Программ-Асс Компьютер дома

Ян Логан
Френк О'Хара

Полное описание ПЗУ ZX-Spectrum

Программ-АСС

The Complete SPECTRUM ROM DISASSEMBLY

BY

Dr Ian Logan & Dr Frank O'Hara

Melbourne House Publishers 1983

Ян Логан и Френк О'Хара

Полное описание ПЗУ

ZX-Spectrum

(перевод с английского)

"Программ-Асс"

Харьков, 1992

Восстановлено в 2011 году,

http://zx.pk.ru/showthread.php?t=14994

ПОЛНОЕ ОПИСАНИЕ ПЗУ КОМПЬЮТЕРА ZX SPECTRUM

THE COMPLETE SPECTRUM ROM DISSASSEMBLY

LOGAN J, O'HARA F.

MELBOURNE HOUSE PUBLISHERS, 1983

ВВЕДЕНИЕ

Монитор Spectrum, объемом 16K, представляет собой сложную программу в машинных кодах Z80. Ее можно разделить на три основные части:
  а. Программы ввода/вывода.
  б. Интерпретатор BASIC.
  в. Вычислительные процедуры.
Однако, для подробного описания эти блоки слишком громоздки, и поэтому монитор разделен еще на 10 частей. Каждая часть будет представлять собой элемент монитора.

ПРОГРАММЫ ИНИЦИАЛИЗАЦИИ И ТАБЛИЦЫ
При старте монитора работают все программы-рестарты, которые выдаются однобайтовыми командами 'RST'. Используются все 'рестарты'. Например, 'рестарт 0008' используется для сообщения об ошибках синтаксиса или ошибках исполняющей системы.

Таблицы в этой части монитора содержат расширенные формы токенов и коды клавиш.

ПРОГРАММЫ РАБОТЫ С КЛАВИАТУРОЙ
Обращение к клавиатуре происходит 50 раз в секунду (U.K. модель) с последующим возвратом требуемого символьного кода. При постоянном нажатии - клавиши 'повторяются', что учитывается программой работы с клавиатурой.

ПРОГРАММЫ РАБОТЫ С ДИНАМИКОМ
В Spectrum встроен один динамик, звук которого создается с помощью повторяемых соответствующих команд 'OUT'. В контроллерной программе большое значение уделяется обеспечению звучания на заданной ноте при соответствующей длительности.

ПРОГРАММЫ ОБРАБОТКИ ИНФОРМАЦИИ С КАССЕТНЫХ ЛЕНТ
Для ZX81 было неудачным то, что только небольшая часть монитора была предназначена для работы с кассетами. Однако, в Spectrum'е для этого есть большой блок программ, с помощью которого достигается высокий уровень работы с кассетами, что является одним из наиболее удачных свойств машины. С помощью блока 'ЗАГОЛОВОК' (17 байт), который записывается <SAVE> первым, обрабатываются программы <BASIC> или блоки данных. Этот 'заголовок' описывает 'блок данных', записанный после него. Недостатком данной системы является невозможность создания программ с 'защитой'.

ПРОГРАММЫ ОБРАЩЕНИЯ К ЭКРАНУ И ПРИНТЕРУ
Все оставшиеся программы ввода/вывода Spectrum проходят через 'информационные каналы'. В стандартном Spectrum-е 'ввод' возможен только через клавиатуру, но 'вывод' можно осуществить через принтер, или на верхнюю или нижнюю часть экрана телевизора. Основной программой 'ввода' в этой части монитора является EDITOR, который дает возможность пользователю вводить символы в нижнюю часть экрана телевизора. Программа <PRINT-OUT> является довольно медленной, т.к. это - программа на 'все случаи жизни'. Например, добавление одного байта в 'область дисплея' предполагает в каждом случае просмотр текущего состояния OVER и INVERSE.

ПРОГРАММЫ ВЫПОЛНЕНИЯ КОМАНД
В этой части монитора находится процедура INITIALISATION и 'основной цикл выполнения' интерпретатора BASIC.
В Spectrum-е строка BASIC, возвращаемая EDITOR-ом проверяется на правильность синтаксиса и затем записывается в программную область, если строка начинается номером строки, или в противном случае 'выполняется'.
Это выполнение может в свою очередь привести к проверке следующего оператора. (Это ясно видно на примере RUN).

ИНТЕРПРЕТАЦИЯ СТРОК И КОМАНД БЕЙСИКА
Эта часть монитора рассматривает BASIC-строку как набор операторов и в свою очередь каждый оператор как начало конкретной команды.
Для каждой команды существует 'командная процедура' и выполнение машинного кода в соответствующей 'командной процедуре' воздействует на 'интерпретацию'.

РАСЧЕТ ВЫРАЖЕНИЙ
В Spectrum-е имеется большой блок вычислений, позволяющий работать с широким диапазоном типов переменных, функций и операций. Опять же это довольно медленная часть монитора, т.к. рассматриваются все возможные альтернативы.
Обработка строк частично управляема. Все простые строки управляются 'динамически' и старые записи исправляются после того, как они были зарезервированы. Это означает, что не происходит 'сбор ненужной информации'.

АРИФМЕТИЧЕСКИЕ ПРОЦЕДУРЫ
Spectrum имеет две формы представления чисел. Целые числа в диапазоне от -65535 до +65535 представляются в 'целостной' или 'короткой' формах, в то время как все остальные числа в форме с плавающей точкой размером в пять байт.
Представленная версия монитора обладает двумя ошибками в данной части.

1. Ошибка деления, при которой во время деления теряется 34-й разряд.
2. Величина -65536 иногда берется в 'короткой' форме, а иногда с 'плавающей точкой', что приводит к некоторым неудобствам.

КАЛЬКУЛЯТОР С ПЛАВАЮЩЕЙ ТОЧКОЙ
КАЛЬКУЛЯТОР обрабатывает числа и строки, а также команды,

задаваемые 'литералами'. Поэтому можно считать, что в КАЛЬКУЛЯТОРе имеется внутренний 'стековый операционный' язык.
Вычисления SIN X, EXP X, LN X & ATN X осуществляются с помощью полиномов Чебышева. Подробности приведены в приложении.

В целом 16К байтный монитор предлагает широкий диапазон различных команд и функций BASIC. Поскольку программистам всегда не хватает 'места', то программы написаны с точки зрения компактности, а не быстродействия.

От издателя:
1) Необходимо отметить, что в СНГ существует множество реализаций знаменитого ZX-Spectrum (как промышленных, так и самодельных). Все они отличаются друг от друга не только конструктивными особенностями, но и ПЗУ. Поэтому у Вас могут возникнуть некоторые недоразумения при освоении этой книги. По этой же причине не работают и некоторые фирменные программы. Однако эти отличия ПЗУ от фирменного обычно незначительны и степень совместимости таких ПК довольна высока. В основном изменения в ПЗУ вносятся при попытках адаптации под различ-ные национальные шрифты.
2) Вы наверняка заметили, что в книгах, содержащих примеры небольших программ, обязательно находятся ошибки в их текстах (обычно это выяс-няется, когда программа введена в компьютер). Естественно, в книге, полностью состоящей из текстов программ, не сделать ошибок было практически невозможно. Надеемся, что Вы будете благосклонны к нам и простите нас за допущенные в тексте и программах опечатки. Мы будем особо благодарны читателем, приславшим свои отзывы и замеченные в тексте книги неточности. Английское фирменное издание также содержит ошибки и поэтому ориентироваться на него нельзя (там ведь тоже люди работают). Практика - критерий истины. Для нас это очень важно, т.к. сейчас мы готовим эту книгу к более массовому изданию (Вы держите в руках книгу из пробного тиража) и хотим избежать данного недостатка. Помогите своим коллегам.

Наш адрес: 310085 Харьков-85 а/я 9207 НПФ "Программ-Асс"

Обратившись с письмом по этому же адресу, Вы можете приобрести также и другую литературу как оптом, так и в розницу. Ждем также предложений от книготорговых организаций на оптовую закупку литературы. У нас Вы можете приобрести уникальную литературу по ZX-Spectrum!

ПРОГРАММЫ ИНИЦИАЛИЗАЦИИ И ТАБЛИЦЫ

'START' ('Старт')
Блокируется маскируемое прерывание и задается регистровая пара DE, содержащая 'вершину возможного ОЗУ'.

```
0000 START        DI                      Блокировка 'прерывания клавиатуры'.
                  XOR   A                 +00 для старта (но +FF для 'NEW').
                  LD    DE,+FFFF          Вершина возможного ПЗУ.
                  JP    11CB,START/NEW    Переход вперед на START-NEW.
```

РЕСТАРТ 'ERROR' ('Ошибка')
Для указания позиций ошибок создается указатель ошибок.

```
0008 ERROR-1      LD    HL,(CH-ADD)       Адрес, достигнутый интерпретатором,
                  LD    (X-PTR),HL        перед продолжением копируется
                  JP    0053,ERROR-2      в указатель ошибок.
```

РЕСТАРТ 'PRINT A CHARACTER' ('Печать символа')
Регистр A содержит коды подлежащего печати символа.

```
0010 PRINT-A-1    JP    15F2,PRINT-A-2    Немедленный переход вперед.
                  DEFB  +FF,+FF,+FF,+FF,+FF Неиспользуемые ячейки
```

РЕСТАРТ 'COLLECT CHARACTER' ('Выбор символа')
Выбирается содержимое ячейки, адресованной в настоящий момент с помощью CH-ADD. Осуществляется возврат, если значение представляет подлежащий печати символ, в противном случае, CH-ADD увеличивается и проверки повторяются.

```
0018 GET-CHAR     LD    HL,(CH-ADD)       Выбор значения, адресуемого
                  LD    A,(HL)            CH-ADD.
001C TEST-CHAR    CALL  007D,SKIP-OVER    Определить, подлежит ли символ печати.
                  RET   NC                Если да, то возврат.
```

РЕСТАРТ 'COLLECT NEXT CHARACTER' ('Выбор следующего символа')
Пока идет интерпретация строки BASIC, эта программа неоднократно вызывается для просмотра строки.

```
0020 NEXT-CHAR    CALL  0074,CH-ADD+1     Необходимо увеличивать CH-ADD
                  JR    001C,TEST-CHAR    Переход назад для проверки
                                          нового значения.
                  DEFB  +FF,+FF,+FF       Неиспользуемые ячейки.
```

РЕСТАРТ 'CALCULATOR' ('Калькулятор')
Калькулятор с плавающей точкой. Форт-подобный язык.

```
0028 FP-CALC      JP    335B,CALCULATE      Немедленный переход вперед.
                  DEFB  +FF,+FF,+FF,+FF,+FF Неиспользуемые ячейки.
```

РЕСТАРТ 'MAKE BC SPACES' ('Создание места')
Эта программа создает свободные ячейки для рабочей области и стека калькулятора. Количество ячеек определяется текущим содержимым регистровой пары BC.

```
0030 BC-SPACES    PUSH  BC                  Записать 'количество'.
                  LD    HL,(WORKSP)         Выбрать текущий адрес начала
                  PUSH  HL                  рабочей области и записать
                  JP    169E,RESERVE        его перед продолжением.
```

ПРОГРАММА 'MASKABLE INTERRUPT' ('Маскируемое прерывание')
Всякий раз при появлении маскируемого прерывания увеличиваются часы реального времени и просматривается клавиатура.

```
0038 MASK-INT     PUSH  AF                  Запись текущих значений, содержащихся
                  PUSH  HL                  в этих регистрах.
                  LD    HL,(FRAMES)         Два младших байта счетчика
                  INC   HL                  данных увеличиваются каждые
                  LD    (FRAMES),HL         20 ms. (U.K.) Старший
                  LD    A,H                 байт счетчика увеличивается
                  OR    L                   только тогда, когда два
                  JR    NZ,0048,KEY-INT     младших байта равны
                  INC   (FRAMES-3)          нулю.
0048 KEY-INT      PUSH  BC                  Записать текущее значение,
                  PUSH  DE                  содержащееся в этих регистрах.
                  CALL  02BF,KEYBOARD       Теперь просмотр клавиатуры.
                  POP   DE                  Восстановить значение.
                  POP   BC
                  POP   HL
                  POP   AF
                  EI                        Маскируемое прерывание
                  RET                       возможно перед возвратом.
```

ПРОГРАММА 'ERROR-2' ('Ошибка-2')
Адрес возврата к интерпретатору указывает на 'DEFB', которое означает появление какой-либо ошибки. Это 'DEFB' выбирается и передается в ERR-NR. Машинный стек перед переходом вперед для очистки стека калькулятора очищается сам.

```
0053 ERROR-2      POP   HL                  Адрес на стеке указывает
                  LD    L,(HL)              на код ошибки.
0055 ERROR-3      LD    (ERR-NR),L          Передано в ERR-NR.
                  LD    SP,(ERR-SP)         Перед выходом через
                  JP    16C5,SET-STK        SET-STK машина очищается.
                  DEFB  +FF,+FF,+FF,+FF     Неиспользуемые ячейки.
                  DEFB  +FF,+FF,+FF
```

ПРОГРАММА 'NON-MASKABLE INTERRUPT' ('Немаскируемое прерывание')
Эта программа в стандартном Spectrum не используется, но допускается код для сброса системы с последующей активизацией строки

NMI (немаскируемое прерывание). Системная переменная в 5CB0, называемая NMIADD, для осуществления сброса должна иметь нулевое значение.

```
0066 RESET        PUSH  AF                 Запись текущего значения, содержащегося
                  PUSH  HL                 в этих регистрах.
                  LD    HL,(NMIADD)        Два байта NMIADD
                  LD    A,H                должны быть нулевыми, чтобы
                  OR    L                  произошел сброс.
                  JR    NZ,0070,NO-RESET   Примечание: Это должно
                                           быть 'JR Z'!
                  JP    (HL)               Переход на START.
0070 NO-RESET     POP   HL                 В этих регистрах восстанавливаются текущие
                  POP   AF                 значения
                  RETN                     и возврат.
```

ПОДПРОГРАММА 'CH-ADD+1'
Выбирается, увеличивается и восстанавливается адрес, содержащийся в CH-ADD. Выбирается содержимое ячейки, адресованной CH-ADD. Точки входа TEMP-PTR1 и TEMP-PTR2 используются для временной установки CH-ADD.

```
0074 CH-ADD+1     LD    HL,(CH-ADD)        Выбор адреса.
0077 TEMP-PTR1    INC   HL                 Увеличить указатель.
0078 TEMP-PTR2    LD    (CH-ADD),HL        Установить CH-ADD.
                  LD    A,(HL)             Выбрать адресованное значение, затем
                  RET                      возврат.
```

ПОДПРОГРАММА 'SKIP-OVER'
Проверяется, можно ли напечатать привнесенное в регистр A значение. Различные специальные коды вводятся в HL, перед этим единожды или дважды увеличенные, и в соответствии с этим изменяется CH-ADD.

```
007D SKIP-OVER    CP    +21                Возврат со сброшенным признаком
                  RET   NC                 переноса, если обычный код символа.
                  CP    +0D                Возврат, если достигнут
                  RET   Z                  конец строки.
                  CP    +10                Возврат с кодами +00 - +0F,
                  RET   C                  но со сброшенным переносом.
                  CP    +18                Возврат с кодами +18 - +20
                  CCF                      и опять со сброшенным переносом.
                  RET   C
                  INC   HL                 Прыгнуть один раз.
                  CP    +16                Переход вперед с кодами +10
                  JR    C,0090,SKIPS       до +15 (INK - OVER).
                  INC   HL                 Еще один прыжок (AT и TAB).
0090 SKIPS        SCF                      Возврат со сброшенным признаком
                  LD    (CH-ADD),HL        переноса и CH-ADD,
                  RET                      содержащим соответствующий адрес.
```

ТАБЛИЦЫ ТОКЕНОВ
При обращении к этой таблице расширяются все токены, используемые в Spectrum. Последний код каждого токена 'инвертируется' с помощью его 7-го разряда.

```
0095  BF 52 4E C4 49 4E 4B 45          '?' R  N 'D' I  N  K  E
009D  59 A4 50 C9 46 CE 50 4F           Y '$' P 'I' F 'N' P  O
00A5  49 4E D4 53 43 52 45 45           I  N 'T' S  C  R  E  E
00AD  4E A4 41 54 54 D2 41 D4           N '$' A  T  T 'R' A 'T'
00B5  54 41 C2 56 41 4C A4 43           T  A 'B' V  A  L '$' C
00BD  4F 44 C5 56 41 CC 4C 45           O  D 'E' V  A 'L' L  E
00C5  CE 53 49 CE 43 4F D3 54          'N' S  I 'N' C  O 'S' T
00CD  41 CE 41 53 CE 41 43 D3           A 'N' A  S 'N' A  C 'S'
00D5  41 54 CE 4C CE 45 58 D0           A  T 'N' L 'N' E  X 'P'
00DD  49 4E D4 53 51 D2 53 47           I  N 'T' S  Q 'R' S  G
00E5  CE 41 42 D3 50 45 45 CB          'N' A  B 'S' P  E  E 'K'
00ED  49 CE 55 53 D2 53 54 52           I 'N' U  S 'R' S  T  R
00F5  A4 43 48 52 A4 4E 4F D4          '$' C  H  R '$' N  O 'T'
00FD  42 49 CE 4F D2 41 4E C4           B  I 'N' O 'R' A  N 'D'
0105  3C BD 3E BD 3C BE 4C 49           <  '=' >  '=' <  '>' L  I
010D  4E C5 54 48 45 CE 54 CF           N 'E' T  H  E 'N' T 'O'
0115  53 54 45 D0 44 45 46 20           S  T  E 'P' D  E  F
011D  46 CE 43 41 D4 46 4F 52           F 'N' C  A 'T' F  O  R
0125  4D 41 D4 4D 4F 56 C5 45           M  A 'T' M  O  V 'E' E
012D  52 41 53 C5 4F 50 45 4E           R  A  S 'E' O  P  E  N
0135  20 A3 43 4C 4F 53 45 20              '#' C  L  O  S  E
013D  A3 4D 45 52 47 C5 56 45          '#' M  E  R  G 'E' V  E
0145  52 49 46 D9 42 45 45 D0           R  I  F 'Y' B  E  E 'P'
014D  43 49 52 43 4C C5 49 4E           C  I  R  C  L 'E' I  N
0155  CB 50 41 50 45 D2 46 4C          'K' P  A  P  E 'R' F  L
015D  41 53 C8 42 52 49 47 48           A  S 'H' B  R  I  G  H
0165  D4 49 4E 56 45 52 53 C5          'T' I  N  V  E  R  S 'E'
016D  4F 56 45 D2 4F 55 D4 4C           O  V  E 'R' O  U 'T' L
0175  50 52 49 4E D4 4C 4C 49           P  R  I  N 'T' L  L  I
017D  53 D4 53 54 4F D0 52 45           S 'T' S  T  O 'P' R  E
0185  41 C4 44 41 54 C1 52 45           A 'D' D  A  T 'A' R  E
018D  53 54 4F 52 C5 4E 45 D7           S  T  O  R 'E' N  E 'W'
0195  42 4F 52 44 45 D2 43 4F           B  O  R  D  E 'R' C  O
019D  4E 54 49 4E 55 C5 44 49           N  T  I  N  U 'E' D  I
01A5  CD 52 45 CD 46 4F D2 47          'M' R  E 'M' F  O 'R' G
01AD  4F 20 54 CF 47 4F 20 53           O     T 'O' G  O     S
01B5  55 C2 49 4E 50 55 D4 4C           U 'B' I  N  P  U 'T' L
01BD  4F 41 C4 4C 49 53 D4 4C           O  A 'D' L  I  S 'T' L
01C5  45 D4 50 41 55 53 C5 4E           E 'T' P  A  U  S 'E' N
01CD  45 58 D4 50 4F 4B C5 50           E  X 'T' P  O  K 'E' P
01D5  52 49 4E D4 50 4C 4F D4           R  I  N 'T' P  L  O 'T'
01DD  52 55 CE 53 41 56 C5 52           R  U 'N' S  A  V 'E' R
01E5  41 4E 44 4F 4D 49 5A C5           A  N  D  O  M  I  Z 'E'
01ED  49 C6 43 4C D3 44 52 41           I 'F' C  L 'S' D  R  A
01F5  D7 43 4C 45 41 D2 52 45          'W' C  L  E  A 'R' R  E
01FD  54 55 52 CE 43 4F 50 D9           T  U  R 'N' C  O  P 'Y'
```

ТАБЛИЦА КЛАВИШ

Представлены шесть отдельных таблиц клавиш. Конечный код символа зависит от нажатой конкретной клавиши и используемого 'режима'.

a) Таблица основных клавиш - режимы L и CAPS SHIFT.

0205	42	48	59	36	35	54	47	56		B	H	Y	6	5	T	G	V
020D	4E	4A	55	37	34	52	46	43		N	J	U	7	4	R	F	C
0215	4D	4B	49	38	33	45	44	58		M	K	I	8	3	E	D	X
021D	0E	4C	4F	39	32	57	53	5A		SYMBOL SHIFT	L	0	9	2	W	S	Z
0225	20	0D	50	30	31	51	41			SPACE	ENTER	P	0	1	Q	A	

b) Расширенный режим. Буквенные клавиши без клавиш смены регистров.

022C	E3	C4	E0	E4	READ	BIN	LPRINT	DATA
0230	B4	BC	BD	BB	TAN	SGN	ABS	SQR
0234	AF	B0	B1	C0	CODE	VAL	LEN	USR
0238	A7	A6	BE	AD	PI	INKEY$	PEEK	TAB
023C	B2	BA	E5	A5	SIN	INT	RESTORE	RND
0240	C2	E1	B3	B9	CHR$	LLIST	COS	EXP
0244	C1	B8			STR$	LN		

c) Расширенный режим. Буквенные клавиши с любой клавишей смены регистра.

0246	7E	DC	DA	5C	~	BRIGHT	PAPER	\
024A	B7	7B	7D	D8	ATN	{	}	CIRCLE
024E	BF	AE	AA	AB	IN	VAL$	SCREEN$	ATTR
0252	DD	DE	DF	7F	INVERSE	OVER	OUT	(c)
0256	B5	D6	7C	D5	ASN	VERIFY	|	MERGE
025A	5D	DB	B6	D9	]	FLASH	ACS	INK
025E	5B	D7	0C	07	[	BEEP		

d) Коды управления. Цифровые клавиши и CAPS SHIFT.

0260	0C	07	06	04	DELETE	EDIT	CAPS LOCK	TRUE VIDEO
0264	05	08	0A	0B	INV VIDEO	К. влево	К. вниз	К. вверх
0268	09	0F			К. вправо	GRAPHICS		

Примечание: символ "К." в этой таблице означает "Курсор".

e) Коды символов. Буквенные клавиши и клавиша смены регистров.

026A	E2	2A	3F	CD	STOP	*	?	STEP
026E	C8	CC	CB	5E	>=	TO	THEN	^
0272	AC	2D	2B	3D	AT	-	+	=
0276	2E	2C	3B	22	.	,	;	"
027A	C7	3C	C3	3E	<=	<	NOT	>
027E	C5	2F	C9	60	OR	/	<>	%
0282	C6	3A			AND	:		

f) Расширенный режим. Цифровые клавиши и клавиша смены регистров.

0284	D0	CE	A8	CA	FORMAT	DEF FN	FN	LINE
0288	D3	D4	D1	D2	OPEN	CLOSE	MOVE	ERASE
028C	A9	CF			POINT	CAT		

ПРОГРАММЫ РАБОТЫ С КЛАВИАТУРОЙ

ПОДПРОГРАММА 'KEYBOARD SCANNING' ('Просмотр клавиатуры')
Эта очень важная подпрограмма вызывается и основной программой клавиатуры, и программой INKEY$ (в SCANNING).
Во всех случаях регистр E возвращается со значением, лежащим в диапазоне +00 - +27, значение различно дли каждой из сорока клавиш клавиатуры, или со значением +FF, если клавиши не используются.
Регистр D возвращается со значением, которое отражает одну нажатую клавишу смены регистров. Если нажаты обе клавиши смены регистров, то регистры D и E возвращаются со значениями для клавиш CAPS SHIFT и SYMBOL SHIFT, соответственно. Если клавиша не нажата, то регистровая пара возвращается со значением +FFFF.
Признак нуля возвращается сброшенным, если нажато более двух клавиш, или если из пары клавиш ни одна не является клавишей смены регистра.

```
028E KEY-SCAN     LD    L,+2F              Начальное значение клавиши
                                           для каждой строки будет + 2F,
                                           +2E,...,+28 (8 строк).
                  LD    DE,+FFFF           Инициализирует DE 'не
                                           клавишей'.
                  LD    BC,+FEFE           C = адрес порта, B = счетчик.
```

Теперь введем цикл. Делается восемь проходов, на каждом проходе имеются разные начальные значения клавиш и просматривается другая строка из пяти клавиш. (Первая строка это GAPS SHIFT, Z, X, C, V).

```
0296 KEY-LINE     IN    A,(C)              Чтение с указанного порта.
                  CPL                      Нажатая клавиша из строки
                  AND   +1F                будет задавать соответствую-
                                           щий разряд (от 0 разряда -
                                           внешняя клавиша, до 4 раз-
                                           ряда - внутренняя).
                  JR    Z,02AB,KEY-DONE    Переход вперед, если в
                                           строке не нажата ни одна из
                                           пяти клавиш.
                  LD    H,A                Клавишный разряд поступает
                  LD    A,L                на регистр H, пока выби-
                                           рается начальное значение
                                           клавиши.
029F KEY-3KEYS    INC   D                  Если на клавиатуре нажаты
                  RET   NZ                 три клавиши, то регистр D
                                           не будет больше содержать
                                           +FF - поэтому если это
                                           случается - возврат.
02A1 KEY-BITS     SUB   +08                Неоднократно вычитается '8'
                  SRL   H                  из значения текущей клавиши
                  JR    NC,02A1,KEY-BITS   до тех пор, пока не обнару-
                                           жится клавишный разряд.
                  LD    D,E                Скопировать любое предыдущее
                                           значение клавиши в регистр D.
                  LD    E,A                Передать новое значение
                                           клавиши в регистр E.
                  JR    NZ,029F,KEY-3KEYS  Если в этой строке имеется
                                           вторая, а возможно и третья
                                           нажатая клавиша, переход
                                           назад.
```

```
02AB KEY-DONE     DEC   L                  Строка просмотрена, поэтому
                                           начальное значение клавиши
                                           уменьшается для следующего
                                           прохода.
                  RLC   B                  Смещается счетчик и осу-
                  JR    C,0296,KEY-LINE    ществляется переход, если
                                           еще имеются подлежащие
                                           просмотру строки.
```

Теперь выполняются четыре теста.

```
                  LD    A,D                Получить любое значение кла-
                  INC   A                  виши, при котором регистр D
                  RET   Z                  содержит +FF, т.е. нажата
                                           одна клавиша или 'нет
                                           клавиш'.
                  CP    +28                Получить значение для пары
                  RET   Z                  клавиш, если клавиша 'D'
                                           является CAPS SHIFT.
                  CP    +19                Получить значение для па-
                  RET   Z                  ры клавиш, если клавиша
                                           'D' является SYMBOL SHIFT.
                  LD    A,E                В паре для SYMBOL SHIFT .
                  LD    E,D                возможна клавиша 'E'.
                  LD    D,A
                  CP    +18
                  RET                      Возврат с установленным
                                           признаком нуля, если была
                                           'другая клавиша' и
                                           SYMBOL SHIFT, иначе, сброс.
```

ПОДПРОГРАММА 'KEYBOARD' ('Клавиатура')
Эта подпрограмма вызывается всегда при появлении маскируемого прерывания. При нормальной работе это происходит каждые 20 мс. Эта подпрограмма предназначена для просмотра клавиатуры и декодирования значений клавиш. Созданный код, если это разрешает статус 'повторения', будет передан в системную переменную LAST-K. Когда код внесен в системную переменную, задается 5 разряд FLAGS, показывающий, что нажата новая клавиша.

```
02BF KEYBOARD     CALL  028E,KEY-SCAN      Выбор значения клавиши в
                  RET   NZ                 регистровой паре DE, но
                                           немедленный возврат, если
                                           сброшен признак нуля.
```

Используется двойная система 'системных переменных KSTATE' (KSTATE0 – KSTATE3 и KSTATE4 – KSTATE7).
Два набора позволяют выявить новую нажатую клавишу (используется один набор) внутри 'периода повторения' предыдущей клавиши (детализируется в другом наборе).
Набор становится свободным для обработки новой клавиши, если она фиксируется в течение 1/10 секунды, т.е. пяти обращений к KEYBOARD.

```
                  LD    HL,KSTATE0         Начинать с KSTATE0.
02C6 K-ST-LOOP    BIT   7,(HL)             Переход вперед, если 'на-
                  JR    NZ,02D1,K-CH-SET   бор свободен', т.е.
                                           KSTATE0/4 содержит +FF.
                  INC   HL                 Если набор не свободен,
```

```
                DEC   (HL)               уменьшать его 'счетчик
                DEC   HL                 5 обращений' и при дос-
                JR    NZ,02D1,K-CH-SET   тижении нуля сигнализи-
                LD    (HL),+FF           ровать, что набор свободен.
```

После рассмотрения первого набора изменяется указатель и рассматривается второй набор.

```
02D1 K-CH-SET   LD    A,L                Выбор младшего байта ад-
                LD    HL,KSTATE4         реса и переход назад,
                CP    L                  если должен рассматрива-
                JR    NZ,02C6,K-ST-LOOP  ться второй набор.
```

Возврат, если значение клавиши обозначает клавишу смены регистра или 'нет-клавиши'.

```
                CALL  031E,K-TEST        Провести необходимые тесты
                RET   NC                 и, если необходимо, возврат.
                                         Кроме того, изменяется зна-
                                         чение клавиши на
                                         'основной код'.
```

Прижатая клавиша, которая повторяется, отделяется от новой прижатой клавиши.

```
                LD    HL,KSTATE0         Посмотреть сначала
                                         KSTATE0.
                CP    (HL)               Переход вперед, если ото-
                JR    Z,0310,K-REPEAT    ждествляются коды - обозна-
                                         чается повтор.
                EX    DE,HL              Запись адреса KSTATE0.
                LD    HL,KSTATE4         Теперь посмотреть KSTATE4.
                CP    (HL)               Переход вперед, если
                JR    Z,0310,K-REPEAT    отождествляются коды -
                                         обозначается повтор.
```

Но, если не 'свободен' один из наборов системных переменных KSTATE, новая клавиша не будет получена.

```
                BIT   7,(HL)             Рассмотреть второй набор.
                JR    NZ,02F1,K-NEW      Если 'свободен', переход
                                         вперед.
                EX    DE,HL              Теперь рассмотрим первый
                                         набор.
                BIT   7,(HL)             Продолжать, если набор
                RET   Z                  'свободен', но, если нет,
                                         выйти из подпрограммы
                                         KEYBOARD.
```

Должна быть получена новая клавиша. Но перед заполнением системной переменной LAST-K, системные переменные KSTATE используемого набора должны быть инициализированы для обработки любых повторений, а коды клавиш должны быть декодированы.

```
02F1 K-NEW      LD    E,A                Код поступает в регистр
                LD    (HL),A             E и в KSTATE0/4.
                INC   HL                 'Счетчик 5 обращений' для
                LD    (HL),+05           этого набора сброшен к '5'.
                INC   HL                 Третья системная переменная
```

```
                LD    A,(REPDEL)          набора содержит значение
                LD    (HL),A              REPDEL (обычно 0.7 сек.).
                INC   HL                  Указать KSTATE3/7.
```

Декодирование 'основного кода' зависит от текущего состояния MODE, 3 разряда FLAGS и 'байта смещения'.

```
                LD    C,(MODE)            Выбор MODE.
                LD    D,(FLAGS)           Выбор FLAGS.
                PUSH  HL                  Запись указателя, пока
                CALL  0333,K-DECODE       декодируется 'основной
                POP   HL                  код'.
                LD    (HL),A              Значение конечного кода
                                          записывается в KSTATE3/7,
                                          откуда оно берется в случае
                                          повторения.
```

Следующие три строки команд являются общими и для 'новых клавиш' и 'клавиш, повторения'.

```
0308 K-END      LD    (LAST-K),A          Ввести значение конечного
                SET   5,(FLAGS)           кода в LAST-K и сигнализиро-
                                          вать - 'новая клавиша'.
                RET                       Возврат.
```

ПОДПРОГРАММА 'REPEATING KEY' ('Повторяющаяся клавиша')
Клавиша будет 'повторяться' при первом случае нажатия после временной задержки REPDEL (обычно 0.7 сек) и при последовательности нажатий после временной задержки PEPPER (обычно 0.1 сек).

```
0310 K-REPEAT   INC   HL                  Указать 'счетчик 5 обра-
                LD    (HL),+05            щений' используемого на-
                                          бора и сбросить его до 5.
                INC   HL                  Указать третью системную
                DEC   (HL)                переменную - значение
                                          REPDEL/REPPER и уменьшить
                                          его.
                RET   NZ                  Если не передано время
                                          задержки, выход из под-
                                          программы KEYBOARD.
                LD    A,(REPPER)          Единожды переданное вре-
                LD    (HL),A              мя задержки для следую-
                                          щего повторения должно
                                          быть REPPER.
                INC   HL                  Повторение получено, по-
                LD    A,(HL)              этому значение конечного
                                          кода выбирается из
                                          KSTATE3/7 и передается в
                JR    0308,K-END          K-END.
```

ПОДПРОГРАММА 'K-TEST' ('Проверка-K')
Проверяется значение клавиши и, если присутствует 'нет-клавиши', или 'только клавиша смены регистра', осуществляется возврат: в противном случае, находится для этой клавиши 'основной код'.

```
031E K-TEST     LD    B,D                 Скопировать байт сдвига.
                LD    D,+00               Очистить регистр D.
```

```
                LD    A,E               Переместить номер клавиши.
                CP    +27               Если значение было для
                RET   NC                'CAPS SHIFT' или 'нет-
                                        клавиши', возврат.
                CP    +18               Переход вперед, если клавиша
                JR    NZ,032C,K-MAIN    'E' не была SYMBOL SHIFT.
                BIT   7,B               Тем не менее, получите
                RET   NZ                SYMBOL SHIFT и другую
                                        клавишу: возврат только
                                        с SYMBOL SHIFT.
```

Найден 'основной код' с помощью индексирования таблицы основных клавиш.

```
032C K-MAIN     LD    HL,+0205          Базовый адрес таблицы.
                ADD   HL,DE             Индексирование таблицы и
                LD    A,(HL)            выбор 'основного кода'.
                SCF                     Перед возвратом сигнал -
                RET                     'правильное нажатие клавиши'.
```

ПОДПРОГРАММА 'KEYBOARD DECODING' ('Декодирование клавиатуры')
Эта подпрограмма вводится с 'основным кодом' в регистре E, значением FLAGS в регистре D, значением MODE в регистре C и 'байтом сдвига' в регистре B.
'Конечный код' создается с помощью рассмотрения этих четырех значений и обращения, если необходимо, к шести таблицам клавиш. Возвращение в регистр A.

```
0333 K-DECODE   LD    A,E               Скопировать основной код.
                CP    +3A               Переход вперед, если рассма-
                JR    C,0367,K-DIGIT    тривалась цифровая клавиша:
                                        а также, SPACE, ENTER и обе
                                        клавиши смены регистров.
                DEC   C                 Уменьшить значение MODE.
                JP    M,034F,K-KLC-LET  При необходимости пере-
                JR    Z,0341,K-E-LET    ход вперед для режимов
                                        'K', 'L', 'C' и 'E'.
```

Остается только 'графический' режим, а 'конечный код' для буквенных клавиш в графическом режиме вычисляется из 'основного кода'.

```
                ADD   A,+4F             Добавить смещение.
                RET                     Возврат с 'конечным ко-
                                        дом'.
```

Следующими рассматриваются буквенные клавиши в расширенном режиме.

```
0341 K-E-LET    LD    HL,+01EB          Базовый адрес для табл.'b'.
                INC   B                 Переход вперед, чтобы ис-
                JR    Z,034A,K-LOOK-UP  пользовать эту таблицу,
                                        если не нажата ни одна из
                                        клавиш смены регистров.
                LD    HL,+0205          В противном случае испо-
                                        льзовать базовый адрес
                                        для таблицы 'c'.
```

Таблицы клавиш 'b-f' просматриваются с помощью подпрограммы поиска. Во всех случаях находится и возвращается 'конечный код'.

```
034A K-LOOK-UP    LD    D,+00              Очистить регистр D.
                  ADD   HL,DE              Проиндексировать требуе-
                  LD    A,(HL)             мую таблицу и выбрать
                                           'конечный код'.
                  RET                      Затем возврат.
```

Теперь рассматриваются буквенные клавиши в режимах 'K', 'L' или 'C'. Но сначала должны быть обработаны специальные коды SYMBOL SHIFT.

```
034F K-KLC-LET    LD    HL,+0229           Базовый адрес таблицы 'e'.
                  BIT   0,B                Переход назад, если исполь-
                  JR    Z,034A,K-LOOK-UP   зуется клавиша SYMBOL SHIFT
                                           и буквенная клавиша.
                  BIT   3,D                Переход вперед, если в на-
                  JR    Z,0364,K-TOKENS    стоящий момент в режиме 'K'.
                  BIT   3,(FLAGS2)         Если установлен CAPS LOCK,
                  RET   NZ                 то возврат с 'основным
                                           кодом'.
                  INC   B                  Также такой возврат, если
                  RET   NZ                 нажата CAPS SHIFT.
                  ADD   A,+20              Однако, если требуются коды
                  RET                      нижнего регистра, то к
                                           'основному коду' необходимо
                                           добавить +20 (шестнадцате-
                                           ричное), чтобы выдать пра-
                                           вильный 'конечный код'.
```

Значения 'конечного кода' для токенов находятся с помощью добавки +A5 к 'основному коду'.

```
0364 K-TOKENS     ADD   A,+A5              Добавить требуемое смещение
                  RET                      и вернуться.
```

Далее рассматриваются цифровые клавиши, SPACE, ENTER и обе клавиши смены регистров.

```
0367 K-DIGIT      CP    +30                Продолжать только с циф-
                                           ровыми клавишами.
                  RET   C                  Т.е. возврат со SPACE (+20),
                                           ENTER (+0D) и двумя клави-
                                           шами смены регистров (+0E).
                  DEC   C                  Теперь цифровые клавиши
                                           делятся на три группы -
                                           в соответствии с режимом.
                  JP    M,039D,K-KLC-DGT   Переход с режимами 'K',
                                           'L' и 'C';
                  JR    NZ,0389,K-GRA-DGT  а также с режимом 'G'.
                                           Продолж. с режимом 'E'.
                  LD    HL,+0254           Базовый адрес таблицы 'f'.
                  BIT   5,B                Использовать эту таблицу для
                  JR    Z,034A,K-LOOK-UP   SYMBOL SHIFT и цифровой кла-
                                           клавиши в расширенном режиме.
                  CP    +38                Переход вперед с цифровыми
                  JR    NC,0382,K-8-&-9    клавишами '8' и '9'.
```

Цифровые клавиши '0' - '7' в расширенном режиме используются для выдачи или 'кода цвета фона', или 'кода цвета шрифта', в зависимости от использования CAPS SHIFT.

```
                  SUB   +20                Уменьшить диапазон +30 - +37,
```

```
                                    выдавая +10 - +17.
                INC   B             Если не использован CAPS
                RET   Z             SHIFT, возврат с 'кодом
                                    цвета фона'.
                ADD   A,+08         Но, если потом использу-
                RET                 ется диапазон +18 - +1F
                                    - 'код цвета шрифта'.
```

Цифровые клавиши '8' и '9' выдают коды 'BRIGHT' и 'FLASH'.

```
0382 K-8-&-9    SUB   +36           +38 и +39 переходят в
                                    +02 и +03.
                INC   B             Возврат с этими кодами,
                RET   Z             если используется CAPS
                                    SHIFT.(Это коды BRIGHT).
                ADD   A,+FE         Вычесть '2', если исполь-
                RET                 зуется CAPS SHIFT; выда-
                                    ются +00 и +01 (FLASH).
```

Цифровые клавиши в графическом режиме выдают символы графического блока (+80 - +8F), код GRAPHICS (+0F) и код DELETE (+0C).

```
0389 K-GRA-DGT  LD    HL,+0230      Базовый адрес таблицы 'd'.
                CP    +39           Таблицу использовать для
                JR    Z,034A,K-LOOK-UP  цифровой клавиши '9', ко-
                CP    +30           торая выдает GRAPHICS, и
                JR    Z,034A,K-LOOK-UP  '0', которая выдает DELETE.
                AND   +07           Для клавиш '1' - '8' соз-
                ADD   A,+80         дать диапазон +80 - +87.
                INC   B             Возврат со значением из это-
                RET   Z             го диапазона, если не нажата
                                    клавиша смены регистра.
                XOR   +0F           Но, если она нажата, соз-
                RET                 дать диапазон +88 - +8F.
```

Наконец рассмотрим цифровые клавиши в режимах 'K', 'L' и 'C'.

```
039D K-KLC-DGT  INC   B             Возврат, если не использует-
                RET   Z             ся клавиша смены регистров
                                    (конечные коды +30 - +39).
                BIT   5,B           Использовать таблицу 'd',
                LD    HL,+0230      если нажата также клавише
                JR    NZ,034A,K-LOOK-UP  CAPS SHIFT.
```

Теперь можно найти коды для цифровых клавиш и SYMBOL SHIFT.

```
                SUB   +10           Уменьшить диапазон, чтобы
                                    выдать +20 - +29.
                CP    +22           Отделить символ '@' от ос-
                JR    Z,03B2,K-@-CHAR  тальных.
                CP    +20           Кроме того, должен быть
                                    отделен символ '-'.
                RET   NZ            Теперь возврат с конечны-
                                    ми кодами +21, +23 - +29.
                LD    A,+5F         Выдать код +5F символа
                RET                 '-'.
03B2 K-@-CHAR   LD    A,+40         Выдать код +40 символа
                RET                 '@'.
```

ПРОГРАММЫ РАБОТЫ С ДИНАМИКОМ

В этом разделе рассматриваются две подпрограммы - это подпрограмма BEEPER, которая фактически управляет динамиком, и командная процедура BEEP.
Динамик активизируется с помощью D4 низкого уровня во время команды OUT, которая использует порт '254'. Когда D4 высокого уровня, в аналогичной ситуации, динамик деактивизируется. Сигнал 'beep' поэтому может создаваться с помощью изменения уровня D4.
Теперь рассмотрим ноту 'middle C' ('среднее до'), которая имеет частоту 261,63 Гц. Для того чтобы получить эту ноту, динамик должен попеременно активизироваться и деактивизироваться каждые 1/523,26 секунды. В SPECTRUM системные часы настроены на 3.5 МГц и нота 'middle C' будет требовать выполнения команды OUT с максимально возможной точностью каждые 6,689 тактов процессора. Это последнее значение, которое может несколько уменьшиться вследствие издержек, представляет 'длину временного цикла' в подпрограмме BEEPER.

ПОДПРОГРАММА 'BEEPER'
Эта подпрограмма вводится с регистровой парой DE, содержащей значение 'f*t', где нота заданной частоты 'f' имеет длительность звучания 't' секунд, и регистровой парой HL, содержащей значение, равное количеству тактов процессора во 'временном цикле', деленное на '4'.
То есть для ноты 'middle C', которую необходимо сделать длительностью в 1 секунду, DE содержит +0105 (INT (261.63*1)) и HL содержит +066A (полученное из 6.689/4 - 30.125).

```
03B5 BEEPER     DI                      Сделать невозможным пре-
                                        рывание во время звучания
                LD    A,L               Временно записать L.
                SRL   L                 Каждая '1' в регистре L учи-
                SRL   L                 тывает '4' такта процессора,
                                        но берется INT (L/4) и учиты-
                                        вается '16' тактов процес-
                                        сора.
                CPL                     Перейти назад к исходному
                AND   +03               значению в L и определить
                LD    C,A               потери после взятия INT
                LD    B,+00             (L/4).
                LD    IX,+03D1          Вазовый адрес временного
                                        цикла.
                ADD   IX,BC             Изменить длину временного
                                        цикла. Использовать пре-
                                        дыдущую начальную точку
                                        для каждой потерянной '1'
                                        после ввода INT (L/4).
                LD    A,(BORDCR)        Выбор текущего граничного
                AND   +38               цвета и пересылка его в
                RRCA                    разряды 2, 1 и 0 регистра
                RRCA                    A.
                RRCA
                OR    +08               Вывод MIC является 'выкл'.
```

Теперь введем цикл генерации звука. Сделаны полные проходы 'DE', т.е. проход для каждого цикла ноты.
Регистр HL содержит 'длину временного цикла' с '16' тактами

процессора, использованными для каждой '1' в регистре L и '1024' тактов процессора для каждой '1' в регистре H.

```
03D1 BE-IX+3      NOP                     Добавить '4' такта процессора
03D2 BE-IX+2      NOP                     для каждой ранее используемой
03D3 BE-IX+1      NOP                     точки входа.
03D4 BE-IX+0      INC   B                 Значения в регистры B и C будут
                  INC   C                 поступать из регистров H и L -
                                          см. ниже.
03D6 BE-H&L-LP    DEC   C                 'Временной цикл' - т.е. 'BC'*'4'
                  JR    NZ,03D6,BE-H&L-LP такта процессора.
                  LD    C,+3F             (Но в точке половины цикла C
                  DEC   B                 будет равно 'L+1').
                  JP    NZ,03D6,BE-H&L-LP
```

Теперь динамик попеременно активизируется и деактивизируется.

```
                  XOR   +10               Перебросить бит 4.

                  OUT   (+FE),A           Выполнить операцию OUT,
                                          оставляя неизменным бордюр.
                  LD    B,H               Сброс регистра B.
                  LD    C,A               Сохранение регистра A.
                  BIT   4,A               Если в точке половины цикла,
                  JR    NZ,03F2,BE-AGAIN  то переход.
```

После полного цикла проверяется регистровая пара DE.

```
                  LD    A,D               Если уже выполнен последний
                  OR    E                 полный проход, переходим
                  JR    Z,03D6,BE-END     вперёд.
                  LD    A,C               Выбор записанного значения.
                  LD    C,L               Сброс регистра C.
                  DEC   DE                Уменьшить счётчик проходов.
                  JP    (IX)              Переход назад на требуемую
                                          ячейку цикла.
```

Установка параметров для второй половины цикла.

```
03F2 BE-AGAIN     LD    C,L               Сброс регистра C.
                  INC   C                 Добавить '16' тактов процессора,
                                          т.к. этот путь короче.
                  JP    (IX)              Переход назад.
```

После завершения 'beep' разрешаем маскируемые прерывания.

```
03F6 BE-END       EI                      Разрешение прерываний.
                  RET                     Окончательный возврат.
```

КОМАНДНАЯ ПРОЦЕДУРА 'BEEP'
Эта подпрограмма вводится с двумя числами на стек калькулятора. Верхнее число представляет 'основной тон' ноты, нижнее - 'длительность'.

```
03F8 BEEP         RST   0028,FP-CALC      Для обработки значений t и P
                                          используется вычислитель с
                                          плавающей точкой.
```

```
                DEFB  +31,duplicate       t,P,P
                DEFB  +27,int             t,P,i (где i = INT P)
                DEFB  +C0,st-mem-0        t,P,i (mem-0 содержит i)
                DEFB  +03,subtract        t,P (где p - дробная
                                          часть P)
                DEFB  +34,stk-data        Занести на стек десятичное
                                          значение 'K'.
                DEFB  +EC,exponent+7C     0.0577622606 (которое
                DEFB  +6C,+98,+1F,+F5     является меньшей 12*(2^0.5)-1)
                DEFB  +04,multiply        t,pK
                DEFB  +A1,stk-one         t,pK,1
                DEFB  +0F,addition        t,pK+1
                DEFB  +38,end-calc
```

Теперь выполняются несколько тестов на i, целую часть 'основного тона'.

```
                LD    HL,+5C92            Это 'mem-0-1st' (MEMBOT).
                LD    A,(HL)              Выбор порядка i.
                AND   A                   Выдача ошибки, если i не в
                JR    NZ,046C,REPORT-B    интегральной форме.
                INC   HL                  Скопировать знаковый байт в
                LD    C,(HL)              регистр C.
                INC   HL                  Скопировать младший байт в
                LD    B,(HL)              регистр B и в регистр A.
                LD    A,B
                RLA                       Опять выдать сообщение B,
                SBC   A,A                 если i не удовлетворяет
                CP    C                   условию -128<=i<=+127
                JR    NZ,046C,REPORT-B
                INC   HL
                CP    (HL)
                JR    NZ,046C,REPORT-B
                LD    A,B                 Выбрать младший байт и
                                          проверить его.
                ADD   A,+3C
                JP    P,0425,BE-i-OK      Получить -60<=i<=67.
                JP    PO,046C,REPORT-B    Отбросить значения
                                          от -128 до -61.
```

Примечание: диапазон от +70 до +127 будет отброшен позднее.

