведет к тому, что компьютер не реагирует на клавишу CS и на SS, хотя на обе, нажатые сразу, реагирует.

9.6. Ошибка указателя бегущей строки

Допустим, что последняя строка в программе имеет номер 1000. При нажатии 1001 и <Enter>, а затем CS/1 - в нижнюю часть экрана будет скопирована строка 1000, но вместе с указателем бегущей строки, который перед записью строки необходимо удалять.

9.7. Ошибка DELETE

При удалении содержимого нижней части экрана с помощью CS/1 - в нижнюю часть экрана сканируется текущая строка программы, и невозможен возврат к контрольному размеру этой области. Требуется вновь нажать <Enter>.

9.8. Ошибка ведущих пробелов

Некоторые ключевые слова во время высвечивания не всегда отделяются пробелами от предшествующих. Попробуйте, например, выполнить PRINT CHR$ 255; CHR$ 13; CHR$ 255.

9.9. Ошибка режима K

После нажатия клавиши в режиме **K** и удерживании в этом положении клавиша начинает размножаться. Курсор меняется на **L** или **C**, но все время печатается символ в режиме **K**.

9.10. Ошибка SCREEN$

В ячейке #2570 должно находиться #C9 вместо #C3. В результате программа вместо ожидаемого числа 12 выведет 22:

```
10 PRINT "1234567890"
20 LET A$=SCREEN$ (0,0)+SCREEN$ (0,1)
30 PRINT A$
```

Еще более удивительную печать мы получим, если в конце строки 20 допишем +SCREEN$ (0,2)+SCREEN$ (0,4). Переменная примет значение 55. Эту ошибку легко обойти, добавляя к A$ значение SCREEN$ (0,1) по очереди, а не одним выражением.

9.11. Ошибка STR$

Работая с числами -1<X<1,но не равными 0, можно "налететь" на необычную "прожорливость" команды STR$. Попробуйте выполнить следующие директивы:

```
PRINT "ALA"+"BUM CUK CUK"+ STR$ .001   или
PRINT 7+ VAL STR$ .001
```

В обоих случаях на экране будет только .001. Вина лежит на Spectrum.

9.12. Ошибка CLOSE

Попытка отключения потока 4...15 от канала, до его подключения, ведет к непредвиденным эффектам, с рестартом системы включительно. А все, потому что в ROM, в таблице, содержащей данные о каналах, с адресом #1716, забыли разместить указатель конца таблицы.

9.13. Ошибка RET

Одна из труднейших в локализации ошибок. Проявляется иногда в ZX-бейсик из программы в машинном коде, вызываемой через USR K. Забыто перед возвратом в интерпретатор восстановление пары регистров HL. Если их использовала ваша программа и модифицировала, то то, что произойдет при возврате в бейсик, дело случая. Обычно наступает зависание системы, но не всегда.

9.14. Ошибка NMI

Это - одна из серьезных ошибок. Переключение одного бита по адресу #006D вызвало то, что Spectrum не в состоянии принимать обслуживающими программами немаскированные прерывания от процедур пользователя. Такие прерывания либо могут игнорироваться, либо рестартовать систему переходом по адресу 0. Это делает невозможным контролируемый рестарт системы после ее сбоя, также и реализацию других возможностей компьютера Устранение этой ошибки требует дорогостоящих доработок.

9.15. Ошибка PAUSE N

Проблема с инструкцией PAUSE заключается в том, что она не всегда выполняется. Ошибка в процедуре, обслуживающей дублирование клавиши. Непосредственно перед выполнением инструкции PAUSE она игнорируется. Например:

```
10 PRINT "Отпусти клавишу, когда услышишь звук"
20 FOR I=1 TO 500: NEXT I
30 BEEP 1,10
40 PAUSE 0
50 PRINT "Конец"
```

Поведение программы даже не намекает на существование в ней строки 40.

9.16. Ошибка CLS

При значениях системной переменной DF_SZ<2 трудно воспринять результат этой команды за очистку экрана. Для DF_SZ=1 это можно считать несущественным, но при DF_SZ=0 это уже трагедия.

Надеемся, что приведенный перечень исчерпывает "неожиданности" при программировании на ZX-Spectrum.