Теперь можно найти частоту 'основного тона' i.

```
0425 BE-i-OK    LD    B,+FA               Начать с '6' октав ниже 'среднего
                                          до'.
0427 BE-OCTAVE  INC   B                   Чтобы найти правильную октаву
                SUB   +0C                 неоднократно понижаем i.
                JR    NC,0427,BE-OCTAVE
                ADD   A,+0C               Добавить назад последнее
                                          вычитание.
                PUSH  BC                  Записать номер октавы.
                LD    HL,+046E            Базовый адрес таблицы полутонов.
                CALL  3406,LOC-MEM        Рассмотреть таблицу и передать
                CALL  33B4,STACK-NUM      'A-тое' значение на стек
                                          калькулятора.
```

Теперь можно рассмотреть дробную часть 'основного тона'.

```
                RST   0028,FP-CALC        t, pK+1, C
                DEFB  +04,multiply        t, C(pK+1)
```

```
                DEFB  +38,end-calc
```

С помощью изменения 'последнего значения' в соответствии с номером октавы находится конечная частота f.

```
                POP   AF                Выбор номера октавы
                ADD   A,(HL)            Умножить последнее значение
                LD    (HL),A            на '2 в степени номер октавы'.
                RST   0028,FP-CALC      t, f
                DEFB  +C0,st-mem-0      На некоторое время частота
                DEFB  +02,delete        заносится в mem-0.
```

Теперь обратимся к 'длительности'.

```
                DEFB  +31,duplicate     t, t
                DEFB  +38,end-calc
                CALL  1E94,FIND-INT1    Значение 'INT t' должно
                CP    +0B               находиться в диапазоне
                                        +00 to +0A.
                JR    NC,046C,REPORT-B
```

Количество полных циклов в 'beep' задаётся 'f*t', т.к. эти значения уже найдены.

```
                RST   0028,FP-CALC      t
                DEFB  +E0,get-mem-0     t, f
                DEFB  +04,multiply      f*t
```

Результат остаётся на стеке калькулятора, пока вычисляется длина 'временного цикла', требуемая для 'beep'.

```
                DEFB  +E0,get-mem-0     f*t, f
                DEFB  +34,stk-data      На вершине стека калькулятора
                DEFB  +80,four bytes    формируется значение
                DEFB  +43,exponent +93  '3.5 * 10^6/8'
                DEFB  +55,+9F,+80,(+00) f*t, f, 437,500 (десятичн.)
                DEFB  +01,exchange      f*t, 437,500, f
                DEFB  +05,division      f*t, 437,500/f
                DEFB  +34,stk-data
                DEFB  +35,exponent +85
                DEFB  +71,(+00,+00,+00) f*t, 437,500/f, 30.125 (деся.)
                DEFB  +03,subtract      f*t, 437,500/f - 30.125
                DEFB  +38,end-calc
```

Примечание: значение '437,500/f' даёт длину 'полуцикла' ноты, а уменьшение его на '30.125' выдаёт '120.5' тактов процессора, в которых реально создать ноту и установить счётчики. Значение можно передать в требуемые регистры.

```
                CALL  1E99,FIND-INT2    Значение 'временного цикла'
                                        заносится в регистровую пару BC
                PUSH  BC                и сохраняется.
```

Примечание: если значение 'временного цикла' оказывается слишком большим, то будет ошибка (возврат через ERROR-1), в связи с этим не допускаются значения 'основного тона' '+70 to +127'.

```
                CALL  1E99,FIND-INT2    Значение 'f*t' заносится в
                                        регистровую пару BC.
```

```
                POP   HL                Переслать значение 'временного
                                        цикла' в HL.
                LD    D,B               Переслать значение 'f*t' в DE
                LD    E,C
```

Однако, перед созданием 'beep' проверьте значение 'f*t'.

```
                LD    A,D               Возврат, если 'f*t'
                OR    E                 выдало результат 'нет требуемых
                RET   Z                 циклов'.
                DEC   DE                Уменьшить номер цикла и перейти
                JP    03B5,BEEPER       на подпрограмму BEEP (создавая
                                        по крайней мере один переход).
```

Сообщение B - 'Целое вне диапазона'.

```
046C REPORT-B   RST   0008,ERROR-1      Вызов подпрограммы обработки
                DEFB  +0A               ошибок.
```

ТАБЛИЦА 'SEMI-TONE'('ТАБЛИЦА ПОЛУТОНОВ')
В таблице содержатся частоты двенадцати полутонов октавы.

```
                      Частота, Гц.          Нота
046E DEFB       +89,+02,+D0,+12,+86     261.63 C
     DEFB       +89,+0A,+97,+60,+75     277.18 C#
     DEFB       +89,+12,+D5,+17,+1F     293.66 D
     DEFB       +89,+1B,+90,+41,+02     311.12 D#
     DEFB       +89,+24,+D0,+53,+CA     329.63 E
     DEFB       +89,+2E,+9D,+36,+B1     349.23 F
     DEFB       +89,+38,+FF,+49,+3E     369.99 F#
     DEFB       +89,+43,+FF,+6A,+73     392 G
     DEFB       +89,+4F,+A7,+00,+54     415.30 G#
     DEFB       +89,+5C,+00,+00,+00     440 A
     DEFB       +89,+69,+14,+F6,+24     466.16 A#
     DEFB       +89,+76,+F1,+10,+05     493.88 B
```

ПОДПРОГРАММА 'PROGRAM NAME' ('ИМЯ ПРОГРАММЫ') (ZX81)
Эта подпрограмма использовалась в ZX81 и не была убрана, когда ПЗУ переписывалось для ZX-SPECTRUM.

```
04AA DEFB       +CD,+FB,+24,+3A
     DEFB       +3B,+5C,+87,+FA
     DEFB       +8A,+1C,+E1,+D0
     DEFB       +E5,+CD,+F1,+2B
     DEFB       +62,+6B,+0D,+F8
     DEFB       +09,+CB,+FE,+C9
```

ПРОГРАММЫ ОБРАБОТКИ ИНФОРМАЦИИ С КАССЕТНЫХ ЛЕНТ

Монитор 16К имеет большой набор программ для работы с лентой. Эти программы используются при работе БЕЙСИК-операторов SAVE, LOAD, VERIFY и MERGE.

Точкой входа в программы является SAVE-ETC (0605). Однако перед этой точкой входа расположены подпрограммы, связанные с непосредственной записью, загрузкой или проверкой байт.

Во всех случаях байты, подлежащие обработке этими подпрограммами, описываются регистровыми парами: в IX содержится начальный адрес в памяти для операции, в DE содержится количество байт для операции (длина блока данных) и регистр A содержит +00 для работы с блоком заголовка или +FF для работы с блоком программа/данные.

ПОДПРОГРАММА 'SA-BYTES'
Эта подпрограмма обращается к SAVE для записи информации заголовка (из 09BA) и затем для записи непосредственно блока программа/данные (из 099E).

```
04C2 SA-BYTES     LD    HL,+053F          Перегрузка машинного стека адресом
                  PUSH  HL                SA/LD-RET.
                  LD    HL,+1F80          Эта константа выдаёт для
                                          заголовка пилоттон длиной
                                          5 секунд.
                  BIT   7,A               Если запись заголовка, то переход
                  JR    Z,04D0,SA-FLAG    вперёд.
                  LD    HL,+0C98          Эта константа выдаёт пилоттон
                                          около 2-х секунд для блока
                                          данных.
04D0 SA-FLAG      EX    AF,A'F'           Сохраняем признак данные/заголовок.
                  INC   DE                Увеличиваем длину блока
                  DEC   IX                и уменьшаем 'базовый адрес' для
                                          запуска признака.
                  DI                      Во время подпрограммы SAVE
                                          запрещаем прерывания.
                  LD    A,+02             Сигнал 'MIC' включен, бордюр
                                          красного цвета.
                  LD    B,A               Сохраняем значение в регистре B.
```

Теперь вводится цикл, чтобы создать импульс пилоттона. Импульсы 'MIC включен' и 'MIC выключен' содержат в длине 2168 тактов процессора. Цвет бордюра меняется с красного на голубой при каждом фронте импульса.

Примечание: фронт импульса - это когда делается переход с 'выкл' на 'вкл' или наоборот - со 'вкл' на 'выкл'.

```
04D8 SA-LEADER    DJNZ  04D8,SA-LEADER    Основной временной период.
                  OUT   (+FE),A           MIC вкл/выкл, бордюр RED/CYAN,
                  XOR   +0F               на каждом переходе.
                  LD    B,+A4             Основная временная константа.
                  DEC   L                 Уменьшить младший байт счетчика.
                  JR    NZ,04D8,SA-LEADER Переход для другого импульса.
                  DEC   B                 Допускается более длинный путь
                                          (уменьшает на 13 тактов
                                          процессора).
                  DEC   H                 Уменьшить старший байт счётчика.
```

```
                JP    P,04D8,SA-LEADER    Переход назад для другого
                                          импульса, пока на завершится
                                          пилоттон.
```

Посылка синхроимпульса.

```
                LD    B,+2F
04EA SA-SYNC-1  DJNZ  04EA,SA-SYNC-1      MIC выкл в течение 667 тактов
                                          процессора 'OUT to OUT'.
                OUT   (+FE),A             MIC вкл, бордюр RED.
                LD    A,+0D               Сигнал 'MIC выкл & CYAN'.
                LD    B,+37               MIC вкл в течение 735 тактов
04F2 SA-SYNC-2  DJNZ  04F2,SA-SYNC-2      процессора 'OUT to OUT'.
                OUT   (+FE),A             Теперь MIC выкл & бордюр CYAN.
```

Заголовок v. признак программа/данные будут первым байтом, подлежащем обработке с помощию SAVE.

```
                LD    BC,+3B0E            +3B является временной константой;
                                          +0E сигнализирует 'MIC off & YELLOW'.
                EX    AF,A'F'             Выбор признака и передача его в L
                LD    L,A                 для 'посылки'.
                JP    0507,SA-START       Переход вперёд в цикл работы SAVE
```

Теперь вводится цикл записи байт. Первый байт, обрабатываемый SAVE, является признаком; за ним следует фактический байт данных, а последний конечный байт является байтом чётности, который создаётся с помощью рассмотрения значений всех предыдущих байт.

```
04FE SA-LOOP    LD    A,D                 Проверяется 'длина' счётчика
                OR    E                   и выполняется переход, когда
                JR    Z,050E,SA-PARITY    она достигает 0.
                LD    L,(IX+00)           Выбор следующего байта, подлежа-
                                          щего обработке с помощью SAVE.
0505 SA-LOOP-P  LD    A,H                 Выбор текущей чётности.
                XOR   L                   Включает текущий байт.
0507 SA-START   LD    H,A                 Восстановить чётность. Отметим,
                                          что на входе значение признака
                                          сигнализирует чётность.
                LD    A,+01               Сигнал 'MIC вкл & BLUE'.
                SCF                       Установить флаг переноса. Это
                                          будет действовать как 'маркер'
                                          для 8 бит одного байта.
                JP    0525,SA-8-BITS      Переход вперёд.
```

Когда наступает время послать байт чётности, он передаётся в регистр L для записи.

```
050E SA-PARITY  LD    L,H                 Взять конечное значение байта.
                                          'чётности'.
                JR    0505,SA-LOOP-P      Переход назад.
```

Последующий цикл создаёт фактические импульсы. Цикл

вводится на SA-BIT-1 с типом записанного разряда, обозначенного признаком переноса. Для каждого разряда делается два перехода цикла, создавая 'выкл. импульса' и 'вкл. импульса'. Импульсы для разряда сброса короче на 855 тактов процессора.

```
0511 SA-BIT-2     LD    A,C                 Сюда делается переход на втором
                                            проходе и выбрано
                                            'MIC выкл. & YELLOW'.
                  BIT   7,B                 Установка флага Z, чтобы указать,
                                            что это 'второй проход'.
0514 SA-BIT-1     DJNZ  0514,SA-BIT-1       Основной временной цикл. Всегда
                                            801 тактов процессора на втором
                                            проходе.
                  JR    NC,051C,SA-OUT      Переход по короткому пути, если
                                            запись '0'.
                  LD    B,+42               Однако если запись '1', то
051A SA-SET       DJNZ  051A,SA-SET         добавить 855 тактов процессора.
051C SA-OUT       OUT   (+FE),A             На первом проходе 'MIC вкл. &
                                            BLUE', а на втором проходе
                                            'MIC выкл. & YELLOW'.
                  LD    B,+3E               Установить временную константу
                                            для второго прохода.
                  JR    NZ,0511,SA-BIT-2    Переход назад в конец первого
                  DEC   B                   прохода: иначе восстановить
                                            13 тактов процессора.
                  XOR   A                   Сбросить флаг переноса и
                  INC   A                   установить значение рег.A, чтобы
                                            содержать +01 (MIC вкл. & BLUE}
                                            перед продолжением в 8-битовом
                                            цикле.
```

Восьмибитовый цикл вводится изначально со всем байтом в регистр L признаков переноса. Однако он вновь водится после каждого записанного бита, пока не достигнет точки, когда 'маркер' передаётся в признак переноса, оставляя регистр L пустым.

```
0525 SA-8-BITS    RL    L                   Переслать 7 разряд в перенос, а
                                            маркер влево.

                  JP    NZ,0514,SA-BIT-1    Запись бита, пока не закончится
                                            байт.
                  DEC   DE                  Уменьшить счётчик.
                  INC   IX                  Передвинуть базовый адрес.
                  LD    B,+31               Установить временную константу
                                            для первого бита следующего
                                            байта.
                  LD    A,+7F               Возврат на SA/LD-RET, если
                  IN    A,(+FE)             нажата клавиша BREAK.
                  RRA
                  RET   NC
                  LD    A,D                 В противном случае проверить
                  INC   A                   счётчик и перейти назад, даже
                  JP    NZ,04FE,SA-LOOP     если достигнут 0 (для того,
                                            чтобы послать байт чётности).
```

```
                  LD    B,+3B               Выход, если счётчик достигает
053C SA-DELAY     DJNZ  053C,SA-DELAY       +FFFF. Но сначала выводится
                  RET                       короткая задержка.
```

Примечание: разряд сброса выдаст импульс 'MIC выкл.' длительностью 855 тактов процессора, за которым следует импульс 'MIC вкл.' длиной 855 тактов процессора. Тогда как разряд установки выдаст импульс двойной длины. Отметим также, что нет интервалов ни между синхроимпульсом и первым разрядом признака, ни между байтами.

ПОДПРОГРАММА 'SA/LD-RET'
Эта подпрограмма является общей и для SAVE и для LOAD. Задаётся исходным цветом граница и в конце проверяется клавиша BREAK.

```
053F SA/LD-RET    PUSH  AF                  Запись признака переноса (сброс
                                            после ошибки при работе LOAD).
                  LD    A,(BORDER)          Выбор исходного цвета бордюра
                  AND   +38                 из системной переменной.
                  RRCA                      Переслать цвет бордюра
                  RRCA                      в биты 2, 1 и 0.
                  OUT   (+FE),A             Задать исходным цветом
                                            окантовку.
                  LD    A.+7F               Читать клавишу BREAK в течение
                  IN    A,(+FE)             последнего времени.
                  RRA
                  EI                        Разрешение прерываний.
                  JR    C,0554,SA/LD-END    Переход, если не была нажата
                                            клавиша BREAK.
```

Сообщение D - 'BREAK-CONT repeats' (Пауза, нажмите CONT для продолжения).

```
0552 REPORT-D     RST   0008,ERROR-1        Вызов подпрограмма обработки
                  DEFB  +0C                 ошибок.
```

Продолжение здесь.

```
0554 SA/LD-END    POP   AF                  Вернуть в прежнее состояние флаг
                                            переноса.
                  RET                       Возврат к вызывающей подпрограмме.
```

ПОДПРОГРАММА 'LD-BYTES'
Эта подпрограмма вызывается, чтобы выполнить LOAD для информации заголовка (из 07BE) и далее LOAD, или VERIFY фактического блока данных (из 0802).

```
0556 LD-BYTES     INC   D                   Сброс флага Z (регистр D не может
                                            содержать +FF в данном случае).
                  EX    AF,A'F'             Регистр A содержит +00 для
                                            заголовка и +FF для блока данных.
                                            Флаг переноса сброшен для VERIFY
                                            и установлен для LOAD.
```

```
                DEC   D                 Восстановить D в его исходное
                                        значение.
                DI                      Запрещаем прерывания.
                LD    A,+0F             Цвет бордюра - WHITE.
                OUT   (+FE),A
                LD    HL,+053F          Перезагрузка машинного стека
                PUSH  HL                с адресом SA/LD-RET.
                IN    A,(+FE)           Начальное прочтение порта '254'
                RRA                     Циклический сдвиг полученного
                AND   +20               байта, оставляем бит EAR.
                OR    +02               Бордюр красный.
                LD    C,A               Занести значение в регистр C
                                        (+22 для 'выкл' и +02 для 'вкл'
                                        - текущее состояние EAR).
                CP    A                 Установка флага Z.
```

Первый этап чтения с ленты включает в себя проверку, что пульсирующий сигнал действительно существует (т.е. наличествуют фронты импульса 'вкл/выкл' или 'выкл/вкл').

```
056B LD-BREAK   RET   NZ                Возврат, если нажата клавиша
                                        BREAK.
056C LD-START   CALL  05E7,LD-EDGE-1    Возврат со сброшенным флагом C,
                JR    NC,056B,LD-BREAK  если нет фронта импульса прибли-
                                        зительно 14000 тактов процессора.
                                        Но если фронт обнаружен, то бордюр
                                        становится голубого цвета.
```

Следующий этап включает в себя время ожидания и затем показывает, что сигнал ещё не пульсирует.

```
                LD    HL,+0415          Длина периода ожидания
0574 LD-WAIT    DJNZ  0574,LD-WAIT      будет длительностью почти
                DEC   HL                одну секунду.
                LD    A,H
                OR    L
                JR    NZ,0574,LD-WAIT
                CALL  05E3,LD-EDGE-2    Продолжение, если обнаружены
                JR    NC,056B,LD-BREAK  два фронта внутри допустимого
                                        временного интервала.
```

Теперь можно получить только сигнал пилоттона.

```
0580 LD-LEADER  LD    B,+9C             Временная константа,
                CALL  05E3,LD-EDGE-2    Продолжать, если обнаружены
                JR    NC,056B,LD-BREAK  два фронта внутри допустимого
                                        временного участка.
                LD    A,+C6             Однако фронты импульса должны
                CP    B                 быть обнаружены внутри 3000
                JR    NC,056C,LD-START  тактов процессора каждый.
                INC   H                 Счёт пар фронтов в регистре H,
                JR    NZ,0580,LD-LEADER пока не будет обнаружено 256 пар.
```

После пилоттона приходят части 'вкл' и 'выкл' синхроимпульса.

```
058F LD-SYNC    LD    B,+C9              Временная константа.
                CALL  05E7,LD-EDGE-1     Рассматривается каждый фронт, пока
                JR    NC,056B,LD-BREAK   не будет обнаружено два фронта
                LD    A,B                вместе - это будут начальный
                CP    +D4                и конечный фронты синхроимпульса
                JR    NC,058F,LD-SYNC    'выкл'.
                CALL  05E7,LD-EDGE-1     Должен существовать конечный фронт
                RET   NC                 синхроимпульса 'вкл' (возврат
                                         сброшенного импульса переноса).
```

Байты заголовка или блока программы/данные теперь могут быть загружены и проверены. Но первый байт является признаком типа.

```
                LD    A,C                Цвета бордюра в дальнейшем будут
                XOR   +03                BLUE & YELLOW.

                LD    C,A
                LD    H,+00              Байт чётности
                                         изначально равен 0.
                LD    B,+B0              Установить временную константу
                                         для байта признака.
                JR    05C8,LD-MARKER     Переход вперёд к циклу загрузки байта
```

Цикл загрузки байта используется для выбора одного байта за один раз. Первый - это байт признака. За ним следуют байты данных и последним будет 'байт чётности'.

```
05A9 LD-LOOP    EX    AF,A'F'            Возвращаем флаги.
                JR    NZ,05B3,LD-FLAG    Переход вперёд только когда
                                         обработан первый байт.
                JR    NC,05BD,LD-VERIFY  Переход вперед, если нужна
                                         операция VERIFY.
                LD    (IX+00),L          Если потребуется, осуществить
                                         фактическую перезагрузку.
                JR    05C2,LD-NEXT       Переход вперёд для загрузки
                                         следующего байта.
05B3 LD-FLAG    RL    C                  Временно сохраним флаг переноса.
                XOR   L                  Возврат, если признак типа не равен
                RET   NZ                 первому загруженному байту
                                         (сброшен флаг переноса).
                LD    A,C                Восстановим флаг переноса.
                RRA
                LD    C,A
                INC   DE                 Увеличиваем счётчик, чтобы
                JR    05CA,LD-DEC        компенсировать уменьшение его
                                         после переноса.
```

Если выполняем проверку блока данных, то загруженные байты не записываются в память, а сверяются с байтами из памяти.

```
05BD LD-VERIFY  LD    A,(IX+00)          Исходный байт в памяти.
                XOR   L                  Сравним с загруженным байтом.
```

```
                                        новым байтом.
                RET   NZ                Если 'нет сопоставления',
                                        возврат. (Сброшен
                                        признак переноса).
```

Теперь можно получить с ленты новый байт.

```
05C2 LD-NEXT    INC   IX                Увеличить 'адрес назначения'.
05C4 LD-DEC     DEC   DE                Уменьшить 'счетчик'.
                EX    AF,A'F'           Записать признаки.
                LD    B,+B2             Установить временную
                                        константу.
05C8 LD-MARKER  LD    L,+01             Очистить регистр 'объекта',
                                        кроме разряда 'маркера'.
```

Цикл 'LD-8-BITS' используется для создания байта в регистре L.

```
05CA LD-8-BITS  CALL  05E3,LD-EDGE-2    Найти длину импульсов 'выкл'
                                        и 'вкл' следующего разряда.
                RET   NC                Возврат, если превышен вре-
                                        менной период (сброшен
                                        признак переноса).
                LD    A,+CB             Сравнить длину с приблизи-
                                        тельно 2400 тактами процес-
                                        сора, сбрасывая признак пе-
                CP    B                 реноса для '0' и устанав-
                                        ливая для '1'.
                RL    L                 Включить новый разряд в
                                        регистре L.
                LD    B,+B0             Задать временную константу
                                        для следующего разряда.
                JP    NC,05CA,LD-8-BITS Переход назад, пока
                                        проверяются разряды.
```

Байт 'сопоставление четности' должен корректироваться с каждым новым байтом.

```
                LD    A,H               Выбор байта 'сопостав-
                XOR   L                 ления четности' и
                                        включение нового байта.
                LD    H,A               Записать его еще раз.
```

Делаются проходы цикла, пока 'счетчик' не достигает нуля. В этой точке байт 'сопоставление четности' должен содержать ноль.

```
                LD    A,D               Проход, если регистровая
                OR    E                 пара DE не содержит
                JR    NZ,05A9,LD-LOOP   ноль.
                LD    A,H               Выбор байта 'сопоставление
                                        четности'.
                CP    +01               Если значение является 0, то
                RET                     возврат с установленным
                                        признаком переноса.
```

ПОДПРОГРАММЫ 'LD-EDGE-2' и 'LD-EDGE-1'
Эти две подпрограммы формируют наиболее важную часть операции LOAD/VERIFY. Подпрограммы вводятся с временной константой в регистр B, предыдущим цветом окантовки и 'типом фронта' в регистра C.

Подпрограммы возвращаются с установленным признаком переноса, если в отведенное время обнаружено требуемое число 'фронтов'.
Если ошибка, то признак переноса будет сброшен. Признак нуля с помощью сброса сигнализирует - 'нажата BREAK', или с помощью установки 'время закончилось'.
Точка входа LD-EDGE-2 используется, когда требуется длина полного импульса, а LD-EDGE-1 используется для того, чтобы найти временной интервал перед следующим 'фронтом'.

```
05E3 LD-EDGE-2    CALL  05E7,LD-EDGE-1      В действительности LD-EDGE-1
                  RET   NC                  вызывается дважды: промежу-
                                            точный возврат, если была
                                            ошибка.
05E7 LD-EDGE-1    LD    A,+16               Ждет 358 тактов про-
05E9 LD-DELAY     DEC   A                   цессора перед вводом
                  JR    NZ,05E9,LD-DELAY    цикла дискретизации.
                  AND   A
```

Теперь вводится цикл дискретизации. Значение в регистре B увеличивается на каждом проходе; выдается 'время закончилось', когда B достигает 0.

```
05ED LD-SAMPLE    INC   B                   Счет каждого прохода.
                  RET   Z                   Если 'время закончилось',
                                            возврат сброшенного переноса
                                            и установленного 0.
                  LD    A,+7F               Считать из порта +7FFE,
                  IN    A,(+FE)             т.е. BREAK & EAR.
                  RRA                       Сдвиг байта.
                  RET   NC                  Если нажата BREAK, возврат
                                            сброшенного переноса и 0.
                  XOR   C                   Теперь проверка
                  AND   +20                 байта с 'последним типом
                  JR    Z,05ED,LD-SAMPLE    Фронта', переход назад,
                                            пока он не изменится.
```

Обнаружен новый 'фронт' внутри отведенного для поиска периода времени. Поэтому изменится цвет окантовки и устанавливается признак переноса.

```
                  LD    A,C                 Изменить 'последний тип
                  CPL                       фронта' и цвет окантовки.
                  LD    C,A
                  AND   +07                 Сохраняется только цвет
                  OR    +08                 окантовки. Сигнал 'MIC выкл'.
                  OUT   (+FE),A             Изменить цвет окантовки
                                            (RED/CYAN или BLUE/YELLOW).
                  SCF                       Сигнал - успешней поиск
                  RET                       перед возвратом.
```

Примечание: программа LD-EDGE-1 использует 465 тактов процессора, плюс дополнительные 58 тактов для каждого успешного прохода по циклу дискретизации.

Например, поэтому, когда ожидается синхроимпульс (см. LD-SYNC на 038F), допускается 10 дополнительных проходов через цикл дискретизации.
Поиск, таким образом, следующего фронта, подлежащего обнаружению, составляет примерно 1100 тактов процессора (405+10*58+издержки). Это подходит для синхроимпульса 'выкл', который следует за длинным 'импульсом начального участка'.

КОМАНДНАЯ ПРОЦЕДУРА 'SAVE, LOAD, VERIFY & MERGE' ('Запись, загрузка, проверка и слияние')

Для всех четырех команд используется точка входа SAVE-ETC. Однако, значение, содержащееся в T-ADDR для 4 команд различно. Первая часть последующей программы имеет отношение к конструкции 'информации о заголовке' в рабочей области.

```
0605 SAVE-ETC     POP   AF                 Удалить адрес SCAN-LOOP.
                  LD    A,(T-ADDR-lo)      Уменьшить T-ADDR на
                  SUB   +E0                +E0; выдавая +00 для
                  LD    (T-ADDR-lo),A      LOAD, +02 для VERIFY
                                           и +03 для MERGE.
                  CALL  1C8C,EXPT-EXP      Передать параметры 'имени'
                                           на стек калькулятора.
                  CALL  2530,SYNTAX-Z      Если проверка синтак-
                  JR    Z,0652,SA-DATA     сиса, переход вперед.
                  LD    BC,+0011           Допускается 17 ячеек
                  LD    A,(T-ADDR-lo)      для заголовка SAVE,
                  AND   A                  но 30 для остальных
                  JR    Z,0621,SA-SPACE    команд.
                  LD    C,+22
0621 SA-SPACE     RST   0030,BC-SPACES     В рабочей области отведено
                                           необходимое место.
                  PUSH  DE                 Скопировать начальный адрес
                  POP   IX                 в регистровую пару IX.
                  LD    B,+0B              Имя программы может
                  LD    A,+20              иметь до 10 символов,
0629 SA-BLANK     LD    (DE),A             но сначала вводится
                  INC   DE                 11 символов-пробелов
                  DJNZ  0629,SA-BLANK      в подготовленную область.
                  LD    (IX+01),+FF        Пустое имя только +FF.
                  CALL  2BF1,STK-FETCH     Выбираются параметры имени
                                           и проверяется его длина.
                  LD    HL,+FFF6           Это '-10'.
                  DEC   BC                 В действительности,
                  ADD   HL,BC              если длина имени не
                  INC   BC                 слишком большая,
                  JR    NC,064B,SA-NAME    переход вперед (т.е.
                                           не больше 10 символов).
                  LD    A,(T-ADDR-lo)      Не допускается обработка
                  AND   A                  с помощью LOAD, VERIFY и
                  JR    NZ,0644,SA-NULL    MERGE программ с 'пустым'
                                           именем или слишком
                                           длинным именем.
```

Сообщение F - 'Неправильное имя файла'.

```
0642 REPORT-F     RST   0008,ERROR-1       Вызов подпрограммы
                  DEFB  +0E                обработки ошибок.
```

Продолжать обработку имени программы.

```
0644 SA-NULL      LD    A,B                Переход вперед, если
                  OR    C                  имеет 'пустую' длину.
                  JR    Z,0652,SA-DATA
                  LD    BC,+000A           Урезаются длинные имена.
```

Теперь имя передается в рабочую область (вторая ячейка вперед).

```
064B SA-NAME      PUSH  IX                  Скопировать начальный
                  POP   HL                  адрес в регистровую пару HL.
                  INC   HL                  Шагнуть ко второй ячейке.
                  EX    DE,HL               Переключить указатели
                  LDIR                      и скопировать имя.
```

Теперь рассматриваются различные параметры, которые следуют за командой. Начинайте с отработки 'xxx "имя" DATA'.

```
0652 SA-DATA      RST   0018,GET-CHAR       Текущий код токена
                  CP    +E4                 'DATA'?
                  JR    NZ,06A0,SA-SCR$     Если нет, переход.
                  LD    A,(T-ADDR-lo)       Невозможно иметь
                  CP    +03                 'MERGE' или 'DATA'.
                  JP    Z,1C8A,REPORT-C
                  RST   0020,NEXT-CHAR      Продвинуть CH-ADD.
                  CALL  28B2,LOOK-VARS      Посмотрите в области
                                            переменных массив.
                  SET   7,C                 Установить 7 разряд имени
                                            массива.
                  JR    NC,0672,SA-V-OLD    Переход, если обрабатывается
                                            существующий массив.
                  LD    HL,+0000            Сигнал 'использование
                                            нового массива'.
                  LD    A,(T-ADDR-lo)       Рассмотреть значение в
                  DEC   A                   T-ADDR и выдать ошибку, если
                  JR    Z,0685,SA-V-NEW     попытка обработать SAVE или
                                            VERIFY новый массив.
```

Сообщение 2 - 'Переменная не найдена'.

```
0670 REPORT-2     RST   0008,ERROR-1        Вызов подпрограммы
                  DEFB  +01                 обработки ошибок.
```

Продолжать обработку существующего массива.

```
0672 SA-V-OLD     JP    NZ,1C8A,REPORT-C    Примечание: это не выпол-
                                            няется, чтобы исключить
                                            пустые строки.
                  CALL  2530,SYNTAX-Z       При проверке синтак-
                  JR    Z,0692,SA-DATA-1    сиса переход вперед.
                  INC   HL                  Указывает 'младший байт
                                            длины' переменной.
                  LD    A,(HL)              Младший байт длины
                  LD    (IX+0B),A           поступает в рабочую
                  INC   HL                  область; за ним сле-
                  LD    A,(HL)              дует старший байт
                  LD    (IX+0C),A           длины.
                  INC   HL                  Шаг за байты длины.
```

Следующая часть является общей для 'старого' и 'нового' массивов. Примечание: ошибка тракта синтаксиса.

```
0685 SA-V-NEW     LD    (IX+0E),C           Скопировать имя массива.
                  LD    A,+01               Допускается массив чисел.
                  BIT   6,C                 Если это так, переход.
                  JR    Z,068F,SA-V-TYPE
                  INC   A                   Это массив символов.
068F SA-V-TYPE    LD    (IX+00),A           Записать 'тип' в первую
```

ячейку области заголовка.

Последняя часть оператора проверяется перед соединением с другими магистралями.

```
0692 SA-DATA-1    EX    DE,HL              Запись указателя в DE.
                  RST   0020,NEXT-CHAR     Следующий символ ')'?
                  CP    +29
                  JR    NZ,0672,SA-V-OLD   Если нет, сообщение C.
                  RST   0020,NEXT-CHAR     Продвижение CH-ADD.
                  CALL  1BEE,CHECK-END     Если проверяется синтаксис,
                                           перемещение на следующий
                                           оператор.
                  EX    DE,HL              Перед переходом вперед
                  JP    075A,SA-ALL        возврат указателя регистро-
                                           вой пары HL. (Указатель пока-
                                           зывает начало содержимого
                                           существующего массива).
```

Теперь рассмотрим 'SCREEN$'.

```
06A0 SA-SCR$      CP    +AA                Текущий код токена
                                           'SCREEN$'?
                  JR    NZ,06C3,SA-CODE    Если нет, переход.
                  LD    A,(T-ADDR-lo)      Невозможно иметь
                  CP    +03                'MEGRE имя SCREEN$'.
                  JP    Z,1C8A,REPORT-C
                  RST   0020,NEXT-CHAR     Продвижение CH-ADD.
                  CALL  1BEE,CHECK-END     Если проверка синтаксиса,
                                           перемещение на следующий
                                           оператор.
                  LD    (IX+0B),+00        Область дисплея и область
                  LD    (IX+0C),+1B        атрибутов занимает +1800
                  LD    HL,+4000           ячеек и эти ячейки ничинают-
                  LD    (IX+0D),L          ся с +4000; эта информация
                  LD    (IX+0E),H          передается в область заго-
                                           ловка в рабочей области.
                  JR    0710,SA-TYPE-3     Переход вперед.
```

Теперь рассмотрим 'CODE'.

```
06C3 SA-CODE      CP    +AF                Текущий код токена 'CODE'?
                  JR    NZ,0716,SA-LINE    Если нет, переход.
                  LD    A,(T-ADDR-lo)      Невозможно иметь
                  CP    +03                'MERGE имя CODE',
                  JP    Z,1C8A,REPORT-C    Продвижение CH-ADD.
                  RST   0020,NEXT-CHAR     Переход вперед, если
                  CALL  2048,PR-ST-END     не закончился оператор.
                  JR    NZ,06E1,SA-CODE-1
                  LD    A,(T-ADDR-lo)      Невозможно иметь
                  AND   A                  'SAVE имя CODE' само
                  JP    Z,1C8A,REPORT-C    по себе.
                  CALL  1CE6,USE-ZERO      Для 'начала' занесите 0
                                           на стек калькулятора.
                  JR    06F0,SA-CODE-2     Переход вперед.
```

Ищется 'начальный адрес'.

```
06E1 SA-CODE-1    CALL  1C82,EXPT-1NUM     Выбор первого числа.
```

```
                    RST   0018,GET-CHAR       Текущий символ ',' или
                    CP    +2C                 нет?
                    JR    Z,06F5,SA-CODE-3    Если да, переход. Число
                                              было 'начальным адресом'.
                    LD    A,(T-ADDR-lo)       Однако, отвергается 'SAVE
                    AND   A                   имя CODE', которое не имеет
                    JP    Z,1C8A,REPORT-C     'начала' и 'длины'.
06F0 SA-CODE-2      CALL  1CE6,USE-ZERO       Занесите 0 на стек
                                              калькулятора - для 'длины'.
                    JR    06F9,SA-CODE-4      Переход вперед.
```

Выбирается 'длина', так как она была определена.

```
06F5 SA-CODE-3      RST   0020,NEXT-CHAR      Продвижение CH-ADD.
                    CALL  1C82,EXPT-1NUM      Выбор 'длины'.
```

Теперь в области заголовка рабочей области запоминаются параметры.

```
06F9 SA-CODE-4      CALL  1BEE,CHECK-END      Если проверка синтаксиса,
                                              перемещение на следующий
                                              оператор.
                    CALL  1E99,FIND-INT2      Занести 'длину' в
                    LD    (IX+0B),C           регистровую пару BC
                    LD    (IX+0C),B           и запомнить ее.
                    CALL  1E99,FIND-INT2      Поместить и запом-
                    LD    (IX+0D),C           нить 'начальный адрес'
                    LD    (IX+0E),B           в регистровой паре BC.
                    LD    H,B                 Передать 'указатель' как
                    LD    L,C                 обычно в регистровую пару HL.
```

'SCREEN$' и 'CODE' представляют 3 типа.

```
0710 SA-TYPE-3      LD    (IX+00),+03         Введите номер 'типа'.
                    JR    075A,SA-ALL         Переобъединить магистрали.
```

Теперь рассмотрим 'LINE'; и 'нет больше параметров'.

```
0716 SA-LINE        CP    +CA                 Текущий код токена 'LINE'?
                    JR    Z,0723,SA-LINE-1    Если да, то переход.
                    CALL  1BEE,CHECK-END      При проверке синтаксиса
                                              перейти на следующий
                                              оператор.
                    LD    (IX+0E),+80         Когда нет больше пара-
                                              метров, вводится +80.
                    JR    073A,SA-TYPE-0      Переход вперед.
```

Выбор 'номера строк', которые должны следовать за 'LINE'.

```
0723 SA-LINE-1      LD    A,(T-ADDR-lo)       Допускается только
                    AND   A                   'SAVE name LINE
                    JP    NZ,1C8A,REPORT-C    number' ('СОХРАНИТЬ имя
                                              номер строки')?
                    RST   0020,NEXT-Char      Продвижение CH-ADD.
                    CALL  1C82,EXPT-1NUM      Передать номер на
                                              стек вычислителя.
                    CALL  1BEE,CHECK-END      При проверке синтак-
                                              сиса, переместиться
                                              на следующий оператор.
                    CALL  1E99,FIND-INT2      Поместить 'номер
```

```
                LD    (IX+0D),C          строки' в регистровую
                LD    (IX+0E),B          пару BC и запомнить его.
```

'LINE' и 'нет больше параметров' являются 0 типом.

```
073A SA-TYPE-0  LD    (IX+00),+00        Введите номер 'типа'.
```

Найдены и запомнены в области заголовка рабочей области параметры, которые описывают программу и ее применение.

```
                LD    HL,(E-LINE)        Указатель конца области
                                         переменных.
                LD    DE,(PROG)          Указатель начала программы
                                         BASIC.
                SCF                      Теперь выполняется вычита-
                SBC   HL,DE              ние, чтобы найти длину
                LD    (IX+0B),L          'программа + переминные';
                LD    (IX+0C),H          запомнить результат.
                LD    HL,(VARS)          Повторить операцию,
                SBC   HL,DE              но это для запоминания
                LD    (IX+0F),L          только длины 'программы'.
                LD    (IX+10),H
                EX    DE,HL              Передать 'указатель'
                                         в регистровую пару HL.
```

Теперь во всех случаях готовится информация заголовка.

Ячейка 'IX+00' содержит номер типа.
Ячейки 'IX+01 to IX+0A' содержат имя (если пустое, то в 'IX+01' содержится +FF).
Ячейки 'IX+0B & IX+0C' содержат число байт, которые должны быть найдены в 'блоке данных'.
Ячейки 'IX+0D to IX+10' содержат различные параметры, точная интерпретация зависит от 'типа'.

Программа продолжается, и первой задачей является отделение SAVE от LOAD, VERIFY и MERGE.

```
075A SA-ALL     LD    A,(T-ADDR-lo)      Переход вперед,
                AND   A                  когда обрабатывается
                JP    Z,0970,SA-CONTRL   SAVE.
```

В случае команд LOAD, VERIFY или MERGE первые 17 байт 'области заголовка' в рабочей области содержат подготовленную информацию, как сказано выше; теперь надо выбрать 'заголовок' с ленты.

```
                PUSH  HL                 Запись указателя 'адрес
                                         назначения'.
                LD    BC,+0011           В регистровой паре IX фор-
                ADD   IX,BC              мируется базовый адрес
                                         'второй области заголовка'.
```

Теперь вводится цикл; выход из него только тогда, когда 'заголовок' обрабатывается LOAD.

```
0767 LD-LOOK-H  PUSH  IX                 Создать копию базового
                                         адреса.
                LD    DE,+0011           Обработать с помощью LOAD
                                         17 байт.
```

```
                XOR   A                 Сигнал 'заголовок'.
                SCF                     Сигнал 'LOAD'.
                CALL  0556,LD-BYTES     Теперь ищется заголовок.
                POP   IX                Отыскивается базовый адрес.
                JR    NC,0767,LD-LOOK-H Прогон цикла до получения
                                        результата.
```

Новый 'заголовок' отображается на экране, но программа будет продолжаться, только если сравнивается 'новый' заголовок со 'старым'.

```
                LD    A,+FE             Обеспечивается откры-
                CALL  1601,CHAN-OPEN    тие канала 'S'.
                LD    (SCR-CT),+03      Установить счетчик просмотра.
                LD    C,+80             Сигнал 'имя не сравнилось'.
                LD    A,(IX+00)         Сравнить 'новый' тип
                CP    (IX-11)           со 'старым'.
                JR    NZ,078A,LD-TYPE   Переход, если типы
                                        не сопоставляются.
                LD    C,+F6             Но, если сопоставляются,
                                        сигнал - '10 символов' сопос-
                                        тавляются.
078A LD-TYPE    CP    +04               Нонсенс, если 'заголо-
                JR    NC,0767,LD-LOOK-H вок' '4 тип или больше'.
```

Печатается соответствующее сообщение - 'Программа:', 'Массив чисел:',' Массив символов:' или 'Байты:'.

```
                LD    DE,+09C0          Базовый адрес блока сооб-
                                        щений.
                PUSH  BC                Запись регистра C,
                CALL  0C0A,PO-MSG       пока печатается соот-
                POP   BC                ветствующее выражение.
```

Печатается 'новое имя', после этого сравниваются 'новое' и 'старое' имена.

```
                PUSH  IX                Регистровая пара DE
                POP   DE                указывает 'новый тип',
                LD    HL,+FFF0          а HL - 'старое имя'.
                ADD   HL,DE
                LD    B,+0A             Необходимо рассмотреть
                                        10 символов.
                LD    A,(HL)            Переход вперед, если
                                        сопоставление осуществ-
                INC   A                 ляется с фактическим
                JR    NZ,07A6,LD-NAME   именем.
                LD    A,C               Но, если 'старое имя
                ADD   A,B               является 'пустым', то
                LD    C,A               сигнал '10 символов
                                        уже отождествлены'.
```

Вводится цикл для печати символов 'нового имени'. Или будет получено, если 'счетчик' достигнет 0.

```
07A6 LD-NAME    INC   DE                Рассмотрим по очереди каж-
                LD    A,(DE)            дый символ 'нового имени'.
                CP    (HL)              Сопоставить их с соот-
                INC   HL                ветствующими символами
                                        'старого имени'.
```

```
                JR    NZ,07AD,LD-CH-PR    Если нет отождествления,
                INC   C                   не считать их.
07AD LD-CH-PR   RST   0010,PRINT-A-1      Печать 'нового'
                                          символа.
                DJNZ  07A6,LD-NAME        Цикл для 10 символов.
                BIT   7,C                 Получить имя, только
                JR    NZ,0767,LD-LOOK-H   если счетчик достигнет 0.
                LD    A,+0D               За 'новым именем'
                RST   0010,PRINT-A-1      следует 'возврат каретки'.
```

Был найден правильный заголовок и теперь настало время рассмотреть три команды LOAD, VERIFY и MERGE раздельно.

```
                POP   HL                  Выбор указателя.
                LD    A,(IX+00)           'SCREEN$ и CODE' обрабаты-
                CP    +03                 ваются с VERIFY.
                JR    Z,07CB,VR-CONTRL
                LD    A,(T-ADDR-lo)       Переход вперед, если
                DEC   A                   используется команда
                JP    Z,0808,LD-CONTRL    LOAD.
                CP    +02                 Переход вперед, если исполь-
                JP    Z,08B6,ME-CONTRL    зуется команда MERGE; про-
                                          должение с командой VERIFY.
```

УПРАВЛЯЮЩАЯ ПОДПРОГРАММА VERIFY
Процесс верификации включает в себя загрузку блока данных, но байты не запоминаются, а только проверяются. Эта процедуре также используется для загрузки блоков данных, которые описаны 'SCREEN$ & CODE'.

```
07CB VR-CONTRL  PUSH  HL                  Запись 'указателя'.
                LD    L,(IX-06)           Выбор 'количества байт', как
                LD    H,(IX-05)           описано в 'старом' заголовке.
                LD    E,(IX+0B)           Кроме того, выбор из
                LD    D,(IX+0C)           'нового' заголовка.
                LD    A,H                 Переход вперед, если
                OR    L                   не определена 'длина'.
                JR    Z,07E9,VR-CONT-1    Например, только 'LOAD имя
                                          CODE' ('Загрузка имя код').
                                          Выдать сообщение R,
                SBC   HL,DE               если осуществлена попытка
                JR    C,0806,REPORT-R     загрузи блок большего раз-
                                          мера, чем требуется.
                                          Получить равные 'длины'.
                JR    Z,07E9,VR-CONT-1    Также выдать сообщение R,
                LD    A,(IX+00)           если сделана попытка
                CP    +03                 верифицировать блоки
                JR    NZ,0806,REPORT-R    неравного размера (Старая
                                          длина' больше 'новой').
```

Программа продолжается, рассматривая 'указатель адреса назначения'.

```
07E9 VR-CONT-1  POP   HL                  Выбор 'указателя', т.е.
                                          'начала'.
                LD    A,H                 Этот 'указатель' будет
                OR    L                   использоваться, если
                JR    NZ,07F4,VR-CONT-2   он не 0, а в случае 0
```

```
                 LD    L,(IX+0D)          будет использоваться
                 LD    H,(IX+0E)          'начало', найденное в
                                          'новом' заголовке.
```

Теперь рассматривается признак VERIFY/LOAD и создается фактическое LOAD.

```
07F4 VR-CONT-2   PUSH  HL                 Переместить указатель
                 POP   IX                 в регистровую пару IX.
                 LD    A,(T-ADDR-lo)      Переход вперед, если
                 CP    +02                используется команда
                 SCF                      VERIFY$ с признакам перено-
                 JR    NZ,0800,VR-CONT-3  са, сигнализирующим 'LOAD'.
                 AND   A                  Сигнал 'VERIFY'.
0800 VR-CONT-3   LD    A,+FF              Сигнал 'получить только
                                          блок данных', перед
                                          загрузкой блока.
```

ПОДПРОГРАММА 'LOAD A DATA BLOCK' ('Загрузка блока данных')
Эта подпрограмма является общей для всех программ загрузки. В случае LOAD и VERIFY она действует как полный возврат из подпрограммы обработки кассет, но в случае MERGE блок данных должен еще быть обработан MERGE.

```
0802 LD-BLOCK    CALL  0556,LD-BYTES      Обработка блока данных с
                                          помощью LOAD/VERIFY.
                 RET   C                  Если нет ошибки, возврат.
```

Сообщение R – 'Ошибка загрузки с ленты'.

```
0806 REPORT-R    RST   0008,ERROR-1       Вызов подпрограммы
                 DEFB  +1A                обработки ошибок.
```

УПРАВЛЯЮЩАЯ ПОДПРОГРАММА 'LOAD'
Эта подпрограмма управляет загрузкой программы BASIC и ее переменных, или массива.

```
0808 LD-CONTRL   LD    E,(IX+0B)          Выбор 'числа байт', как
                 LD    D,(IX+0C)          задано в 'новом заголовке'.
                 PUSH  HL                 Запись 'указателя
                                          адреса назначения'.
                 LD    A,H                Переход вперед, если нет
                 OR    L                  попытки обработать LOAD пре-
                 JR    NZ,0819,LD-CONT-1  дыдущий необъявленный массив.
                 INC   DE                 К длине добавить 3
                 INC   DE                 байта-1 для имени,
                 INC   DE                 младший и старший байты
                 EX    DE,HL              длины новой переменной.
                 JR    0825,LD-CONT-2     Переход вперед.
```

Теперь рассмотрим, достаточно ли места для нового блока данных.

```
0819 LD-CONT-1   LD    L,(IX-06)          Выбор размера суще-
                 LD    H,(IX-05)          ствующих 'программа +
                 EX    DE,HL              переменные или массив'.
                 SCF                      Переход вперед, если не тре-
                 SBC   HL,DE              буется дополнительного мес-
                 JR    C,082E,LD-DATA     та, учесть восстановление
```

```
                                        используемой в настоящий
                                        момент памяти.

Фактическая проверка места.

0825 LD-CONT-2    LD    DE,+0005        Допускаются лишние
                  ADD   HL,DE           5 байт.
                  LD    B,H             Пересылка результата
                  LD    C,L             в регистровую пару
                  CALL  1F05,TEST-ROOM  BC и создание места.

Теперь загрузка массивов.

082E LD-DATA      POP   HL              Опять выбор 'указателя'.
                  LD    A,(IX+00)       Переход вперед, если
                  AND   A               загружена программа
                  JR    Z,0873,LD-PROG  BASIC.
                  LD    A,H             Переход вперед, если
                  OR    L               загружается новый
                  JR    Z,084C,LD-DATA-1 массив.
                  DEC   HL              Выбор длины существующего
                  LD    B,(HL)          массива,
                  DEC   HL              отобрав длины байт
                  LD    C,(HL)          из области переменных.
                  DEC   HL              Указать старое имя.
                  INC   BC              Добавить к длине 3
                  INC   BC              байта-1 для имени, 2
                  INC   BC              для длины.
                  LD    (X-PTR),IX      Временно записать
                  CALL  19E8,RECLAIM-2  регистровую пару IX,
                  LD    IX,(X-PTR)      пока восстанавливается
                                        старый массив.

Для нового массива создается место - в конце текущей области переменных.

084C LD-DATA-1    LD    HL,(E-LINE)     Найти указатель маркера
                  DEC   HL              конца области переменных -
                                        '80 байт'.
                  LD    C,(IX+0B)       Выбор 'длины' нового
                  LD    B,(IX+0C)       массива.
                  PUSH  BC              Записать эту 'длину'.
                  INC   BC              Добавить 3 байта-1
                  INC   BC              для имени и 2 для
                  INC   BC              'длины'.
                  LD    A,(IX-03)       'IX+0E' старого заголовка
                                        выдает имя массива.
                  PUSH  AF              Записывается имя, пока соз-
                  CALL  1655,MAKE-ROOM  дается соответствующее коли-
                  INC   HL              чество места. В действительнос-
                  POP   AF              ти места 'BC' перед 'новым
                                        80 байтом'.
                  LD    (HL),A          Вводится имя.
                  POP   DE              Выбирается длина и
                  INC   HL              вводятся 2 ее байта.
                  LD    (HL),E
                  INC   HL
                  LD    (HL),D
                  INC   HL              HL теперь указывает на пер-
                                        вую ячейку, которая запол-
```

```
                                    няется данными с ленты.
                PUSH  HL            Этот адрес пересылается в
                POP   IX            регистровую пару IX;
                SCF                 установлен признак переноса
                LD    A,+FF         сигнал 0 'блока данных'; и
                JP    0802,LD-BLOCK обработка блока с помощью
                                    LOAD.
```

Теперь LOAD обрабатывает программу BASIC и ее переменные.

```
0873 LD-PROG    EX    DE,HL             Запись 'указателя адреса
                                        назначения'.
                LD    HL,(E-LINE)       Найти адрес маркера конца
                DEC   HL                текущей области переменных -
                                        '80 байт'.
                LD    (X-PTR),IX        Временно записать IX.
                LD    C,(IX+0B)         Выбор 'длины' нового
                LD    B,(IX+0C)         блока данных.
                PUSH  BC                Сохранить копию 'длины'
                CALL  19E5,RECLAIM-1    пока исправляется текущая
                POP   BC                программа и область
                PUSH  HL                переменных. Запись указателя
                PUSH  BC                программной области и длина
                                        нового блока данных.
                CALL  1655,MAKE-ROOM    Создать достаточное место,
                                        необходимое для новой
                                        программы и ее переменных.
                LD    IX,(X-PTR)        Восстановить пару IX.
                INC   HL                Для новой программы также
                LD    C,(IX+0F)         должна быть установлена
                LD    B,(IX+10)         системная переменная
                ADD   HL,BC             VARS.
                LD    (VARS),HL
                LD    H,(IX+0E)         Если был задан номер
                LD    A,H               строки, то его необ-
                AND   +C0               ходимо рассмотреть.
                JR    NZ,08AD,LD-PROG-1 Если 'нет номера',
                LD    L,(IX+0D)         переход; иначе уста-
                LD    (NEWPPC),HL       новить NEWPPC и
                LD    (NSPPC),+00       NSPPC.
```

Теперь можно загрузить блок данных.

```
08AD LD-PROG-1  POP   DE                Выбор 'длины'.
                POP   IX                Выбор 'начала'.
                SCF                     Сигнал 'LOAD'.
                LD    A,+FF             Сигнал - только 'блок
                                        данных'.
                JP    0802,LD-BLOCK     Теперь обработать
                                        его командой LOAD.
```

УПРАВЛЯЮЩАЯ ПОДПРОГРАММА 'MERGE'
У этой подпрограммы имеются 3 основные части:

1. Загрузка блока данных в рабочую область (командой LOAD).
2. Объединение строки новой программы со строками старой программы (MERGE).
3. Объединение новых переменных со старыми переменными (MERGE).

Начнем поэтому с загрузки блока данных.

```
08B6 ME-CONTRL    LD    C,(IX+0B)           Выбор 'длины' блока
                  LD    B,(IX+0C)           данных.
                  PUSH  BC                  Записать копию 'длины'.
                  INC   BC                  Теперь в рабочей области
                  RST   0030,BC-SPACES      создайте ячейки 'длина+1'.
                  LD    (HL),+80            Разместить маркер конца
                                            в дополнительной ячейке
                  EX    DE,HL               Переместить указатель 'нача-
                                            ла' в регистровую пару HL.
                  POP   DE                  Выбор исходной 'длины'.
                  PUSH  HL                  Запись копии 'начала'.
                  PUSH  HL                  Теперь установите регистро-
                  POP   IX                  вую пару IX для фактической
                                            команды LOAD.
                  SCF                       Сигнал 'LOAD'.
                  LD    A,+FF               Сигнал только 'блок данных'.
                  CALL  0802,LD-BLOCK       Загрузка блока данных.
```

Строки новой программы объединяются со строками старой.

```
                  POP   HL                  Выбор 'начала' новой
                                            программы.
                  LD    DE,(PROG)           Инициализировать DE в 'начало'
                                            старой программы.
```

Ввести цикл для работы со строками новой программы.

```
08D2 ME-NEW-LP    LD    A,(HL)              Выбрать и проверить
                  AND   +C0                 номер строки.
                  JR    NZ,08F0,ME-VAR-LP   После окончания всех строк,
                                            переход.
```

Теперь вводится внутренний цикл для строк старой программы.

```
08D7 ME-OLD-LP    LD    A,(DE)              Выбрать старший байт номера
                  INC   DE                  строки и сравнить его.
                  CP    (HL)                Если он не отождествляется,
                  INC   HL                  переход вперед, но в любом
                  JR    NZ,08DF,ME-OLD-L1   случае продвижение обоих
                                            указателей.
                  LD    A,(DE)              Повторить сравнение для
                  CP    (HL)                младшего байта номера строки.
08DF ME-OLD-L1    DEC   DE                  Теперь переобработка
                  DEC   HL                  указателей.
                  JR    NC,08EB,ME-NEW-L2   Если, обнаруженное для новой
                                            программы, место правильно,
                                            переход вперед.
                  PUSH  HL                  В противном случае
                  EX    DE,HL               находится адрес начала
                                            следующей строки.
                  CALL  19B8,NEXT-ONE
                  POP   HL
                  JR    08D7,ME-OLD-LP      Прогон цикла для каждой
                                            'старой строки'.
08EB ME-NEW-L2    CALL  092C,ME-ENTER       Ввести 'новую строку' и
                  JR    08D2,ME-NEW-LP      опять прогон внешнего цикла.
```

Подобным образом переменные новой программы обрабатываются MERGE с

переменными старой программы. Для поочередной обработки каждой новой переменной вводится новый цикл.

```
08F0 ME-VAR-LP    LD    A,(HL)              Поочередный выбор каждого
                  LD    C,A                 имени переменной и его про-
                                            верка.
                  CP    +80                 После рассмотрения
                  RET   Z                   всех переменных, возврат.
                  PUSH  HL                  Запись текущего нового
                                            указателя.
                  LD    HL,(VARS)           Выбор VARS (для
                                            старой программы).
```

Теперь введем внутренний цикл для поиска существующей области переменных.

```
08F9 ME-OLD-VP    LD    A,(HL)              Выбор и проверка
                  CP    +80                 каждого имени переменной.
                  JR    Z,0923,ME-VAR-L2    При переходе вперед обнару-
                                            живается маркер конца.
                                            (Выполнить 'сложение').
                  CP    C                   Сравнить имена (1-е байты).
                                            Переход вперед для дальней-
                  JR    Z,0909,ME-OLD-V2    шего их рассмотрения, возв-
                                            рат, если нет полного
                                            отождествления.
0901 ME-OLD-V1    PUSH  BC                  Запись нового имени
                  CALL  19B8,NEXT-ONE       переменной, пока рас-
                  POP   BC                  полагается следующая
                                            'старая переменная'.
                  EX    DE,HL               Восстановить указатель
                  JR    08F9,ME-OLD-VP      регистровой пары DE и
                                            опять прогнать цикл.
```

Старые и новые переменные отождествляются по их первым байтам, но переменные с длинными именами необходимо будет полностью сопоставлять.

```
0909 ME-OLD-V2    AND   +E0                 Рассматривать только
                                            разряды 7, 6 и 5.
                  CP    +A0                 Получить все типы перемен-
                  JR    NZ,0921,ME-VAR-L1   ных, кроме переменных с
                                            'длинными именами'.
                  POP   DE                  DE указывает первый
                  PUSH  DE                  символ 'нового имени'.
                  PUSH  HL                  Запись указателя 'старого
                                            имени'.
```

Ввести цикл для сравнения букв длинных имен.

```
0912 ME-OLD-V3    INC   HL                  Обновить и 'старый' и
                  INC   DE                  'новый' указатели.
                  LD    A,(DE)              Сравнить две буквы.
                  CP    (HL)
                  JR    NZ,091E,ME-OLD-V4   Если сопоставление не сра-
                                            батывает, переход вперед.
                  RLA                       Прогон цикла, пока не
                  JR    NC,0912,ME-OLD-V3   обнаружится 'последний
                                            символ'.
                  POP   HL                  Выбор указателя начала 'ста-
```

```
                  JR    0921,ME-VAR-L1      рого имени' и, в случае
                                            успеха, переход вперед.
091E ME-OLD-V4    POP   HL                  Выбор указателя и переход
                  JR    0901,ME-OLD-V1      назад, в случае неуспеха.
```

Переход на это место, если найдено сопоставление.

```
0921 ME-VAR-L1    LD    A,+FF               Сигнал 'замена' переменной.
```

А если нет, то сюда. (А содержит +80 – переменная, которую необходимо 'добавить').

```
0923 ME-VAR-L2    POP   DE                  Выбор указателя 'нового'
                                            имени.
                  EX    DE,HL               Переключить регистры.
                  INC   A                   Признак 0 должен быть уста-
                                            новлен, если необходима
                                            'замена'; сброшен для
                                            'сложения'.
                  SCF                       Сигнал – 'переменные
                                            обработки'.
                  CALL  092C,ME-ENTER       Теперь создается элемент.
                  JR    08F0,ME-VAR-LP      Прогнать цикл, чтобы
                                            рассмотреть следующую новую
                                            переменную.
```

ПОДПРОГРАММА 'MERGE A LINE OR VARIABLE'
Эта подпрограмма вводится со следующими параметрами:

```
  Признак переноса        сброшен   – строка BASIC обрабатывается MERGE.
   (Carry flag)        установлен   – переменная обрабатывается MERGE.
  Ноль                    сброшен   – будет 'сложение'.
   (Zero)              установлен   – 'замена'.
  Регистровая пара HL              – указывает начало нового элемента.
  Регистровая пара DE              – указывает, где применяется MERGE.
```

```
092C ME-ENTER     JR    NZ,093E,ME-ENT-1    Переход, если обра-
                                            батывается 'сложение'.
                  EX    AF,A'F'             Запись признаков.
                  LD    (X-PTR),HL          Запись 'нового' ука-
                  EX    DE,HL               зателя, пока восста-
                  CALL  19B8,NEXT-ONE       навливается 'старая'
                  CALL  19E8,RECLAIM-2      строка или переменная.
                  EX    DE,HL
                  LD    HL,(X-PTR)
                  EX    AF,A'F'             Восстановить признаки.
```

Теперь можно создать новый элемент.

```
093E ME-ENT-1     EX    AF,A'F'             Записать признаки.
                  PUSH  DE                  Создать копию указателя
                                            'адреса назначения'.
                  CALL  19B8,NEXT-ONE       Найти длину 'новой'
                                            переменной/строки.
                  LD    (X-PTR),HL          Записать указатель 'новой'
                                            переменной/строки.
                  LD    HL,(PROG)           Выбор PROG, чтобы
```

```
                                          избежать искажений.
                  EX    (SP),HL           Запись PROG на стек и выбор
                                          'нового' указателя.
                  PUSH  BC                Записать длину.
                  EX    AF,A'F'           Отыскать признаки.
                  JR    C,0955,ME-ENT-2   Если добавляется новая
                                          переменная, переход вперед.
                  DEC   HL                Перед ячейкой 'адреса
                                          назначения' добавляется
                                          новая строка.
                  CALL  1655,MAKE-ROOM    Создать место для новой
                                          строки.
                  INC   HL
                  JR    0958,ME-ENT-3     Переход вперед.
0955 ME-ENT-2     CALL  1655,MAKE-ROOM    Создать место для новой
                                          переменной.
0958 ME-ENT-3     INC   HL                Указать 1-ю новую ячейку.
                  POP   BC                Отыскать длину.
                  POP   DE                Отыскать PROG и занести
                  LD    (PROG),DE         на правильное место.
                  LD    DE,(X-PTR)        Также выбрать 'новый
                                          указатель'.
                  PUSH  BC                Опять записать длину и
                  PUSH  DE                'новый' указатель.
                  EX    DE,HL             Переключить указатели и
                  LDIR                    скопировать 'новую'
                                          переменную/строку в
                                          созданное для нее место.
```

Теперь 'новая' переменная/строка должна быть убрана из рабочей области.

```
                  POP   HL                Выбор 'нового' указателя.
                  POP   BC                Выбор длины.
                  PUSH  DE                Записать 'старый' указатель
                                          (Указать ячейку после 'добав-
                                          ленной' переменной/строки).
                  CALL  19E8,RECLAIM-2    Убрать переменную/строку из
                                          рабочей области.
                  POP   DE                Возврат со 'старым' указате-
                  RET                     лем в регистровую пару DE.
```

УПРАВЛЯЮЩАЯ ПОДПРОГРАММА 'SAVE'
Операция SAVE для программ и блоков данных является очень простой.

```
0970 SA-CONTRL    PUSH  HL                Запись 'указателя'.
                  LD    A,+FD             Обеспечивается открытие
                  CALL  1601,CHAN-OPEN    канала 'K'.
                  XOR   A                 Сигнал - 'первое сообщение'.
                  LD    DE,+09A1          Печать сообщения - 'Начало
                  CALL  0C0A,PO-MSG       ленты, нажмите любую
                                          клавишу'.
                  SET   5,(TV-FLAG)       Сигнал 'необходимо очистить
                                          экран'.
                  CALL  15D4,WAIT-KEY     Ожидание нажатия клавиши.
```

Запись 'заголовка' идет до нажатия клавиши.

```
                PUSH  IX                Запись базового адреса 'за-
                                        головка' на машинный стек.
                LD    DE,+0011          Должны быть записаны 17 байт.
                XOR   A                 Сигнал - 'это заголовок'.
                CALL  04C2,SA-BYTES     Послать 'заголовок': с го-
                                        ловным байтом 'типа' и кон-
                                        кретным байтом 'четности'.
```

Перед записью блока программа/данные следует короткая задержка.

```
                POP   IX                Восстановить указатель
                                        'заголовка'.
                LD    B,+32             Задержка для 50 прерываний,
0991 SA-1-SEC   HALT                    т.е. 1 секунда.
                DJNZ  0991,SA-1-SEC
                LD    E,(IX+0B)         Выбор длины блока данных,
                LD    D,(IX+0C)         который должен быть записан.
                LD    A,+FF             Сигнал - 'блок данных'.
                POP   IX                Выбор 'начала указа-
                JP    04C2,SA-BYTES     теля блока' и обра-
                                        ботка блока каждой SAVE.
```

СООБЩЕНИЯ

Каждое сообщение выдается с последним инвертированным символом (+80, шестнадцатеричное):

```
09A1 DEFB +80     - Перешагнули через начальный байт.
09A2 DEFM         - Start tape, then press any key.
                    Начало ленты, нажмите любую клавишу.
09C1 DEFM         - 'carriage return' - Program:
                                        Программа.
09CB DEFM         - 'carriage return' - Number array:
                                        Массив чисел.
09DA DEFM         - 'carriage return' - Character array:
                                        Массив символов.
09EC DEFM         - 'carriage return' - Bytes:
                                        Байты.
```

ПРОГРАММЫ ОБРАЩЕНИЯ К ЭКРАНУ И ПРИНТЕРУ

ПРОГРАММА 'PRINT-OUT' ('ВЫВОД ДАННЫХ')
С помощью этой программы обрабатывается принтер, нижняя и основная часть экрана. Программа PRINT-OUT вводится с регистра A, содержащим код для управляющего символа, символа или токена, подлежащих печати.

```
09F4 PRINT-OUT    CALL  0B03,PO-FETCH       Текущая позиция печати.
                  CP    +20                 Если код представляет сим-
                  JP    NC,0AD9,PO-ABLE     вол, подлежащий печати,
                                            переход.
                  CP    +06                 Печать знака вопроса для
                  JR    C,0A69,PO-QUEST     кодов диапазона +00 - +05.
                  CP    +18                 То же для кодов
                  JR    NC,0A69,PO-QUEST    +18 - +1F.
                  LD    HL,+0A0B            База таблицы 'управления'.
                  LD    E,A                 Переслать код в ре-
                  LD    D,+00               гистровую пару DE.
                  ADD   HL,DE               Проиндексировать таблицу и
                  LD    E,(HL)              выбрать смещение (offset).
                  ADD   HL,DE               Добавить смещение и
                  PUSH  HL                  осуществить непрямой
                  JP    0B03,PO-FETCH       переход в соответствующую
                                            подпрограмму.
```

ТАБЛИЦА 'СИМВОЛЫ УПРАВЛЕНИЯ'

адрес	смещение	символы	адрес	смещение	символы
0A11	4E	PRINT запятая	0A1A	4F	не используется
0A12	57	EDIT	0A1B	5F	INK control
0A13	10	курсор влево	0A1C	5E	PAPER control
0A14	29	курсор вправо	0A1D	5D	FLASH control
0A15	54	курсор вниз	0A1E	5C	BRIGHT control
0A16	53	курсор вверх	0A1F	5B	INVERSE control
0A17	52	DELETE	0A20	5A	OVER control
0A18	37	ENTER	0A21	54	AT control
0A19	50	не используется	0A22	53	TAB control

ПОДПРОГРАММА 'CURSOR LEFT' ('Курсор влево')
Подпрограмма вызывается с регистром B, содержащим текущий номер строки и регистром C с текущим номером столбца.

```
0A23 PO-BACK-1    INC   C                   Переместиться влево
                                            на один столбец.
                  LD    A,+22               Изменение, если
                  CP    C                   с левой стороны.
                  JR    NZ,0A3A,PO-BACK-3
                  BIT   1,(FLAGS)           Если работа с прин-
                  JR    NZ,0A38,PO-BACK-2   тером, переход вперед.
                  INC   B                   Переход на одну строку.
                  LD    C,+02               Задать значения столбца.
                  LD    A,+18               Проверка по верхней строке.
```

```
                  CP    B                   Примечание: Должно быть +19.
                  JR    NZ,0A3A,PO-BACK-3   Получить изменение, если не
                                            на вершине экрана.
                  DEC   B                   Не получается, поэтому вниз
                                            на строку.
0A38 PO-BACK-2    LD    C,+21               Установить на левый столбец.
0A3A PO-BACK-3    JP    0DD9,CL-SET         Осуществить непрямой возврат
                                            через CL-SET & PO-STORE.
```

ПОДПРОГРАММА 'CURSOR RIGHT' ('Курсор вправо')
Эта подпрограмма выполняет операцию, идентичную оператору BASIC - PRINT OVER 1;CHR$ 32; -.

```
0A3D PO-RIGHT     LD    A,(P-FLAG)          Выбор и запись на
                  PUSH  AF                  машинный стек P-FLAG.
                  LD    (P-FLAG),+01        Установить P-FLAG и OVER.
                  LD    A,+20               'Пробел'.
                  CALL  0B65,PO-CHAR        Печать символа.
                  POP   AF                  Выбор старого значе-
                  LD    (P-FLAG),A          ния P-FLAG.
                  RET                       Окончание.
                                            Примечание: Программист
                                            забыл выйти через PO-STORE.
```

ПОДПРОГРАММА 'CARRIAGE RETURN' ('Возврат каретки')
Если обработанные данные идут на принтер, то символ возврата каретки поступает в очищенный буфер принтера. Если же они идут на экран, то вычисляется тест для 'scroll?' ('прокрутка?') перед уменьшением номера строки.

```
0A4F PO-ENTER     BIT   1,(FLAGS)           Если обрабатывается
                  JP    NZ,0ECD,COPY-BUFF   принтер, то переход вперед.
                  LD    C,+21               Установить на левый столбец.
                  CALL  0C55,PO-SCR         Если необходимо, прокрутка.
                  DEC   B                   Теперь вниз на строку.
                  JP    0DD9,CL-SET         Осуществить непрямой возврат
                                            через CL-SET & PO-STORE.
```

ПОДПРОГРАММА 'PRINT COMMA' ('Печать запятой')
Обрабатывается текущее значение столбца и задается регистр A, чтобы содержать +00 (для TAB 0) или +10 (для TAB 16).

```
0A5F PO-COMMA     CALL  0B03,PO-FETCH       Почему опять?
                  LD    A,C                 Номер текущего столбца.
                  DEC   A                   Перемещение вправо на два
                  DEC   A                   столбца, а затем проверка.
                  AND   +10                 Регистр A будет +00 или +10.
                  JR    0AC3,PO-FILL        Выход через PO-FILL.
```

ПОДПРОГРАММА 'PRINT A QUESTION MARK' ('Печать знака вопроса')
Знак вопроса распечатывается в случае попытки напечатать не подлежащий печати код.

```
0A69 PO-QUEST     LD    A,+3F               Символ '?'.
                  JR    0AD9,PO-ABLE        Напечатать этот символ.
```

ПРОГРАММА 'CONTROL CHARACTERS WITH OPERANDS' ('Управляющие символы с операндами')
Управляющие символы от INK до OVER требуют единственный операнд, тогда как управляющие символы AT и TAB требуют за собой два операнда. Настоящая программа приводит к коду управляющего символа, записанного в TVDATA-hi, 1-му операнду в TVDATA-hi - или в регистр A, если затребован только один операнд, и второму операнду в регистре A.

```
0A6D PO-TV-2      LD    DE,+0A87            Записать первый операнд в
                  LD    (TVDATA-hi),A       TVDATA-hi и изменить адрес
                  JR    0A80,PO-CHANGE      программы 'вывода' в
                                            PO-CONT (+0A87).
```

Ввод здесь при обработке символов AT и TAB

```
0A75 PO-2-OPER    LD    DE,+0A6D            Код символа 'будет занесен
                  JR    0A7D,PO-TV-1        в TVDATA-lo,
                                            в адрес измененной программы
                                            'вывода' в PO-TV-2 (+0A6D).
```

Ввод здесь при обработке элементов цвета - от INK к OVER.

```
0A7A PO-1-OPER    LD    DE,+0A87            Должна быть изменена прог-
                                            рамма 'вывода' в PO-CONT
                                            (+0A87).
0A7D PO-TV-1      LD    (TVDATA-lo),A       Запись кода символов
                                            управления.
```

Текущий адрес программы 'вывода' изменяется временно.

```
0A80 PO-CHANGE    LD    HL,(CURCHL)         HL будет указывать адрес
                                            программы 'вывода'.
                  LD    (HL),E              Ввести новый адрес
                                            программы 'вывода'
                  INC   HL                  с тем, чтобы форси-
                                            ровать рассмотрение
                  LD    (HL),D              следующего кода сим-
                  RET                       вола как операнда.
```

Программа продолжается, когда собраны операнды.

```
0A87 PO-CONT      LD    DE,+09F4            Восстановить исходный адрес
                  CALL  0A80,PO-CHANGE      для PRINT-OUT (+09F4).
                  LD    HL,(TVDATA)         Выбрать управляющий код и
                                            первый операнд, если их в
                                            действительности два.
                  LD    D,A                 Пересылаются 'последний'
                  LD    A,L                 операнд и управляющий код.
                  CP    +16                 Если обрабатываются
                  JP    C,2211,CO-TEMPS     INK-OVER, переход вперед.
                  JR    NZ,0AC2,PO-TAB      Если обрабатывается
                                            TAB, переход вперед.
```

Теперь работа с управляющим символом AT.

```
                  LD    B,H                 Номер строки.
                  LD    C,D                 Номер столбца.
                  LD    A,+1F               Перестановка номера;
```

```
                  SUB   C                  столбца, т.е. +00 - +1F
                                           становится +1F - +00.
                  JR    C,0AAC,PO-AT-ERR   Должно быть в диапазоне.
                  ADD   A,+02              Добавить смещение, чтобы
                  LD    C,A                выдать C, содержащее
                                           +21 - +22.
                  BIT   1,(FLAGS)          Если обработка прин-
                  JR    NZ,0ABF,PO-AT-SET  тера, переход вперед.
                  LD    A,+16              Перестановка номера
                  SUB   B                  строки: т.е. +00 - +15
                                           становится +16 - +01.
0AAC PO-AT-ERR    JP    C,1E9F,REPORT-B    Если соответствует,
                                           переход вперед.
                  INC   A                  Диапазон +16 - +01 ста-
                  LD    B,A                новится +17 - +02.
                  INC   B                  A теперь +18 - +03.
                  BIT   0,(TV-FLAG)        Если печать в нижней части
                  JP    NZ,0C55,PO-SCR     экрана, то рассмотреть,
                                           необходима ли прокрутка.
                  CP    (DF-SZ)            Если требуется, выдать
                  JP    C,0C86,REPORT-5    сообщение 5 - вне экрана.
0ABF PO-AT-SET    JP    0D09,CL-SET        Возврат через CL-SET и
                                           PO-STORE.
```

И управляющий символ TAB.

```
0AC2 PO-TAB       LD    A,H                Выбор операнда.
0AC3 PO-FILL      CALL  0B03,PO-FETCH      Текущая позиция печати.
                  ADD   A,C                Добавить текущее значение
                                           столбца.
                  DEC   A                  Найти сколько требуется
                  AND   +1F                'пробелов' по модулю 32, и,
                  RET   Z                  если результат 0, возврат.
                  LD    D,A                Использовать как счетчик D.
                  SET   0,(FLAGS)          Подавить 'головные пробелы'.
0AD0 PO-SPACE     LD    A,+20              Печатать 'число D'
                  CALL  0C3B,PO-SAVE       пробелов.
                  DEC   D
                  JR    NZ,0AD0,PO-SPACE
                  RET                      Теперь окончание.
```

КОДЫ ПЕЧАТАЕМЫХ СИМВОЛОВ
Требуемый символ (или символы) печатаются с помощью вызова PO-ANY, за которой следует PO-STORE.

```
0AD9 PO-ABLE      CALL  0B24,PO-ANY        Напечатать символ(ы) и
                                           продолжить в PO-STORE.
```

ПОДПРОГРАММА 'POSITION STORE' ('Запоминание позиции')
Новое значение позиции 'строка и столбец' и адрес запоминаются в соответствующих системных переменных.

```
0ADC PO-STORE     BIT   1,(FLAGS)          Если обработка прин-
                  JR    NZ,0AFC,PO-ST-PR   тера, переход вперед.
                  BIT   0,(TV-FLAG)        Если обработка нижней части
                  JR    NZ,0AF0,PO-ST-E    экрана, переход вперед.
                  LD    (S-POSN),BC        Запись значений, относящихся
```

```
                                        к основной части
                LD    (DF-CC),HL        экрана.
                RET                     Затем возврат.
0AF0 PO-ST-E    LD    (S-POSNL),BC      Запись значений, от-
                LD    (ECHO-E),BC       носящихся к нижней
                LD    (DF-CCL),HL       части экрана.
                RET
0AFC PO-ST-PR   LD    (P-POSN),C        Запись значений, относящихся
                LD    (PR-CC),HL        к буферу принтера.
                RET                     Затем возврат.
```

ПОДПРОГРАММА 'POSITION FETCH' ('Выбор позиции')
Из соответствующих системных переменных выбираются параметры текущей позиции.

```
0B03 PO-FETCH   BIT   1,(FLAGS)         Если обработка принтера,
                JR    NZ,0B1D,PO-F-PR   переход вперед.
                LD    BC,(S-POSN)       Выбор значений, относящихся
                LD    HL,(DF-CC)        к основной части экрана и,
                BIT   0,(TV-FLAG)       если это было намеренно,
                RET   Z                 возврат.
                LD    BC,(S-POSNL)      В противном случае выбрать
                LD    HL,(DF-CCL)       значение, относящееся к
                RET                     нижней части экрана.
0B1D PO-F-PR    LD    C,(P-POSN)        Выбор значений,
                LD    HL,(PR-CC)        относящихся к буферу
                RET                     принтера.
```

ПОДПРОГРАММА 'PRINT ANY CHARACTER(S)' ('Печать любого символа(ов)')
Коды обычных символов, коды токенов и коды графических символов, определенных пользователем, коды графики обрабатываются раздельно.

```
0B24 PO-ANY     CP    +80               Переход вперед с ко-
                JR    C,0B65,PO-CHAR    дами обычных символов.
                CP    +90               Переход вперед с кодами
                JR    NC,0B52,PO-T&UDG  токенов и кодами графических
                                        символов, определенных
                                        пользователем.
                LD    B,A               Переместить код графики.
                CALL  0B38,PO-GR-1      Создать графическую форму.
                CALL  0B03,PO-FETCH     HL разрушалась, поэтому
                                        опять 'выбор'.
                LD    DE,+5C92          Создать DE для указания
                                        начала графической
                                        формы, т.е. MEMBOT.
                JR    0B7F,PO-ALL       Переход вперед, чтобы напе-
                                        чатать графический символ.
```

Графические символы конструируются специально подобранным для этого способом в области памяти калькулятора, т.е. в MEM-0 и MEM-1.

```
0B38 PO-GR-1    LD    HL,+5C92          Это MEMBOT.
                CALL  0B3E,PO-GR-2      В действительности вызов
                                        подпрограммы происходит
                                        дважды.
0B3E PO-GR-2    RR    B                 Определить 0 разряд
```

```
                                                (и позднее 2 разряд) кода
                    SBC   A,A                   графического символа.
                    AND   +0F                   Регистр A будет содержать
                                                +00 или +0F, зависящее от
                                                разряда в коде.
                    LD    C,A                   Запись результата в C.
                    RR    B                     Определить 1 разряд
                    SBC   A,A                   (и позднее 3 разряд) кода
                                                графического символа.
                    AND   +F0                   Регистр A будет со-
                                                держать +00 или +0F.
                    OR    C                     Комбинируются два
                                                результата.
                    LD    C,+04                 Регистр A содержит половину
0B4C PO-GR-3        LD    (HL),A                формы символа и должен быть
                    INC   HL                    использован 4 раза.
                    DEC   C                     Это делается для
                    JR    NZ,0B4C,PO-GR-3       верхней половины формы
                    RET                         символа, затем для нижней.
```

Теперь разделяются коды токенов и коды графических символов, определенных пользователем (ГСОП или UDG).

```
0B52 PO-T&UDG       SUB   +A5                   Переход вперед с кодами
                    JR    NC,0B5F,PO-T          токенов.
                    ADD   A,+15                 Коды ГСОП теперь +00 - +0F.
                    PUSH  BC                    Записать текущее значение
                                                позиции на машинный стек.
                    LD    BC,(UDG)              Выбор базового адреса облас-
                    JR    0B6A,PO-CHAR-2        ти ГСОП и переход вперед.
0B5F PO-T           CALL  0C10,PO-TOKENS        Теперь печать токена и
                    JP    0B03,PO-FETCH         возврат через PO-FETCH.
```

Идентифицируется форма требуемого символа.

```
0B65 PO-CHAR        PUSH  BC                    Записывается текущая
                                                позиция.
                    LD    BC,(CHARS)            Выбирается базовый адрес
                                                области символов.
0B6A PO-CHAR-2      EX    DE,HL                 Записывается адрес печати.
                    LD    HL,+5C3B              Это FLAGS.
                    RES   0,(HL)                Разрешить начальные пробелы.
                    CP    +20                   Если символ не 'пробел',
                    JR    NZ,0B76,PO-CHAR-3     переход вперед.
                    SET   0,(HL)                Если 'пробел', 'подавить'.
0B76 PO-CHAR-3      LD    H,+00                 Теперь передать код символа
                    LD    L,A                   в регистровую пару HL.
                    ADD   HL,HL                 Код символа в дейс-
                    ADD   HL,HL                 твительности умно-
                    ADD   HL,HL                 жается на 8.
                    ADD   HL,BC                 Найден базовый адрес формы
                                                символа.
                    POP   BC                    Выбирается текущая позиция,
                    EX    DE,HL                 а базовый адрес передается
                                                в регистровую пару DE.
```

ПОДПРОГРАММА 'PRINT ALL CHARACTERS' ('Печать всех символов')
Эта подпрограмма используется для печати всех '8*8' разрядов символов.

На входе регистровая пара DE содержит базовый адрес формы символа, регистр HL – адрес назначения и регистровая пара BC текущее значение 'строка и столбец'.

```
0B7F PR-ALL       LD    A,C                 Выбор номера столбца.
                  DEC   A                   Переместить один столбец
                                            вправо.
                  LD    A,+21               Если не обозначена новая
                  JR    NZ,0893,PR-ALL-1    строка, переход вперед.
                  DEC   B                   Переместиться вниз на
                                            одну строку.
                  LD    C,A                 Номер столбца +21.
                  BIT   1,(FLAGS)           Если обработка экра-
                  JR    Z,0B93,PR-ALL-1     на, переход вперед.
                  PUSH  DE                  Запись базового ад-
                  CALL  0ECD,COPY-BUFF      реса, пока очищается
                  POP   DE                  буфер принтера.
                  LD    A,C                 Скопировать номер
                                            нового столбца.
0B93 PR-ALL-1     CP    C                   Проверить, является
                                            ли новая строка уже
                  PUSH  DE                  использованной. Если
                  CALL  Z,0C55,PO-SCR       да, дисплей требует
                  POP   DE                  прокрутку.
```

Теперь рассмотрим текущее состояние INVERSE и OVER.

```
                  PUSH  BC                  Запись значения позиции и
                  PUSH  HL                  адреса назначения на машин-
                                            ный стек.
                  LD    A,(P-FLAG)          Выбрать P-FLAG и прочитать
                                            0 разряд.
                  LD    B,+FF               Подготовить 'OVER-
                  RRA                       маску' в регистре B,
                  JR    C,0BA4,PR-ALL-2     т.е. OVER 0 = +00 и
                  INC   B                   OVER 1 – +FF.
0BA4 PR-ALL-2     RRA                       Прочитать 2 разряд
                  RRA                       P-FLAG и подготовить
                  SBC   A,A                 'INVERSE-маску' в
                  LD    C,A                 регистре C, т.е. INVERSE 0
                                            = +00 и INVERSE 1 = +FF.
                  LD    A,+08               Установить регистр A чтобы
                  AND   A                   содержать счетчик чисел
                                            строки и очистить признак
                                            переноса.
                  BIT   1,(FLAGS)           Если обрабатывается
                  JR    Z,0BB6,PR-ALL-3     экран, переход вперед.
                  SET   1,(FLAGS2)          Сигнал – 'буфер принтера
                                            не пустой'.
                  SCF                       Установить признак переноса,
                                            чтобы показать, что принтер
                                            использовался.
0BB6 PR-ALL-3     EX    DE,HL               Перед вводом цикла обменять
                                            адрес назначения с базовым
                                            адресом.
```

Теперь можно напечатать символ. Выполняются 8 проходов цикла – один для каждой 'пиксел-строки'.

```
0BB7 PR-ALL-4     EX    AF,A'F'             При использовании принтера
                                            устанавливается признак
                                            переноса. Записать этот
                                            признак в F.
                  LD    A,(DE)              Выбор существующей
                                            'пиксел-строки'.
                  AND   B                   Использовать 'OVER-маску' и
                  XOR   (HL)                затем обработать с помощью
                                            XOR результат с 'пиксел-
                                            строкой' формы символа.
                  XOR   C                   В конце рассмотреть
                                            'INVERSE-маску'.
                  LD    (DE),A              Ввести результат.
                  EX    AF,A'F'             Выбрать признак принтера,
                  JR    C,0BD3,PR-ALL-6     и если требуется, переход
                                            вперед.
                  INC   D                   Обновить адрес назначения.
0BC1 PR-ALL-5     INC   HL                  Обновить 'пиксел-строку'
                                            формы символа.
                  DEC   A                   Уменьшить счетчик и, если он
                  JR    NZ,0BB7,PR-ALL-4    не является 0, вернуться
                                            к началу цикла.
```

Символ печатается один раз, если необходимо, устанавливается байт атрибута.

```
                  EX    DE,HL               Сделать регистр H, содержа-
                  DEC   H                   щим правильный старший адрес
                                            для области символов.
                  BIT   1,(FLAGS)           Установить байт ат-
                  CALL  Z,0BDB,PO-ATTR      рибута, если только
                                            обрабатывается экран.
                  POP   HL                  Восстановить исход-
                  POP   BC                  ный адрес назначения
                                            и значения позиций.
                  DEC   C                   Перед возвратом умень-
                  INC   HL                  шить номер столбца и
                  RET                       увеличить адрес назначения.
```

Когда используется принтер адрес назначения должен обновиться увеличением на +20.

```
0BD3 PR-ALL-6     EX    AF,A'F'             Опять запись признака
                                            принтера.
                  LD    A,+20               Требуемое значение
                                            приращения.
                  ADD   A,E                 Добавить значение и
                  LD    E,A                 передать результат в рег. E.
                  EX    AF,A'F'             Выбор признака.
                  JR    0BC1,PR-ALL-5       Переход назад в цикл.
```

ПОДПРОГРАММА 'SET ATTRIBUTE BYTE' ('Установить байт атрибута')
Идентифицируется и выбирается соответствующий байт атрибута. Новое значение формируется с помощью обработки старого значения, ATTR-T, MASK-T и P-FLAG. В конце новое значение копируется в область атрибутов.

```
0BDB PO-ATTR      LD    A,H                 Старший байт адреса назна-
```

```
                  RRCA                    чения делится на 8 и
                  RRCA                    обрабатывается AND с +03,
                  RRCA                    чтобы определить, какая
                  AND   +03               треть экрана была адресо-
                                          вана, т.е. 00, 01 или 02.
                  OR    +58               Затем формируется старший
                  LD    H,A               байт для области атрибутов.
                  LD    DE,(ATTR-T)       D содержит ATTR-T, а
                                          B содержит MASK-T.
                  LD    A,(HL)            Старое значение атрибута.
                  XOR   E                 Берутся в расчет зна-
                  AND   D                 чения MASK-T и ATTR-T.
                  XOR   E                 Если не работаете с PAPER 9,
                  BIT   6,(P-FLAG)        переход
                  JR    Z,0BFA,PO-ATTR-1  вперед.
                  AND   +C7               Игнорируется старый цвет
                  BIT   2,A               фона; зависимость от цвета
                                          символов: темные или светлые.
                  JR    NZ,0BFA,PO-ATTR-1 Новый цвет фона будет черным
                  XOR   +38               (000) или белым (111).
0BFA PO-ATTR-1    BIT   4,(P-FLAG)        Если не работаете с
                  JR    Z,0C08,PO-ATTR-2  INK 9, переход вперед.
                  AND   +F8               Игнорируется старый цвет
                                          изображения; зависимость
                                          от цвета фона: светлый или
                  BIT   5,A               темный.
                  JR    NZ,0C08,PO-ATTR-2 Новый цвет символов
                  XOR   +07               будет черным (000) или белым
                                          (111).
0C08 PO-ATTR-2    LD    (HL),A            Ввести новое значение
                  RET                     атрибута и возврат.
```

ПОДПРОГРАММА 'MESSAGE PRINTING' ('Печать сообщений')
Эта подпрограмма используется для печати сообщений и токенов. Регистр A содержит 'входной номер' сообщения или токена в таблице. Регистровая пара DE содержит базовый адрес таблицы.

```
0C0A PO-MSG       PUSH  HL               Старший байт последнего
                  LD    H,+00            элемента на машинном стеке
                  EX    (SP),HL          задан 0, чтобы подавить
                                         конечные пробелы (см.ниже).
                  JR    0C14,PO-TABLE    Переход вперед.
```

При обработке кодов токенов ввод осуществляется в этом месте.

```
0C10 PO-TOKENS    LD    DE,+0095         Базовый адрес таблицы токенов.
                  PUSH  AF               Записать код на стек. (Диапа-
                                         зон +00 - +5A: RND-COPY).
```

Исследуется таблица и печатается правильный элемент.

```
0C14 PO-TABLE     CALL  0C41,PO-SEARCH   Расположить запрошенный
                                         элемент.
                  JR    C,0C22,PO-EACH   Печать сообщения (токена).
                  LD    A,+20            Если требуется, бу-
                  BIT   0,(FLAGS)        дет напечатан 'пробел'
                  CALL  Z,0C3B,PO-SAVE   перед сообщением/токеном.
```

Символы сообщений/токенов печатаются по очереди.

```
0C22 PO-EACH      LD    A,(DE)              Выбрать код.
                  AND   +7F                 Отменить любой 'инвертиро-
                                            ванный разряд'.
                  CALL  0C3B,PO-SAVE        Печать символа.
                  LD    A,(DE)              Снова выбрать код.
                  INC   DE                  Продвинуть указатель.
                  ADD   A,A                 'Инвертированный разряд' идет
                  JR    NC,0C22,PO-EACH     к признаку переноса и
                                            сигнализирует о конце
                                            сообщения/токена,
                                            в противном случае переход назад.
```

Теперь рассмотрим, требуются ли 'конечные пробелы'.

```
                  POP   DE                  Для сообщений - D содержит
                                            +00; для токенов - D содержит
                                            +00 - +5A.
                  CP    +48                 Если последний символ
                  JR    Z,0C35,PO-TR-SP     был '$', переход вперед.
                  CP    +82                 Если последний символ перед
                  RET   C                   'A' был другой - возврат.
0C35 PO-TRSP      LD    A,D                 Проверить значение в D и,
                  CP    +03                 если оно отображает сообще-
                  RET   C                   ние, RND, INKEY$ или PI,
                                            возврат.
                  LD    A,+20               Во всех других случаях пот-
                                            ребуется 'конечный пробел'.
```

ПОДПРОГРАММА 'PO-SAVE'
Эта подпрограмма позволяет 'рекурсивно' печатать символы. Пока вызывается 'PRINT-OUT' записываются соответствующие регистры.

```
0C3B PO-SAVE      PUSH  DE                  Записать регистровую пару DE.
                  EXX                       Записать HL и BC.
                  RST   0010,PRINT-A-1      Печать одного символа.
                  EXX                       Восстановить HL и BC.
                  POP   DE                  Восстановить DE.
                  RET                       Окончание.
```

ПОДПРОГРАММА 'TABLE SEARCH' ('Поиск таблицы')
Подпрограмма возвращается с регистровой парой DE, указывающей на начальный символ требуемого элемента, и сброшенным признаком переноса, если рассматривается 'начальный пробел'.

```
0C41 PO-SEARCH    PUSH  AF                  Записать 'номер элемента'.
                  EX    DE,HL               Теперь HL содержит базовый
                                            адрес.
                  INC   A                   Сделать диапазон +01 - ?.
0C44 PO-STEP      BIT   7,(HL)              Ждите 'инвертирован-
                  INC   HL                  ный символ'.
                  JR    Z,0C44,PO-STEP
                  DEC   A                   Счет элементов, пока
                  JR    NZ,0C44,PO-STEP     не будет найден правильный.
                  EX    DE,HL               DE указывает на начальный
                                            символ.
```

```
                POP   AF                Выбор 'номера элемента'
                CP    +20               и возврат со сброшенным
                RET   C                 признаком переноса для
                                        первых 32 элементов.
                LD    A,(DE)            Однако, если начальный
                SUB   +41               символ является буквой, то
                RET                     может быть необходим
                                        начальный пробел.
```

ПОДПРОГРАММА 'TEST FOR SCROLL' ('Тест для прокрутки')
Эта подпрограмма вызывается в случае необходимости прокрутки информации на дисплее. Это возможно в 3 случаях: 1) когда обрабатывается символ 'возврат каретки'; 2) когда используется AT в строке INPUT; 3) когда заполнена текущая строка и должна использоваться следующая.
На входе регистр B содержит номер строки.

```
0C55 PO-SCR     BIT   1,(FLAGS)         Немедленный возврат, если
                RET   NZ                принтер уже использовался.
                LD    DE,+0DD9          Перезагрузка машинного
                PUSH  DE                стека адресом 'CL-SET'.
                LD    A,B               Передать номер строки.
                BIT   0,(TV-FLAG)       Если рассматривается 'INPUT ... AT ..',
                JP    NZ,0D02,PO-SCR-4  переход вперед.
                                        Возврат через CL-SET, если
                CP    (DF-SZ)           номер строки больше значения
                JR    C,0C86,REPORT-6   DF-SZ; если меньше, выдать
                RET   NZ                сообщение 5; иначе,
                                        продолжить.
                BIT   4,(TV-FLAG)       Если не работаете с
                JR    Z,0C88,PO-SCR-2   'автоматической распечат-
                                        кой', переход вперед.
                LD    E,(BREG)          Выбор счетчика строк.
                DEC   E                 Уменьшить этот счетчик.
                JR    Z,0CD2,PO-SCR-3   Если должна быть просмотрена
                                        распечатка, переход вперед.
                LD    A,+00             Иначе, открыть
                CALL  1601,CHAN-OPEN    канал 'K', восстановить
                LD    SP,(LIST-SP)      указатель стека, признак,
                RES   4,(TV-FLAG)       который закончил
                RET                     автоматический просмотр и
                                        вернуться через CL-SET.
```

Сообщение 5 - 'Вне экрана'.

```
0C86 REPORT-5   RST   0008,ERROR-1      Вызов подпрограммы
                DEFB  +04               обработки ошибок.
```

Теперь рассмотрим, требуется ли приглашение 'scroll?' ('просмотреть').

```
0C88 PO-SCR-2   DEC   (SCR-CT)          Уменьшить счетчик просмотров
                JR    NZ,0CD2,PO-SCR-3  и продолжать вызывать
                                        приглашение.
```

Продолжайте выдавать сообщение о приглашении.

```
                LD    A,+18             Счетчик сброшен.
```

```
                SUB   B
                LD    (SCR-CT),A
                LD    HL,(ATTR-T)         Записаны текущие зна-
                PUSH  HL                  чения ATTR-T и MASK-T.
                LD    A,(P-FLAG)          Записано текущее зна-
                PUSH  AF                  чение P-FLAG.
                LD    A,+FD               Открыт канал 'K'.
                CALL  1601,CHAN-OPEN
                XOR   A                   Сообщение 'scroll?' явл.
                LD    DE,+0CF8            сообщением '0'.
                CALL  0C0A,PO-MSG         Теперь это сообщение
                                          напечатано.
                SET   5,(TV-FLAG)         Сигнал - 'очистить нижнюю
                                          часть экрана после нажатия
                                          клавиши'.
                LD    HL,+5C3B            Это FLAGS.
                SET   3,(HL)              Сигнал - 'режим L'.
                RES   5,(HL)              Сигнал - 'нет еще клавиши'.
                EXX                       Примечание: DE должно быть
                                          помещено на стек.
                CALL  15D4,WAIT-KEY       Выбор одного клавишного кода.
                EXX                       Восстановить регистры.
                CP    +20                 Если была нажата кла-
                JR    Z,0D00,REPORT-D     виша 'BREAK','STOP',
                CP    +E2                 'N' или 'n' переход
                JR    Z,0D00,REPORT-D     вперед на REPORT-D -
                OR    +20                  'BREAK - CONT repeats' в
                CP    +6E                 противном случае
                JR    Z,0D00,REPORT-D     нажата клавиша для
                                          просмотра изображения.
                LD    A,+FE               Открыть канал 'S'.
                CALL  1601,CHAN-OPEN
                POP   AF                  Восстановить значе-
                LD    (P-FLAG),A          ние P-FLAG.
                POP   HL                  Восстановить значе-
                LD    (ATTR-T),HL         ние ATTR-T и MASK-T.
```

Теперь просмотр изображения.

```
0CD2 PO-SCR-3   CALL  0DFE,CL-SC-ALL      Просмотрено все изображение.
                LD    B,(DF-SZ)           Найдены и записаны номера
                INC   B                   строки и столбца для
                LD    C,+21               начала строки над нижней
                PUSH  BC                  частью изображения.
                CALL  0E9B,CL-ADDR        Теперь находится
                LD    A,H                 соответствующий байт
                RRCA                      атрибутов для этой
                RRCA                      области символов.
                RRCA                      Регистровая пара HL
                AND   +03                 содержит адрес байта.
                OR    +58
                LD    H,A
```

Рассматриваемая строка будет иметь значения атрибутов 'нижней части', а новая строка в нижней части изображения может иметь значения 'ATTR-P', поэтому значения атрибутов меняются.

```
                LD    DE,+5AE0            DE указывает на первый байт
                                          атрибута нижней строки.
```

```
                  LD    A,(DE)              Выбирается значение.
                  LD    C,(HL)              Значение 'нижней части'.
                  LD    B,+20               Имеется 32 байта.
                  EX    DE,HL               Обмен указателей.
0CF0 PO-SCR-3A    LD    (DE),A              Сделать первый обмен
                  LD    (HL),C              и затем продолжать
                  INC   DE                  использовать то же
                  INC   HL                  значение для 32 байт
                  DJNZ  0CF0,PO-SCR-3A      атрибутов двух уже
                                            обработанных строк.
                  POP   BC                  Номера строки и столбца
                                            нижней строки 'верхней
                                            части' выбираются перед
                  RET                       возвратом.

СООБЩЕНИЕ 'scroll?' ('прокрутка?')

0CF8              DEFB  +80                 Перейти через начальный
                                            маркер.
                  DEFB  +73,+63,+72,+6F     s-c-r-o
                  DEFB  +6C,+6C,+BF         l-l-? (инвертированный).

Сообщение D - 'BREAK - CONT repeats'.

0D00 REPORT-D     RST   0008,ERROR-1        Вызов подпрограммы
                  DEFB  +0C                 обработки ошибок.

Нижняя часть изображения обрабатывается следующим образом:

0D02 PO-SCR-4     CP    +02                 Выдается ошибка 'вне
                  JR    C,0C86,REPORT-5     экрана', если нижняя часть
                  ADD   A,(DF-SZ)           ожидается 'слишком большой'
                  SUB   +19                 и делается возврат, если
                  RET   NC                  необходима прокрутка.
                  NEG                       Теперь регистр A будет со-
                                            держать 'число подлежащих
                                            выполнению просмотров'.
                  PUSH  BC                  Теперь записываются номера
                                            строки и столбца.
                  LD    B,A                 Записываются номера
                  LD    HL,(ATTR-T)         просмотра, ATTR-T,
                  PUSH  HL                  MASK-T и P-FLAG.
                  LD    HL,(P-FLAG)
                  PUSH  HL
                  CALL  0D40,TEMPS          Должны использоваться эле-
                                            менты 'постоянного' цвета.
                  LD    A,B                 Выбирается 'номер прокрутки'.

Нижняя часть экрана теперь прокручивается 'A' число раз.

0D1C PO-SCR-4A    PUSH  AF                  Записать 'число'.
                  LD    HL,+5C6B            Это DF-SZ.
                  LD    B,(HL)              Значение в DF-SZ увеличи-
                  LD    A,B                 вается, устанавливается
                  INC   A                   регистр B, чтобы содержать
                  LD    (HL),A              значение формирователя, и
                                            регистр A - новое значение.
                  LD    HL,+5C89            Это S-POSN-hi.
                  CP    (HL)                Переход, если должна быть
```

```
                JR    C,0D2D,PO-SCR-4B    прокручена только нижняя часть
                                          изображения (B = старое DF-SZ).
                INC   (HL)                Иначе S-POSN-hi увеличивается
                LD    B,+18               и прокручивается всё изображение
                                          (B = +18).
0D2D PO-SCR-4B  CALL  0E00,CL-SCROLL      Прокрутить 'B' строк.
                POP   AF                  Выбрать и уменьшить
                DEC   A                   'номер прокрутки'.
                JR    NZ,0D1C,PO-SCR-4A   Пока не закончено, переход
                                          назад.
                POP   HL                  Восстановить значение
                LD    (P-FLAG),L          P-FLAG.
                POP   HL                  Восстановить значения ATTR-T
                LD    (ATTR-T),HL         и MASK-T.
                LD    BC,(S-POSN)         В случае, когда было изменено
                RES   0,(TV-FLAG)         S-POSN, вызывается CL-SET,
                CALL  0DD9,CL-SET         чтобы выдать значение
                                          отождествления  в DF-CC.
                SET   0,(TV-FLAG)         Сбросить признак, отображающий,
                POP   BC                  что нижняя часть экрана уже
                RET                       обработана, выбрать номера строки
                                          и столбца, а затем возврат.
```

ПОДПРОГРАММА 'TEMPORARY COLOUR ITEMS' ('Временные элементы цвета')
Это очень важная подпрограмма. Она используется всякий раз, когда требуются 'постоянные' детали для копирования во 'временную' системную переменную. Сначала рассмотрим ATTR-T и MASK-T.

```
0D4D TEMPS      XOR   A                   Заносим +00 в регистр A.
                LD    HL,(ATTR-P)         Берём текущие значения ATTR-P
                BIT   0,(TV-FLAG)         и MASK-P.
                JR    Z,0D5B,TEMPS-1      Если обрабатывается основная
                                          часть экрана, то переход вперёд.
                LD    H,A                 В противном случае использовать
                LD    L,(BORDCR)          +00 и значение BORDER.
0D5B TEMPS-1    LD    (ATTR-T),HL         Теперь установить ATTR-T и MASK-T.
```

Далее рассматривается P-FLAG.

```
                LD    HL,+5C91            Это P-FLAG.
                JR    NZ,0D65,TEMPS-2     Если обрабатывается нижняя часть
                                          экрана (A= +00), то переход
                                          вперёд.
                LD    A,(HL)              В противном случае выбрать
                RRCA                      значение P-FLAG и переслать
                                          нечётные разряды к чётным.
0D65 TEMPS-2    XOR   (HL)                Продолжать копировать чётные
                AND   +55                 разряды A в P-FLAG.
                XOR   (HL)
                LD    (HL),A
                RET
```

ПРОГРАММА 'CLS COMMAND' ('Очистка экрана')
В первой проверке весь дисплей очищается – пикселы все сбрасываются, и байты атрибутов устанавливаются равными значению в ATTR-P, затем преобразуется нижняя часть экрана дисплея.

```
0D6B CLS          CALL  0DAF,CL-ALL         Очищается весь дисплей
0D6E CLS-LOWER    LD    HL,+5C3C            Это TV-FLAG.
                  RES   5,(HL)              Устанавливаем флаг 'после нажатия
                                            клавиши не очищать нижнюю часть
                                            экрана'
                  SET   0,(HL)              Флаг 'нижняя часть'.
                  CALL  0D4D,TEMPS          Использовать постоянные значения,
                                            т.е. ATTR-T копируется из BORDER.
                  LD    B,(DF-SZ)           Нижняя часть экрана теперь
                  CALL  0E44,CL-LINE        очищается с этими значениями.
```

Атрибуты для строк в нижней части изображения, кроме атрибутов для строк 22 и 23, необходимо сделать равными значению из ATTR-P.

```
                  LD    HL,+5AC0            Адрес атрибута в начале строки 22.
                  LD    A,(ATTR-P)          Значение ATTR-P.
                  DEC   B                   Счётчик строк.
                  JR    0D8E,CLS-3          Переход вперёд в цикл.
0D87 CLS-1        LD    C,+20               +20 символов на строку.
0D89 CLS-2        DEC   HL                  Пройти назад вдоль строки,
                  LD    (HL),A              задавая значения атрибутов.
                  DEC   C
                  JR    NZ,0D89,CLS-2
0D8E CLS-3        DJNZ  0D87,CLS-1          Повторяем, пока не закончим все
                                            строки.
```

Размер нижней части изображения можно зафиксировать.

```
                  LD    (DF-SZ),+02         Нижняя часть изображения равна
                                            по высоте двум строкам.
```

Теперь остаётся выполнить некоторые вспомогательные задачи.

```
0D94 CL-CHAN      LD    A,+FD               Открыть канал 'K'.
                  CALL  1601,CHAN-OPEN
                  LD    HL,(CURCHL)         Взять адрес текущего канала и
                  LD    DE,+09F4            сделать адрес вывода +09F4
                  AND   A                   (= PRINT-OUT) и адрес ввода
0DA0 CL-CHAN-A    LD    (HL),E              +10A8 (= KEY-INPUT)
                  INC   HL
                  LD    (HL),D
                  INC   HL
                  LD    DE,+10A8
                  CCF                       Первый адрес вывода, затем
                  JR    C,0DA0,CL-CHAN-A    адрес ввода. Т.к. нижняя часть.
                  LD    BC,+1721            изображения отработана, нижней
                                            строкой печати будет строка 23.
                  JR    0DD9,CL-SET         Возврат через CL-SET.
```

ПОДПРОГРАММА 'CLEARING THE WHOLE DISPLAY AREA' ('Очистка всего дисплейного пространства')
Эта подпрограмма вызывается из:
1) командной процедуры CLS;
2) главной исполняющей программы;
3) программы автоматического вывода распечаток.

```
0DAF CL-ALL       LD    HL,+0000            Установим системную переменную
                  LD    (C00RDS),HL         C00RDS равной 0.
                  RES   0,(FLAGS2)          Флаг 'экран очищен'.
                  CALL  0D94,CL-CHAN        Выполнить 'вспомогательные задачи'
                  LD    A,+FE               Открыть канал 'S'.
                  CALL  1601,CHAN-OPEN
                  CALL  0D4D,TEMPS          Использовать 'постоянные'
                                            значения.
                  LD    B,+18               Очистим 24 строки экрана.
                  CALL  0E44,CL-LINE
                  LD    HL,(CURCHL)         Обеспечивается текущий адрес
                  LD    DE,+09F4            вывода +09F4
                  LD    (HL),E              (PRINT-OUT).
                  INC   HL
                  LD    (HL),D
                  LD    (SCR-CT),+01        Сброс счётчика прокруток.
                  LD    BC,+1821            Т.к. верхняя часть изображения
                                            обработана, 'верхней строкой
                                            печати' будет строка 0.
                                            Продолжение в CL-SET.
```

ПОДПРОГРАММА 'CL-SET'
Входные данные в подпрограмму передаются через регистровую пару BC (номера строк и столбцов областей символов) или через регистр C (номер столбца внутри буфера принтера). Затем находится соответствующий адрес первого разряда символа. Подпрограмма возвращается через PO-STORE так, чтобы запомнить все значения в требуемых системных переменных.

```
0DD9 CL-SET       LD    HL,+5B00            Начальный адрес буфера принтера.
                  BIT   1,(FLAGS)           Если обрабатывается буфер принтера,
                  JR    NZ,0DF4,CL-SET-2    то переход вперёд.
                  LD    A,B                 Передать номер строки.
                  BIT   0,(TV-FLAG)         Если обрабатывается основная часть
                  JR    Z,0DEE,CL-SET-1     изображения, то переход вперёд.
                  ADD   A,(DF-SZ)           Верхняя строка нижней части
                  SUB   +18                 изображения вызывается как
                                            'строка +18' и должна быть
                                            преобразована.
0DEE CL-SET-1     PUSH  BC                  Сохраняем номера строк и столцов
                  LD    B,A                 Перемещаем номер строки.
                  CALL  0E9B,CL-ADDR        Адрес для начала строки
                                            формируется в HL.
                  POP   BC                  Восстанавливаем номера строк
                                            и столцов.
0DF4 CL-SET-2     LD    A,+21               Номер столбца теперь
                  SUB   C                   отреверсирован и передан в
                  LD    E,A                 регистровую пару DE.
```

```
                LD    D,+00
                ADD   HL,DE               Теперь сформирован требуемый адрес;
                JP    0ADC,PO-STORE       и адрес, и номера строк и столбцов
                                          запоминаются с переходом в
                                          PO-STORE.
```

ПОДПРОГРАММА 'SCROLLING' ('Прокрутка')
Номера строк изображения, которые надо прокрутить, должны содержаться на входе основной програмы в регистре B.

```
0DFE CL-SC-ALL  LD    B,+17               Точка входа после прокрутки
                                          'scroll?'.
```

Основная точка входа - сверху и при прокрутке INPUT..AT.

```
0E00 CL-SCROLL  CALL  0E9B,CL-ADDR        Найти начальный адрес строки.
                LD    C,+08               Восьмипиксельные строки в полную
                                          строку.
```

Теперь введём основной цикл прокрутки.
Входные данные:
- регистр B содержит число верхних строк, которые должны быть прокручены;
- регистровая пара HL - начальный адрес в области изображения этой строки;
- регистр C - счётчик пиксел-строк.

```
0E05 CL-SCR-1   PUSH  BC                  Сохраним оба счётчика.
                PUSH  HL                  Сохраним начальный адрес.
                LD    A,B                 Переход вперёд, если в настоящий
                AND   +07                 момент не работаем с 'третью'
                LD    A,B                 изображения.
                JR    NZ,0E19,CL-SCR-3
```

Пиксел-строки верхней строки 'трети' изображения должны быть перемещены через 2-килобайтовые границы (каждая 'треть' экрана занимает 2 КБ).

```
0E0D CL-SCR-2   EX    DE,HL               Результат этих манипуляций
                LD    HL,+F8E0            оставляет HL неизвенной, а
                ADD   HL,DE               DE, указывает требуемый адрес
                EX    DE,HL               назначения.
                LD    BC,+0020            Имеется +20 символов.
                DEC   A                   Уменьшить счётчик, т.к.
                                          обработана одна строка.
                LDIR                      Переместить 32 байта.
```

Теперь можно прокрутить пиксел-строки внутри 'трети' экрана. Регистр A содержит на первом проходе +01 - +07, +09 - +0F или +11 - +17.

```
0E19 CL-SCR-3   EX    DE,HL               DE указывет требуемый адрес
                LD    HL,+FFE0            назначения.
                ADD   HL,DE               На этот раз сдвиг только 32 ячеек.
                EX    DE,HL               Записать в B конец строки.
                LD    B,A                 Теперь находим количество
                AND   +07                 символов, остающихся в 'трети'
                RRCA                      экрана.
                RRCA
                RRCA
```

```
                LD    C,A               Передать сумму символов в
                                        регистр C.
                LD    A,B               Выбрать номер строки.
                LD    B,+00             BC содержит 'сумму символов' и
                LDIR                    пиксел-строку от каждого
                                        просмотренного символа.
                LD    B,+07             Теперь подготовим для увеличения
                                        адрес, чтобы перейти границу
                                        'трети' экрана.
                ADD   HL,BC             Увеличить HL на +0700.
                AND   +F8               Если остались какие-либо
                JR    NZ,0E0D,CL-SCR-2  'трети' для рассмотрения, то
                                        переход назад.
```

Теперь, если цикл был использован 8 раз, находим один раз для каждой пиксел-строки.

```
                POP   HL                Восстановить исходный адрес.
                INC   H                 Адресовать следующую пиксел-строку.
                POP   BC                Восстановить счётчик.
                DEC   C                 Уменьшить счётчик пиксел-строки
                JR    NZ,0E05,CL-SCR-1  и перейти назад, если не
                                        перемещены 8 строк.
```

Прокручены следующие байты атрибутов Отметим, что регистр B ещё содержит число строк, подлежащих просмотру, а регистр C содержит 0.

```
                CALL  0E88,CL-ATTR      Найден требуемый адрес в области
                                        атрибутов и число символов в
                                        'B' строках.
                LD    HL,+FFE0          Смещением для всех байтов
                ADD   HL,DE             атрибутов является сдвиг 32 ячеек
                EX    DE,HL
                LDIR                    Прокручиваем байты атрибутов.
```

Теперь остаётся только почистить нижнюю строку изображения.

```
                LD    B,+01             Регистр B загружается с +01 и
                                        вводится CL-LINE.
```

ПОДПРОГРАММА 'CLEAR LINES' ('Очистка строк')
Эта подпрограмма очищает нижние B строк экрана.

```
0E44 CL-LINE    PUSH  BC                Сохраняем количество строк
                                        и номер строки (регистр C).
                CALL  0E9B,CL-ADDR      Получаем в HL начальный адрес
                                        строки.
                LD    C,+08             Работаем с восемью пиксел-
                                        строками.
```

Цикл для очистки всех пиксел-строк.

```
0E4A CL-LINE-1  PUSH  BC                Сохраним номер строки и счётчик
                                        пиксел-строк.
```

```
                PUSH  HL                 Сохраним адрес строки.
                LD    A,B                Номер строки переносим в A.
0E4D CL-LINE-2  AND   +07                Найти сколько символов
                RRCA                     содержится в 'B mod 8' строках.
                RRCA                     Передать результат в регистр C
                RRCA                     (в нём сейчас содержится +00,
                LD    C,A                т.е. 256dec. для 'трети' экрана)
                LD    A,B                Возьмём номер строки.
                LD    B,+00              В регистровую пару BC запишем
                DEC   C                  число символов-1
                LD    D,H                DE будет указывать на первый
                LD    E,L                символ.
                LD    (HL),+00           Очистить пиксел-байт первого
                                         символа.
                INC   DE                 DE указывает на следующий символ.
                LDIR                     Очистить пиксел-байты следующих
                                         символов.
                LD    DE,+0701           Для каждой трети изображения адрес
                ADD   HL,DE              в HL должен увеличиваться на +0701.
                DEC   A                  Уменьшим номер строки.
                AND   +F8                Убрать любые дополнительные строки
                LD    B,A                и передать подсчёт 'третей' в B.
                JR    NZ,0E4D,CL-LINE-2  Переход назад, если ещё необходимо
                                         работать с 'третями' экрана.
```

Цикл повторяется 8 раз.

```
                POP   HL                 Восстановим адрес пиксель-строки.
                INC   H                  Адрес следующей пиксель-строки.
                POP   BC                 Восстановим счётчик пиксель-строк.
                DEC   C                  Уменьшить счётчик.
                JR    NZ,0E4A,CL-LINE-1  Если ещё не все строки очистили,
                                         то переход назад.
```

Если требуется, устанавливаются следующие байты атрибутов. Значение в ATTR-P будет использоваться, когда обрабатывается основная часть изображения, а значение в BORDER - когда обрабатывается нижняя часть экрана.

```
                CALL  0E88,CL-ATTR       Получим адрес первого байта
                                         атрибута и число байт.
                LD    H,D                HL будет указывать на первый
                LD    L,E                байт атрибута, в DE - на второй.
                INC   DE
                LD    A,(ATTR-P)         Возьмём значение ATTR-P.
                BIT   0,(TV-FLAG)        Если обрабатывается основная часть
                JR    Z,0E80,CL-LINE-3   экрана, то переход вперёд.
                LD    A,(BORDER)         В противном случае используем
                                         BORDER.
0E80 CL-LINE-3  LD    (HL),A             Запишем байт атрибута.
                DEC   BC                 Обработали один байт.
                LDIR                     Теперь значение атрибута копируем
                                         во все остальные байты атрибутов.
                POP   BC                 Восстановим номер строки.
                LD    C,+21              Установить номер столбца в левый
                RET                      столбец и возврат.
```

ПОДПРОГРАММА 'CL-ATTR'
Подпрограмма выполняет две функции:

1) Для данного адреса области изображения в HL возвращает в DE адрес соответствующего атрибута. Отметим, что значение адреса на входе должно указывать на 'девятую' строку символов.
2) Для номера строки в регистре B возвращает в BC количество символов экрана от начала экрана и до указанной строки.

```
0E88 CL-ATTR    LD    A,H               Старший байт адреса.
                RRCA                    Умножаем на 32
                RRCA
                RRCA
                DEC   A                 Возвращаемся к строке 8.
                OR    +50               Адресовать область атрибутов.
                LD    H,A               Полученное значение занесём в
                                        старший байт адреса
                EX    DE,HL             и перенесём адрес в DE.
                LD    H,C               На входе C=0.
                LD    L,B               Номер строки.
                ADD   HL,HL             Умножим на 32.
                ADD   HL,HL
                ADD   HL,HL
                ADD   HL,HL
                ADD   HL,HL
                LD    B,H               Результат перенесём в BC
                LD    C,L
                RET
```

ПОДПРОГРАММА 'CL-ADDR'
Возвращает в HL экранный адрес строки, номер которой указан в регистре B.

```
0E9B CL-ADDR    LD    A,+18             Преобразуем номер строки на
                SUB   B                 обратную нумерацию.
                LD    D,A               Сохраним результат в D.
                RRCA                    В действительности
                                        получим '(A mod 8) * 32'.
                RRCA                    В каждой трети экрана младший
                RRCA                    байт каждой строки будет такой:
                AND   +E0               1-я строка = +00,
                                        2-я строка = +20, и т.д.
                LD    L,A               Младший байт сохраним в L.
                LD    A,D               Берём истинный номер строки.
                AND   +18               В действительности
                                        '64 +8 * INT (A/8)'
                OR    +40               Для верхней трети экрана старший
                                        байт равен = +40,
                                        для средней трети 'third' = +48,
                                        и для нижней трети = +50.
                LD    H,A               Старший байт адреса занесём в H.
                RET
```

КОМАНДНАЯ ПРОЦЕДУРА 'COPY' ('Копировать')
176 пиксель-строк обрабатываются одна за одной.

```
0EAC COPY         DI                      Запретить прерывания.
                  LD    B,+B0             176 строк.
                  LD    HL,+4000          Начальный адрес экранной области.

Начало цикла.

0EB2 COPY-1       PUSH  HL                Сохраним адрес.
                  PUSH  BC                Сохраним номер строки.
                  CALL  0EF4,COPY-LINE    Подпрограмма вызывается 176 раз.
                  POP   BC                Восстановим номер строки и
                  POP   HL                экранный адрес.
                  INC   H                 Адрес следующей строки
                  LD    A,H               Переход вперёд и отсюда опять
                  AND   +07               прогон цикла для 8 пиксел-строк
                  JR    NZ,0EC9,COPY-2    одной строки символов.

Для каждой новой строки символов должен обновляться адрес этой строки.

                  LD    A,L               Младший байт адреса.
                  ADD   A,+20             Увеличиваем на 32 символа.
                  LD    L,A               Если мы находимся внутри 'трети'
                                          экрана, то флаг C будет сброшен.
                  CCF                     Инвертируем флаг C.
                  SBC   A,A               Регистр A будет содержать +F8,
                  AND   +F8               когда адрес находится внутри
                                          'трети' экрана, и +00, если
                                          достигнута следующая 'треть'.
                  ADD   A,H               Новое значение старшего байта
                  LD    H,A               адреса.
0EC9 COPY-2       DJNZ  0EB2,COPY-1       Возврат, пока не перебраны все
                                          176 строк экрана.
                  JR    0EDA,COPY-END     Переход в конец программы.

ПОДПРОГРАММА 'COPY-BUFF'('Копирование в буфер')
Эта подпрограмма вызывается всякий раз, когда буфер принтера готов принять информацию.

0ECD COPY-BUFF    DI                      Запрет прерываний.
                  LD    HL,+5B00          Адрес буфера принтера.
                  LD    B,+08             8 пиксел-строк
0ED3 COPY-3       PUSH  BC                Сохраним номер строки.
                  CALL  0EF4,COPY-LINE    Подпрограмма вызывается 8 раз.
                  POP   BC                Восстановим номер строки.
                  DJNZ  0ED3,COPY-3       Переход назад, пока не напечатаем
                                          8 строк.

Продолжение в программе COPY-END.

0EDA COPY-END     LD    A,+04             Остановка двигателя принтера.
```

```
                OUT   (+FB),A
                EI                      Разрешить прерывания.
                                        Продолжение в подпрограмме
                                        CLEAR-PRB.
```

ПОДПРОГРАММА 'CLEAR PRINTER BUFFER' ('Очистить буфер принтера')
С помощью этой подпрограммы очищается буфер принтера.

```
0EDF CLEAR-PRB  LD    HL,+5B00          Адрес буфера принтера.
                LD    (PR-CC-lo),L      Сброс столбца принтера.
                XOR   A                 Очистка регистра A.
                LD    B,A               Заодно очистка регистра B
                                        (на самом деле считаем, что
                                        регистр B содержит 256)
0EE7 PRB-BYTES  LD    (HL),A            Очищение 256 байт буфера
                INC   HL                принтера
                DJNZ  0EE7,PRB-BYTES
                RES   1,(FLAGS2)        Установим флаг 'буфер принтера
                                        пуст'.
                LD    C,+21             Установить позицию принтера и
                JP    0DD9,CL-SET       возврат через CL-SET & P0-STORE.
```

ПОДПРОГРАММА 'COPY-LINE'
На входе в подпрограмму в HL должен быть адрес 32-байтной пиксел-строки, а в регистре B должен содержаться номер пиксел-строки.

```
0EF4 COPY-LINE  LD    A,B               Копируем номер пиксел-строки.
                CP    +03               В регистре A будет +00, пока не
                SBC   A,A               будут обработаны две последние
                AND   +02               строки.

                OUT   (+FB),A           Замедлить двигатель принтера только
                                        для двух последних пиксел-строк.
                LD    D,A               Регистр D будет содержать или
                                        +00 или +02.
```

Перед любой печатью должны быть сделаны три теста.

```
0EFD COPY-L-1   CALL  1F54,BREAK-KEY    Если не была нажата клавиша
                JR    C,0F0C,COPY-L-2   BREAK, то переход вперёд.
                LD    A,+04             При нажатии BREAK остановить
                OUT   (+FB),A           двигатель,
                EI                      разрешить прерывания,
                CALL  0EDF,CLEAR-PRB    очистить буфер принтера и выйти
                RST   0008,ERROR-1      чере подпрограмму обработки
                DEFB  +0C               ошибок с сообщением
                                        'BREAK-CONT repeats'.
0F0C COPY-L-2   IN    A,(+FB)           Чтение статуса принтера
                ADD   A,A
                RET   M                 Если принтер не готов, или
                                        его нет, то возврат.
                JR    NC,0EFD,COPY-L-1  Ожидание готовности принтера.
                LD    C,+20             32 байта.
```

Для обработки этих 32 байт делаем цикл.

```
0F14 COPY-L-3     LD    E,(HL)              Выбор байта.
                  INC   HL                  Указатель в буфере принтера
                                            установим на следующий байт.
                  LD    B,+08               8 бит в одном байте.
0F18 COPY-L-4     RL    D                   Сдвиг D влево.
                  RL    E                   Сдвиг 7 бита E в флаг переноса.
                  RR    D                   Сдвиг D обратно с учётом флага
                                            переноса (с учётом 7 бита из E).
0F1E COPY-L-5     IN    A,(+FB)             Чтение статуса принтера и
                  RRA                       ожидание сигнала от кодирующего
                  JR    NC,0F1E,COPY-L-5    устройства.
                  LD    A,D                 Передача байта в принтер.
                  OUT   (+FB),A             Управляющие биты:
                                            бит 2=0 - старт двигателя
                                            бит 1=1 - замедление двигателя
                                            бит 7=1 - непосредственно печать
                  DJNZ  0F18,COPY-L-4       Печатать каждый бит.
                  DEC   C                   Уменьшить счётчик байтов.
                  JR    NZ,0F14,COPY-L-3    Переход назад, если есть ещё
                  RET                       байты для печати.
```

ПРОГРАММА 'EDITOR' ('Редактор')
Редактор вызывается в двух случаях:
1) Из основной исполняющей программы для того, чтобы пользователь мог ввести BASIC-строку в программу;
2) Из командной строки при работе оператора INPUT.

Сначала сохраняется 'указатель стека ошибок' и вместо него подставляется альтернативный адрес.

```
0F2C EDITOR       LD    HL,(ERR-SP)         Текущее значение сохраняется
                  PUSH  HL                  в стеке.
0F30 ED-AGAIN     LD    HL,+107F            Обработчик ED-ERROR.
                  PUSH  HL                  Любое событие, приводящее к
                  LD    (ERR-SP),SP         использованию подпрограммы
                                            обработки ошибок, будет передавать
                                            управление в ED-ERROR.
```

Для обработки каждого нажатия клавиши используется цикл.

```
0F38 ED-LOOP      CALL  15D4,WAIT-KEY       Возврат, если нажата клавиша.
                  PUSH  AF                  Сохраним код клавиши.
                  LD    D,+00               Выбор длительности срабатывания
                  LD    E,(PIP)             клавиатуры (нажать и отпустить).
                  LD    HL,+00C8            Выбор тона звучания клавиатуры.
                  CALL  03B5,BEEPER         Делаем звук 'пип'.
                  POP   AF                  Восстановить код нажатой клавиши.
                  LD    HL,+0F38            Сохранить на стек адрес ED-LOOP.
                  PUSH  HL
```

Теперь анализируем полученный код клавиши.

```
                CP    +18               Получить все коды символов,
                JR    NC,0F81,ADD-CHAR  коды графических символов
                                        и токены.
                CP    +07               Также получить ','.
                JR    C,0F81,ADD-CHAR
                CP    +10               Переход вперёд; код представляет
                JR    C,0F92,ED-KEYS    клавишу редактирования.
```

Рассмотрим управляющие клавиши - INK to TAB.

```
                LD    BC,+0002          INK и PAPER потребуют
                                        две ячейки.
                LD    D,A               Скопировать код в D.
                CP    +16               Переход вперёд с INK и
                JR    C,0F6C,ED-CONTR   PAPER
```

AT и TAB обрабатываются следующим образом:

```
                INC   BC                Требуются три ячейки.
                BIT   7,(FLAGX)         Переход вперёд, если не
                JP    Z,101E,ED-IGNORE  обрабатывается INPUT LINE... .
                CALL  15D4,WAIT-KEY     Получить второй код
                LD    E,A               и поместить его в E.
```

Выбираем другие байты для управляющих символов.

```
0F6C ED-CONTR   CALL  15D4,WAIT-KEY     Получить другой код.
                PUSH  DE                Сохранение предыдущих кодов.
                LD    HL,(K-CUR)        Выбор K-CUR.
                RES   0,(MODE)          Сигнал 'режим K'.
                CALL  1655,MAKE-ROOM    Создать два или три места.
                POP   BC                Восстановить предыдущие коды.
                INC   HL                Указать первую ячейку.
                LD    (HL),B            Ввести первый код.
                INC   HL                Затем ввести второй код,
                LD    (HL),C            который будет наложен, если
                                        имеются 2 кода, т.е.
                                        с INK и PAPER.
                JR    0F8B,ADD-CH-1     Переход вперёд.
```

ПОДПРОГРАММА 'ADDCHAR' ('Добавление знака')
Эта подпрограмма добавляет код к текущей строке EDIT или INPUT.

```
0F81 ADD-CHAR   RES   0,(MODE)          Сигнал 'режим K'.
                LD    HL,(K-CUR)        Выбор позиции курсора.
                CALL  1652,ONE-SPACE    Создание единичного места.
0F8B ADD-CH-1   LD    (DE),A            Ввести на это место код и
                INC   DE                сигнализировать о том, что курсор
                LD    (K-CUR),DE        занимает следующую ячейку.
                RET                     Затем непрямой возврат в ED-LOOP.
```

Клавиши редактирования обрабатываются следующим образом:

```
0F92 ED-KEYS    LD    E,A               Передаём код в DE.
                LD    D,+00
```

```
                LD    HL,+0F99          Начальный адрес таблицы клавиш.
                ADD   HL,DE             Адресуем нужный элемент таблицы
                LD    E,(HL)            и заносим его значение в E.
                                        Элемент таблицы является адресом
                                        смещения искомой подпрограммы
                                        обработки относительно текущего
                                        элемента таблицы.
                ADD   HL,DE             Суммируем смещение с текущим
                                        адресом в таблице и получаем адрес
                                        подпрограммы обработки,
                PUSH  HL                который кладём на стек.
                LD    HL,(K-CUR)        Положение курсора заносим в HL
                RET                     и переходим на требуемую
                                        подпрограмму.
```

ТАБЛИЦА 'EDITING KEYS' ('Клавиши редактирования')

Адрес	Смещение	Символ	Адрес	Смещение	Символ
0FA0	09	EDIT	0FA5	70	DELETE
0FA1	66	cursor left	0FA6	7E	ENTER
0FA2	6A	cursor right	0FA7	CF	SYMBOL SHIFT
0FA3	50	cursor down	0FA8	D4	GRAPHICS
0FA4	85	cursor up			

ПОДПРОГРАММА 'EDIT KEY' ('Клавиша EDIT')
В режиме редактирования нажатая клавиша EDIT будет обрабатывать текущую BASIC-строку, а в режиме INPUT нажатая клавиша EDIT вызовет очистку текущей введенной информации и ввод новых символов.

```
0FA9 ED-EDIT    LD    HL,(E-PPC)        Текущий номер строки.
                BIT   5,(FLAGX)         Если мы в режиме INPUT, то переход
                JP    NZ,1097,CLEAR-SP  вперёд.
                CALL  196E,LINE-ADDR    Найти адрес начала текущей строки
                CALL  1695,LINE-NO      и по нему определить номер строки
                LD    A,D               Если номер строки равен 0, то
                OR    E                 просто очистить область
                JP    Z,1097,CLEAR-SP   редактирования.
                PUSH  HL                Сохранить адрес строки.
                INC   HL                Определим длину строки
                LD    C,(HL)
                INC   HL
                LD    B,(HL)
                LD    HL,+000A          К длине строки добавим +0A и
                ADD   HL,BC             проверить, достаточно ли места
                LD    B,H               для копирования строки в буфер.
                LD    C,L
                CALL  1F05,TEST-ROOM
                CALL  1097,CLEAR-SP     Очистить область редактирования.
                LD    HL,(CURCHL)       Выбор адреса текущего канала
                EX    (SP),HL           и замена его адресом строки.
                PUSH  HL                Сохраним его.
                LD    A,+FF             Открыть канал 'R', чтобы строка
                CALL  1601,CHAN-OPEN    копировалась в область
                                        редактирования.
                POP   HL                Восстановим адрес строки.
```

```
                DEC   HL                  Адрес -1 относительно начала
                                          строки.
                DEC   (E-PPC-lo)          Уменьшить номер строки, чтобы
                                          избежать печатания курсора.
                CALL  1855,OUT-LINE       печать BASIC-строки.
                INC   (E-PPC-lo)          Увеличить обратно номер текущей
                                          строки.
                                          Замечание: уменьшение номера
                                          строки не всегда помогает,
                                          когда надо избежать печати курсора.
                LD    HL,(E-LINE)         Выбор начала строки в области
                INC   HL                  редактирования и установка адреса
                INC   HL                  за номер строки и её длину, чтобы
                INC   HL                  определить адрес для K-CUR.
                INC   HL
                LD    (K-CUR),HL
                POP   HL                  Восстановить старый адрес канала
                CALL  1615,CHAN-FLAG      и установка соответствующих
                RET                       признаков перед возвратом
                                          в ED-LOOP.

ПОДПРОГРАММА 'CURSOR DOWN EDITING' ('Курсор вниз при редактировании')
0FF3 ED-DOWN    BIT   5,(FLAGX)           Если мы в режиме INPUT, то
                JR    NZ,1001,ED-STOP     переход вперёд.
                LD    HL,+5C49            Это E-PPC.
                CALL  190F,LN-FETCH       Поиск номера следующей строки
                JR    106E,ED-LIST        и автоматическая перерисовка
                                          листинга программы.
1001 ED-STOP    LD    (ERR-NR),+10        сообщение 'STOP in INPUT'.
                JR    1024,ED-ENTER       Переход вперёд.

ПОДПРОГРАММА 'CURSOR LEFT EDITING' ('Курсор влево при редактировании')
1007 ED-LEFT    CALL  1031,ED-EDGE        Перемещение курсора.
                JR    1011,ED-CUR         Переход вперёд.

ПОДПРОГРАММА 'CURSOR RIGHT EDITING' ('Курсор вправо при редактировании')
100C ED-RIGHT   LD    A,(HL)              Проверка текущего символа.
                CP    +0D                 Если это 'возврат каретки',
                RET   Z                   то возврат из попрограммы.
                INC   HL                  Поставить курсор на следующий
                                          символ.
1011 ED-CUR     LD    (K-CUR),HL          Установить системную переменную
                RET                       K-CUR.

ПОДПРОГРАММА 'DELETE EDITING' ('Редактирование клавишей DELETE')
1015 ED-DELETE  CALL  1031,ED-EDGE        Передвинуть курсор влево.
                LD    BC,+0001            Убрать текущий символ.
                JP    19E8,RECLAIM-2
```

```
ПОДПРОГРАММА 'ED-IGNORE'
101E ED-IGNORE    CALL  15D4,WAIT-KEY       Игнорируются следующие два кода
                  CALL  15D4,WAIT-KEY       из программы ввода клавиш.

ПОДПРОГРАММА 'ENTER EDITING' ('Редактирование клавишей ENTER')
1024 ED-ENTER     POP   HL                  Отбрасываем адреса ED-LOOP
                  POP   HL                  и ED-ERROR.
1026 ED-END       POP   HL                  Восстанавливаем старое значение
                  LD    (ERR-SP),HL         ERR-SP.
                  BIT   7,(ERR-NR)          Если нет ошибок, то возврат
                  RET   NZ
                  LD    SP,HL               В противном случае делаем непрямой
                  RET                       переход на программу обработки
                                            ошибок.
```

ПОДПРОГРАММА 'ED-EDGE'
Адрес курсора задаётся на входе в HL и будет уменьшен, если курсор не в начале строки. Также уделено внимание тому, чтобы курсор не оказался помещённым между управляющими символами и их параметрами.

```
1031 ED-EDGE      SCF                       DE будет содержать или E-LINE
                  CALL  1195,SET-DE         (для редактирования) или WORKSP
                                            (для ввода через INPUT).
                  SBC   HL,DE               Установить флаг переноса, если
                                            курсор должен уже быть в начале
                  ADD   HL,DE               строки.
                  INC   HL                  Коррекция для вычитания.

                  POP   BC                  Удалить адрес возврата.
                  RET   C                   Возврат через ED-LOOP, если
                                            установлен флаг переноса.
                  PUSH  BC                  Восстановить адрес возврата.
                  LD    B,H                 Переслать текущий адрес курсора
                  LD    C,L                 в BC.
```

Теперь вводится цикл, проверяющий, что управляющие символы не отделены курсором от их параметров.

```
103E ED-EDGE-1    LD    H,D                 HL будет указывать на символ
                  LD    L,E                 в строке, потом он адресуется
                  INC   HL                  регистровой парой DE.
                  LD    A,(DE)              Взять код символа.
                  AND   +F0                 Если код не представляется, то
                  CP    +10                 переход вперёд
                  JR    NZ,1051,ED-EDGE-2   INK к TAB.
                  INC   HL                  Допускается для одного
                                            параметра
                  LD    A,(DE)              Выбрать новый код.
                  SUB   +17                 Для TAB сброшен флаг переноса.
                  ADC   A,+00               Примечание: этим отделяются
                                            AT и TAB, но AT и TAB в этой
                                            форме реализуются как попало,
                                            поэтому не делают различий.
```

```
                JR    NZ,1051,ED-EDGE-2   Переход вперёд, если не
                INC   HL                  обрабатываются AT и TAB, которые
                                          при использовании должны иметь
                                          два параметра.
1051 ED-EDGE-2  AND   A                   Подготовиться к истинному
                                          вычитанию.
                SBC   HL,BC               Когда обновлённый указатель
                ADD   HL,BC               достигает K-CUR, сбрасывается
                                          флаг переноса.
                EX    DE,HL               Для следующего цикла используем
                JR    C,103E,ED-EDGE-1    обновлённый указатель, но если
                RET                       выходим, то используем текущий
                                          указатель для K-CUR.

ПОДПРОГРАММА 'CURSOR UP EDITING' ('Курсор вверх при редактировании')
1059 ED-UP      BIT   5,(FLAGX)           Возврат, если режим INPUT.
                RET   NZ
                LD    HL,(E-PPC)          Берём номер текущей строки
                CALL  196E,LINE-ADDR      и её начальный адрес.
                EX    DE,HL               HL теперь указывает на
                                          предыдущую строку.
                CALL  1695,LINE-NO        Выбрали номер предыдущей
                                          строки.
                LD    HL,+5C4A            Это E-PPC-hi.
                CALL  1910,LN-STORE       Запомнить номер строки.
106E ED-LIST    CALL  1795,AUTO-LIST      Автолистинг результата
                                          перемещения курсора к предыдущей
                                          строке.
                LD    A,+00               Переоткрывает канал 'K' перед
                JP    1601,CHAN-OPEN      возвратом в ED-LOOP.

ПОДПРОГРАММА 'ED-SYMBOL'
Если использовались коды SYMBOL и GRAPHICS, то они будут обработаны следующим образом:

1076 ED-SYMBOL  BIT   7,(FLAGX)           Переход назад, если не
                JR    Z,1024,ED-ENTER     обрабатываются INPUT LINE.
107C ED-GRAPH   JP    0F81,ADD-CHAR       Переход назад.

ПОДПРОГРАММА 'ED-ERROR' ('Ошибка редактирования')
Сюда осуществляется переход при возникновении какой-либо ошибки при редактировании.

107F ED-ERROR   BIT   4,(FLAGS2)          Переход назад, если не
                JR    Z,1026,ED-END       используется канал 'K'.

                LD    (ERR-NR),+FF        Отменить номер ошибки и
                LD    D,+00               выдать 'rasp' (дребезжащий
                LD    E,(RASP)            звук) перед повторным
                LD    HL,+1A90            обходом редактора
                CALL  0385,BEEPER
                JP    0F30,ED-AGAIN

ПОДПРОГРАММА 'CLEAR-SP' ('Очистить область')
Очищаются область редактирования или рабочая область.
```

```
1097 CLEAR-SP     PUSH  HL                Сохраняем указатель области.
                  CALL  1190,SET-HL       DE будет указывать на первый
                                          символ, а HL - на последний.
                  DEC   HL                Восстанавливается правильное
                  CALL  19E5,RECLAIM-1    количество.
                  LD    (K-CUR),HL        Системные переменные K-CUR
                  LD    (MODE),+00        и MODE ('режим K')
                  POP   HL                инициализируются перед
                  RET                     выбором указателя и возвратом.
```

ПОДПРОГРАММА 'KEYBOARD INPUT' ('Ввод с клавиатуры')
Эта важная подпрограмма возвращает код последней нажатой клавиши. Но отметим, что нажатия на CAPS LOCK, изменение режима и параметры управления цветом обрабатываются внутри программы.

```
10A8 KEY-INPUT    BIT   3,(TV-FLAG)       Скопировать строку редактирования
                  CALL  NZ,111D,ED-COPY   или INPUT-строку на экран, если
                                          изменился режим.
                  AND   A                 Возврат со установленным флагом Z
                  BIT   5,(FLAGS)         и регистром A=+0, если не нажата
                  RET   Z                 новая клавиша.
                  LD    A,(LAST-K)        В противном случае выбор кода
                  RES   5,(FLAGS)         и установка флага, подтверждающего,
                                          что код взят.
                  PUSH  AF                Временно сохранить код.
                  BIT   5,(TV-FLAG)       Если необходимо, очистить лишнюю
                  CALL  NZ,0D6E,CLS-LOWER часть изображения; например,
                                          после 'scroll?';
                  POP   AF                Восстановить код клавиши.
                  CP    +20               Переход, если код >=+20, т.е.
                  JR    NC,111B,KEY-DONE
                  CP    +10               Переход вперёд с большинством
                  JR    NC,10FA,KEY-CONTR управляющих кодов.
                  CP    +06               Переход вперёд с кодами режимов
                  JR    NC,10DB,KEY=M&CL  и кодом CAPS LOCK.
```

Теперь работа с кодами FLASH, BRIGHT и INVERSE.

```
                  LD    B,A               Сохраним код.
                  AND   +01               Оставляем 0-й бит кода.
                  LD    C,A               C будет +00 (= выкл) или
                                          C будет +01 (= вкл).
                  LD    A,B               Восстановить код.
                  RRA                     Сдвиг вправо (убираем бит 0).
                  ADD   A,+12             Увеличить на +12, получая
                  JR    1105,KEY-DATA     FLASH - +12, BRIGHT - +13
                                          и INVERSE - +14.
```

Код CAPS LOCK и коды режимов обрабатываются отдельно.

```
10DB KEY-M&CL     JR    NZ,10E6,KEY-MODE    Переход вперёд с кодами режимов.
                  LD    HL,+5C6A            FLAGS2.
                  LD    A,+08               Инвертируем 3 бит FLAGS2, это
                  XOR   (HL)                признак CAPS LOCK.
                  LD    (HL),A
                  JR    10F4,KEY-FLAG       Переход вперёд.
10E6 KEY-MODE     CP    +0E                 Проверка нижней границы кода.

                  RET   C
                  SUB   +0D                 Сужение диапазона.
                  LD    HL,+5C41            MODE.
                  CP    (HL)                Изменено?
                  LD    (HL),A              Ввести новый код режима.
                  JR    NZ,10F4,KEY-FLAG    Переход, если изменился;
                  LD    (HL),+00            Иначе сделать его режимом 'L'.
10F4 KEY-FLAG     SET   3,(TV-FLAG)         Установка флага 'режим
                                            изменился'.
                  CP    A                   Сброс флага переноса
                  RET                       и возврат.
```

Обработка кодов управляющих клавиш (отдельно от FLASH, BRIGHT и INVERSE).

```
10FA KEY-CONTR    LD    B,A                 Сохранить код.
                  AND   +07                 Оставляем от кода значения 0-7
                  LD    C,A                 и заносим его в регистр C.
                  LD    A,+10               В регистре A содержится код INK.
                  BIT   3,B                 Но если код был 'несмещённым',
                  JR    NZ,1105,KEY-DATA    то A содержит код PAPER.
                  INC   A
```

Параметр записывается в K-DATA, и изменяется адрес канала с KEY-INPUT на KEY-NEXT.

```
1105 KEY-DATA     LD    (K-DATA),C          Запись.
                  LD    DE,+110D            KEY-NEXT.
                  JR    1113,KEY-CHAN       Переход вперёд.
```

Примечание: на первом проходе ввода на KEY-INPUT регистр A возвращается с содержанием 'кода управления' и затем на следующем проходе на KEY-NEXT, это возвращаемый параметр.

```
110D KEY-NEXT     LD    A,(K-DATA)          Выбор параметра.
                  LD    DE,+10A8            KEY-INPUT.
```

Теперь установить адрес ввода в первой канальной области.

```
1113 KEY-CHAN     LD    HL,(CHANS)          Выбор адреса канала.
                  INC   HL
                  INC   HL
                  LD    (HL),E              Теперь установить адрес ввода.
                  INC   HL
                  LD    (HL),D
```

В конце выход с требуемым кодом в регистре A.

```
111B KEY-DONE     SCF                       Признак, что код обнаружен.
                  RET                       Возврат.
```

ПОДПРОГРАММА 'LOWER SCREEN COPYING' ('Копирование нижней части экрана')
Эта подпрограмма вызывается всякий раз, когда строка в области редактирования или области INPUT должна печататься в нижней части экрана.

```
111D ED-COPY      CALL  0D4D,TEMPS          Использовать постоянные цвета.
                  RES   3,(TV-FLAG)         Флаг 'режим должен рассматриваться
                  RES   5,(TV-FLAG)         изменённым' и нижняя часть экрана
                                            не должна очищаться
                  LD    HL,(S-POSNL)        Сохраним текущее значение
                  PUSH  HL                  S-POSNL.
                  LD    HL,(ERR-SP)         Сохранить текущее значение
                  PUSH  HL                  ERR-SP.
                  LD    HL,+1167            Это ED-FULL.
                  PUSH  HL                  Поместить этот адрес на стек,
                  LD    (ERR-SP),SP         чтобы выполнить ED-FULL.
                                            Точка входа следует за ошибкой.
                  LD    HL,(ECHO-E)         Поместить значение ECHO-E
                  PUSH  HL                  на стек.
                  SCF                       HL указывает на начало области,
                  CALL  1195,SET-HL         а DE - на конец.
                  EX    DE,HL
                  CALL  187D,OUT-LINE2      Печать строки.
                  EX    DE,HL               Поменять указатели и напечатать
                  CALL  18E1,OUT-CURS       курсор.
                  LD    HL,(S-POSNL)        Берём текущее значение S-POSNL
                  EX    (SP),HL             и меняем его с ECHO-E.
                  EX    DE,HL               Передать ECHO-E в DE.
                  CALL  0D4D,TEMPS          Опять выбрать постоянный цвет.
```

Остаток любой строки, которая начата, теперь завершается пробелами, напечатанными с 'постоянным' цветом PAPER.

```
1150 ED-BLANK     LD    A,(S-POSNL-hi)      Выбрать текущий номер строки
                  SUB   D                   и вычесть старый номер.
                  JR    C,117C,ED-C-DONE    Переход вперёд, если нет
                                            'бланкирования' требуемых строк.
                  JR    NZ,115E,ED-SPACES   Переход вперёд, если находимся
                                            не на той же самой строке.
                  LD    A,E                 Выбрать старый номер столбца
                  SUB   (S-POSNL-lo)        и вычесть новый номер.
                  JR    NC,117C,ED-C-DONE   Переход, если нет требуемых
                                            пробелов.
115E ED-SPACES    LD    A,+20               Код 'пробел'.
                  PUSH  DE                  Запись старых значений.
                  CALL  09F4,PRINT-OUT      Печать.
                  POP   DE                  Выбор старых значений.
                  JR    1150,ED-BLANK       Возврат назад.
```

Теперь обработка любых ошибок.

```
1167 ED-FULL      LD    D,+00               Выдача сигнала 'rasp'.
                  LD    E,(RASP)
                  LD    HL,+1A90
                  CALL  03B5,BEEPER
                  LD    (ERR-NR),+FF        Отменить номер ошибки.
                  LD    DE,(S-POSNL)        Выбор текущего значения
                  JR    117E,ED-C-END       S-POSNL и переход вперёд.
```

Обычный выход на завершение копирования через строку редактирования или INPUT-строку.

```
117C ED-C-DONE    POP   DE                  Новое значение позиции.
                  POP   HL                  Адрес ошибки.
```

Сюда переходить после ошибки.

```
117E ED-C-END     POP   HL                  Восстанавливаем старое значение
                  LD    (ERR-SP),HL         ERR-SP.
                  POP   BC                  Старое значение S-POSNL.
                  PUSH  DE                  Сохранить новое значение
                                            позиции.
                  CALL  0DD9,CL-SET         Установить системные переменные.
                  POP   HL                  Старое значение S-POSNL
                  LD    (ECHO-E),HL         идёт в ECHO-E.
                  LD    (X-PTR-hi),+00      Соответствующим образом
                  RET                       очищается X-PTR и выполняется
                                            возврат.
```

ПОДПРОГРАММЫ 'SET-HL' И 'SET-DE'
Эти подпрограммы возвращают в HL указатель на первую ячейку и в DE указатель на последнюю ячейку или области редактирования или рабочей области.

```
1190 SET-HL       LD    HL,(WORKSP)         Указывает на последнюю ячейку
                  DEC   HL                  области редактирования.
                  AND   A                   Сброс флага переноса.
1195 SET-DE       LD    DE,(E-LINE)         Указывает на начало области
                  BIT   5,(FLAGX)           редактирования, и если мы в
                  RET   Z                   режиме редактирования, то возврат.
                  LD    DE,(WORKSP)         В противном случае меняем DE
                  RET   C                   и возврат.
                  LD    HL,(STKBOT)         Выбор STKBOT и возврат.
                  RET
```

ПОДПРОГРАММА 'REMOVE-FP'
Эта подпрограмма переводит невидимые формы с плавающей точкой в BASIC-строку.

```
11A7 REMOVE-FP    LD    A,(HL)              Проверяется по очереди каждый
                                            символ.
                  CP    +0E                 Это маркер числа?
                  LD    BC,+0006            Он будет занимать 6 байт.
                  CALL  Z,19E8,RECLAIM-2    Восстановить число с плавающей
                                            точкой.
```

```
LD    A,(HL)              Опять выбор кода.
INC   HL                  Обновить указатель.
CP    +0D                 Код 'возврат каретки'?
JR    NZ,11A7,REMOVE-FP   Если нет, переход назад,
RET                       но если да, то простой возврат.
```

ПРОГРАММЫ ВЫПОЛНЕНИЯ КОМАНД

ПОДПРОГРАММА 'INITIALISATION' ('Инициализация')
Главная точка входа в эту программу находится в START/NEW (11CB). При входе из START (0000), как и при первом включении питания, регистр A содержит 0, а регистр DE значение +FFFF. Однако, главной точки входа также можно достичь, если следовать выполнению командной процедуры NEW.

ПРОГРАММА 'NEW COMMAND' ('Команда NEW')

```
11B7 NEW          DI                      Невозможно маскируемое
                                          прерывание.
                  LD    A,+FF             Признак NEW.
                  LD    DE,(RAMTOP)       Сохраняется существу-
                                          ющее значение RAMTOP.
                  EXX                     Загрузите альтерна-
                  LD    BC,(P-RAMT)       тивные регистры сле-
                  LD    DE,(RASP/PIP)     дующими системными
                  LD    HL,(UDG)          переменными, которые
                  EXX                     также будут сохранены.
```

ПОДПРОГРАММА ИНИЦИАЛИЗАЦИИ
Главная точка входа.

```
11CB START/NEW    LD    B,A               Для дальнейшего записать
                                          признак.
                  LD    A,+07             Сделать окантовку белым
                                          цветом.
                  OUT   (+FE),A
                  LD    A,+3F             Установить регистр I чтобы
                  LD    I,A               содержать значение +3F.
                  DEFB  +00,+00,+00       Ждать 24 такта про-
                  DEFB  +00,+00,+00       цессора.
```

Теперь проверяется память.

```
11DA RAM-CHECK    LD    H,D               Передать значение в DE
                  LD    L,E               (START = +FFFF, NEW = RAMTOP).
11DC RAM-FILL     LD    (HL),+02          Ввести значение +02
                  DEC   HL                в каждую ячейку выше
                  CP    H                 +3FFF.
                  JR    NZ,11DC,RAM-FILL
11E2 RAM-READ     AND   A                 Подготовка для истин-
                                          ного вычитания.
                  SBC   HL,DE             По достижении вершины
                                          сбрасывается признак
                  ADD   HL,DE             переноса.
                  INC   HL                Обновить указатель.
                  JR    NC,11EF,RAM-DONE  Когда на вершине, переход.
                  DEC   (HL)              +02 становится +01.
                  JR    Z,11EF,RAM-DONE   Но, если 0 то ПЗУ неисправно.
                                          Как вершину используйте HL.
                  DEC   (HL)              +01 становится +00.
                  JR    Z,11E2,RAM-READ   Перейти к следующему тесту,
                                          если нет ошибки.
```

```
11EF RAM-DONE     DEC   HL                HL указывает на последнюю
                                          исправную ячейку
```

Далее восстанавливаются 'сохраненные' системные переменные. (Не имеет значения, если пришли из START).

```
                  EXX                     Переключить регистры.
                  LD    (P-RAMT),BC       Восстановить P-RAMT,
                  LD    (RASP/PIP),DE     RASP/PIP и UDG.
                  LD    (UDG),HL
                  EXX
                  INC   B                 Проверить флаг START/NEW.
                  JR    Z,1219,RAM-SET    Если пришли из NEW,
                                          переход вперед.
```

Когда пришли из START, затереть системные переменные и инициализировать область определенных пользователем графических символов (UDG).

```
                  LD    (P-RAMT),HL       Вершина физического ОЗУ.
                  LD    DE,+3EAF          Последний байт 'U'
                                          в наборе символов.
                  LD    BC,+00A8          Это кол-во байт в 21 букве.
                  EX    DE,HL             Переключить указатели.
                  LDDR                    Теперь скопировать символь-
                                          ные формы букв от 'A' до 'U'.
                  EX    DE,HL             Переключить указатели назад.
                  INC   HL                Указать первый байт.
                  LD    (UDG),HL          Теперь задать UDG.
                  DEC   HL                Вниз на 1 ячейку.
                  LD    BC,+0040          Задать системные
                  LD    (RASP/PIP),BC     переменные RASP и PIP.
```

Остаток программы является общим для операций START и NEW.

```
1219 RAM-SET      LD    (RAMTOP),HL       Установить RAMTOP.
                  LD    HL,+3C00          Инициализировать сис-
                  LD    (CHARS),HL        темную переменную CHARS.
```

Далее устанавливается машинный стек.

```
                  LD    HL,(RAMTOP)       Создается верхняя ячейка,
                  LD    (HL),+3E          содержащая +3E.
                  DEC   HL                Следующая ячейка ос-
                                          тается содержащей 0.
                  LD    SP,HL             Эти 2 ячейки представляют
                                          'последний элемент'.
                  DEC   HL                Шаг на 2 ячейки вниз
                  DEC   HL                чтобы найти правиль-
                  LD    (ERR-SP),HL       ное значение для ERR-SP.
```

Программа инициализации продолжается с:

```
                  IM    1                 Используется режим
                                          прерывания 1.
                  LD    IY,+5C3A          IY всегда содержит +ERR-NR.
                  EI                      Теперь возможно маскируемое
                                          прерывание. Обновляются
                                          часы реального времени и
```

```
                                клавиатура опрашивается
                                каждую 1/50 секунды.
LD      HL,+5CB6                Базовый адрес области
LD      (CHANS),HL              канальной информации.
LD      DE,15AF                 Начальные данные ка-
LD      BC,+0015                нала пересылаются из
EX      DE,HL                   таблицы (15AF) в область
LDIR                            канальной информации.
EX      DE,HL                   Создана системная пе-
DEC     HL                      ременная DATADD, чтобы
LD      (DATADD),HL             указывать на последнюю
                                ячейку данных канала.
INC     HL                      После этого PROG и
LD      (PROG),HL               VARS в ячейку.
LD      (VARS),HL
LD      (HL),+80                Маркер конца области
                                переменных.
INC     HL                      Переместиться на 1
                                ячейку, чтобы найти
LD      (E-LINE),HL             значение для E-LINE.
LF      (HL),+0D                Создать строку редактирова-
INC     HL                      ния из одного символа
                                'возврат каретки'.
LD      (HL),+80                Теперь ввести маркер конца.
INC     HL                      Переместиться на 1
LD      (WORKSP),HL             ячейку, чтобы найти
LD      (STKBOT),HL             значения WORKSP, STKBOT
LD      (STKEND),HL             и STKEND.
LD      A,+38                   Инициализировать сис-
LD      (ATTR-P),A              темные переменные
LD      (ATTR-T),A              цвета в: FLASH 0, BRIGHT 0,
LD      (BORDCR),A              PAPER 7 и INK 0.
LD      HL,+0523                Инициализировать сис-
LD      (REPDEL),HL             темные переменные
                                REPDEL и REPPER.
DEC     (KSTATE-0)              Сделать KSTATE-0
                                содержащей +FF.
DEC     (KSTATE-4)              Сделать KSTATE-4
                                содержащей +FF.
LD      HL,+15C6                Далее переместить
LD      DE,+5C10                начальные потоковые
LD      BC,+000E                данные из их таблицы
LDIR                            в потоковую область.
SET     1,(FLAGS)               Сигнал - 'используется
CALL    0EDF,CLEAR-PRB          принтер', и очищается
                                буфер принтера.
LD      (DF-SZ),+02             Задать размер нижней
CALL    0D6B,CLS                части дисплея и очистить
                                весь дисплей.
XOR     A                       Теперь печатается
LD      DE,+1538                сообщение '(c) 1982
CALL    0C0A,PO-MSG             Sinclair Research
                                LTD' в нижней строке.
SET     5,(TV-FLAG)             Сигнал - 'нижнюю часть
                                необходимо будет очистить'.
JR      12A9,MAIN-1             Переход вперед на основной
                                цикл исполнения.
```

ЦИКЛ 'MAIN EXECUTION' ('Основное исполнение')
Основной цикл простирается от ячейки 12A2 до ячейки 15AE и он управляет 'режимом редактирования', выполнением прямых команд и созданием сообщений.

```
12A2 MAIN-EXEC    LD    (DF-SZ),+02         Нижняя часть экрана должна
                                            иметь размер двух строк.
                  CALL  1795,AUTO-LIST      Сделать автоматический
                                            просмотр денных.
12A9 MAIN-1       CALL  16B0,SET-MIN        Все области от E-LINE вперед
                                            выдаются в их минимальной
                                            конфигурации.

12AC MAIN-2       LD    A,+00               Перед вызовом EDITOR
                  CALL  1601,CHAN-OPEN      открывается канал 'K'.
                  CALL  0F2C,EDITOR         EDITOR вызывается, чтобы
                                            разрешить пользователю
                                            создать BASIC-строку.
                  CALL  1B17,LINE-SCAN      Текущая строка просматри-
                                            вается на правильность
                                            синтаксиса.
                  BIT   7,(ERR-NR)          Если синтаксис пра-
                  JR    NZ,12CF,MAIN-3      вильный, переход вперед.
                  BIT   4,(FLAGS2)          Переход вперед, если
                  JR    Z,1303,MAIN-4       не используется канал 'K'.
                  LD    HL,(E-LINE)         Указывает на начало строки
                                            с ошибкой.
                  CALL  11A7,REMOVE-FP      Убрать формы с плавающей
                                            точкой из строки.
                  LD    (ERR-NR),+FF        Сброс ERR-NR и переход
                  JR    12AC,MAIN-2         назад к MAIN-2, оставляя
                                            листинг неизменным.
```

'Строка редактирования' прошла проверку синтаксиса, и три возможных типа строк должны отличаться один от другого.

```
12CF MAIN-3       LD    HL,(E-LINE)         Указать начало строки.
                  LD    (CH-ADD),HL         Также установить на начало
                                            CH-ADD.
                  CALL  19FB,E-LINE-NO      Выбрать любой номер строки
                                            в BC.
                  LD    A,B                 Номер строки пра-
                  OR    C                   вильный?
                  JR    NZ,155D,MAIN-ADD    Если да, то переход, и
                                            добавить новую строку в
                                            существующую программу.
                  RST   0018                Выберите первый символ
                  CP    +0D                 строки и посмотрите,
                                            является ли строка 'только
                                            возвратом каретки'.
                  JR    Z,12A2,MAIN-EXEC    Если да, переход назад.
```

'Строка редактирования' должна начинаться прямой командой BASIC, т.к. эта строка становится первой строкой, подлежащей обработке.

```
                  BIT   0,(FLAGS2)          Очищается все изображение,
                  CALL  NZ,0DAF,CL-ALL      если флаг не сообщает,
                                            что в этом нет необходи-
                                            мости.
```

```
                CALL  0D6E,CLS-LOWER      Очистить нижнюю часть
                                          в любом случае.
                LD    A,+19               Задать соответствующее
                SUB   (S-POSN-hi)         значение для счетчика
                                          прокруток.
                LD    (SCR-CT),A
                SET   7,(FLAGS)           Сигнал 'выполнение строки'.
                LD    (ERR-NR),+FF        Обеспечивает правильность
                                          ERR-NR.
                LD    (NSPPC),+01         Обработка первого оператора
                                          в строке.
                CALL  1B8A,PROG-RUN       Теперь обрабатывается строка.
                                          Примечание: Адрес 1303 помещается
                                          на машинный стек и адресуется
                                          с помощью ERR-SP.
```

После того, как строка обработана и все её действия последовательно завершены, выполняется возврат в MAIN-4, так что может быть создано сообщение.

```
1303 MAIN-4     HALT                      Маскируемое прерывание
                                          должно быть разрешено.
                RES   5,(FLAGS)           Сигнал - 'готов для новой
                                          клавиши'.
                BIT   1,(FLAGS2)          Освободить буфер принтера,
                CALL  NZ,0ECD,COPY-BUFF   если он использовался.
                LD    A,(ERR-NR)          Выбрать номер ошибки
                INC   A                   и увеличить его.
1313 MAIN-G     PUSH  AF                  Запись нового значения.
                LD    HL,+0000            В системные переменные
                LD    (FLAGX),H           FLAGX, X-PTR-hi и
                LD    (X-PTR-hi),H        DEFAND записывается 0.
                LD    (DEFADD),HL
                LD    HL,+0001            Обеспечивает, что поток +00
                LD    (STRMS-6),HL        указывает на канал 'K'.
                CALL  16B0,SET-MIN        Очистить все рабочие области
                                          и стек калькулятора.
                RES   5,(FLAGX)           Сигнал 'режим редактиро-
                                          вания'.
                CALL  0D6E,CLS-LOWER      Очистить нижнюю часть
                                          экрана.
                SET   5,(TV-FLAG)         Сигнал 'нижнюю часть экрана
                                          потребуется очистить'.
                POP   AF                  Выбрать значение сообщения.
                LD    B,A                 Сделать копию в B.
                CP    +0A                 Переход вперед с но-
                JR    C,133C,MAIN-5       мерами сообщений '0-9'.
                ADD   A,+07               Добавить значение
                                          смещения для буквы ASCII.
133C MAIN-5     CALL  15EF,OUT-CODE       Печать кода сообще-
                LD    A,+20               ния и следом за
                RST   0010,PRINT-A-1      ним 'пробела'.
                LD    A,B                 Выбор значения сообщения
                LD    DE,+1391            и использование его для
                CALL  0C0A,PO-MSG         идентификации требуемого
                                          сообщения.
                XOR   A                   Печать сообщения и
                LD    DE,+1536            следом за ними
                CALL  0C0A,PO-MSG         'запятой' и 'пробела'.
                LD    BC,(PPC)            Теперь выбор текущего
                CALL  1A1B,OUT-NUM1       номера строки и печать его.
```

```
                LD    A,+3A             Следом за ним ':'.
                RST   0010,PRINT-A-1
                LD    C,(SUBPPC)        Выбор текущего номера
                LD    B,+00             оператора в регистровой
                CALL  1A1B,OUT-NUM1     паре BC и печать его.
                CALL  1097,CLEAR-SP     Очистить область ре-
                                        дактирования.
                LD    A,(ERR-NR)        Опять выбрать номер ошибки.
                INC   A                 Увеличить ее обычным путем.
                JR    Z,1386,MAIN-9     Если программа успешно
                                        завершена, не может быть
                                        никакого 'продолжения',
                                        поэтому переход.
                CP    +09               Если программа оста-
                JR    Z,1373,MAIN-6     новлена операторами
                CP    +15               'STOP' или 'BREAK в
                                        программе', продолже-
                JR    NZ,1376,MAIN-7    ние будет со следую-
                                        щего оператора.
1373 MAIN-6     INC   (SUBPPC)          В противном случае
                                        SUBPPC не изменяется.
1376 MAIN-7     LD    BC,+0003          Системные переменные OLDPPC
                LD    DE,+5C70          и OSPCC должны теперь
                                        содержать строку для CONT
                                        и номера операторов.
                LD    HL,+5C44          Используемое значение будет
                BIT   7,(NSPPC)         в PPC и SUBPPC если NSPPC не
                JR    Z,1384,MAIN-8     отражает, что 'прерывание'
                ADD   HL,BC             появляется перед 'переходом'
1384 MAIN-8     LDDR                    (т.е. после оператора GO TO).
1386 MAIN-9     LD    (NSPPC),+FF       Чтобы отобразить 'нет пере-
                                        хода', сбрасывается NSPPC.
                RES   3,(FLAGS)         Выбран 'режим K'.
                JP    12AC,MAIN-2       И в завершение выполняется
                                        переход назад, но программный
                                        листинг не появляется до тех
                                        пор, пока не будет затребован.
```

СООБЩЕНИЯ
Каждое сообщение выдается с инвертированным последним символом (+80, шестнадцатеричное).

```
1391 DEFB +80     - перейти через начальный байт.
1392 Report 0     - 'OK'
1394 Report 1     - 'NEXT without FOR'      ('NEXT без FOR')
13A4 Report 2     - 'Variable not found'    ('Переменная не обнаружена')
13B6 Report 3     - 'Subscript wrong'       ('Описание неверное')
13C6 Report 4     - 'Out of memory'         ('Вне памяти')
13D2 Report 5     - 'Out of screen'         ('Вне экрана')
13DF Report 6     - 'Number too big'        ('Слишком большое число')
13ED Report 7     - 'RETURN without GOSUB'  ('RETURN без GOSUB')
1401 Report 8     - 'End of file'           ('Конец файла')
140C Report 9     - 'STOP statement'        ('Оператор STOP')
141A Report A     - 'Invalid argument'      ('Неправильный аргумент')
142A Report B     - 'Integer out of range'  ('Целое вне диапазона')
143E Report C     - 'Nonsense in BASIC'     ('Нонсенс в BASIC')
144F Report D     - 'BREAK - CONT repeats'  ('BREAK - CONT для повтора')
```

```
1463 Report E     - 'Out of DATA'           ('Вне оператора DATA')
146E Report F     - 'Invalid file name'     ('Направильное имя файла')
147F Report G     - 'No room for line'      ('Нет места для строки')
148F Report H     - 'STOP in INPUT'         ('STOP в INPUT')
149C Report I     - 'FOR without NEXT'      ('FOR без NEXT')
14AC Report J     - 'Invalid I/O device'    ('Неисправно устройство ввода/вывода')
14BE Report K     - 'Invalid colour'        ('Неправильный цвет')
14CC Report L     - 'BREAK into program'    ('BREAK в программе')
14DE Report M     - 'RAMTOP no good'        ('Не подходит RAMTOP')
14EC Report N     - 'Statement lost'        ('Потерян оператор')
14FA Report O     - 'Invalid stream'        ('Неправильный поток')
1508 Report P     - 'FN without DEF'        ('FN без DEF')
1516 Report Q     - 'Parameter error'       ('Ошибка параметра')
1525 Report R     - 'Tape loading error'    ('Ошибка при загрузке с ленты')
```

Еще два сообщения.

```
1537              ', '                      - 'запятая' и 'пробел'
1539              '(c) 1982 Sinclair Research Ltd'
```

Сообщение G - 'Нет места для строки'.

```
1555 REPORT-G     LD    A,+10               'G' имеет код '10+07+30'
                  LD    BC,+0000            Очистить BC.
                  JP    1313,MAIN-G         Переход назад для выдачи сообщения.
```

ПОДПРОГРАММА 'MAIN-ADD' ('Добавление к основному')
Эта подпрограмма разрешает добавить новую BASIC-строку в существующую BASIC-программу в программной области. Если строка имеет и старую, и новую версии, то старая версия 'восстанавливается'. Новая строка, которая состоит только из номера строки, не поступает в программную область.

```
155D MAIN-ADD     LD    (E-PPC),BC          Создать новый номер
                                            строки 'текущая строка'.
                  LD    HL,(CH-ADD)         Выбрать CH-ADD и за-
                  EX    DE,HL               писать адрес в DE.
                  LD    HL,+1555            Поместить адрес
                  PUSH  HL                  REPORT-G на стек.
                                            ERR-SP теперь будет
                                            указывать на REPORT-G.
                  LD    HL,(WORKSP)         Выбрать WORKSP.
                  SCF                       Найти длину строки от
                  SBC,  HL,DE               номера строки до символа
                                            'возврат каретки'
                                            включительно.
                  PUSH  HL                  Записать длину.
                  LD    H,B                 Поместить номер строки
                  LD    L,C                 в регистровую пару HL.
                  CALL  196E,LINE-ADDR      Строка с таким номером
                                            существует?
                  JR    NZ,157D,MAIN-ADD1   Если нет, переход.
                  CALL  19B8,NEXT-ONE       Найти длину 'старой'
                  CALL  19E8,RECLAIM-2      строки и восстановить ее.
157D MAIN-ADD1    POP   BC                  Выбрать длину 'новой'
                  LD    A,C                 строки и перейти вперед,
                  DEC   A                   если это только
```

```
                    OR    B                   'номер строки
                    JR    15AB,MAIN-ADD2      и возврат каретки'.
                    PUSH  BC                  Запись длины.
                    INC   BC                  Необходимы 4 допол-
                    INC   BC                  нительные ячейки.
                    INC   BC                  Т.е. 2 для числа и 2
                    INC   BC                  для длины.
                    DEC   HL                  Сделать HL указывающей
                                              ячейку перед 'адресом
                                              назначения'.
                    LD    DE,(PROG)           Записать текучее значение
                    PUSH  DE                  PROG, чтобы избежать
                                              разрушения при добавлении
                                              первой строки.
                    CALL  1655,MAKE-ROOM      Создается место для новой
                                              строки.
                    POP   HL                  Выбирается и восстанавли-
                    LD    (PROG),HL           вается старое значение PROG.
                    POP   BC                  Берется копия длины
                    PUSH  BC                  строки (без параметров).
                    INC   DE                  Создается DE, указыва-
                    LD    HL,(WORKSP)         ющая конечную ячейку
                    DEC   HL                  новой области, и HL,
                    DEC   HL                  указывающая символ 'возврат
                                              каретки' новой строки в
                                              области редактирования.
                    LDDR                      Теперь копирование строки.
                    LD    HL,(E-PPC)          Выбор номера строки.
                    EX    DE,HL               Адрес назначения в HL,
                                              а номер в DE.
                    POP   BC                  Выбор новой длины строки.
                    LD    (HL),B              Старший байт длины.
                    DEC   HL
                    LD    (HL),C              Младший байт длины.
                    DEC   HL
                    LD    (HL),E              Младший байт номера
                                              строки.
                    DEC   HL
                    LD    (HL),D              Старший байт номера
                                              строки.
15AB MAIN-ADD2      POP   AF                  Отбросить адрес REPORT-G.
                    JP    12A2,MAIN-EXEC      Переход назад и в это же
                                              время сделать автоматический
                                              листинг.
```

'НАЧАЛЬНАЯ ИНФОРМАЦИЯ О КАНАЛАХ'
Изначально существуют 4 канала - 'K', 'S', 'R' и 'P' для работы с 'клавиатурой', 'экраном', 'рабочей областью' и 'принтером'. Для каждого канала адрес программы вывода идет перед адресом программы ввода и кодом канала.

```
15AF DEFB         F4 09                   - PRINT-OUT
     DEFB         A8 10                   - KEY-INPUT
     DEFB         4B                      - 'K'
15B4 DEFB         F4 09                   - PRINT-OUT
     DEFB         C4 15                   - REPORT-J
     DEFB         53                      - 'S'
15B9 DEFB         81 0F                   - ADD-CHAR
```

```
     DEFB         C4 15                    - REPORT-J
     DEFB         52                       - 'R'
15BE DEFB         F4 09                    - PRINT-OUT
     DEFB         C4 15                    - REPORT-J
     DEFB         50                       - 'P'
15C3 DEFB         80                       - Маркер конца.
```

Сообщение J - 'Неисправно устройство ввода/вывода'.

```
15C4 REPORT-J     RST   0008,ERROR-1       Вызов подпрограммы
                  DEFB  +12                обработки ошибок.
```

'НАЧАЛЬНЫЕ ДАННЫЕ ПОТОКА'
Изначально существуют 7 потоков - +FD - +03.

```
15C6 DEFB         01 00 - поток +FD ведет к каналу 'K'
15C8 DEFB         06 00 - поток +FE ведет к каналу 'S'
15CA DEFB         0B 00 - поток +FF ведет к каналу 'R'
15CC DEFB         01 00 - поток +00 ведет к каналу 'K'
15CE DEFB         01 00 - поток +01 ведет к каналу 'K'
15D0 DEFB         06 00 - поток +02 ведет к каналу 'S'
15D2 DEFB         10 00 - поток +03 ведет к каналу 'P'
```

ПОДПРОГРАММА 'WAIT-KEY'
Эта подпрограмма является управляющей подпрограммой во время вызова текущей подпрограммы ввода.

```
15D4 WAIT-KEY     BIT   5,(TV-FLAG)         Переход вперед, если признак
                  JR    NZ,15DE,WAIT-KEY1  означает, что нижняя часть
                                           экрана не требует очистки.
                  SET   3,(TV-FLAG)        В противном случае, сигнал
                                           'рассмотреть измененный
                                           режим'.
15DE WAIT-KEY1    CALL  15E6,INPUT-AD      Вызвать подпрограмму ввода
                                           косвенно через INPUT-AD.
                  RET   C                  Возврат с допустимыми кодами.
                  JR    Z,15DE,WAIT-KEY1   Флаги переноса и нуля
                                           сбрасываются, если 'нет
                                           нажатой клавиши', иначе,
                                           сигнал об ошибке.
```

Сообщение 8 - 'Конец файла'.

```
15E4 REPORT-8     RST   0008,ERROR-1       Вызов подпрограммы
                  DEFB  +07                обработки ошибок.
```

ПОДПРОГРАММА 'INPUT-AD' ('Адреса ввода')
Сохраняются регистры и HL будет указывать на адрес ввода.

```
15E6 INPUT-AD     EXX                      Сохранить регистры.
                  PUSH  HL
                  LD    HL,(CURCHL)        Выбор базового адреса для
                                           текущей канальной информации.
                  INC   HL                 Шаг за адрес вывода.
                  INC   HL
                  JR    15F7,CALL-SUB      Переход вперед.
```

ПОДПРОГРАММА 'MAIN PRINTING' ('Основная печать')
Подпрограмма вызывается или с абсолютным значением или кодом символа в регистре A.

```
15EF OUT-CODE     LD    E,+30              Увеличить значение в
                  ADD   A,E                регистре A на +30.
15F2 PRINT-A-2    EXX                      Опять записать регистр.
                  PUSH  HL
                  LD    HL,(CURCHL)        Выбор базового адреса для
                                           текущего канала. Укажет
                                           адрес вывода.
```

Теперь вызов нужной подпрограммы. HL указывает, в зависимости от направления, адрес ввода и вывода.

```
15F7 CALL-SUB     LD    E,(HL)             Выбор младшего байта.
                  INC   HL
                  LD    D,(HL)             Выбор старшего байта.
                  EX    DE,HL              Переместить адрес
                                           в регистровую пару HL.
                  CALL  162C,CALL-JUMP     Вызов реальной подпрограммы.
                  POP   HL                 Восстановить регистры.
                  EXX
                  RET                      Будет возврат, если
                                           не появляется ошибка.
```

ПОДПРОГРАММА 'CHAN-OPEN' ('Открыть канал')
Эта подпрограмма вызывается с регистром A, содержащим правильный номер потока - обычно от +FD до +03. Потом, в зависимости от потоковых данных, определенный канал станет текущим каналом.

```
1601 CHAN-OPEN    ADD   A,A                Значение в регистре A
                  ADD   A,+16              дублируется, а затем
                  LD    L,A                увеличивается на +16.
                                           Результат помещается в L.
                  LD    H,+5C              Адрес 5C16 является базовым
                                           адресом для потока +00.
                  LD    E,(HL)             Выбор первого байта
                  INC   HL                 требуемых данных по-
                  LD    D,(HL)             тока; затем второй байт.
                  LD    A,D                Выдача ошибки, если
                  OR    E                  оба байта = 0; иначе,
                  JR    NZ,1610,CHAN-OP-1  переход вперед.
```

Сообщение O - 'Неправильный поток'.

```
160E REPORT-O     RST   0008,ERROR-1       Вызов подпрограммы
                  DEFB  +17                обработки ошибок.
```

Используя потоковые данные находим базовый адрес канальной информации, связанной с этим потоком.

```
1610 CHAN-OP-1    DEC   DE                 Уменьшить потоковые данные.
                  LD    HL,(CHANS)         Базовый адрес всей области
                                           канальной информации.
                  ADD   HL,DE              В этой области формируется
                                           требуемый адрес.
```

ПОДПРОГРАММА 'CHAN-FLAG' ('Признак канала')
Этой подпрограммой задаются соответствующие признаки для различных каналов.

```
1615 CHAN-FLAG    LD    (CURCHL),HL         Регистровая пара HL содержит
                                            базовый адрес для конкрет-
                                            ного канала.
                  RES   4,(FLAGS2)          Сигнал - 'использование
                                            другого, а не 'K' канала'.
                  INC   HL                  Шаг за адрес вывода
                  INC   HL                  и ввода и сделать HL
                  INC   HL                  указывающим на код
                  INC   HL                  канала.
                  LD    C,(HL)              Выбор кода.
                  LD    HL,+162D            Базовый адрес 'таблицы
                                            поиска кодов каналов'.
                  CALL  16DC,INDEXER        Проиндексировать таблицу и
                                            расположить требуемое
                                            смещение;
                  RET   NC                  но возврат, если нет сопо-
                                            ставимого кода требуемого
                                            смещения канала.
                  LD    D,+00               Передать смещение в
                  LD    E,(HL)              регистровую пару DE.
                  ADD   HL,DE               Переход вперед к программе
1162C CALL-JUMP   JP    (HL)                установки соответствующего
                                            признака.
```

ТАБЛИЦА ПОИСКА КОДОВ КАНАЛА

```
162D DEFB         4B 06 - канал 'K',      смещение +06, адрес 1634
162F DEFB         53 12 - канал 'S',      смещение +12, адрес 1642
1631 DEFB         50 1B - канал 'P',      смещение +1B, адрес 164D
1633 DEFB         00    - маркер конца.
```

ПОДПРОГРАММА 'CHANNEL 'K' FLAG' ('Признак канала 'K'')

```
1634 CHAN-K       SET   0,(TV-FLAG)         Сигнал 'использование
                                            нижней части экрана'.
                  RES   5,(FLAGS)           Сигнал 'готов к нажатию
                                            клавиши'.
                  SET   4,(FLAGS2)          Сигнал 'использование
                                            канала 'K''.
                  JR    1646,CHAN-S-1       Переход вперед.
```

ПОДПРОГРАММА 'CHANNEL 'S' FLAG' ('Признак канала 'S'')

```
1642 CHAN-S       RES   0,(TV-FLAG)         Сигнал 'использование
                                            основного экрана'.
1646 CHAN-S-1     RES   1,(FLAGS)           Сигнал 'принтер не
                                            использовался'.
                  JP    0D4D,TEMPS          Выход через TEMPS, чтобы
                                            установить системные
                                            переменные цвета.
```

ПОДПРОГРАММА 'CHANNEL 'P' FLAG' ('Признак канала 'P'')

```
164D CHAN-P       SET   1,(FLAGS)          Сигнал 'использование
                                           принтера'.
                  RET
```

ПОДПРОГРАММА 'MAKE-ROOM' ('Создание места')
Это очень важная подпрограмма. Она вызывается во многих случаях, чтобы 'выделить' 'область' памяти. Во всех случаях регистровая пара HL указывает ячейку, после которой требуется выделить 'область', а регистровая пара BC содержит необходимую длину этой 'области'.
Когда требуется выделить область памяти только для одного пробела, то вход в подпрограмму выполняется через ONE-SPACE.

```
1652 ONE-SPACE    LD    BC,+0001           Требуется только одна
                                           дополнительная ячейка.
1655 MAKE-ROOM    PUSH  HL                 Записать указатель.
                  CALL  1F05,TEST-ROOM     Убедиться, что для
                                           рассматриваемой задачи
                                           достаточно памяти.
                  POP   HL                 Восстановить указатель.
                  CALL  1664,POINTERS      Перед созданием области
                                           изменить все указатели.
                  LD    HL,(STKEND)        Создать HL содержащей
                                           новое STKEND.
                  EX    DE,HL              Переключить 'старое'
                                           и 'новое'.
                  LDDR                     Теперь создаем 'область'
                  RET                      и возврат.
```

Примечание: Эта подпрограмма возвращается с регистровой парой HL, указывающей на ячейку перед новой 'областью', и регистровой парой DE, указывающей на конец новых ячеек. Новая 'область', поэтому, имеет описание: от '(HL)+1' до '(DE)' включительно. Однако, так как 'новые ячейки' еще сохраняют 'старые значения', то возможно рассмотрение новой области как созданной от исходной ячейки '(HL)' и поэтому имеющей описание: от '(HL)+2' до '(DE)+1'.
В действительности, программисты предпочитают 'второе описание', что может привести к путанице.

ПОДПРОГРАММА 'POINTERS' ('Указатели')
Всякий раз, когда должна быть 'создана' или 'восстановлена' область, системные переменные, которые адресуют ячейки за пределами 'позиции' изменения, должны уточняться. На входе регистровая пара BC содержит число затронутых байт, а регистровая пара HL адресует ячейки перед 'позицией'.

```
1664 POINTERS     PUSH  AF                 Записаны эти регистры.
                  PUSH  HL                 Скопировать адрес 'позиции'.
                  LD    HL,+5C4B           Это VARS, первый из
                  LD    A,+0E              14 указателей.
```

Теперь вводится цикл, чтобы по очереди рассмотреть каждый указатель. Изменяются только те указатели, которые указывают за пределы 'позиции'.

```
166B PTR-NEXT     LD    E,(HL)             Выбрать два байта
```

```
                INC   HL                текущего указателя.
                LD    D,(HL)
                EX    (SP),HL           Поменять системную перемен-
                                        ную с адресом 'позиции'.
                AND   A                 Признак переноса будет
                SBC   HL,DE             установлен, если
                ADD   HL,DE             обновится адрес системной
                                        переменной.
                EX    (SP),HL           Восстановить 'позицию'.
                JR    NC,167F,PTR-DONE  Переход вперед, если должен
                                        остаться указатель, иначе,
                                        изменить его.
                PUSH  DE                Записать старое значение.
                EX    DE,HL             Теперь добавить значение
                ADD   HL,BC             в BC к старому значению.
                EX    DE,HL
                LD    (HL),D            Ввести в системную
                DEC   HL                переменную новое значение -
                LD    (HL),E            старший байт перед младшим.
                INC   HL                Снова указать старший байт.
                POP   DE                Выбор старого значения.
167F PTR-DONE   INC   HL                Указать следующую
                DEC   A                 системную переменную
                JR    NZ,166B,PTR-NEXT  и перейти назад, пока
                                        не будут рассмотрены все 14.
```

Теперь находим размер перемещаемого блока.

```
                EX    DE,HL             Поместить старое значение
                POP   DE                STKEND в HL и восстановить
                POP   AF                регистры.
                AND   A                 Теперь найти 'разность'
                SBC   HL,DE             между старым значением
                LD    B,H               STKEND и 'позицией'.
                LD    C,L               Передать результат в BC
                INC   BC                и прибавить '1' для включаемого
                                        байта.
                ADD   HL,DE             Преобразовать старое значение
                EX    DE,HL             STKEND и передать его в DE
                RET                     перед возвратом.
```

ПОДПРОГРАММА 'COLLECT A LINE NUMBER' ('Выбрать номер строки')
На входе регистровая пара HL указывает на рассматриваемую ячейку. Если ячейка содержит значение, которое включает соответствующий старший байт номера строки, то номер строки возвращается в DE. Однако, если это не так, то вместо нее проверяется ячейка, адресуемая DE, и в случае неудачного номера строки возвращается 0.

```
168F LINE-ZERO  DEFB  +00               Номер строки 0.
                DEFB  +00
1691 LINE-NO-A  EX    DE,HL             Рассматривается другой указатель.
                LD    DE,+168F          Использовать 0 как номер строки.
```

Обычной точкой входа является LINE-NO.

```
1695 LINE-NO    LD    A,(HL)            Выбрать старший байт
```

```
                AND   +C0                 и проверить его.
                JR    NZ,1691,LINE-NO-A   Если не подходит, переход
                                          назад.
                LD    D,(HL)              Выбрать старший байт.
                INC   HL
                LD    E,(HL)              Выбрать младший байт
                RET                       и возврат.
```

ПОДПРОГРАММА 'RESERVE' ('Резервирование')
Эта подпрограмма обычно вызывается подпрограммой RST 0030, BC-SPACES. На входе здесь последнем значением на машинном стеке является WORKSP, а значение выше него - это количество пробелов, которое 'резервируется'.
Эта подпрограмма всегда создает 'место' между существующей рабочей областью и стеком калькулятора.

```
169E RESERVE    LD    HL,(STKBOT)         Выбор текущего значения
                DEC   HL                  STKBOT и уменьшение его,
                                          чтобы получить последнюю
                                          ячейку рабочей области.
                CALL  1655,MAKE-ROOM      Теперь создать 'BC ячеек
                                          нового места'.
                INC   HL                  Указать на первое новое место,
                INC   HL                  а затем на второе.
                POP   BC                  Выбор старого значения
                LD    (WORKSP),BC         WORKSP и восстановление его.
                POP   BC                  Восстановить BC - число мест.
                EX    DE,HL               Переключить указатели.
                INC   HL                  Создать HL указывающим
                                          на первый из перемещенных байт.
                RET                       Теперь возврат.
```

Примечание: Это также можно рассматривать как то, что программа возвращается с регистровой парой DE, указывающей 'первый дополнительный байт' и регистровой парой HL, указывающей на 'последний дополнительный байт', эти дополнительные байты добавляются после исходной '(HL)+1' ячейки.

ПОДПРОГРАММА 'SET-MIN' ('Установка минимума')
Эта подпрограмма устанавливает размеры области редактирования и областей после нее минимальными. Фактически она 'очищает' эти области.

```
16B0 SET-MIN    LD    HL,(E-LINE)         Выбор E-LINE.
                LD    (HL),+0D            Создать область редактирова-
                LD    (K-CUR),HL          ния, содержащую только
                                          символ 'возврат каретки' и
                                          маркер конца.
                INC   HL
                LD    (HL),+80
                INC   HL                  Переместиться, чтобы
                LD    (WORKSP),HL         очистить рабочую область.
```

Здесь будет 'очищаться' рабочая область и стек калькулятора.

```
16BF SET-WORK   LD    HL,(WORKSP)         Выбор WORKSP.
                LD    (STKBOT),HL         Этим очищается рабочая
                                          область.
```

Если вход здесь, будет 'очищаться' только стек калькулятора.

```
16C5 SET-STK      LD    HL,(STKBOT)         Выбрать STKBOT.
                  LD    (STKEND),HL         Этим очищается стек.
```

Во всех случаях переменная MEM указывает на адрес области памяти калькулятора.

```
                  PUSH  HL                  Записать STKEND.
                  LD    HL,+5C92            Базовый адрес области памяти.
                  LD    (MEM),HL            Установить в MEM этот адрес.
                  POP   HL                  Восстановить STKEND в HL
                  RET                       перед возвратом.
```

ПОДПРОГРАММА 'RECLAIM THE EDIT-LINE' ('Восстановление строки редактирования')

```
16D4 REC-EDIT     LD    DE,(E-LINE)         Выбор E-LINE.
                  JP    19E5,RECLAIM-1      Восстановить память.
```

ПОДПРОГРАММА 'INDEXER' ('Индексатор')
Эта подпрограмма используется в нескольких случаях для просмотра таблиц. Точка входа - INDEXER.

```
16DB INDEXER-1    INC   HL                  Переместиться, чтобы рас-
                                            смотреть следующую пару
                                            элементов.
16DC INDEXER      LD    A,(HL)              Выбор первой пары
                  AND   A                   элементов, но воз-
                  RET   Z                   врат, если 0 - маркер конца.
                  CP    C                   Сравнить с предложенным
                                            кодом.
                  INC   HL                  Указывает второй элемент.
                  JR    NZ,16DB,INDEXER-1   Переход назад, если не найден
                                            правильный элемент.
                  SCF                       Признак переноса установлен
                  RET                       при успешном поиске.
```

КОМАНДНАЯ ПРОЦЕДУРА 'CLOSE #'
Эта команда позволяет пользователю закрыть (CLOSE) потоки. Однако для потоков +00 - +03 восстанавливаются 'начальные' данные, поэтому эти потоки не могут быть закрыты командой CLOSE.

```
16E5 CLOSE        CALL  171E,STR-DATA       Выбираются существующие
                                            данные для потока.
                  CALL  1701,CLOSE-2        Проверяется код в этом
                                            канале потока.
                  LD    BC,+0000            Подготовка, чтобы обнулить
                                            данные потока.
                  LD    DE,+A3E2            Подготовка для определения
                  EX    DE,HL               использования
                                            потоков +00 - +03.
                  ADD   HL,DE               Признак переноса будет
                                            установлен для потоков
                                            +04 - +0F.
                  JR    C,16FC,CLOSE-1      Переход вперед для этих
```

```
                  LD    BC,+15D4            потоков; иначе, найти
                  ADD   HL,BC               правильный элемент в таблице
                                            'начальных потоковых данных'.
                  LD    C,(HL)              Выбрать начальные данные
                  INC   HL                  для потоков
                  LD    B,(HL)              +00 - +03.
16FC CLOSE-1      EX    DE,HL               Теперь ввод данных; либо
                  LD    (HL),C              ноль и ноль, либо начальные
                  INC   HL                  значения.
                  LD    (HL),B
                  RET
```

ПОДПРОГРАММА 'CLOSE-2'
Код канала, связанный с закрытым потоком, должен быть 'K', 'S' или 'P'.

```
1701 CLOSE-2      PUSH  HL                  Запись адреса данных потока.
                  LD    HL,(CHANS)          Выбрать базовый адрес
                  ADD   HL,BC               информационной области
                                            канала и найти данные
                                            канала для закрытого потока.
                  INC   HL                  Шаг за адреса под-
                  INC   HL                  программы и бодборка
                  INC   HL                  кодов для этого
                  LD    C,(HL)              канала.
                  EX    DE,HL               Запись указателя.
                  LD    HL,+1716            Базовый адрес справочной
                                            таблицы 'закрытых потоков'.
                  CALL  16DC,INDEXER        Проиндексировать таблицу и
                                            выделить требуемое смещение.
                  LD    C,(HL)              Передать смещение в
                  LD    B,+00               регистровую пару BC.
                  ADD   HL,BC               Переход вперед на соответ-
                  JP    (HL)                ствующую подпрограмму.
```

ТАБЛИЦА ЗАКРЫТЫХ ПОТОКОВ

```
1716              DEFB  4B 05               - канал 'K', смещение +05, адрес 171C
1718              DEFB  53 03               - канал 'S', смещение +03, адрес 171C
171A              DEFB  50 01               - канал 'P', смещение +01, адрес 171C
```

Примечание: В конце таблицы нет маркера конца.

ПОДПРОГРАММА 'CLOSE STREAM' ('Закрыть поток')

```
171C CLOSE-STR    POP   HL                  Выбор указателя на информацию
                  RET                       о канале и возврат.
```

ПОДПРОГРАММА 'STREAM DATA' ('Данные потока')
Эта подпрограмма возвращает в регистровой паре BC данные для заданного потока.

```
171E STR-DATA     CALL  1E94,STK-TO-A       Номер заданного потока
                                            снимается со стека калькулятора.
                  CP    +10                 Выдать ошибку, если
```

```
                JR    C,1727,STR-DATA1    номер потока больше +0F.
```

Сообщение O - 'Неправильный поток'.

```
1725 REPORT-O   RST   0008,ERROR-1        Вызов подпрограммы
                DEFB  +17                 обработки ошибок.
```

Продолжение с правильными номерами потоков.

```
1727 STR-DATA1  ADD   A,+03               Теперь диапазон +03 - +12,
                RLCA                      и теперь +06 - +24.
                LD    HL,+5C10            Базовый адрес области
                                          данных о потоках.
                LD    C,A                 Переместить код
                                          потока в регистровую
                LD    B,+00               пару BC.
                ADD   HL,BC               Проиндексировать область
                LD    C,(HL)              данных и выбрать
                INC   HL                  два байта данных
                LD    B,(HL)              в регистровой паре BC.
                DEC   HL                  Сделать указатель адресующим
                RET                       первые байты данных перед
                                          возвратом.
```

КОМАНДНАЯ ПРОЦЕДУРА 'OPEN #' ('Открыть')
Эта команда позволяет пользователю открывать потоки. Должен применяться код канала, и он должен быть 'K', 'k', 'S', 's', 'P', or 'p'.
Отметим, что не делается попытка выдать потокам +00 to +03 их начальных данных.

```
1736 OPEN       RST   0028,FP-CALC        Использовать CALCULATOR.
                DEFB  +01,exchange        Поменять номер пото-
                DEFB  +38,end-calc        ка и код канала.
                CALL  171E,STR-DATA       Выбор данных для потока.
                LD    A,B                 Переход вперед, если оба
                OR    C                   байта данных 0,
                JR    Z,1756,OPEN-1       т.е. поток был закрыт.
                EX    DE,HL               Запись DE.
                LD    HL,(CHANS)          Выбрать CHANS - базовый
                ADD   HL,BC               адрес информации
                INC   HL                  о каналах и найти
                INC   HL                  код канала,
                INC   HL                  связанного с потоком,
                LD    A,(HL)              который открывается OPEN.
                EX    DE,HL               Возврат DE.
                CP    +4B                 Код, выбранный из инфор-
                JR    Z,1756,OPEN-1       мационной области о каналах
                CP    +53                 должен быть 'K', 'S' или 'P';
                JR    Z,1756,OPEN-1       иначе, ошибка.
                CP    +50
                JR    NZ,1725,REPORT-O
1756 OPEN-1     CALL  175D,OPEN-2         Собрать соответствующие данные
                                          в DE.
                LD    (HL),E              Ввести данные в 2 байта
                INC   HL                  в информационной области
                LD    (HL),D              о каналах.
                RET                       Окончательный возврат.
```

ПОДПРОГРАММА 'OPEN-2'
Находятся соответствующие байты данных потока, которые связаны с открытым потоком.

```
175D OPEN-2       PUSH  HL                  Запись HL.
                  CALL  2BF1,STK-FETCH      Выбор параметром кода канала.
                  LD    A,B                 Выдается ошибка, если
                  OR    C                   дополненное выражение явл.
                  JR    NZ,1767,OPEN-3      пустым, т.е. OPEN #5,"".
```

Сообщение F - 'Неправильное имя файла'.

```
1765 REPORT-F     RST   0008,ERROR-1        Вызов подпрограммы
                  DEFB  +0E                 обработки ошибок.
```

Если нет ошибки, продолжение.

```
1767 OPEN-3       PUSH  BC                  Записана длина выражения.
                  LD    A,(DE)              Выбор первого символа.
                  AND   +DF                 Преобразовать коды нижнего
                                            регистра в верхний.
                  LD    C,A                 Переместить код в регистр C.
                  LD    HL,+177A            Базовый адрес таблицы
                                            'Поиск открытого потока'.
                  CALL  16DC,INDEXER        Проиндексировать таблицу и
                                            выделить требуемое смещение.
                  JR    NC,1765,REPORT-F    Если не найдено, переход
                                            назад.
                  LD    C,(HL)              Передать смещение в
                  LD    B,+00               регистровую пару BC.
                  ADD   HL,BC               Сделать HL указывающей
                                            начало соответствующей
                                            программы.
                  POP   BC                  Выбрать длину выражения
                  JP    (HL)                перед переходом к под-
                                            программе.
```

ТАБЛИЦА 'ПОИСК ОТКРЫТОГО КАНАЛА'

```
177A              DEFB  4B 06               - канал 'K', смещение +06, адрес 1781
177C              DEFB  53 08               - канал 'S', смещение +08, адрес 1785
177E              DEFB  50 0A               - канал 'P', смещение +0A, адрес 1789
1780              DEFB  00                  - маркер конца.
```

ПОДПРОГРАММА 'OPEN-K'

```
1781 OPEN-K       LD    E,+01               Байты данных будут
                  JR    178B,OPEN-END       +01 и +00.
```

ПОДПРОГРАММА 'OPEN-S'

```
1785 OPEN-S       LD    E,+06               Байты данных будут
                  JR    178B,OPEN-END       +06 и +00.
```

ПОДПРОГРАММА 'OPEN-P'

```
1789 OPEN-P       LD    E,+10               Байты данных будут +10 и +00.
178B OPEN-END     DEC   BC                  Уменьшить длину вы-
                  LD    A,B                 ражения и выдать
                  OR    C                   ошибку, если не один
                  JR    NZ,1765,REPORT-F    символ, в противном
                  LD    D,A                 случае очистить ре-
                  POP   HL                  гистр D, выбрать HL и
                  RET                       вернуться.
```

КОМАНДНАЯ ПРОЦЕДУРА 'CAT, ERASE, FORMAT & MOVE' ('Каталог', 'Уничтожить', 'Форматировать' и 'Переместить')
В стандартной системе SPECTRUM использование этих команд приводит к сообщению O - неправильный поток.

```
1793 CAT-ETC.     JR    1725,REPORT-O       Выдача этого сообщения.
```

КОМАНДНАЯ ПРОЦЕДУРА 'LIST & LLIST'
Программы в этой части монитора используются для создания распечаток BASIC-программ. Каждая строка должна иметь свой вычисленный номер, свои токены и расположенные по позициям соответствующие курсоры.
Точка входа AUTO-LIST используется и программой MAIN EXECUTION и EDITOR для создания одной страницы распечатки.

```
1795 AUTO-LIST    LD    (LIST-SP),SP        Записывается указатель
                                            стека, допускающий сброс
                                            машинного стека при
                                            окончании распечатки
                                            (см. PO-SCR,0C55).
                  LD    (TV-FLAG),+10       Сигнал - 'автоматический
                                            просмотр данных (распечатка)
                                            на основном экране'.
                  CALL  0DAF,CL-ALL         Очистка этой части экрана.
                  SET   0,(TV-FLAG)         Переключение на область
                                            резервирования.
                  LD    B,(DF-SZ)           Теперь очистить ниж-
                  CALL  0E44,CL-LINE        нюю часть экрана.
                  RES   0,(TV-FLAG)         Обратное переключение.
                  SET   0,(FLAGS2)          Сигнал - 'экран чист'.
                  LD    HL,(E-PPC)          Теперь выбор 'текущего'
                  LD    DE,(S-TOP)          номера строки и 'автомати-
                                            ческого' номера строки.
                  AND   A                   Если 'текущий' номер
                  SBC   HL,DE               меньше 'автоматичес-
                  ADD   HL,DE               кого', то переход впе-
                  JR    C,17E1,AUTO-L-2     ред для обновления
                                            'автоматического' номера.
```

Теперь изменяется 'автоматический' номер для выдачи распечатки с 'текущей' строкой, появляющейся рядом с нижней частью экрана.

```
                  PUSH  DE                  Запись 'автоматического'
                                            номера.
                  CALL  196E,LINE-ADDR      Найти адрес начала
```

```
                LD    DE,+02C0          'текущей' строки и создать
                EX    DE,HL             адрес, приблизительно, 'экран
                SBC   HL,DE             перед ней' (инвертированный).
                EX    (SP),HL           Пока находится 'автомати-
                CALL  196E,LINE-ADDR    ческий' адрес строки (в HL),
                                        записать 'результат' на
                                        машинный стек.
                POP   BC                'Результат' идет в
                                        регистровую пару BC.
```

Теперь вводится цикл. 'Автоматический' номер строки увеличивается на каждом проходе, пока не станет похожим на тот, который показывает в листинге 'текущая' строка.

```
17CE AUTO-L-1   PUSH  BC                Запись 'результата'.
                CALL  19B8,NEXT-ONE     Найти адрес начала строки
                                        после текущей 'автоматичес-
                                        кой' строки (в DE).
                POP   BC                Восстановить 'результат'.
                ADD   HL,BC             Выполнить вычисление и при
                JR    C,17E4,AUTO-L-3   окончании, переход вперед.
                EX    DE,HL             Переместить следующий
                LD    D,(HL)            адрес строки в регистровую
                INC   HL                пару HL и выбрать номер
                                        строки.
                LD    E,(HL)
                DEC   HL
                LD    (S-TOP),DE        Теперь, можно обновить S-TOP
                JR    17CE,AUTO-L-1     и повторить текст с новой
                                        строкой.
```

Теперь можно создать 'автоматический' просмотр данных (листинг).

```
17E1 AUTO-L-2   LD    (S-TOP),HL        Когда E-PPC меньше, чем
                                        S-TOP.
17E4 AUTO-L-3   LD    HL,(S-TOP)        Выбирается наиболь-
                CALL  196E,LINE-ADDR    ший номер строки и
                                        отсюда его адрес.
                JR    Z,17ED,AUTO-L-4   Если строку нельзя
                EX    DE,HL             обнаружить, используется DE.
17ED AUTO-L-4   CALL  1833,LIST-ALL     Создан листинг.
                RES   4,(TV-FLAG)       Сюда будет произведен воз-
                RET                     врат, если не нужна прокрут-
                                        ка для показа текущей строки.
```

ТОЧКА ВХОДА 'LLIST'
Канал принтера должен быть открытым.

```
17F5 LLIST      LD    A,+03             Используйте поток +03.
                JR    17FB,LIST-1       Переход вперед.
```

ТОЧКА ВХОДА 'LIST'
Канал 'основного экрана' должен быть открытым.

```
17F9 LIST       LD    A,+02             Используйте поток +02.
17FB LIST-1     LD    (TV-FLAG),+00     Сигнал 'обычный листинг
                                        в основной части экрана'.
```

```
                  CALL  2530,SYNTAX-Z       Открыть канал, если не
                  CALL  NZ,1601,CHAN-OPEN   проверяется синтаксис.
                  RST   0018,GET-CHAR       С текущим символом в
                  CALL  2070,STR-ALTER      регистре A посмотрите,
                                            нужно ли изменить поток.
                  JR    C,181F,LIST-4       Если не изменяется,
                                            переход вперед.
                  RST   0018,GET-CHAR       Текущий символ ';' ?
                  CP    +38
                  JR    Z,1814,LIST-2       Если да, переход.
                  CP    +2C                 Это ',' ?
                  JR    NZ,181A,LIST-3      Если нет, переход.
1814 LIST-2       RST   0020,NEXT-CHAR      Должно следовать цифровое
                  CALL  1C82,EXPT-1NUM      выражение, например
                                            LIST # 5.20.
                  JR    1822,LIST-5         С ним переход вперед
181A LIST-3       CALL  1CE6,USE-ZERO       В противном случае
                  JR    1822,LIST-5         использовать 0 и тоже
                                            перейти вперед.

Если поток не изменился, перейти сюда.

181F LIST-4       CALL  1CDE,FETCH-NUM      Если ничего не поступило,
                                            выбрать любую строку или
                                            использовать 0.
1822 LIST-5       CALL  1BEE,CHECK-END      Если проверяется синтаксис
                                            строки редактирования, пе-
                                            рейти на следующий оператор.
                  CALL  1E99,FIND-INT       Номер строки в BC.
                  LD    A,B                 Старший байт в A.
                  AND   +3F                 Ограничить старший байт
                  LD    H,A                 нужным диапазоном и передать
                  LD    L,C                 весь номер строки в HL.
                  LD    (E-PPC),HL          Установить E-PPC и найти
                  CALL  196E,LINE-ADDR      адрес начала этой строки или
                                            первую строку после нее если
                                            фактическая строка не
                                            существует.
                  LD    E,+01               Признак - 'перед текущей
                                            строкой'.

Теперь вводится управляющий цикл для печати последовательности строк.

1835 LIST-ALL     CALL  1855,OUT-LINE       Печать всей BASIC-строки.
                  RST   0010,PRINT-A-1      Это будет 'возврат каретки'.
                  BIT   4,(TV-FLAG)         Если не работаете с
                  JR    Z,1835,LIST-ALL     автоматическим просмотром
                                            данных, переход назад.
                  LD    A,(DF-SZ)           Если имеется еще часть
                  SUB   (S-POSN-hi)         основного экрана, которую
                  JR    NZ,1835,LIST-ALL    можно использовать, тоже
                                            осуществляется переход
                                            назад.
                  XOR   E                   Может быть сделан возврат в
                  RET   Z                   эту точку, если экран
                                            заполнен и напечатана
                                            текущая строка (E=+00).
                  PUSH  HL                  Однако, если текущая
                  PUSH  DE                  строка в просмотре
```

```
                LD    HL,+5C6C          пропускается, то S-
                                        TOP должен обновиться
                                        и напечататься сле-
                CALL  190F,LN-FETCH     дующая строка (ис-
                POP   DE                пользуя прокрутку).
                POP   HL
                JR    1835,LIST-ALL
```

ПОДПРОГРАММА 'PRINT A WHOLE BASIC LINE' ('Печать полной BASIC-строки')
Регистровая пара HL указывает начало строки - ячейку, содержащую старший байт номера строки. Перед печатью номера строки, он проверяется, чтобы определить, приходит ли он перед 'текущей' строкой или приходит после нее.

```
1855 OUT-LINE     LD    BC,(E-PPC)          Выбрать и сравнить
                  CALL  1980,CP-LINES       номер 'текущей' строки.
                  LD    D,+3E               Перезагрузить регистр.
                                            курсором текущей строки.
                  JR    Z,1865,OUT-LINE1    Если печатается текущая
                                            строка, переход вперед.
                  LD    DE,+0000            Загрузить регистр D
                  RL    E                   0 (это не курсор) и
                                            установить E, содержащим
                                            +01, если строка перед
                                            'текущей' строкой, и +00,
                                            если после (признак переноса
                                            поступает из CP-LINES).
1865 OUT-LINE1    LD    (BREG),E            Запись маркера строки.
                  LD    A,(HL)              Выбрать старший байт
                  CP    +40                 номера строки и осу-
                  POP   BC                  ществить полный воз-
                  RET   NC                  врат, если окончен
                  PUSH  BC                  просмотр данных.
                  CALL  1A29,OUT-NUM-2      Теперь можно напечатать
                                            номер строки - с начальными
                                            пробелами.
                  INC   HL                  Переместить указа-
                  INC   HL                  тель, чтобы адресовать
                  INC   HL                  код первой команды в строке.
                  RES   0,(FLAGS)           Сигнал 'допускаются
                                            начальные пробелы'.
                  LD    A,D                 Выбор кода курсора и
                  AND   A                   переход вперед, если
                                            курсор не подлежит
                  JR    Z,1881,OUT-LINE3    печатанию.
                  RST   0010,PRINT-A-1      Итак, теперь печать курсора.
187D OUT-LINE2    SET   0,(FLAGS)           Сигнал - 'теперь нет
                                            начальных пробелов'.
1881 OUT-LINE3    PUSH  DE                  Запись регистров.
                  EX    DE,HL               Переместить указатель в DE.
                  RES   2,(FLAGS2)          Сигнал -' не в кавычках'.
                  LD    HL,+5C3B            Это FLAGS.
                  RES   2,(HL)              Сигнал - 'печать в режиме K'.
                  BIT   5,(FLAGX)           Если не в режиме
                  JR    Z,1894,OUT-LINE-4   INPUT, переход вперед.
                  SET   2,(HL)              Сигнал - 'печать в режиме L'.
```

Теперь введем цикл, чтобы напечатать все коды в остальных BASIC-строках - переход, если необходимо, через формы с плавающей точкой.

```
1894 OUT-LINE4    LD    HL,(X-PTR)          Выбор указателя синтаксичес-
                  AND   A                   ких ошибок и переход вперед,
                  SBC   HL,DE               если не время печатать маркер
                  JR    NZ,18A1,OUT-LINE5   ошибки.
                  LD    A,+3F               Теперь печатается маркер ошибки.
                  CALL  18C1,OUT-FLASH      Мигание '?'
18A1 OUT-LINE5    CALL  18E1,OUT-CURS       Рассматривается, печатать ли
                                            курсор.
                  EX    DE,HL               Теперь указатель переме-
                                            щается в HL.
                  LD    A,(HL)              По очереди выбирается каждый
                                            символ.
                  CALL  18B6,NUMBER         Если символ является 'марке-
                                            ром числа', то не печатаются
                                            невидимые формы с плавающей
                                            точкой.
                  INC   HL                  Обновить указатель для
                                            следующего прохода.
                  CP    +0D                 Символ 'возврат каретки'?
                  JR    Z,18B4,OUT-LINE6    Переход, если это так.
                  EX    DE,HL               Переключить указатель DE.
                  CALL  1937,OUT-CHAR       Напечатать символ.
                  JR    1894,OUT-LINE4      Програть цикл, по крайней
                                            мере, еще раз.
```

Теперь строка напечатана.

```
18B4 OUT-LINE6    POP   DE                  Восстановление регистровой
                  RET                       пары DE и возврат.
```

ПОДПРОГРАММА 'NUMBER' ('Число')
Если регистр A содержит 'маркер числа', то регистровая пара HL продвигается за форму с плавающей точкой.

```
18B6 NUMBER       CP    +0E                 Символ 'маркер числа'?
                  RET   NZ                  Если нет, возврат.
                  INC   HL                  Продвинуть указатель
                  INC   HL                  6 раз так, что шаг-
                  INC   HL                  нуть за 'маркер чис-
                  INC   HL                  ла' и 5 ячеек, содер-
                  INC   HL                  жащих форму с плава-
                  INC   HL                  ющей точкой.
                  LD    A,(HL)              Перед возвратом вы-
                  RET                       брать текущий код.
```

ПОДПРОГРАММА 'PRINT A FLASHING CHARACTER' ('Печать мигающего символа')
С использованием этой подпрограммы печатаются 'курсор ошибки' и 'курсоры режимов'.

```
18C1 OUT-FLASH    EXX                       Запись текущего регистра.
                  LD    HL,(ATTR-T)         Запись ATTR-T и
                  PUSH  HL                  MASK-T на машинный стек.
```

```
                RES   7,H               Обеспечивается ак-
                SET   7,L               тивность FLASH.
                LD    (ATTR-T),HL       Эти измененные значения
                                        используются для ATTR-T и
                                        MASK-T.
                LD    HL,+5C91          Это P-FLAG.
                LD    D,(HL)            На машинный стек записать
                PUSH  DE                также P-FLAG.
                LD    (HL),+00          Обеспечивает INVERSE 0, OVER
                                        0, а не PAPER, не INK9.
                CALL  09F4,PRINT-OUT    Печатается символ.
                POP   HL                Восстановлено значе-
                LD    (P-FLAG),H        ние формирователя F-FLAG.
                POP   HL                Кроме того, перед воз-
                LD    (ATTR-T),HL       аратом восстанавли-
                EXX                     ваются значения фор-
                RET                     мирователя ATTR-T и MASK-T.
```

ПОДПРОГРАММА 'PRINT THE CURSOR' ('Печать курсора')
Осуществляется возврат, если нет места, чтобы напечатать курсор, но, если оно есть, будет печататься или 'C', 'E', 'G', 'K', или 'L'.

```
18E1 OUT-CURS   LD    HL,(K-CUR)        Выбирается адрес
                AND   A                 курсора, но осущест-
                SBC   HL,DE             вляется возврат, если для
                RET   NZ                него не определено место.
                LD    A,(MODE)          Выбирается и дублируется
                RLC   A                 текущее значение MODE.
                JR    Z,18F3,OUT-C-1    Если не работает Расширенный
                                        режим и Графика,
                                        переход вперед.
                ADD   A,+43             Чтобы выдать 'E' или 'G',
                                        добавить соответствующее
                                        смещение.
                JR    1909,OUT-C-2      Для печати переход вперед.
18F3 OUT-C-1    LD    HL,+5C3B          Это FLAGS.
                RES   3,(HL)            Сигнал 'режим K'.
                LD    A,+4B             Символ 'K'.
                BIT   2,(HL)            Чтобы напечатать 'K',
                JR    Z,1909,OUT-C-2    переход вперед, если
                                        'печатание должно осущест-
                                        вляться в режиме K'.
                SET   3,(HL)            'Печать должна осуществляться
                                        в режиме L', поэтому сигнал -
                                        'L-MODE'.
                INC   A                 Формируется символ 'L'.
                BIT   3,(FLAGS2)        Если не в 'режиме C',
                JR    Z,1909,OUT-C-2    переход вперед.
                LD    A,+43             Символ 'C'.
1909 OUT-C-2    PUSH  DE                Запись регистровой
                CALL  18C1,OUT-FLASH    пары DE, пока мигает
                POP   DE                напечатанный курсор.
                RET                     Возврат.
```

Примечание: Этим определяется, какой буквенный курсор, определяющий режим, должен быть напечатан - 'K' или 'L/C'.

ПОДПРОГРАММА 'LN-FETCH'
Эта подпрограмма вводится с регистровой парой, адресующей системную переменную S-TOP или E-PPC.
Подпрограмма возвращается с системной переменной, содержащей номер следующей строки.

```
190F LN-FETCH     LD    E,(HL)              Отбирается номер
                  INC   HL                  строки, содержащейся
                  LD    D,(HL)              в системной переменной.
                  PUSH  HL                  Записывается указатель.
                  EX    DE,HL               Номер строки перемещается
                  INC   HL                  в регистровую пару HL и
                                            увеличивается.
                  CALL  196E,LINE-ADDR      Находится адрес начала этой
                                            строки, или следующей, если
                                            не используется фактический
                                            номер строки.
                  CALL  1695,LINE-NO        Выбирается номер строки.
                  POP   HL                  Восстановлен указатель
                                            системной переменной.
```

EDITOR использует точку входа LN-STORE.

```
191C LN-STORE     BIT   5,(FLAGX)           Если в 'режиме INPUT',
                  RET   NZ                  возврат; в противном
                  LD    (HL),D              случае, продолжать вводить
                  DEC   HL                  номер строки в 2 ячейки
                  LD    (HL),E              системной переменной.
                  RET                       Когда это сделано, возврат.
```

ПОДПРОГРАММА 'PRINTING CHARACTERS IN A BASIC LINE' ('Печать символов в BASIC-строку')
С помощью неоднократных вызовов этой подпрограммы печатаются все коды символов/токенов в BASIC-строке.
Точка входа OUT-SP-NO используется, когда печатаются номера строк, которые могут потребовать начальных пробелов.

```
1925 OUT-SP-2     LD    A,E                 Регистр A будет содержать
                                            +20 для пробела или +FF
                                            для не пробела.
                  AND   A                   Проверка значения и
                  RET   M                   возврат, если нет пробела.
                  JR    1937,OUT-CHAR       Для печати пробела, переход
                                            вперед.
192A OUT-SP-NO    XOR   A                   Очистить регистр A.
```

Регистровая пара HL содержит номер строки, а регистр BC значение для 'повторяемого вычитания'. (BC содержит '-1000, -100 или -10').

```
192B OUT-SP-1     ADD   HL,BC               'Пробное вычитание'.
                  INC   A                   Счет каждой 'пробы'.
                  JR    C,192B,OUT-SP-1     Пока не закончится, переход
                                            назад.
                  SBC   HL,BC               Восстановить послед-
                  DEC   A                   нее 'вычитание'.
                  JR    Z,1925,OUT-SP-2     Воли 'вычитания' были
                                            невозможны, переход
```

```
                                        назад, чтобы увидеть,
                                        напечатан ли пробел.
                JP    15EF,OUT-CODE     В противной случае,
                                        напечатать цифру.
```

Точка входа OUT-CHAR используется для всех символов, токенов и символов управления.

```
1937 OUT-CHAR   CALL  2D1B,NUMERIC      Если обрабатывается цифровой
                                        код, возврат сброшенного
                                        переноса.
                JR    NC,196C,OUT-CH-3  Переход вперед для печати
                                        цифры.
                CP    +21               Печать управляющих
                JR    C,196C,OUT-CH-3   символов и 'пробела'.
                RES   2,(FLAGS)         Сигнал - 'печать в режиме K'.
                CP    +CB               Если работа с токеном
                JR    Z,196C,OUT-CH-3   'THEN', переход вперед.
                CP    +3A               Если не работаете
                JR    NZ,195A,OUT-CH-1  с ':',переход вперед.
                BIT   5,(FLAGX)         Если в 'режиме INPUT',
                JR    NZ,1968,OUT-CH-2  переход вперед, чтобы
                                        напечатать ':'.
                BIT   2,(FLAGS2)        Если ':' 'не в
                JR    Z,196C,OUT-CH-3   кавычках', переход вперед;
                                        т.е. межоперационный маркер.
                JR    1968,OUT-CH-2     ':' являются внутренними
                                        кавычками и могут быть
                                        теперь напечатаны.
195A OUT-CH-1   CP    +22               Для печати получить
                JR    NZ,1968,OUT-CH-2  все символы, кроме '"'
                PUSH  AF                Запись символьного кода, пока
                                        изменяется 'режим кавычек'.
                LD    A,(FLAGS2)        Выбрать FLAGS2 и пе-
                XOR   +04               ребросить 2 разряд.
                LD    (FLAGS2),A        Ввести исправленное значение
                POP   AF                и восстановить код символа.
1968 OUT-CH-2   SET   2,(FLAGS)         Сигнал - 'следующий символ
                                        печатается в режиме L'.
196C OUT-CH-3   RST   0010,PRINT-A-1    Перед возвратом печатается
                RET                     текущий символ.
```

Примечание: Последовательность проверок текущего символа, которая определяет, "печатается ли следующий символ в режиме 'K' или 'L'".
Также заметьте, как программа игнорирует ':' в операторе REM.

ПОДПРОГРАММА 'LINE-ADDR' ('Адрес строки')
Для заданного номера строки в регистровой паре HL, эта подпрограмма возвращает начальный адрес этой строки или 'первой строки после' в регистровой паре HL, и начало предыдущей строки в регистровой паре DE.
Если использовался номер строки, то будет установлен признак 0. Однако, если заменяется 'первая строка после', то признак 0 возвращается сброшенным.

```
196E LINE-ADDR  PUSH  HL                Запись заданного номера
                                        строки.
                LD    HL,(PROG)         Выбрать системную
                LD    D,H               переменную PROG и
```

```
                LD    E,L               передать адрес в ре-
                                        гистровую пару DE.
```

Теперь введем цикл для проверки номера каждой строки программы с заданным номером, пока номера не сопоставятся или превысятся.

```
1974 LINE-AD-1  POP   BC                Заданный номер строки.
                CALL  1980,CP-LINES     Сравнить заданный номер
                                        строки с адресованным.
                RET   NC                Возврат, если сброшен
                PUSH  BC                перенос; в противном случае,
                CALL  19B8,NEXT-ONE     адресовать следующий
                                        номер строки.
                EX    DE,HL             Переключить указатели и
                JR    1974,LINE-AD-1    перейти назад, чтобы
                                        рассмотреть следующую строку
                                        программы.
```

ПОДПРОГРАММА 'COMPARE LINE NUMBERS' ('Сравнить номера срок')
Заданный в регистровой паре BC номер строки сравнивается с адресованным номером.

```
1980 CP-LINES   LD    A,(HL)            Выбрать старший байт
                CP    B                 адресованного номера строки
                                        и сравнить его. Если они не
                RET   NZ                сравниваются, возврат.
                INC   HL                Следующими сравниваются
                                        младшие байты.
                LD    A,(HL)            Возврат с установлен-
                DEC   HL                ным признаком переноса,
                CP    C                 если адресованный номер
                RET                     строки еще должен достигнуть
                                        заданного номера.
```

ПОДПРОГРАММА 'FIND EACH STATEMENT' ('Найти каждый оператор')
1. Она может использоваться для нахождения 'D'-го оператора в BASIC-строке - возвращаясь с регистровой парой HL, адресующей ячейку перед началом оператора, и установленным признаком переноса.

2. Кроме того, подпрограмма может использоваться для нахождения оператора, который начинается с заданного кода токена (в регистре E).

```
1988            INC   HL                Не используется.
                INC   HL
                INC   HL
198B EACH-STMT  LD    (CH-ADD),HL       Установить CH-ADD в
                                        текущий байт.
                LD    C,+00             Установить признак
                                        'вне кавычек'.
```

Вводится цикл для обработки каждого оператора в BASIC-строке.

```
1990 EACH-S-1   DEC   D                 Уменьшить 'D' и вер-
                RET   Z                 нуться, если обнаружен
                                        требуемый оператор.
                RST   0020,NEXT-CHAR    Выбор следующего ко-
                CP    E                 да символа и переход,
                JR    NZ,199A,EACH-S-3  если он не сопоставляется
```

```
                                        с заданным кодом токена.
                AND   A                 Но если сопоставляется,
                RET                     возврат со сброшенными
                                        признаками переноса и 0.
```

Теперь вводится другой цикл, чтобы рассмотреть отдельные символы в строке и найти место, где заканчивается оператор.

```
1998 EACH-S-2   INC   HL                Обновить указатель и
                LD    A,(HL)            выбрать новый код.
199A EACH-S-3   CALL  18B6,NUMBER       Сделать произвольный шаг.
                LD    (CH-ADD),HL       Обновить CH-ADD.
                CP    +22               Если символ не '"',
                JR    NZ,19A5,EACH-S-4  переход вперед.
                DEC   C                 В противном случае задать
                                        'признак кавычек'.
19A5 EACH-S-4   CP    +3A               Если символ ':', пе-
                JR    Z,19AD,EACH-S-5   реход вперед.
                CP    +CB               Если код не является токеном
                JR    NZ,19B1,EACH-S-6  'THEN', переход вперед.
19AD EACH-S-5   BIT   0,C               Прочитать 'признак кавычек'
                JR    Z,1990,EACH-S-1   и перейти назад к концу
                                        каждого оператора (в том
                                        числе после 'THEN').
19B1 EACH-S-6   CP    +0D               Переход назад, если
                JR    NZ,1998,EACH-S-2  не в конце BASIC-строки.
                DEC   D                 Уменьшить счетчик операторов
                                        и перед возвратом задать
                                        признак переноса.
                SCF
                RET
```

ПОДПРОГРАММА 'NEXT-ONE' ('Следующий')
Эта подпрограмма используется для обнаружения 'следующей строки' в программной области или 'следующей переменной' в области переменных. Подпрограмма обеспечивает 6 различных типов переменных, которые используются я системе SPECTRUM.

```
19B8 NEXT-ONE   PUSH  HL                Запись адреса текущей
                                        строки переменной.
                LD    A,(HL)            Выбор первого байта.
                CP    +40               Если ищется 'следующая
                JR    C,19D5,NEXT-O-3   строка', переход вперед.
                BIT   5,A               Если ищется следующая строка
                JR    Z,19D6,NEXT-O-4   (как последовательность
                                        литер) или переменная
                                        массива, переход вперед.
                ADD   A,A               Переход вперед с
                                        простой числовой
                JP    M,19C7,NEXT-O-1   переменной или
                                        переменной FOR-NEXT.
                CCF                     Удлиняется только имя
                                        числовой переменной.
19C7 NEXT-O-1   LD    BC,+0005          Числовая переменная
                JR    NC,19CE,NEXT-O-2  займет 5 ячеек, а
                                        управляющей переменной FOR-
                LD    C,+12             NEXT потребуется 18 ячеек.
19CE NEXT-O-2   RLA                     Признак переноса становится
```

```
                                        сброшенным только для пере-
                                        менных с длинным именем;
                                        пока конечный символ длинно-
                                        го имени не будет достигнут.
                INC   HL                Увеличить указатель
                LD    A,(HL)            и выбрать новый код.
                JR    NC,19CE,NEXT-O-2  Переход назад, если предыду-
                                        щий код не является послед-
                                        ним кодом имени переменной.
                JR    19DB,NEXT-O-5     Теперь переход вперед
                                        (BC = +0005 или +0012).
19D5 NEXT-O-3   INC   HL                Шаг за младший байт
                                        номера строки.
19D6 NEXT-O-4   INC   HL                Теперь указывается
                                        младший байт длины.
                LD    C,(HL)            Выбрать длину в ре-
                INC   HL                гистровую пару BC.
                LD    B,(HL)
                INC   HL                Допускается для вклю-
                                        чающего байта.
```

Во всех случаях находится адрес 'следующей' строки или переменной.

```
19DB NEXT-O-5   ADD   HL,BC             Указать первый байт
                                        'следующей' строки
                                        или переменной.
                POP   DE                Выбрать адрес предыдущей
                                        строки или переменной и
                                        продолжать в подпрограмме
                                        'разность'.
```

ПОДПРОГРАММА 'DIFFERENCE' ('Разность')
В регистровой паре BC формируется 'длина' между двумя 'началами'. Указатели переформируются, при возврате заменяются.

```
19DD DIFFER     AND   A                 Подготовка для истинного
                                        вычитания.
                SBC   HL,DE             Найти длину от одного
                LD    B,H               'начала' до следующего и
                LD    C,L               передать ее в регистровую
                                        пару BC.
                ADD   HL,DE             Переформировать адре-
                EX    DE,HL             са и заменить их перед
                RET                     возвратом.
```

ПОДПРОГРАММА 'RECLAIMING' ('Восстановление')
Точка входа RECLAIM-1 используется, когда адрес первой ячейки, подлежащей восстановлению, находится в регистровой паре DE, а адрес первой ячейки, которая остается одна, находится в регистровой паре HL. Точка входа RECLAIM-2 используется, когда регистровая пара HL указывает первую ячейку, подлежащую восстановлению, а регистровая пара BC содержит число подлежащих восстановлению байт.

```
19E5 RECLAIM-1  CALL  19DD,DIFFER       Используйте подпрограмму
                                        'разность', чтобы выработать
                                        соответствующие значения.
19E8 RECLAIM-2  PUSH  BC                Заменить количество байт,
```

```
                                        подлежащих восстановлению.
                LD    A,B               Перед изменением все
                CPL                     указатели системных
                LD    B,A               переменных, которые
                LD    A,C               выше области, должны
                CPL                     выть уменьшены на
                LD    C,A               'BC', т.к. это число
                INC   BC                явл. дополнением до двух.
                CALL  1664,POINTERS
                EX    DE,HL             Возврат адреса 'первой
                POP   HL                ячейки' в регистровую пару
                ADD   HL,DE             DE и переформирование адреса
                                        первой остающейся ячейки.
                PUSH  DE                Запись первой ячейки,
                LDIR                    пока ведется фактичес-
                POP   HL                кое восстановление.
                RET                     Теперь возврат.
```

ПОДПРОГРАММА 'E-LINE-NO'
Эта подпрограмма используется для чтения номера строки, находящейся в области редактирования. Если номера нет, т.е. прямая BASIC-строка, то номер строки рассматривается как 0.
Во всех случаях номер строки возвращается а регистровую пару BC.

```
19FB E-LINE-NO  LD    HL,(E-LINE)       Перевести указатель в
                                        строку редактирования.
                DEC   HL                Установить CH-ADD, чтобы
                LD    (CH-ADD),HL       указывать ячейку перед
                                        любым номером.
                RST   0020,NEXT-CHAR    Передача первого кода
                                        в регистр A.
                LD    HL,+5C92          Однако, перед рассмотрением
                LD    (STKEND),HL       кода сделать область памяти
                                        калькулятора временной об-
                                        ластью стека калькулятора.
                CALL  2D3B,INT-TO-FP    Теперь читаются цифры
                                        номера строки. Возврат 0,
                                        если номер не существует.
                CALL  2DA2,FP-TO-BC     Поместить номер строки в
                                        регистровую пару BC.
                JR    C,1A15,E-L-1      Если номер превышает
                                        '65536', переход вперед.
                LD    HL,+D8F0          Иначе, проверить его
                                        с '10000'.
                ADD   HL,BC
1A15 E-L-1      JP    C,1C8A,REPORT-C   Выдать сообщение C, если
                                        свыше '9999'.
                JP    16C5,SET-STK      Возврат через SET-STK, кото-
                                        рая восстанавливает стек
                                        калькулятора на его месте.
```

ПОДПРОГРАММА 'REPORT AND LINE NUMBER PRINTING' ('Печать сообщений и номеров строк')
Точка входа OUT-NUM-1 приведет к тому, что число в регистровой паре BC будет напечатано. Однако, любое значение свыше '9.999' не будет печататься правильно.
Точка входа OUT-NUM-2 приводит к тому, что напечатается число

непрямо адресованное регистровой паре HL. На этот раз появится необходимое количество начальных пробелов. Числа будут правильно печататься до величины '9999'.

```
1A1B OUT-NUM-1    PUSH  DE                 Записать через под-
                  PUSH  HL                 программу других регистров.
                  XOR   A                  Очистить регистр A.
                  BIT   7,B                Переход вперед, чтобы
                                           напечатать 0 вместо
                  JR    NZ,1A42,OUT-NUM-4  '-2', когда сообщение на
                                           строке редактирования.
                  LD    H,B                Переслать номер в ре-
                  LD    L,C                гистровую пару HL.
                  LD    E,+FF              Признак 'нет начальных
                                           пробелов'.
                  JR    1A30,OUT-NUM-3     Чтобы напечатать чис-
                                           ло, переход вперед.
1A28 OUT-NUM-2    PUSH  DE                 Записать регистровую пару DE.
                  LD    D,(HL)             Выбрать номер в регистровую
                  INC   HL                 пару DE и записать указатель
                  LD    E,(HL)             (обновленный).
                  PUSH  HL
                  EX    DE,HL              Переслать в регистровую пару
                  LD    E,+20              HL номер и признак -
                                           'начальные пробелы необходи-
                                           мо напечатать'.
```

Теперь печатается номер в виде целого числа, находящегося в HL.

```
1A30 OUT-NUM-3    LD    BC,+FC18           Это '-1000'.
                  CALL  192A,OUT-SP-NO     Печать первой цифры.
                  LD    BC,+FF9C           Это '-100'.
                  CALL  192A,OUT-SP-NO     Печать второй цифры.
                  LD    C,+F6              Это '-10'.
                  CALL  192A,OUT-SP-NO     Печать третьей цифры.
                  LD    A,L                Переместить оставшуюся
                                           часть номера в регистр A.
1A42 OUT-NUM-4    CALL  15EF,OUT-CODE      Печать цифры.
                  POP   HL                 Перед возвратом вос-
                  POP   DE                 становить регистры.
                  RET
```

ИНТЕРПРЕТАЦИЯ СТРОК И КОМАНД БЕЙСИКА

ТАБЛИЦЫ СИНТАКСИСА

1. Таблица смещений
Представлены значения смещений для каждой из пятидесяти команд BASIC.

	Команда	Адрес		Команда	Адрес
1A48 DEFB +B1	DEF FN	1AF9	1A61 DEFB +94	BORDER	1AF5
1A49 DEFB +CB	CAT	1B14	1A62 DEFB +56	CONTINUE	1AB8
1A4A DEFB +BC	FORMAT	1B06	1A63 DEFB +3F	DIM	1AA2
1A4B DEFB +BF	MOVE	1B0A	1A64 DEFB +41	REM	1AA5
1A4C DEFB +C4	ERASE	1B10	1A65 DEFB +2B	FOR	1A90
1A4D DEFB +AF	OPEN #	1AFC	1A66 DEFB +17	GO TO	1A7D
1A4E DEFB +B4	CLOSE #	1B02	1A67 DEFB +1F	GO SUB	1A86
1A4F DEFB +93	MERGE	1AE2	1A68 DEFB +37	INPUT	1A9F
1A50 DEFB +91	VERIFY	1AE1	1A69 DEFB +77	LOAD	1AE0
1A51 DEFB +92	BEEP	1AE3	1A6A DEFB +44	LIST	1AAE
1A52 DEFB +95	CIRCLE	1AE7	1A6B DEFB +0F	LET	1A7A
1A53 DEFB +98	INK	1AEB	1A6C DEFB +59	PAUSE	1AC5
1A54 DEFB +98	PAPER	1AEC	1A6D DEFB +2B	NEXT	1A98
1A55 DEFB +98	FLASH	1AED	1A6E DEFB +43	POKE	1AB1
1A56 DEFB +98	BRIGHT	1AEE	1A6F DEFB +2D	PRINT	1A9C
1A57 DEFB +98	INVERSE	1AEF	1A70 DEFB +51	PLOT	1AC1
1A58 DEFB +98	OVER	1AF0	1A71 DEFB +3A	RUN	1AAB
1A59 DEFB +98	OUT	1AF1	1A72 DEFB +6D	SAVE	1ADF
1A5A DEFB +7F	LPRINT	1AD9	1A73 DEFB +42	RANDOMIZE	1AB5
1A5B DEFB +81	LLIST	1ADC	1A74 DEFB +0D	IF	1A81
1A5C DEFB +2E	STOP	1A8A	1A75 DEFB +49	CLS	1ABE
1A5D DEFB +6C	READ	1AC9	1A76 DEFB +5C	DRAW	1AD2
1A5E DEFB +6E	DATA	1ACC	1A77 DEFB +44	CLEAR	1ABB
1A5F DEFB +70	RESTORE	1ACF	1A78 DEFB +15	RETURN	1A8D
1A60 DEFB +48	NEW	1AA8	1A79 DEFB +5D	COPY	1AD6

2. Таблица параметров
Для каждой из пятидесяти команд BASIC представлены до восьми элементов в таблице параметров. Эти элементы охватывают детали классов команд, требуемые разделители и, где необходимо, адреса командных процедур.

```
1A7A P-LET        DEFB  +01              CLASS-01
                  DEFB  +3D              '='
                  DEFB  +02              CLASS-02
1A7D P-GO-TO      DEFB  +06              CLASS-06
                  DEFB  +00              CLASS-00
                  DEFB  +67,+1E          GO-TO,1E67
1A81 P-IF         DEFB  +06              CLASS-06
                  DEFB  +CB              'THEN'
                  DEFB  +05              CLASS-05
                  DEFB  +F0,+1C          IF,1CF0
1A86 P-GO-SUB     DEFB  +06              CLASS-06
                  DEFB  +00              CLASS-00
                  DEFB  +ED,+1E          GO-SUB,1EED
1A8A P-STOP       DEFB  +00              CLASS-00
                  DEFB  +EE,+1C          STOP,1CEE
1A8D P-RETURN     DEFB  +00              CLASS-00
```

```
                DEFB  +23,+1F           RETURN,1F23
1A90 P-FOR      DEFB  +04               CLASS-04
                DEFB  +3D               '='
                DEFB  +06               CLASS-06
                DEFB  +CC               'TO'
                DEFB  +06               CLASS-06
                DEFB  +05               CLASS-05
                DEFB  +03,+1D           FOR,1D03
1A98 P-NEXT     DEFB  +04               CLASS-04
                DEFB  +00               CLASS-00
                DEFB  +AB,+1D           NEXT,1DAB
1A9C P-PRINT    DEFB  +05               CLASS-05
                DEFB  +CD,+1F           PRINT,1FCD
1A9F P-INPUT    DEFB  +05               CLASS-05
                DEFB  +89,+20           INPUT,2089
1AA2 P-DIM      DEFB  +05               CLASS-05
                DEFB  +02,+2C           DIM,2C02
1AA5 P-REM      DEFB  +05               CLASS-05
                DEFB  +B2,+1B           REM,1BB2
1AA8 P-NEW      DEFB  +00               CLASS-00
                DEFB  +B7,+11           NEW,11B7
1AAB P-RUN      DEFB  +03               CLASS-03
                DEFB  +A1,+1E           RUN,1EA1
1AAE P-LIST     DEFB  +05               CLASS-05
                DEFB  +F9,+17           LIST,17F9
1AB1 P-POKE     DEFB  +08               CLASS-08
                DEFB  +00               CLASS-00
                DEFB  +80,+1E           POKE,1E80
1AB5 P-RANDOM   DEFB  +03               CLASS-03
                DEFB  +4F,+1E           RANDOMIZE,1E4F
1AB8 P-CONT     DEFB  +00               CLASS-00
                DEFB  +5F,+1E           CONTINUE,1E5F
1ABB P-CLEAR    DEFB  +03               CLASS-03
                DEFB  +AC,+1E           CLEAR,1EAC
1ABE P-CLS      DEFB  +00               CLASS-00
                DEFB  +6B,+0D           CLS,0D6B
1AC1 P-PLOT     DEFB  +09               CLASS-09
                DEFB  +00               CLASS-00
                DEFB  +DC,+22           PLOT,22DC
1AC5 P-PAUSE    DEFB  +06               CLASS-06
                DEFB  +00               CLASS-00
                DEFB  +3A,+1F           PAUSE,1F3A
1AC9 P-READ     DEFB  +05               CLASS-05
                DEFB  +ED,+1D           READ,1DED
1ACC P-DATA     DEFB  +05               CLASS-05
                DEFB  +27,+1E           DATA,1E27
1ACF P-RESTORE  DEFB  +03               CLASS-03
                DEFB  +42,+1E           RESTORE,1E42
1AD2 P-DRAW     DEFB  +09               CLASS-09
                DEFB  +05               CLASS-05
                DEFB  +82,+23           DRAW,2382
1AD6 P-COPY     DEFB  +00               CLASS-00
                DEFB  +AC+0E            COPY,0EAC
1AD9 P-LPRINT   DEFB  +05               CLASS-05
                DEFB  +C9,+1F           LPRINT,1FC9
1ADC P-LLIST    DEFB  +05               CLASS-05
                DEFB  +F5,+17           LLIST,17F5
1ADF P-SAVE     DEFB  +0B               CLASS-0B
1AE0 P-LOAD     DEFB  +0B               CLASS-0B
```

```
1AE1 P-VERIFY     DEFB  +0B               CLASS-0B
1AE2 P-MERGE      DEFB  +0B               CLASS-0B
1AE3 P-BEEP       DEFB  +08               CLASS-08
                  DEFB  +00               CLASS-00
                  DEFB  +F8,+03           BEEP,03F8
1AE7 P-CIRCLE     DEFB  +09               CLASS-09
                  DEFB  +05               CLASS-05
                  DEFB  +20,+23           CIRCLE,2320
1AEB P-INK        DEFB  +07               CLASS-07
1AEC P-PAPER      DEFB  +07               CLASS-07
1AED P-FLASH      DEFB  +07               CLASS-07
1AEE P-BRIGHT     DEFB  +07               CLASS-07
1AEF P-INVERSE    DEFB  +07               CLASS-07
1AF0 P-OVER       DEFB  +07               CLASS-07
1AF1 P-OUT        DEFB  +08               CLASS-08
                  DEFB  +00               CLASS-00
                  DEFB  +7A,+1E           OUT,1E7A
1AF5 P-BORDER     DEFB  +06               CLASS-06
                  DEFB  +00               CLASS-00
                  DEFB  +94,+22           BORDER,2294
1AF9 P-DEF-FN     DEFB  +05               CLASS-05
                  DEFB  +60,+1F           DEF-FN,1F60
1AFC P-OPEN       DEFB  +06               CLASS-06
                  DEFB  +2C               ','
                  DEFB  +0A               CLASS-0A
                  DEFB  +00               CLASS-00
                  DEFB  +36,+17           OPEN,1736
1B02 P-CLOSE      DEFB  +06               CLASS-06
                  DEFB  +00               CLASS-00
                  DEFB  +E5,+16           CLOSE,16E5
1B06 P-FORMAT     DEFB  +0A               CLASS-0A
                  DEFB  +00               CLASS-00
                  DEFB  +93,+17           CAT-ETC,1793
1B0A P-MOVE       DEFB  +0A               CLASS-0A
                  DEFB  +2C               ','
                  DEFB  +0A               CLASS-0A
                  DEFB  +00               CLASS-00
                  DEFB  +93,+17           CAT-ETC,1793
1B10 P-ERASE      DEFB  +0A               CLASS-0A
                  DEFB  +00               CLASS-00
                  DEFB  +93,+17           CAT-ETC,1793
1B14 P-CAT        DEFB  +00               CLASS-00
                  DEFB  +93,+17           CAT-ETC,1793
```

Примечание: Для разных командных классов приведены следующие требования:

```
CLASS-00          -     В дальнейшем отсутствуют операнды.
CLASS-01          -     Используется в LET. Требуется переменная.
CLASS-02          -     Используется в LET. Должно следовать числовое или
                        строковое выражение.
CLASS-03          -     Может следовать числовое выражение. По умолчанию
                        используется 0.
CLASS-04          -     Должна следовать переменная из одного символа.
CLASS-05          -     Может вызываться набор элементов.
CLASS-06          -     Должно следовать числовое выражение.
CLASS-07          -     Обрабатывает элементы цвета.
CLASS-08          -     Должны следовать два, разделенных запятыми,
                        числовых выражения.
```

```
CLASS-09        -     То же, что и для CLASS-08, но элементы цвета могут
                      предшествовать выражениям.
CLASS-0A        -     Должно следовать строковое выражение.
CLASS-0B        -     Обрабатывает программы для работы с магнитофоном.
```

'ГЛАВНАЯ ПРОГРАММА СИНТАКСИЧЕСКОГО АНАЛИЗА' ИНТЕРПРЕТАТОРА BASIC
Программа синтаксического анализа интерпретатора BASIC вводится на LINE-SCAN при проверке синтаксиса и на LINE-RUN, если в программе BASIC выполнен один или несколько операторов.
Каждый оператор рассматривается по очереди и системная переменная CH-ADD используется, чтобы указывать код каждого оператора, который появляется в программной области или области редактирования.

```
1B17 LINE-SCAN    RES   7,(FLAGS)          Сигнал - 'проверка син-
                                            таксиса'.
                  CALL  19FB,E-LINE-NO     CH-ADD указывает первый код
                                            после номера строки.
                  XOR   A                  Системная переменная
                  LD    (SUBPPC),A         SUBPPC инициализиро-
                  DEC   A                  вана +00, а ERR-NR - +FF.
                  LD    (ERR-NR),A
                  JR    1B29,STMT-L-1      Переход вперед, чтобы
                                            рассмотреть первый оператор
                                            в строке.
```

ОПЕРАТОРНЫЙ ЦИКЛ
Каждый оператор по очереди рассматривается, пока не будет достигнут конец строки.

```
1B28 STMT-LOOP    RST   0020,NEXT-CHAR     CH-ADD двигается
1B29 STMT-L-1     CALL  16BF,SET-WORK      вдоль строки.
                  INC   (SUBPPC)           При каждом проходе цикла
                                            увеличивается SUBPPC.
                  JP    M,1C8A,REPORT-C    В одной строке допускается
                                            только '127' операторов.
                  RST   0018,GET-CHAR      Выбор символа.
                  LD    B,+00              Очистить регистр.
                  CP    +0D                Символ 'возврат каретки'?
                  JR    Z,1BB3,LINE-END    Если да, переход.
                  CP    +3A                Опять прогон цикла,
                  JR    Z,1B28,STMT-LOOP   если символ ':'.
```

Оператор идентифицирован, поэтому сначала рассматривается его начальная команда.

```
                  LD    HL,+1B76           Загрузка машинного стека
                  PUSH  HL                 адресом возврата STMT-RET.
                  LD    C,A                Временно записать команду в C,
                  RST   0020,NEXT-CHAR     пока продвигается CH-ADD.
                  LD    A,C                Уменьшить командный код
                  SUB   +CE                на +CE; выдавая диапазон для
                                            пятидесяти команд +00 - +31.
                  JP    C,1C8A,REPORT-C    Выдать соответствующую ошибку,
                                            если нет кода команды.
                  LD    C,A                Переместить код команды
                                            в регистровую пару BC
                                            (B содержит +00).
```

```
                LD    HL,+1A48          Базовый адрес таблицы
                                        смещений синтаксиса.
                ADD   HL,BC             Требуемое смещение
                LD    C,(HL)            передается в регистр C и
                ADD   HL,BC             используется для вычисления
                                        базового адреса для элемен-
                                        тов команд в таблице
                                        параметров.
                JR    1B55,GET-PARAM    Переход вперед с этим
                                        адресом в цикл просмотра.
```

Каждая программа класса команд, пригодная для данной команды, исполняется по очереди. Любые из требуемых разделителей также рассматриваются.

```
1B52 SCAN-LOOP  LD    HL,(T-ADDR)       Временный указатель элемен-
                                        тов в таблице параметров.
1B55 GET-PARAM  LD    A,(HL)            Выбор по очереди каждого
                                        элемента.
                INC   HL                Обновить указатель элементов
                LD    (T-ADDR),HL       для следующего прохода.
                LD    BC,+1B52          Перезагрузить машинный стек
                PUSH  BC                адресом возврата SCAN-LOOP.
                LD    C,A               Скопировать элемент
                                        в регистр C.
                CP    +20               Переход вперед, если
                JR    NC,1B6F,SEPARATOR элементом является 'сепаратор'
                                        (разделитель).
                LD    HL,+1C01          Базовый адрес таблицы
                                        'класс команд'.
                LD    B,+00             Очистить регистр B и
                ADD   HL,BC             проиндексировать таблицу.
                LD    C,(HL)            Выбор смещения и вычисление
                ADD   HL,BC             начального адреса требуемой
                                        программы классов команд.
                PUSH  HL                Поместить адрес на машинный
                                        стек.
                RST   0018,GET-CHAR     Перед выполнением непрямого
                DEC   B                 перехода к программе классов
                RET                     команд передайте код команды
                                        в регистр A и задайте
                                        регистр B +FF.
```

ПОДПРОГРАММА 'SEPARATOR' ('Разделитель')
Сообщение 'Nonsense in BASIC' выдается, если отсутствует требуемый разделитель. Но отметим, что при проверке синтаксиса сообщение на экране не появляется – только 'маркер ошибки'.

```
1B6F SEPARATOR  RST   0018,GET-CHAR     Выбирается и сравнивается
                CP    C                 текущий символ с элементом
                                        в таблице параметров.
                JP    NZ,1C8A,REPORT-C  Если не сопоставляется,
                                        сообщение об ошибке.
                RST   0020,NEXT-CHAR    Шаг за правильный
                RET                     символ и возврат.
```

ПОДПРОГРАММА 'STMT-RET'
После правильной интерпретации оператора осуществляется возврат на эту точку входа.

```
1B76 STMT-RET     CALL  1F54,BREAK-KEY      После каждого оператора
                                            проверяется клавиша BREAK.
                  JR    C,1B7D,STMT-R-1     Если не нажата, переход
                                            вперед.
```

Сообщение L - 'BREAK в программе'.

```
1B7B REPORT-L     RST   0008,ERROR-1        Вызов подпрограммы
                  DEFB  +14                 обработки ошибок.
```

Т.к. клавиша BREAK не была нажата, продолжать в этом месте.

```
1B7D STMT-R-1     BIT   7,(NSPPC)           Переход вперед, если не дол-
                  JR    NZ,1BF4,STMT-NEXT   жен быть выполнен 'переход'.
                  LD    HL,(NEWPPC)         Выбор номера 'новой строки'
                  BIT   7,H                 и переход вперед, если не
                  JR    Z,1B9E,LINE-NEW     обрабатывается следующий
                                            оператор в области
                                            редактирования.
```

ТОЧКА ВХОДА 'LINE-RUN'
Эта точка входа используется всякий раз, когда строка в области редактирования должна 'запускаться'. В этом случае будет установлен признак синтаксис/запуск (7 разряд FLAGS).
Точка входа, также, используется при проверке синтаксиса строки в области редактирования, которая имеет более одного оператора (7 разряд FLAGS будет сброшен).

```
1B8A LINE-RUN     LD    HL,+FFFE            Строка в области ре-
                                            дактирования рассмат-
                  LD    (PPC),HL            ривается, как строка '-2'.
                  LD    HL,(WORKSP)         Сделать HL указываю-
                  DEC   HL                  щей маркер конца об-
                  LD    DE,(E-LINE)         ласти редактирования
                  DEC   DE                  и DE-ячейку перед на-
                                            чалом этой области.
                  LD    A,(NSPPC)           Выбор номера следующего,
                  JR    1BD1,NEXT-LINE      подлежащего обработке
                                            оператора, перед переходом
                                            вперед.
```

ПОДПРОГРАММА 'LINE-NEW'
Осуществляется переход в программу, при этом должен быть найден начальный адрес новой строки.

```
1B9E LINE-NEW     CALL  196E,LINE-ADDR      Найден начальный адрес стро-
                                            ки или 'первой строки после'.
                  LD    A,(NSPPC)           Отобрать номер оператора.
                  JR    Z,1BBF,LINE-USE     Если требуемая строка обна-
                  AND   A                   ружена, переход вперед;
                  JR    NZ,1BEC,REPORT-N    в противном случае проверьте
                                            достоверность номера
                                            оператора - должен быть 0.
                  LD    B,A                 Проверьте также, что
                  LD    A,(HL)              'первая строка после' -
```

```
                AND   +C0                 не после фактического
                LD    A,B                 'конца программы'.
                JR    Z,1BBF,LINE-USE     Переход вперед с правильным
                                          адресом; в противном случае
                                          сообщение 'OK'.
```

Сообщение 0 - 'OK'.

```
1BB0 REPORT-0   RST   0008,ERROR-1        Использовать подпро-
                DEFB  +FF                 грамму обработки ошибок.
```

Примечание: Очевидно, что это не ошибка в прямом смысле, но в ином случае переход за программу.

КОМАНДНАЯ ПРОЦЕДУРА 'REM'
Адрес возврата в STMT-RET сбрасывается, что приводит к игнорированию остатка строки.

```
1BB2 REM        POP   BC                  Удалить адрес STMT-RET.
```

ПРОГРАММА 'LINE-END'
При проверке синтаксиса осуществляется простой возврат, но при 'выполнении' программы адрес, содержащийся в NXTLIN, должен быть проверен до использования.

```
1BB3 LINE-END   CALL  2530,SYNTAX-Z       Возврат, если проверя-
                RET   Z                   ется синтаксис; выб-
                LD    HL,(NXTLIN)         рать адрес в NXTLIN.
                LD    A,+C0               Возврат, если адрес
                AND   (HL)                после конца програм-
                RET   NZ                  мы - 'запуск' закончен.
                XOR   A                   Перед продолжением
                                          сигнал '0 оператора'.
```

ПРОГРАММА 'LINE-USE'
Эта короткая программа выполняет три функции:
1. Изменяет оператор 0 на оператор '1';
2. Находит номер новой строки и вводит его в PPC;
3. Формирует адрес начала последующей строки.

```
1BBF LINE-USE   CP    +01                 оператор 0 стано-
                ADC   A,+00               вится '1'.
                LD    D,(HL)              Номер используемой
                INC   HL                  строки выбирается и
                LD    E,(HL)              передается в PPC.
                LD    (PPC),DE
                INC   HL                  Теперь находится
                LD    E,(HL)              'длина' строки.
                INC   HL
                LD    D,(HL)
                EX    DE,HL               Переключить значения.
                ADD   HL,DE               Формируется адрес начала
                INC   HL                  следующей строки в HL и
                                          ячейка перед первым символом
                                          'следующей' строки в DE.
```

ПРОГРАММА 'NEXT-LINE'
На входе регистровая пара HL указывает ячейку после конца 'следующей' строки, подлежащей обработке, а регистровая пара DE - ячейку перед первым символом строки. Происходит обращение к строкам в программной области, а также к строке в области редактирования - где следующая строка будет такой же строкой, пока интерпретируются операторы.

```
1BD1 NEXT-LINE    LD    (NXTLIN),HL         Установить 1 раз NXTLIN при
                                            завершении текущей строки.
                  EX    DE,HL               Как обычно CH-ADD
                  LD    (CH-ADD),HL         указывает ячейку перед пер-
                                            вым рассматриваемым символом.
                  LD    D,A                 Выбирается номер оператора.
                  LD    E,+00               Если используется EACH-STMT,
                                            очищается регистр E.
                  LD    (NSPPC),+FF         Сигнал 'нет перехода'.
                  DEC   D                   Номер оператора минус 1
                  LD    (SUBPPC),D          идет в SUBPPC.
                  JP    Z,1B28,STMT-LOOP    Теперь может быть
                                            рассмотрен первый оператор.
                  INC   D                   Тем не менее должен
                  CALL  198B,EACH-STMT      быть найден 'начальный
                                            адрес' для последующих
                                            операторов.
                  JR    Z,1BF4,STMT-NEXT    Переход вперед, если
                                            оператор не существует.
```

Сообщение N - 'Потерян оператор'.

```
1BEC REPORT-N     RST   0008,ERROR-1        Вызов подпрограммы
                  DEFB  +16                 обработки ошибок.
```

ПОДПРОГРАММА 'CHECK-END'
Эта важная подпрограмма вызывается из разных мест в программе-мониторе при проверке синтаксиса в строке редактирования. Цель программы - выдать сообщение об ошибке, если не достигнут конец оператора и переместиться на следующий оператор, если синтаксис правильный.

```
1BEE CHECK-END    CALL  2530,SYNTAX-Z       Не продолжать, если
                  RET   NZ                  не проверяется синтаксис.
                  POP   BC                  Удалить адреса
                  POP   BC                  SCAN-LOOP & STMT-RET
                                            перед продолжением в STMT-NEXT.
```

ПРОГРАММА 'STMT-NEXT'
Если текущий символ 'возврат каретки', то 'следующий оператор' находится в 'следующей строке'; если ':', то он в той же строке; но если обнаружен любой другой символ, то это ошибка синтаксиса.

```
1BF4 STMT-NEXT    RST   0018,GET-CHAR       Выбор текущего символа.
                  CP    +0D                 Рассмотреть 'следующую
                  JR    Z,1BB3,LINE-END     щую строку', если это
                                            'возврат каретки'.
                  CP    +3A                 Рассмотреть 'следующий
                  JP    Z,1B28,STMT-LOOP    оператор', если это ':'.
                  JP    1C8A,REPORT-C       Иначе, ошибка синтаксиса.
```

ТАБЛИЦА 'КОМАНДНЫХ КЛАССОВ'

Адрес	Смещение	Номер класса	Адрес	Смещение	Номер класса
1C01	0F	CLASS-00,1C10	1C07	7B	CLASS-06,1C82
1C02	1D	CLASS-01,1C1F	1C08	8E	CLASS-07,1C96
1C03	4B	CLASS-02,1C4E	1C09	71	CLASS-08,1C7A
1C04	09	CLASS-03,1C0D	1C0A	B4	CLASS-09,1CBE
1C05	67	CLASS-04,1C6C	1C0B	81	CLASS-0A,1C8C
1C06	0B	CLASS-05,1C11	1C0C	CF	CLASS-0B,1CDB

'КОМАНДНЫЕ КЛАССЫ - 00, 03 И 05'
За командами класса 03 может, или не может следовать число. Например, RUN и RUN 200.

```
1C0D CLASS-03     CALL  1CDE,FETCH-NUM      Выбирается число, но по
                                            умолчанию используется 0.
```

Команды класса 00 не должны иметь операнды. Например, COPY и CONTINUE.

```
1C10 CLASS-00     CP    A                   Установить признак 0
                                            для дальнейшего.
```

За командами класса 05 может следовать набор элементов. Например, PRINT и PRINT "222".

```
1C11 CLASS-05     POP   BC                  Во всех случаях сбрасывается
                                            адрес SCAN-LOOP.
                  CALL  Z,1BEE,CHECK-END    Если обрабатываются команды
                                            классов 00 и 03 и проверен
                                            синтаксис, то переместиться,
                                            чтобы рассмотреть следующий
                                            оператор.
                  EX    DE,HL               Запись указателя строк
                                            в регистровую пару DE.
```

ПРОГРАММА 'JUMP-C-R'
После рассмотрения элементов командных классов и разделительных элементов в таблице параметров, выполняется переход на соответствующую командную процедуру.

```
1C16 JUMP-C-R     LD    HL,(T-ADDR)         Выбор указателя элементов
                  LD    C,(HL)              в таблице параметров и
                  INC   HL                  выбор адреса требуемой
                  LD    B,(HL)              командной процедуры.
                  EX    DE,HL               Командный обмен указателей
                  PUSH  BC                  и выполнение непрямого перехода
                  RET                       на командную процедуру.
```

'КОМАНДНЫЕ КЛАССЫ - 01, 02 И 04'
Эти три командных класса используются командами обработки переменных - LET, FOR и NEXT и косвенно READ и INPUT.
Командный класс 01 связан с идентификацией переменных в LET, READ или INPUT.

```
1C1F CLASS-01     CALL  28B2,LOOK-VARS      Смотрит в область системных
                                            переменных, чтобы определить
                                            использовалась или нет
                                            переменная
```

ПОДПРОГРАММА 'VARIABLE IN ASSIGNMENT' ('Присваивание значения переменной')
Эта подпрограмма вырабатывает соответствующие значения для системных переменных DEST и STRLEN.

```
1C22 VAR-A-1      LD    (FLAGX),+00         Инициализировать FLAGX в +00.
                  JR    NC,1C30,VAR-A-2     Если переменная использовалась
                                            раньше, переход вперёд.
                  SET   1,(FLAGX)           Флаг 'новая переменная'.
                  JR    NZ,1C46,VAR-A-3     Выдаётся ошибка при попытке
                                            использовать безразмерный массив.
```

Сообщение 2 - 'Переменная не найдена'.

```
1C2E REPORT-2     RST   0008,ERROR-1        Вызов подпрограммы обработки
                  DEFB  +01                 ошибок.
```

Продолжение с обработкой существующих переменных.

```
1C30 VAR-A-2      CALL  Z,2996,STK-VARS     Параметры простых строковых
                                            переменных и всех переменных
                                            массивов передаются на стек
                                            калькулятора (STK-VARS будет,
                                            если требуется, 'вырезать' строку.
                  BIT   6,(FLAGS)           Если обрабатывается числовая
                  JR    NZ,1C46,VAR-A-3     переменная, то переход вперёд.
                  XOR   A                   Очистить регистр A.
                  CALL  2530,SYNTAX-Z       Если не проверен синтаксис,
                  CALL  NZ,2BF1,STK-FETCH   выбираются параметры строки или
                                            переменная строкового массива.
                  LD    HL,+5C71            Это FLAGX.

                  OR    (HL)                Бит 0 устанавливается только
                  LD    (HL),A              когда после обработки завершаются
                                            простые строки, сигналазирующие
                                            'должна быть уничтожена старая
                                            копия'.
                  EX    DE,HL               Теперь HL указывает на строки или
                                            элемент массива.
```

Теперь устанавливаются STRLEN и DEST. Для всех числовых переменных и 'новых' строковых переменных и переменных строковых массивов STRLEN-io содержит 'букву' имени переменной. Но для старых строковых переменных и переменных строковых массивов завершённых или 'вырезанных' она содержит 'длину' при 'присваивании'.

```
1C46 VAR-A-3      LD    (STRLEN),BC         Установить STRLEN.
```

DEST содержит адрес для 'адреса назначения' 'старой переменной', но в действительности 'источник' для 'новой' переменной.

```
                  LD    (DEST),HL           Установить DEST и выйти.
                  RET
```

Командный класс 02 связан с фактическим вычислением значений, подлежащих присваиванию в операторе LET.

```
1C4E CLASS-02     POP   BC                 Отбрасывается адрес SCAN-LOOP
                  CALL  1C56,VAL-FET-1     Выполняется присваивание.
                  CALL  1BEE,CHECK-END     Переместиться на следующий
                                           оператор через CHECK-END, если
                  RET                      проверяется синтаксис, или через
                                           STMT-RET, если в 'исполняющей
                                           системе'.
```

ПОДПРОГРАММА 'FETCH A VALUE' ('Выбор значения')
Эта подпрограмма используется LET, READ и INPUT, чтобы сначала вычислить, а затем присвоить значение ранее созданной переменной.
Точка входа VAL-FET-1 используется LET и READ и рассматривается FLAGS, тогда как точка входа VAL-FET-2 используется INPUT и рассматривает FLAGX.

```
1C56 VAL-FET-1    LD    A,(FLAGS)          Используется FLAGS.
1C59 VAL-FET-2    PUSH  AF                 Запись FLAGS или FLAGX.
                  CALL  24FB,SCANNING      Вычисляется следующее выражение.
                  POP   AF                 Выбор старого FLAGS или FLAGX.
                  LD    D,(FLAGS)          Выбор нового FLAGS.
                  XOR   D                  Должны сопоставляться числовая
                  AND   +40                или строковая переменная и
                                           выражение.
                  JR    NZ,1C8A,REPORT-C   Выдать сообщение C, если они
                                           не сопоставляются.
                  BIT   7,D                Переход вперёд, чтобы выполнить
                  JP    NZ,2AFF,LET        фактическое присваивание, если
                                           не проверяется синтаксис, при
                  RET                      котором просто возврат.
```

ПРОГРАММА 'COMMAND CLASS 04' ('Командный класс 04')
Эта точка входа используется операторами FOR и NEXT.

```
1C6C CLASS-04     CALL  28B2,LOOK-VARS     Ищется в области переменных
                                           использованная переменная.
                  PUSH  AF                 Сохранить AF пока проверяется
                  LD    A,C                байт дискриминатора, чтобы
                  OR    +9F                обеспечить, что переменная
                  INC   A                  является управляющей FOR-NEXT.
                  JR    NZ,1C8A,REPORT-C
                  POP   AF                 Восстановить AF и перейти назад,
                  JR    1C22,VAR-A-1       чтобы создать переменную, которая
                                           обнаружила 'переменную в
                                           присваивании'.
```

ПОДПРОГРАММА 'EXPECT NUMERIC/STRING EXPRESSIONS' ('Ожидаются числовые/строковые выражения')
Эта последовательность коротких подпрограмм, которые используются для выборки результата вычислений следующего выражения. Результат из одного выражения возвращается как 'последнее значение' на стек калькулятора.
Точка входа NEXT-2NUM используется, когда CH-ADD нуждается в обновлении, чтобы указать начало первого выражения.

```
1C79 NEXT-2NUM    RST   0020,NEXT-CHAR      Продвижение CH-ADD.
```

Точка входа EXPT-2NUM (EQU. CLASS-08) допускается для двух числовых выражений, разделённых запятой и подлежащих вычислению.

```
1C7A EXPT-2NUM    CALL  1C82,EXPT-1NUM      Вычислить каждое выражение. По
     (CLASS-08)                             очереди вычисляется первое.
                  CP    +2C                 Выдать сообщение об ошибке, если
                  JR    NZ,1C8A             разделитель не является запятой.
                  RST   0020,NEXT-CHAR      Продвижение CH-ADD.
```

Точка входа EXPT-1NUM (EQU. CLASS-06) допускается для одного числового выражения, подлежащего вычислению.

```
1C82 EXPT-1NUM    CALL  24FB,SCANNING       Вычислить следущее выражение.
     (CLASS-06)
                  BIT   6,(FLAGS)           Пока результат числовой, возврат.
                  RET   NZ                  Иначе - ошибка.
```

Сообщение C - 'Nonsense in BASIC' (Бессмыслица в БЕЙСИК-программе).

```
1C8A REPORT-C     RST   0008,ERROR-1        Вызов подпрограммы обработки
                  DEFB  +0B                 ошибок.
```

Точка входа EXPT-EXP (EQU. CLASS-0A) допускается для одного строкового выражения, подлежащего вычислению.

```
1C8C EXPT-EXP     CALL  24FB,SCANNING       Вычислить следующее выражение.
     (CLASS-0A)
                  BIT   6,(FLAGS)           Если результат обозначает строку,
                  RET   Z                   то возврат. Иначе выдаётся
                  JR    1C8A,REPORT-C       сообщение об ошибке.
```

ПОДПРОГРАММА 'SET PERMANENT COLOURS' SUBROUTINE (EQU. CLASS-07) ('Установить постоянный цвет')
Эта подпрограмма допускается для текущих временных цветов, которые должны стать постоянными. Командный класс 07 в действительности является командной процедурой для команд элементов цвета.

```
1C96 PERMS        BIT   7,(FLAGS)           Флаг синтаксис/запуск.
     (CLASS-07)
                  RES   0,(TV-FLAG)         Флаг 'основной экран'.
                  CALL  NZ,0D4D,TEMPS       Только во время запуска вызывается
                                            TEMPS, чтобы обеспечить временные
                                            цвета, как цвета основного
                                            экрана.
```

```
                POP   AF                Отбросить адрес возврата -
                                        SCAN-LOOP.
                LD    A,(T-ADDR)        Выбор младшего байта T-ADDR
                                        и вычитание +13, чтобы выдать
                SUB   +13               диапазон от +D9 до +DE, который
                                        является кодами токенов для
                                        INK - OVER.
                CALL  21FC,CO-TEMP-4    Переход вперёд, чтобы изменить,
                                        как предписано, временные цвета
                                        оператором BASIC.
                CALL  1BEE,CHECK-END    Если проверяется синтаксис,
                                        переместиться на следующий
                                        оператор
                LD    HL,(ATTR-T)       Теперь значения временных цветов
                LD    (ATTR-P),HL       сделаем постоянными
                                        (both ATTR-P & MASK-P).
                LD    HL,+5C91          Это P-FLAG; он тоже должен
                LD    A,(HL)            быть рассмотрен.
```

Следующие команды копируют чётные разряды байта в нечётные. В действительности создаются постоянные разряды такие же, как и временные.

```
                RLCA                    Переместить маску влево.
                XOR   (HL)              Отобразить на маске только
                AND   +AA               чётные разряды другого байта.
                XOR   (HL)
                LD    (HL),A            Восстановить результат.
                RET
```

ПРОГРАММА 'COMMAND CLASS 09' ('Командный класс 09')
Эта программа используется операторами PLOT, DRAW и CIRCLE для того, чтобы определить условия по умолчанию 'FLASH 8; BRIGHT 8; PAPER 8;'. Они установлены перед любыми рассмотренными встроенными элементами цвета.

```
1CBE CLASS-09   CALL  2530,SYNTAX-Z     Если проверка синтаксиса, то
                JR    Z,1CD6,CL-09-1    переход вперёд.
                RES   0,(TV-FLAG)       Флаг 'основной экран'.
                CALL  0D4D,TEMPS        Установить временные цвета для
                                        основного экрана.
                LD    HL,+5C90          Это MASK-T.
                LD    A,(HL)            Выбор его текущего значения, но
                OR    +F8               только держать его часть INK
                                        'немаскируемой'.
                LD    (HL),A            Восстановить значение, которое
                                        теперь обозначает 'FLASH 8;
                                        BRIGHT 8; PAPER 8;'.
                RES   6,(P-FLAG)        Обеспечивается также
                                        NOT 'PAPER 9'.
                RST   0018,GET-CHAR     Выбрать текущий символ перед
                                        продолжением, чтобы обработать
                                        встроенные элементы цвета.
1CD6 CL-09-1    CALL  21E2,CO-TEMP      Работа с локально доминирующими
                                        элементами цветов.
                JR    1C7A,EXPT-2NUM    Теперь получить первые два
                                        операнда для PLOT, DRAW или CIRCLE.
```

ПРОГРАММА 'COMMAND CLASS 0B' ('Командный класс 0B')
Эта программа используется операторами SAVE, LOAD, VERIFY и MERGE.

```
1CDB CLASS-0B     JP    0605,SAVE-ETC       Переход к программе
                                            работы с лентой.
```

ПОДПРОГРАММА 'FETCH A NUMBER' ('Выбор числа')
Эта подпрограмма приводит к вычислению числового выражения, но, если оно отсутствует, используется ноль.

```
1CDE FETCH-NUM    CP    +0D                 Если в конце строки, переход
                  JR    Z,1CE6,USE-ZERO     вперёд.
                  CP    +3A                 Но если не в конце оператора,
                  JR    NZ,1C82,EXPT-1NUM   переход к EXPT-1NUM.
```

Теперь для добавления на стек калькулятора нулевого значения используется калькулятор.

```
1CE6 USE-ZERO     CALL  2530,SYNTAX-Z       Если проверяется синтаксис,
                  RET   Z                   операцию не выполнять.
                  RST   0028,FP-CALC        Использовать калькулятор.
                  DEFB  +A0,stk-zero        Теперь 'последнее значение'
                  DEFB  +38,end-calc        равно 0.
                  RET                       Возврат с добавленным 0 на стек
                                            калькулятора.
```

КОМАНДНЫЕ ПРОЦЕДУРЫ
Раздел монитора от 1CEE до 23FA содержит большинство командных процедур интерпретатора BASIC.

КОМАНДНАЯ ПРОЦЕДУРА 'STOP'
Командная процедура STOP содержит только обращение к подпрограмме обработки ошибок.

```
1CEE STOP         RST   0008,ERROR-1        Вызов подпрограммы обработки
     (REPORT-9)   DEFB  +08                 ошибок.
```

КОМАНДНАЯ ПРОЦЕДУРА 'IF' ('Если')
На входе значение выражения между IF и THEN является 'последним значением' на стеке калькулятора. Если присутствует логическая истина, то рассматривается следующий оператор, в противном случае строка рассматривается для её завершения.

```
1CF0 IF           POP   BC                  Обрабатывается адрес возврата
                                            - STMT-RET.
                  CALL  2530,SYNTAX-Z       Если проверяется синтаксис,
                  JR    Z,1D00,IF-1         то переход вперёд.
```

Теперь используется калькулятор, чтобы удалить последнее значение на стеке калькулятора, но оставляется регистровая пара DE, адресующая первый байт значения.

```
                  RST   0028,FP-CALC        Используется калькулятор.
                  DEFB  +02,delete          Удаляется текущее последнее
                  DEFB  +38,end-calc        значение со стека калькулятора.
                  EX    DE,HL               Сделать HL указывающим на первый
                  CALL  34E9,TEST-ZERO      байт и вызвать TEST-ZERO.
                  JP    C,1BB3,LINE-END     Если значение было 'FALSE',
                                            переход на следующую строку.
1D00 IF-1         JP    1B29,STMT-L-1       Но если было 'TRUE', переход на
                                            следующий после THEN оператор.
```

КОМАНДНАЯ ПРОЦЕДУРА 'FOR' ('Для')
Командная процедура вводится с VALUE и LIMIT (значение и граница оператора FOR) уже на вершине стека калькулятора.

```
1D03 FOR          CP    +CD                 Если не выдан 'STEP', переход
                  JR    NZ,1D10,F-USE-1     вперёд.
                  RST   0020,NEXT-CHAR      Продвижение CH-ADD и выбор
                  CALL  1C82,EXPT-1NUM      значения STEP.
                  CALL  1BEE,CHECK-END      Переместиться на следующий
                  JR    1D16,F-REORDER      оператор, если идёт проверка
                                            синтаксиса, иначе переход вперёд.
```

STEP не было, поэтому должно использоваться значение '1'

```
1D10 F-USE-1      CALL  1BEE,CHECK-END      Если идёт проверка синтаксиса,
                                            переместиться на следующий
                  RST   0028,FP-CALC        оператор, иначе использовать
                  DEFB  +A1,stk-one         калькулятор, чтобы разместить
                  DEFB  +38,end-calc        '1! на стеке калькулятора.
```

На стеке калькулятора находятся три значения – VALUE (v), the LIMIT (l) и STEP (s). Эти значения должны обрабатываться.

```
1D16 F-REORDER    RST   0028,FP-CALC        v, l, s
                  DEFB  +C0,st-mem-0        v, l, s (mem-0 = s)
                  DEFB  +02,delete          v, l
                  DEFB  +01,exchange        l, v
                  DEFB  +E0,get-mem-0       l, v, s
                  DEFB  +01,exchange        l, s, v
                  DEFB  +38,end-calc
```

Управляющая переменная FOR теперь устанавливается и обрабатывается как временная область памяти калькулятора.

```
                  CALL  2AFF,LET            Обнаружена, или, если необходимо,
                                            создана переменная
                  LD    (MEM),HL            (используется V).
```

Переменная, которая была обнаружена, может быть и простой числовой переменной, использующей только шесть ячеек, и в этом случае её необходимо расширить.

```
                DEC   HL                Выбор односимвольного имени
                LD    A,(HL)            переменной.
                SET   7,(HL)            Установка 7-го бита.
                LD    BC,+0006          По крайней мере будет 6 байт.
                ADD   HL,BC             Создать HL, указывающую место
                                        на них.
                RLCA                    Циклически сдвинуть имя и
                JR    C,1D34,F-L&S      перейти, если уже была переменная
                                        FOR.
                LD    C,+0D             Иначе создать дополнительно
                CALL  1655,MAKE-ROOM    13 байт.
                INC   HL                Опять создать HL, указывающую
                                        позицию LIMIT.
```

Теперь добавляются начальные значения для LIMIT и STEP.

```
1D34 F-L&S      PUSH  HL                Сохраним указатель.
                RST   0028,FP-CALC      1, s
                DEFB  +02,delete        1
                DEFB  +02,delete        -
                DEFB  +38,end-calc      DE ещё указывает '1'.
                POP   HL                Восстановим указатель,
                EX    DE,HL             заменяются оба указателя.
                LD    C,+0A             Пересылаются ещё 10 байт LIMIT
                LDIR                    и STEP.
```

Теперь вводятся номер строки и номер оператора цикла.

```
                LD    HL,(PPC)          Текущий номер строки.
                EX    DE,HL             Замена регистров перед
                LD    (HL),E            добавлением номера строки в
                INC   HL                управляющую переменную FOR.
                LD    (HL),D
                LD    D,(SUBPPC)
                INC   D
                INC   HL
                LD    (HL),D
```

Подпрограмма NEXT-LOOP вызывается для проверки возможности 'передачи' и, если она возможна, выполняется возврат; иначе, оператор после цикла FOR - NEXT должен быть идентифицирован.

```
                CALL  1DDA,NEXT-LOOP    'Передача' возможна ?
                RET   NC                Если да, возврат.
                LD    B,(STRLEN-lo)     Выбор имени переменной.
                LD    HL,(PPC)          Скопировать текущий номер строки
                LD    (NEWPPC),HL       в NEWPPC.
                LD    A,(SUBPPC)        Выбор текущего номера оператора
                NEG                     и дополнение его до двух.
                LD    D,A               Передача результата в регистр D.
                LD    HL,(CH-ADD)       Выбор текущего значения CH-ADD.
                LD    E,+F3             Для 'NEXT' будет поиск.
```

Теперь в программной области осуществляется поиск от текущей точки впереди для первого появления NEXT, за которыми следует правильная переменная.

```
1D64 F-LOOP       PUSH  BC                 Запись имени переменной.
                  LD    BC,(NXTLIN)        Выбор текущего значения
                                           NXTLIN.
                  CALL  1D86,LOOK-PROG     Теперь исследуется программная
                                           область и будет меняться BC
                                           с каждой новой проверенной
                                           строкой.
                  LD    (NXTLIN),BC        До возврата запись указателя.
                  POP   BC                 Восстановить имя переменной.
                  JR    C,1D84,REPORT-I    Если нет дальнейших NEXT, выдача
                                           ошибки.
                  RST   0020,NEXT-CHAR     Продвинутся за обнаруженный NEXT.
                  OR    +20                Допускается для букв верхнего и
                  CP    B                  и нижнего регистров до проверки
                                           нового имени переменной.
                  JR    Z,1D7C,F-FOUND     Если оно сопоставляется, то
                                           переход вперёд.
                  RST   0020,NEXT-CHAR     Опять продвижения CH-ADD и, если
                  JR    1D64,F-LOOP        переменная неправильная, то
                                           переход назад.
```

NEWPPC содержит номер строки, в которой обнаружен правильный NEXT. Теперь должен быть найден номер оператора и занесен в NSPPC.

```
1D7C F-FOUND      RST   0020,NEXT-CHAR     Продвижение CH-ADD.
                  LD    A,+01              Счётчик операторов в регистре D
                  SUB   D                  считает операторы в обратную
                                           сторону от 0, поэтому он должен
                                           вычитаться из '1'.
                  LD    (NSPPC),A          Запомним результат.
                  RET                      Возврат в STMT-RET.
```

Сообщение I - 'FOR без NEXT'.

```
1D84 REPORT-I     RST   0008,ERROR-1       Вызов подпрограммы обработки
                  DEFB  +11                ошибок.
```

ПОДПРОГРАММА 'LOOK-PROG'
Эта подпрограмма используется, чтобы обнаружить появление или DATA, DEF FN или NEXT. На входе соответствующий код оператора находится в регистре E, а регистровая пара HL указывает на начало области поиска.

```
1D86 LOOK-PROG    LD    A,(HL)             Выбор текущего символа.
                  CP    +3A                Переход вперёд, если это ':',
                  JR    Z,1DA3,LOOK-P-2    который будет обозначать
                                           дополнительные операторы в
                                           текущей строке.
```

Теперь для проверки каждой из последующих строки в программе вводится цикл.

```
1D8B LOOK-P-1     INC   HL                 Выбор старшего байта номера строки
                  LD    A,(HL)             и возврат установленным флагом
```

```
                AND   +C0               переноса, в программе нет
                SCF                     дальнейших строк.
                RET   NZ
                LD    B,(HL)            Выбирается номер строки и
                INC   HL                передаётся в NEWPPC.
                LD    C,(HL)
                LD    (NEWPPC),BC
                INC   HL                Затем отбирается строка.
                LD    C,(HL)
                INC   HL
                LD    B,(HL)
                PUSH  HL                Пока в BC формируется адрес
                ADD   HL,BC             конца строки, записывается
                LD    B,H               указатель.
                LD    C,L
                POP   HL                Восстанавливаем указатель.
                LD    D,+00             Задать счётчик указателя в 0.
1DA3 LOOK-P-2   PUSH  BC                Запускается указатель конца
                CALL  198B,EACH-STMT    строки, пока проверяются
                POP   BC                операторы в строке.
                RET   NC                Если было 'появление', выполнить
                JR    1D8B,LOOK-P-1     возврат; в противном случае
                                        рассмотреть следующую строку.
```

КОМАНДНАЯ ПРОЦЕДУРА 'NEXT' ('Следующий')
Уже определена 'переменная в присваивании' (см. CLASS-04,1C6C); и она остаётся для изменения VALUE.

```
1DAB NEXT       BIT   1,(FLAGX)         Если переменная не обнаружена,
                JP    NZ,1C2E,REPORT-2  переход для выдачи сообщения
                                        об ошибке.
                LD    HL,(DEST)         Выбирается адрес переменной
                BIT   7,(HL)            и далее проверяется имя.
                JR    Z,1DD8,REPORT-1
```

Потом калькулятором обрабатывается VALUE и STEP переменной.

```
                INC   HL                Шаг за имя.
                LD    (MEM),HL          Сделать переменную временной
                                        областью памяти.
                RST   0028,FP-CALC      -
                DEFB  +E0,get-mem-0     v
                DEFB  +E2,get-mem-2     v, s
                DEFB  +0F,addition      v+s
                DEFB  +C0,st-mem-0      v+s
                DEFB  +02,delete        -
                DEFB  +38,end-calc      -
```

Теперь проверяется результат сложения VALUE и STEP с LIMIT с помощью вызове NEXT-LOOP.

```
                CALL  1DDA,NEXT-LOOP    Проверка нового VALUE
                                        с LIMIT
                RET   C                 Возврат, если завершён
                                        цикл FOR-NEXT.
```

В противном случае выбрать номер строки и оператор организации циклов.

```
                LD    HL,(MEM)          Найти адрес младшего байта
                LD    DE,+000F          номера строки организации
                ADD   HL,DE             циклов.
                LD    E,(HL)            Теперь выбираем этот номер.
                INC   HL
                LD    D,(HL)
                INC   HL
                LD    H,(HL)            За ним следует номер оператора.
                EX    DE,HL             Перед переходом вперёд заменить
                JP    1E73,GO-TO-2      номера, чтобы обработать их
                                        как строку назначения
                                        команды GO TO.
```

Сообщение 1 - 'NEXT без FOR'

```
1DD8 REPORT-1   RST   0008,ERROR-1      Вызов подпрограммы обработки
                DEFB  +00               ошибок.
```

ПОДПРОГРАММА 'NEXT-LOOP'
Эта подпрограмма используется для определения текущим VALUE завышено ли LIMIT. Должен учитываться знак STEP.
Подпрограмма возвращается с установленным флагом переноса, если LIMIT завышено.

```
1DDA NEXT-LOOP  RST   0028,FP-CALC      -
                DEFB  +E1,get-mem-1     l
                DEFB  +E0,get-mem-0     l, v
                DEFB  +E2,get-mem-2     l, v, s
                DEFB  +36,less-0        l, v,(1/0)
                DEFB  +00,jump-true     l, v,(1/0)
                DEFB  +02,to NEXT-1     l, v,(1/0)
                DEFB  +01,exchange      v, l
1DE2 NEXT-1     DEFB  +03,subtract      v-l or l-v
                DEFB  +37,greater-0     (1/0)
                DEFB  +00,jump-true     (1/0)

                DEFB  +04,to NEXT-2     -
                DEFB  +38,end-calc      -
                AND   A                 Сбросить флаг перенос и
                RET                     возврат - возможен цикл.
```

Однако, если цикл невозможен, должен быть установлен флаг переноса.

```
1DE9 NEXT-2     DEFB  +38,end-calc      -
                SCF                     Установка флага переноса
                RET                     и возврат.
```

КОМАНДНАЯ ПРОЦЕДУРА 'READ' ('Читать')
Команда READ предназначена для чтения списка DATA и аналогична последовательности операторов LET.
Каждое присваивание внутри одного оператора READ выполняется по очереди. Системная переменная X-PTR используется как ячейка памяти для указателя оператора READ, пока используется CH-ADD, чтобы прошагать вдоль стека DATA.

```
1DEC READ-3     RST   0020,NEXT-CHAR    Приходить сюда на каждом
                                        проходе после первого,
```

```
                                        чтобы перемещаться вдоль
                                        оператора READ.
1DED READ       CALL  1C1F,CLASS-01     Рассматривается, использова-
                                        лась ли переменная раньше;
                                        если да, найти существующий
                                        элемент.
                CALL  2530,SYNTAX-Z     Если проверяется синтаксис,
                JR    Z,1E1E,READ-2     переход вперед.
                RST   0018,GET-CHAR     Запись текущего ука-
                LD    (X-PTR),HL        зателя CH-ADD в X-PTR.
                LD    HL,(DATADD)       Выбор текущего ука-
                LD    A,(HL)            зателя списка DATA и
                CP    +2C               переход вперед, если
                JR    Z,1E0A,READ-1     новый оператор DATA
                                        не должен быть найден.
                LD    E,+E4             Поиск 'DATA'.
                CALL  1D86,LOOK-PROG    Если поиск удачен,
                JR    NC,1E0A,READ-1    переход вперед.
```

Сообщение E - 'Вне DATA'.

```
1E08 REPORT-E   RST   0008,ERROR-1      Вызов подпрограммы
                DEFB  +0D               обработки ошибок.
```

Продолжение - получить значение из списка DATA.

```
1E0A READ-1     CALL  0077,TEMP-PTR1    Продвижение указателя
                                        вдоль списка DATA и
                                        установка CH-ADD.
                CALL  1C56,VAL-FET-1    Выбор значения и присвоение
                                        его переменной.
                RST   0018,GET-CHAR     Выбрать текущее значение
                LD    (DATADD),HL       CH-ADD и запомнить его в
                                        DATADD.
                LD    HL,(X-PTR)        Выбрать указатель оператора
                LD    (X-PTR-hi),+00    READ и очистить X-PTR.
                CALL  0078,TEMP-PTR2    Сделать один раз CH-ADD,
                                        указывающим оператор READ.
1E1E READ-2     RST   0018,GET-CHAR     Получить текущий символ и
                CP    +2C               посмотреть, является ли
                                        он ','.
                JR    Z,1DEC,READ-3     Если да, переход назад,
                                        т.к. есть другие элементы; в
                CALL  1BEE,CHECK-END    противном случае возврат или
                RET                     через CHECK-END (если прове-
                                        ряется синтаксис), или через
                                        команду RET (в STMT-RET).
```

КОМАНДНАЯ ПРОЦЕДУРА 'DATA' ('Данные')
Во время проверки синтаксиса проверяется оператор DATA, этим обеспечивается то, что он содержит последовательность достоверных выражений, разделенных запятыми. Но оператор передается в 'использующую систему'.

```
1E27 DATA       CALL  2530,SYNTAX-Z     Переход вперед, если
                JR    NZ,1E37,DATA-2    синтаксис не проверяется.
```

Теперь вводится цикл для обработки каждого выражения в операторе DATA.

```
1E2C DATA-1       CALL  24FB,SCANNING       Просмотр следующего
                                            выражения.
                  CP    +2C                 Проверке правильности
                                            разделителя - ','.
                  CALL  NZ,1BEE,CHECK-END   Но если он не отождест-
                                            вляется, переместиться
                                            на следующий оператор.
                  RST   0020,NEXT-CHAR      Пока еще проверяются
                  JR    1E2C,DATA-1         операторы, прогнать цикл.
```

Оператор DATA должен передаваться в 'исполняющую систему'.

```
1E37 DATA-2       LD    A,+E4               Оператор 'DATA', который
                                            должен быть передан.
```

ПОДПРОГРАММА 'PASS-BY' ('Передача')
На входе регистр A будет содержать или токен 'DATA', или токен 'DEF FN', в зависимости от типа оператора, который передается.

```
1E39 PASS-BY      LD    B,A                 Сделать регистровую пару BC
                                            содержащей самый большой
                                            номер.
                  CPDR                      Ищете токен, переходя
                                            назад вдоль оператора.
                  LD    DE,+0200            Теперь вдоль строки после
                  JP    198B,EACH-STMT      оператора. ('D-1'-ый оператор
                                            от текущей позиции).
```

КОМАНДНАЯ ПРОЦЕДУРА 'RESTORE' ('Восстановление')
Операнд для команды RESTORE берется как номер строки, если операнд не выдается, используется 0.
Точка входа REST-RUN используется командной процедурой RUN.

```
1E42 RESTORE      CALL  1E99,FIND-INT2      Поместить операнд в
                                            регистровую пару BC.
1E45 REST-RUN     LD    H,B                 Передать результат в
                  LD    L,C                 регистровую пару HL.
                  CALL  196E,LINE-ADDR      Теперь находится адрес этой
                                            строки или 'первой строки
                                            после'.
                  DEC   HL                  Сделать DATADD ука-
                  LD    (DATADD),HL         зывающим 'ячейку до'.
                  RET                       Возврат.
```

КОМАНДНАЯ ПРОЦЕДУРА 'RANDOMIZE'
Еще раз операнд помещается в регистровую пару BC и передается в требуемую системную переменную. Однако, если операнд является 0, используется взамен значение в FRAMES1 и FRAMES2.

```
1E4F RANDOMIZE    CALL  1E99,FIND-INT2      Выбор операнда.
                  LD    A,B                 Переход вперед, если
                  OR    C                   значение операнда не
                  JR    NZ,1E5A,RAND-1      ноль.
                  LD    BC,(FRAMES1)        Выбор взамен двух байт
                                            FRAMES.
1E5A RAND-1       LD    (SEED),BC           Теперь перед возвратом
```

```
                RET                     введем результат в системную
                                        переменную SEED.
```

КОМАНДНАЯ ПРОЦЕДУРА 'CONTINUE' ('Продолжение')
Требуемый номер строки и номер оператора внутри этой строки сделаны объектом перехода.

```
1E5F CONTINUE   LD    HL,(OLDPPC)       Номер строки.
                LD    D,(OSPPC)         Номер оператора.
                JR    1E73,GO-TO-2      Переход вперед.
```

КОМАНДНАЯ ПРОЦЕДУРА 'GO TO'
Операндом GO TO должен быть номер строки в диапазоне '1-9999', но фактическая проверка проводится с верхним значением '61439'.

```
1E67 GO-TO      CALL  1E99,FIND-INT2    Выбрать операнд и пе-
                LD    H,B               редать его в регист-
                LD    L,C               ровую пару HL.
                LD    D,+00             Задать номер оператора 0.
                LD    A,H               Выдать сообщение об ошибке -
                CP    +F0               целое вне диапазона -
                JR    NC,1E9F,REPORT-B  со строкой свыше '61439'.
```

Точка входа GO-TO-2 используется для определения номера следующей строки, подлежащей обработке в нескольких отдельных случаях.

```
1E73 GO-TO-2    LD    (NEWPPC),HL       Вводится номер строки и
                LD    (NSPPC),D         затем номер оператора.
                RET                     Возврат в STMT-RET.
```

КОМАНДНАЯ ПРОЦЕДУРА 'OUT' ('Вывод')
Из стека калькулятора для команды OUT выбираются два параметра и, как это предписано, используются.

```
1E7A OUT        CALL  1E85,TWO-PARAM    Выбираются операнды.
                OUT   (C),A             Фактическая команда OUT.
                RET                     Возврат в STMT-RET.
```

КОМАНДНАЯ ПРОЦЕДУРА 'POKE'
Операция POKE выполняется аналогичным образом.

```
1E80 POKE       CALL  1E85,TWO-PARAM    Выбираются операнды.
                LD    (BC),A            Фактическая операция POKE.
                RET                     Возврат в STMT-RET.
```

ПОДПРОГРАММА 'TWO-PARAM' ('Два параметра')
Самый верхний параметр на стеке вычислителя должен быть помещен регистр. Если он отрицательный, выполняется дополнение до двух. Второй параметр должен быть помещен в регистровую пару.

```
1E85 TWO-PARAM  CALL  2DD5,FP-TO-A      Выбираем параметр.
                JR    C,1E9F,REPORT-B   Если это очень большое
                                        число, выдается ошибка.
                JR    Z,1E8E,TWO-P-1    Если число положительное,
```

```
                NEG                     переход вперед, если отрица-
                                        тельное - дополнение до двух.
1E8E TWO-P-1    PUSH  AF                Запись первого параметра,
                CALL  1E99,FIND-INT2    пока выбирается второй.
                POP   AF                Перед возвратом восстанавли-
                RET                     вается первый параметр.
```

ПОДПРОГРАММЫ 'FIND INTEGERS' ('Поиск целых чисел')
Выбирается 'последнее значение' на стеке калькулятора и помещается в один регистр или регистровую пару с помощью ввода FIND-INT1 и FINT-INT2 соответственно.

```
1E94 FIND-INT1  CALL  2DD5,FP-TO-A      Выбор 'последнего значения'.
                JR    1E9C,FIND-I-1     Переход вперед.
1E99 FIND-INT2  CALL  2DA2,FP-TO-BC     Выбор 'последнего значения'.
1E9C FIND-I-1   JR    C,1E9F,REPORT-B   В обоих случаях переполне-
                                        ние обозначается установ-
                                        ленным признаком переноса.
                RET   Z                 Возврат со всеми положи-
                                        тельными числами, какие
                                        есть в диапазоне.
```

Сообщение B - 'Целое вне диапазона'.

```
1E9F REPORT-B   RST   0008,ERROR-1      Вызов подпрограммы
                DEFB  +0A               обработки ошибок.
```

КОМАНДНАЯ ПРОЦЕДУРА 'RUN' ('Запуск')
Параметр команды RUN передается в NEWPPC с помощью командной процедуры GO TO. Затем, перед возвратом, выполняются операторы 'RESTORE 0' и 'CLEAR 0'.

```
1EA1 RUN        CALL  1E67,GO-TO        Задается как это требуется
                                        NEWPPC.
                LD    BC,+0000          Теперь выполняется
                CALL  1E45,REST-RUN     'RESTORE 0'.
                JR    1EAF,CLEAR-1      Выход через командную
                                        процедуру CLEAR.
```

КОМАНДНАЯ ПРОЦЕДУРА 'CLEAR' ('Очищать')
Эта программа допускает очистку области переменных, очищенную областью изображений и перемещенную RAMTOP. В результате последней операции машинный стек перестраивается с тем, чтобы очистить стек GO SUB.

```
1EAC CLEAR      CALL  1E99,FIND-INT2    Выбор операнда - по
                                        умолчанию используется 0.
1EAF CLEAR-RUN  LD    A,B               Если операнд отличен
                OR    C                 от 0, переход вперед.
                JR    NZ,1EB7,CLEAR-1   Когда вызывается из
                                        RUN, перехода нет.
                LD    BC,(RAMTOP)       Если 0, используется сущест-
                                        вующее значение в RAMTOP.
1EB7 CLEAR-1    PUSH  BC                Запись значения.
                LD    DE,(VARS)         Далее восстанавливаются
                LD    HL,(E-LINE)       все байты текущей
```

```
                DEC   HL                  области переменных.
                CALL  19E5,RECLAIM-1
                CALL  0D6B,CLS            Очистить область изображений.
```

Значение в регистровой паре BC, которое будет использоваться как RAMTOP, проверяется, чтобы обеспечить его значение как не очень малое и не очень большое.

The value in the BC register pair which will be used as RAMTOP is tested to ensure it is neither too low nor too high.

```
                LD    HL,(STKEND)         Текущее значение
                LD    DE,+0032            STKEND уменьшено перед
                ADD   HL,DE               проверкой на '50'.
                POP   DE                  Этим формируется нижняя
                                          граница.
                SBC   HL,DE
                JR    NC,1EDA,REPORT-M    RAMTOP также будет очень
                                          маленьким.
                LD    HL,(P-RAMT)         Для проверки верхней границы
                AND   A                   значение проверяется для
                SBC   HL,DE               RAMTOP с P-RAMT.
                JR    NC,1EDC,CLEAR-2     Если приемлемо, переход
                                          вперед.
```

Сообщение M - 'Не подходит RAMTOP'.

```
1EDA REPORT-M   RST   0008,ERROR-1        Вызов подпрограммы
                DEFB  +15                 обработки ошибок.
```

Продолжение с операцией с CLEAR.

```
1EDC CLEAR-2    EX    DE,HL               Теперь значение фактически
                LD    (RAMTOP),HL         может быть передано
                                          в RAMTOP.
                POP   DE                  Выбор адреса STMT-RET.
                POP   BC                  Выбор 'адреса ошибки'.
                LD    (HL),+3E            Ввести маркер конца стека
                                          GO SUB.
                DEC   HL                  Оставить одну ячейку.
                LD    SP,HL               Сделать указатель стека,
                                          указывающим пустой стек
                                          GO SUB.
                PUSH  BC                  Далее передать 'адрес ошиб-
                LD    (ERR-SP),SP         ки' в стек и записать адрес
                                          в ERR-RET.
                EX    DE,HL               Теперь непрямой возврат
                JP    (HL)                в STMT-RET.
```

Примечание: Когда программа вызывается из RUN, значения NEWPPC и NSPPC будут затронуты и операторы, поступающие после RUN, не смогут быть обнаружены перед переходом.

КОМАНДНАЯ ПРОЦЕДУРА 'GO SUB'
Текущее значение PPC и увеличенное значение SUBPPC заносятся в стек GO SUB.

```
1EED GO-SUB     POP   DE                  Запись адреса - STMT-RET.
                LD    H,(SUBPPC)          Выбрать номер опера-
                INC   H                   тора и увеличить его.
                EX    (SP),HL             Обменять 'адрес ошибки' с
```

```
                                        номером оператора.
                INC   SP                Восстановить использование
                                        ячейки.
                LD    BC,(PPC)          Далее запись текущего
                PUSH  BC                номера строки.
                PUSH  HL                Вернуть 'адрес сшибки' на
                LD    (ERR-SP),SP       машинный стек и для того,
                                        чтобы задать это, сбросить
                                        ERR-SP.
                PUSH  DE                Возврат адреса - STMT-RET.
                CALL  1E67,GO-TO-1      Теперь задать NEWPPC и NSPPC
                                        требуемыми значениями.
                LD    BC,+0014          Но перед переходом
                                        проверить место.
```

ПОДПРОГРАММА 'TEST-ROOM' ('Тест-памяти')
Выполняется серия тестов, чтобы убедиться в достаточности свободной памяти для предпринятой задачи.

```
1F05 TEST-ROOM  LD    HL,(STKEND)       Увеличить значение, взятое
                ADD   HL,BC             из STKEND, на значение,
                                        перенесенное в программу
                                        регистровой парой BC.
                JR    C,1F15,REPORT-4   Переход вперед, если
                EX    DE,HL             результат больше +FFFF.
                LD    HL,+0050          Попытаться еще раз
                ADD   HL,DE             использовать для
                JR    C,1F15,REPORT-4   дальнейших 80 байт.
                SBC   HL,SP             В конце проверка значения с
                                        адресом машинного стека.
                RET   C                 Если удовлетворительно,
                                        возврат.
```

Сообщение 4 - 'Вне памяти'.

```
1F15 REPORT-4   LD    L,+03             Это ошибка исполняющей
                JP    0055,ERROR-3      системы и маркер ошибки
                                        не используется.
```

ПОДПРОГРАММА 'FREE MEMORY' ('Свободная память')
В SPECTRUM нет команды BASIC 'FRE', но подпрограмма выполняет эту задачу. Оценка количества свободного места может проводиться с помощью 'PRINT 65536-USR 7962'.

```
1F1A FREE-MEM   LD    BC,+0000          Не допускаются лишние
                                        издержки.
                CALL  1F05,TEST-ROOM    Выполнить тест и пе-
                LD    B,H               редать результат в
                LD    C,L               регистр BC перед
                RET                     возвратом.
```

КОМАНДНАЯ ПРОЦЕДУРА 'RETURN' ('Возврат')
Номер строки и номер оператора, которые должны быть объектами 'возврата', выбираются из стека GO SUB.

```
1F23 RETURN     POP   BC                Выбор адреса STMT-RET.
```

```
                POP   HL                  Выбор адреса ошибки.
                POP   DE                  Выбор последнего элемента
                                          на стеке GOSUB.
                LD    A,D                 Проверен элемент, чтобы
                CP    +3E                 убедиться, является ли
                JR    Z,1F36,REPORT-7     он маркером конца стека
                                          GOSUB; если да, переход.
                DEC   SP                  Полный элемент использует
                                          только три ячейки.
                EX    (SP),HL             Обменять номер оператора
                                          с 'адресом ошибки'.
                EX    DE,HL               Пересылать номера оператора.
                LD    (ERR-SP),SP         Сбросить указатель ошибок.
                PUSH  BC                  Заменить адрес STMT-RET.
                JP    1E73,GO-TO-2        Переход назад, чтобы
                                          уменьшить NEWPPC и NSPPC.
```

Сообщение 7 - 'RETURN без GOSUB'.

```
1F36 REPORT-7   PUSH  DE                  Заменить конец маркера
                PUSH  HL                  и 'адрес ошибок'.
                RST   0008,ERROR-1        Вызов подпрограммы
                DEFB  +06                 обработки ошибок.
```

КОМАНДНАЯ ПРОЦЕДУРА 'PAUSE' ('Пауза')
Время PAUSE определяется подсчетом количества маскируемых прерываний, поскольку они появляется каждую 1/50 секунды.
PAUSE заканчивается или после соответствующего количества прерываний, или с помощью системной переменной FLAGS, обозначающей, что нажата клавиша.

```
1F3A PAUSE      CALL  1E99,FIND-INT2      Выбор операнда.
1F3D PAUSE-1    HALT                      Ждите маскируемое прерывание.
                DEC   BC                  Уменьшить счетчик.
                LD    A,B                 Если счетчик уменьшен до 0,
                OR    C                   PAUSE заканчивается.
                JR    Z,1F4F,PAUSE-END    Если операнд был 0,
                LD    A,B                 BC будет содержать
                AND   C                   +FFFF и это значение
                INC   A                   будет возвращено в 0.
                JR    NZ,1F49,PAUSE-2     Переход будет для всех
                INC   BC                  других значений операндов.
1F49 PAUSE-2    BIT   5,(FLAGS)           Переход назад, если
                JR    Z,1F3D,PAUSE-1      не нажата клавиша.
```

Время PAUSE теперь окончено.

```
1F4F PAUSE-END  RES   5,(FLAGS)           Сигнал 'нет нажатой клавиши'.
                RET                       Теперь возврат - в STMT-RET.
```

ПОДПРОГРАММА 'BREAK-KEY' ('Клавиша BREAK')
Эта подпрограмма вызывается в нескольких случаях для чтения клавиши BREAK. Признак переноса возвращается только сброшенным, если нажаты обе клавиши SHIFT и BREAK.

```
1F54 BREAK-KEY  LD    A,+7F               Формируется адрес порта
                IN    A,(+FE)             +7FFE и считывается байт.
                RRA                       Проверяется нулевой разряд
```

```
                                        с помощью сдвига его на
                                        позицию перекоса.
                RET     C               Возврат, если клавиша
                                        BREAK не была нажата.
                LD      A,+FE           Формируется адрес порта
                IN      A,(+FE)         +FEFE и считывается байт.
                RRA                     Опять проверяется 0 разряд.
                RET                     Возврат со сброшенным
                                        признаком переноса.
```

КОМАНДНАЯ ПРОЦЕДУРА 'DEF FN' ('Определить функцию')
Во время проверки синтаксиса проверяется оператор DEF FN, чтобы убедиться, что он имеет правильное значение. Создается место для результата вычисления функции. Но оператор DEF FN передается в 'исполняющую систему'.

```
1F60 DEF-FN     CALL    2530,SYNTAX-Z       Если проверка синтак-
                JR      Z,1F6A,DEF-FN-1     сиса, переход вперед.
                LD      A,+CE               В противном случае
                JP      1E39,PASS-BY        передача оператора 'DEF FN'.
```

Сначала рассматривается переменная функция.

```
1F6A DEF-FN-1   SET     6,(FLAGS)           Сигнал - 'числовая
                                            переменная'.
                CALL    2C8D,ALPHA          Проверка, что текущий
                                            код является буквой.
                JR      NC,1F89,DEF-FN-4    Если нет, переход вперед.
                RST     0020,NEXT-CHAR      Выбор следующего символа.
                CP      +24                 Переход вперед, если
                JR      NZ,1F7D,DEF-FN-2    не '$'.
                RES     6,(FLAGS)           Т.к. строковая переменная,
                                            изменяется разряд 6.
                RST     0020,NEXT-CHAR      Выбор следующего символа.
1F7D DEF-FN-2   CP      +28                 За именем переменной
                JR      NZ,1FBD,DEF-FN-7    должно следовать '('.
                RST     0020,NEXT-CHAR      Выбор следующего символа.
                CP      +29                 Переход вперед, если ')',
                JR      Z,1FA6,DEF-FN-6     т.к. нет параметров функции.
```

Теперь вводится цикл для обработки каждого параметра по очереди.

```
1F86 DEF-FN-3   CALL    2C8D,ALPHA          Текущий код должен
1F89 DEF-FN-4   JP      NC,1C8A,REPORT-C    быть буквой.
                EX      DE,HL               Запись указателя в DE.
                RST     0020,NEXT-CHAR      Выбор следующего символа.
                CP      +24                 Переход вперед, если
                JR      NZ,1F94,DEF-FN-5    это не '$'.
                EX      DE,HL               В противном случае записать
                                            в DE новый указатель.
                RST     0020,NEXT-CHAR      Выбор следующего символа.
1F94 DEF-FN-5   EX      DE,HL               Переместить указатель
                                            последнего символа имени
                                            в регистровую пару HL.
                LD      BC,+0006            Теперь создать 6 ячеек
                CALL    1655,MAKE-ROOM      после последнего
                INC     HL                  символа и ввести
                INC     HL                  'числовой маркер' в
```

```
                LD    (HL),+0E          первую из новых ячеек.
                CP    +2C               Если текущий символ ',', то
                JR    NZ,1FA6,DEF-FN-6  переход назад, т.к. должен
                RST   0020,NEXT-CHAR    быть следующий параметр;
                JR    1F86,DEF-FN-3     иначе, выход из цикла.
```

Далее рассматривается описание функции.

```
1FA6 DEF-FN-6   CP    +29               Проверить, что ')'
                JR    NZ,1FBD,DEF-FN-7  существует.
                RST   0020,NEXT-CHAR    Выбирается следующий
                                        символ.
                CP    +3D               Это должен быть '='.
                JR    NZ,1FBD,DEF-FN-7
                RST   0020,NEXT-CHAR    Выбор следующего символа.
                LD    A,(FLAGS)         Запись содержания -
                PUSH  AF                числового или строкового -
                                        переменной.
                CALL  2F4B,SCANNING     Теперь рассмотрим описание,
                                        как выражение.
                POP   AF                Выбор содержания переменной
                XOR   (FLAGS)           и проверка, что она того же
                AND   +40               типа что и описание.
1FBD DEF-FN-7   JP    NZ,1C8A,REPORT-C  Если требуется, выдать
                                        сообщение об ошибке.
                CALL  1BEE,CHECK-END    Выход через подпрограмму
                                        CHECK-END. (В связи с этим
                                        перемещение для рассмотрения
                                        следующего оператора
                                        в строке).
```

ПОДПРОГРАММА 'UNSTACK-Z'
Эта подпрограмма вызывается в нескольких случаях для того, чтобы сделать 'ранний возврат' из подпрограммы при проверке синтаксиса. Причиной этого является необходимость избежать фактической печати символов или передачу символов в/из стека калькулятора.

```
1FC3 UNSTACK-Z  CALL  2530,SYNTAX-Z     Синтаксис проверяется.
                POP   HL                Выбор адреса возврата, но
                RET   Z                 игнорирование его во 'время
                JP    (HL)              проверки синтаксиса'.
                                        Осуществить простой возврат
                                        к вызывающей программе
                                        в 'исполняющую систему'.
```

КОМАНДНЫЕ ПРОЦЕДУРЫ 'LPRINT и PRINT'
Открывается, как это необходимо, соответствующий канал и рассматриваются элементы, подлежащие печати.

```
1FC9 LPRINT     LD    A,+03             Подготовить к открытию
                                        канал 'P'.
                JR    1FCF,PRINT-1      Переход вперед.
1FCD PRINT      LD    A,+02             Подготовить к открытию
                                        канал 'S'.
1FCF PRINT-1    CALL  2530,SYNTAX-Z     Если не проверяется
                CALL  NZ,1601,CHAN-OPEN синтаксис, открыть канал.
                CALL  0D4D,TEMPS        Установить временную
```

```
                                        переменную цвета.
                CALL  1FDF,PRINT-2      Вызов управляющей
                                        подпрограммы печати.
                CALL  1BEE,CHECK-END    Переместиться, чтобы
                RET                     рассмотреть следующий
                                        оператор; через CHECK-END,
                                        если проверка синтаксиса.
```

Управляющая подпрограмма печати вызывается командными процедурами PRINT, LPRINT и INPUT.

```
1FDF PRINT-2    RST   0018,GET-CHAR     Получить первый символ.
                CALL  2045,PR-END-Z     Переход вперед, если уже
                JR    Z,1FF2,PRINT-4    в конце списка элементов.
```

Теперь ввести цикл, чтобы обработать 'контроллеры позиций' и напечатать элементы.

```
1FE5 PRINT-3    CALL  204E,PR-POSN-1    Работа с любыми последова-
                JR    Z,1FE5,PRINT-3    тельными контроллерами
                                        позиций.
                CALL  1FFC,PR-ITEM-1    Работа с одним элементом
                                        печати.
                CALL  204E,PR-POSN-1    Проверки следующих парамет-
                JR    Z,1FE5,PRINT-3    ров позиций и печать элемен-
                                        тов, пока ни один не
                                        останется.
1FF2 PRINT-4    CP    +29               Теперь возврат, если текущий
                RET   Z                 символ ')'; иначе, рассмот-
                                        реть выполнение 'возврата
                                        каретки'.
```

ПОДПРОГРАММА 'PRINT A CARRIAGE RETURN' ('Печать возврата каретки')

```
1FF5 PRINT-CR   CALL  1FC3,UNSTACK-Z    Если проверяется синтаксис,
                                        возврат.
                LD    A,+0D             Печать символа воз-
                RST   0010,PRINT-A-1    врата каретки, а затем
                RET                     возврат.
```

ПОДПРОГРАММА 'PRINT ITEMS' ('Печать элементов')
Эта подпрограмма вызывается из командных процедур PRINT, LPRINT и INPUT.
Идентифицируются и печатаются различные типы элементов печати.

```
1FFC PR-ITEM-1  RST   0018,GET-CHAR     Выбирается первый символ.
                CP    +AC               Переход вперед, если
                JR    NZ,200E,PR-ITEM-2 это не 'AT'.
```

Теперь работа с 'AT'.

```
                CALL  1C79,NEXT-2NUM    Два параметра передаются
                                        на стек калькулятора.
                CALL  1FC3,UNSTACK-Z    Теперь возврат, если
                                        проверяется синтаксис.
                CALL  2307,STK-TO-BC    Параметры помещаются
                                        в регистровую пару BC.
                LD    A,+16             Управляющий символ AT перед
                                        выполнением перехода загру-
```

```
                JR    201E,PR-AT-TAB      жается в регистр A.
```

Далее ищется 'TAB'.

```
200E PR-ITEM-2  CP    +AD                 Переход вперёд, если это
                JR    NZ,2024,PR-ITEM-3   не 'TAB'.
```

Теперь работа с 'TAB'.

```
                RST   0020,NEXT-CHAR      Получить следующий символ.
                CALL  1C82,EXPT-1NUM      Передать один параметр на
                                          стек калькулятора.
                CALL  1FC3,UNSTACK-Z      Теперь, если проверка синтаксиса,
                                          то возврат.
                CALL  1E99,FIND-INT2      Значение помещается в регистровую
                                          пару BC.
                LD    A,+17               Управляющий символ TAB загружается
                                          в регистр A
```

Элементы печати 'AT' и 'TAB' печатаются с помощью трёх обращений к PRINT-OUT.

```
201E PR-AT-TAB  RST   0010,PRINT-A-1      Печать управляющего символа.
                LD    A,C                 Следовать за ним с первым
                RST   0010,PRINT-A-1      символом.
                LD    A,B                 В конце напечатать второе
                RST   0010,PRINT-A-1      значение и возврат.
                RET
```

Далее рассматриваются встроенные элементы цвета.

```
2024 PR-ITEM-3  CALL  21F2,CO-TEMP-3      Возврат со сброшенным флагом
                                          переноса, если обнаружены
                                          элементы цвета.
                RET   NC                  Продолжение, если ничего
                                          не обнаружено.
                CALL  2070,STR-ALTER      Далее рассматривается, должен
                                          ли быть обнаружен поток..
                RET   NC                  Продолжать, если он не был
                                          изменён.
```

Элемент печати теперь должен быть или строковым, или числовым выражением.

```
                CALL  24FB,SCANNING       Вычисляется выражение, но, если
                CALL  1FC3,UNSTACK-Z      проверяется синтаксис, возврат.
                BIT   6,(FLAGS)           Проверка содержания выражения.
                CALL  Z,2BF1,STK-FETCH    Если это строка, то выбираются
                                          необходимые параметры, но если
                JP    NZ,2DE3,PRINT-FP    она числовая, выход через PRINT-FP.
```

Теперь запускается цикл для поочерёдной обработки каждого символа в строке.

```
203C PR-STRING  LD    A,B                 Возврат, если в строке не
```

```
                OR    C                 остаётся символов, иначе
                DEC   BC                уменьшить счётчик.
                RET   Z
                LD    A,(DE)            Выбор кода и увеличение
                INC   DE                указателя.
                RST   0010,PRINT-A-1    Печатается код и выполняется
                JR    203C,PR-STRING    переход, чтобы рассмотреть
                                        любые следующие символы.
```

ПОДПРОГРАММА 'END OF PRINTING' ('Конец печати')
Будет установлен флаг Z, если в дальнейшем нет необходимости в печати.

```
2045 PR-END-Z   CP    +29               Возврат, если символ ')'.
                RET   Z
2048 PR-ST-END  CP    +0D               Возврат, если символ
                RET   Z                 'возврат каретки'.
                CP    +3A               Осуществить конечный тест с ':'
                RET                     перед возвратом.
```

ПОДПРОГРАММА 'PRINT POSITION' ('Позиция печати')
Этой подпрограммой рассматриваются различные управляющие символы позиций.

```
204E PR-POSN-1  RST   0018,GET-CHAR     Прочитать текущий символ.
                CP    +3B               Переход вперёд, если он
                JR    Z,2067,PR-POSN-3  ';'.
                CP    +2C               Кроме того переход вперёд
                JR    NZ,2061,PR-POSN-2 с символом ',', но если проверка
                CALL  2530,SYNTAX-Z     синтаксиса символ
                JR    Z,2067,PR-POSN-3  в действительности печатается.
                LD    A,+06             Загрузка регистра A управляющим
                RST   0010,PRINT-A-1    кодом 'запятая' и её печать,
                JR    2067,PR-POSN-3    затем переход вперёд.
2061 PR-POSN-2  CP    +27               Это ' ' '?
                RET   NZ                Возврат, если нет какого-либо
                                        контроллера позиций.
                CALL  1FF5,PR-CR        Если не проверяется синтаксис,
                                        напечатать 'возврат каретки'.
2067 PR-POSN-3  RST   0020,NEXT-CHAR    Выбор следующего символа.
                CALL  2045,PR-END-Z     Если не в конце оператора печати,
                JR    NZ,206E,PR-POSN-4 переход вперёд, в противном
                POP   BC                случае возврат к вызывающей
206E PR-POSN-4  CP    A                 программе.
                RET                     Будет сброшен флаг Z, если не
                                        достигнут конец оператора
                                        печати.
```

ПРОГРАММА 'ALTER STREAM' ('Изменение потока')
Эта подпрограмма вызывается всякий раз, когда есть необходимость рассмотреть, желает ли пользователь использовать различные потоки.

```
2070 STR-ALTER  CP    +23               Если текущий символ не '#',
```

```
                SCF                     возврат с установленным флагом
                RET   NZ                переноса.
                RST   0020,NEXT-CHAR    Продвижение CH-ADD.
                CALL  1C82,EXPT-1NUM    Передать параметры на стек
                                        калькулятора.
                AND   A                 Сбросить флаг переноса.
                CALL  1FC3,UNSTACK-Z    Если проверка синтаксиса, то
                                        возврат.
                CALL  1E94,FIND-INT1    В регистр A передано значение.
                CP    +10               Выдать сообщение 0, если
                JP    NC,160E,REPORT-O  значение больше, чем #FF.
                CALL  1601,CHAN-OPEN    Использовать канал для
                                        запрашиваемого потока.
                AND   A                 Сброс флага переноса и
                RET                     возврат.
```

КОМАНДНАЯ ПРОЦЕДУРА 'INPUT' ('Ввод')
Эта подпрограмма допускает присвоение переменным значений, введенных с клавиатуры. Также, возможно использование встроенных в оператор INPUT элементов печати и эти элементы печатаются в нижней части экрана.

```
2089 INPUT      CALL  2530,SYNTAX-Z     Переход вперёд, если проверяется
                JR    Z,2096,INPUT-1    синтаксис.
                LD    A,+01             Открыть канал 'K'.
                CALL  1601,CHAN-OPEN
                CALL  0D6E,CLS-LOWER    Очищается нижняя часть
                                        изображения.
2096 INPUT-1    LD    (TV-FLAG),+01     Задать размер нижней части
                                        экрана в одну строку.
                CALL  20C1,IN-ITEM-1    Вызов подпрограммы для обработки
                                        элементов INPUT.
                CALL  1BEE,CHECK-END    Если проверка синтаксиса, то
                                        переместиться на следующий
                                        оператор.
                LD    BC,(S-POSN)       Выбор текущей позиции печати.
                LD    A,(DF-SZ)         Переход вперёд, если текущая
                CP    B                 позиция выше нижней части экрана.
                JR    C,20AD,INPUT-2
                LD    C,+21
                LD    B,A
20AD INPUT-2    LD    (S-POSN),BC       Сброс S-POSN.
                LD    A,+19             Теперь устанавливается счётчик
                SUB   B                 прокруток.
                LD    (SCR-CT),A
                RES   0,(TV-FLAG)       Флаг 'основной экран'.
                CALL  0DD9,CL-SET       Установить системную переменную
                JP    0D6E,CLS-LOWER    и выход через CLS-LOWER.
```

Элементы INPUT и встроенные элементы PRINT обрабатываются по очереди в следующем цикле.

```
20C1 IN-ITEM-1  CALL  204E,PR-POSN-1    Рассматривается первый любой
                JR    Z,20C1,IN-ITEM-1  символ управления позицией.
                CP    +28               Переход вперёд, если текущий
                JR    NZ,20D8,IN-ITEM-2 символ не '('.
```

```
                RST   0020,NEXT-CHAR      Выбор следующего символа.
                CALL  1FDF,PRINT-2        Теперь вызов командной процедуры
                                          PRINT для обработки элементов
                                          в скобках.
                RST   0018,GET-CHAR       Выбор текущего символа.
                CP    +29                 Выдать сообщение C, если
                JP    NZ,1C8A,REPORT-C    символ не ')'.
                RST   0020,NEXT-CHAR      Выбор следующего символа и
                JP    21B2,IN-NEXT-2      переход вперёд, чтобы посмотреть,
                                          есть ли ещё элементы INPUT.
```

Теперь смотрим, использовался ли INPUT LINE.

```
20D8 IN-ITEM-2  CP    +CA                 Переход вперёд, если
                JR    NZ,20ED,IN-ITEM-3   не 'LINE'.
                RST   0020,NEXT-CHAR      Продвижение CH-ADD.
                CALL  1C1F,CLASS-01       Определить для переменной
                                          адрес назначения.
                SET   7,(FLAGX)           Флаг 'использование INPUT LINE'.
                BIT   6,(FLAGS)           Выдать ошибку C, если не
                JP    NZ,1C8A,REPORT-C    используется строковая переменная.
                JR    20FA,IN-PROMPT      Переход вперёд для выдачи
                                          наводящего сообщения.
```

Продолжение обработки простых переменных INPUT.

```
20ED IN-ITEM-3  CALL  2C8D,ALPHA          Переход для просмотра работающего
                JP    NC,21AF-IN-NEXT-1   цикла, если текущий символ
                                          не является буквой.
                CALL  1C1F,CLASS-01       Определяет адрес назначения
                                          переменной.
                RES   7,(FLAGX)           Флаг 'не INPUT LINE'
                                          ('не входная строка')
```

Теперь наводящее сообщение встраивается в рабочую область.

```
20FA IN-PROMPT  CALL  2530,SYNTAX-Z       Переход вперёд только в случае
                JP    Z,21B2,IN-NEXT-2    проверки синтаксиса.
                CALL  16BF,SET-WORK       Рабочая область пустая.
                LD    HL,+5C71            Это FLAGX.
                RES   6,(HL)              Флаг 'string result'
                                          ('строковый результат')
                SET   5,(HL)              Флаг 'INPUT mode'.
                LD    BC,+0001            Разрешает наводящему сообщению
                                          занимать только одну ячейку.
                BIT   7,(HL)              Переход вперёд при использовании
                JR    NZ,211C,IN-PR-2     'LINE'.
                LD    A,(FLAGS)           Переход вперёд,  если ожидается
                AND   +40                 числовой ввод.
                JR    NZ,211A,IN-PR-1
                LD    C,+03               Ввод строки требует трёх
                                          ячеек.
211A IN-PR-1    OR    (HL)                Бит 6 FLAGX будет установлен
```

```
                LD    (HL),A              для числового ввода.
211C IN-PR-2    RST   0030,BC-SPACES      Создаем необходимое количество
                                          ячеек.
                LD    (HL),+0D            Код 'возврат каретки' переходит
                                          в последнюю ячейку.
                LD    A,C                 Проверка бита 6 регистра C
                RRCA                      и переход вперёд, если
                RRCA                      требовалась только одна ячейка.
                JR    NC,2129,IN-PR-3
                LD    A,+22               Символ 'двойные кавычки'
                LD    (DE),A              переходит в первую и вторую
                DEC   HL                  ячейки.
                LD    (HL),A
2129 IN-PR-3    LD    (K-CUR),HL          Может быть записана позиция
                                          курсора.
```

В случае INPUT LINE EDITOR может вызываться без дальнейшей подготовки но для других типов INPUT стек ошибки изменится для прерывания ошибки.

```
                BIT   7,(FLAGX)           Переход вперёд с 'INPUT
                JR    NZ,215E,IN-VAR-3    LINE'.
                LD    HL,(CH-ADD)         Сохранить текущие значения
                PUSH  HL                  CH-ADD и ERR-SP на стеке.
                LD    HL,(ERR-SP)
                PUSH  HL
213A IN-VAR-1   LD    HL,+213A            В случае ошибки это будет
                PUSH  HL                  точкой возврата.
                BIT   4,(FLAGS2)          При использовании канала 'K'
                JR    Z,2148,IN-VAR-2     изменяется только указатель
                LD    (ERR-SP),SP         стека ошибки.
2148 IN-VAR-2   LD    HL,(WORKSP)         Установка HL на начало строки
                CALL  11A7,REMOVE-FP      INPUT и удаление любой формы с
                                          плавающей точкой. Не может быть
                                          никаких исключений.
                LD    (ERR-NR),+FF        Сигнал 'ошибки ещё нет'.
                CALL  0F2C,EDITOR         Получение INPUT и с признаком
                RES   7,(FLAGS)           синтаксис (запуск, показывающий
                CALL  21B9,IN-ASSIGN      синтаксис), проверка INPUT на
                JR    2161,IN-VAR-4       ошибки; если всё в порядке,
                                          переход, если нет - возврат
                                          на IN-VAR-1.
215E IN-VAR-3   CALL  0F2C,EDITOR         Получение 'LINE'.
```

Все системные переменные должны быть сброшены перед фактическим присваиванием значений.

```
2161 IN-VAR-4   LD    (K-CUR-hi),+00      Сброс адреса курсора.
                CALL  21D6,IN-CHAN-K      Осуществляется переход при
                JR    NZ,2174,IN-VAR-5    использовании канала,
                                          отличного от 'K'.
                CALL  111D,ED-COPY        Входная строка выведена на
                LD    BC,(ECHO-E)         дисплей и позиция в ECHO-E
                CALL  0DD9,CL-SET         сделала текущую позицию внизу
                                          экрана.
2174 IN-VAR-5   LD    HL,+5C71            Это FLAGX.
                RES   5,(HL)              Флаг 'режим редактирования'.
                BIT   7,(HL)              Переход вперёд при обработке
```

```
                RES   7,(HL)              INPUT LINE.
                JR    NZ,219B,IN-VAR-6
                POP   HL                  Удаление адреса IN-VAR-1.
                POP   HL                  Возврат ERR-SP на исходный
                LD    (ERR-SP),HL         адрес.
                POP   HL                  Запись исходного адреса CH-ADD
                LD    (X-PTR),HL          в X-PTR.
                SET   7,(FLAGS)           Теперь с флагом 'синтаксис/
                CALL  21B9,IN-ASSIGN      запуск', показывающим запуск,
                                          осуществляется присваивание.
                LD    HL,(X-PTR)          Восстановление исходного адреса
                LD    (X-PTR-hi),+00      в CH-ADD и очистка X-PTR.
                LD    (CH-ADD),HL
                JR    21B2,IN-NEXT-2      Если имеются ещё элементы
                                          INPUT, то переход вперёд.
219B IN-VAR-6   LD    HL,(STKBOT)         Найдена длина 'LINE' в рабочей
                LD    DE,(WORKSP)         области.
                SCF
                SBC,  HL,DE
                LD    B,H                 DE указывает на начало, а
                LD    C,L                 BC содержит длину.
                CALL  2AB2,STK-ST-$       Эти параметры пакуются и
                CALL  2AFF,LET            осуществляется фактическое
                                          присваивание.
                JR    21B2,IN-NEXT-2      Переход вперёд для просмотра
                                          следующих элементов.
```

Рассматриваются следующие элементы INPUT.

```
21AF IN-NEXT-1  CALL  1FFC,PR-ITEM-1      Обработка элементов печати.
21B2 IN-NEXT-2  CALL  204E,PR-POSN-1      Обработка контроллеров позиций.
                JP    Z,20C1,IN-ITEM-1    Обход цикла ещё раз в случае
                                          наличия следующих элементов;
                RET                       иначе возврат.
```

ПОДПРОГРАММА 'IN-ASSIGN'
Данная подпрограмма вызывается дважды для каждого значения INPUT. Один раз при признаке синтаксис/запуск, в случае 'синтаксис' сбрасывается и один раз в случае 'запуск' устанавливается.

```
21B9 IN-ASSIGN  LD    HL,(WORKSP)         Установка CH-ADD для указания
                LD    (CH-ADD),HL         первой ячейки рабочей области и
                RST   0018,GET-CHAR       выбор символа.
                CP    +E2                 Это 'STOP'?
                JR    Z,21D0,IN-STOP      Переход, если так.
                LD    A,(FLAGX)           В противном случае сделайте
                CALL  1C59,VAL-FET-2      присваивание 'значения'
                                          переменной.
                RST   0018,GET-CHAR       Получение текущего символа
                CP    +0D                 и проверка является ли он
                RET   Z                   'возвратом каретки'. Возврат,
                                          если это так.
```

Сообщение C - 'Nonsense in BASIC'.

```
21CE REPORT-C     RST   0008,ERROR-1        Вызов подпрограммы обработки
                  DEFB  +0B                 ошибок.
```

Идите сюда, если строка INPUT начинается с оператора 'STOP'.

```
21D0 IN-STOP      CALL  2530,SYNTAX-Z       Но не давайте сообщение об
                  RET   Z                   ошибке при просмотре синтаксиса.
```

Сообщение H - 'STOP в INPUT'

```
21D4 REPORT-H     RST   0008,ERROR-1        Вызов подпрограммы обработки
                  DEFB  +10                 ошибок.
```

ПОДПРОГРАММА 'IN-CHAN-K'
Эта подпрограмма возвращается со сброшенным флагом Z при использовании канала 'K'.

```
21D6 IN-CHAN-K    LD    HL,(CURCHL)         Выбирается базовый адрес
                  INC   HL                  информации 0 канала для текущего
                  INC   HL                  канала, а код канала сравнивается
                  INC   HL                  с символом 'K'.
                  INC   HL
                  LD    A,(HL)
                  CP    +4B
                  RET                       Возврат.
```

ПРОГРАММА 'COLOUR ITEM' ('Элементы цвета')
Этот набор программ можно разделить на две части:
1. Драйвер 'встроенных цветовых элементов';
2. Драйвер 'цветовых системных переменных'.

1. Встроенные цветовые элементы обрабатываются с помощью вызова подпрограммы PRINT-OUT. Обработка элементов осуществляется поочерёдно в цикле. Точкой входа является CO-TEMP-2.

```
21E1 CO-TEMP-1    RST   0020,NEXT-CHAR      Изучение следующего символа
                                            в операторе BASIC.
21E2 CO-TEMP-2    CALL  21F2,CO-TEMP-3      Переход вперёд, чтобы посмотреть,
                                            представляет ли текущий код
                                            встроенный 'рабочий' цветовой
                                            элемент.
                  RET   C                   Возврат с установленным флагом C,
                                            если цветовой элемент отсутствует.
                  RST   0018,GET-CHAR       Выбор текущего символа.
                  CP    +2C                 Переход назад, если наличествует
                  JR    Z,21E1,CO-TEMP-1    ',' или ';', иначе - ошибка.
                  CP    +3B
                  JR    Z,21E1,CO-TEMP-1
                  JP    1C8A,REPORT-C       Выход через 'сообщение C'.
21F2 CO-TEMP-3    CP    +D9                 Возврат с установленным флагом C.
                  RET   C
                  CP    +DF                 Задать, если код не лежит в
                                            диапазоне от +D9 до +DE (INK-
                                            OVER).
```

```
                CCF
                RET   C
                PUSH  AF                Код цветового элемента
                RST   0020,NEXT-CHAR    сохраняется, пока перемещается
                POP   AF                CH-ADD для адресации параметра,
                                        который следует за ним.
```

Код цветового элемента и параметр 'печатаются' с помощью PRINT-OUT.

```
21FC CO-TEMP-4  SUB   +C9               Диапазон токенов (+D9 - +DE)
                                        уменьшен до диапазона символов
                                        управления (+10 - +15).
                PUSH  AF                Код символов управления
                CALL  1C82,EXPT-1NUM    сохраняется, пока параметр
                POP   AF                пересылается на стек калькулятора.
                AND   A                 В случае проверки синтаксиса
                CALL  1FC3,UNSTACK-Z    возврат на эту точку.
                PUSH  AF                Код символов управления
                CALL  1E94,FIND-INT1    сохраняется, пока параметр
                LD    D,A               пересылается в регистр D.
                POP   AF
                RST   0010,PRINT-A-1    Послан символ управления.
                LD    A,D               Затем выбираемый параметр
                RST   0010,PRINT-A-1    печатается.
                RET
```

2. Цветовые системные переменные ATTR-T, MASK-T & P-FLAG изменяются в соответствии с требованиями. Эта подпрограмма вызывается с помощью PRINT-OUT. На входе код символов управления находится в регистре A, а параметр - в регистре D. Отметим, что все изменения относятся к 'рабочим' системным переменным.

```
2211 CO-TEMP-5  SUB   +11               Уменьшения диапазона и переход
                ADC   A,+00             вперёд с INK и PAPER.
                JR    Z,2234,CO-TEMP-7
                SUB   +02               Уменьшение диапазона и переход
                ADC   A,+00             с FLASH и BRIGHT.
                JR    Z,2273,CO-TEMP-C
```

Код управления цветом теперь будет +01 для INVERSE и +02 для OVER, а системная переменная P-FLAG соответственно изменится.

```
                CP    +01               Подготовка к переходу с OVER.
                LD    A,D               Выбор параметра.
                LD    B,+01             Подготовка маски для OVER.
                JR    NZ,2228,CO-TEMP-6 Теперь переход.
                RLCA                    Бит 2 регистра A сбрасывается
                RLCA                    для INVERSE 0 и устанавливается
                LD    B,+04             для INVERSE 1; для маски
                                        устанавливается 2 бит регистра B.
2228 CO-TEMP-6  LD    C,A               На время проверки диапазона
                                        сохраним регистр A.
                LD    A,D               Для INVERSE и OVER правильным
                CP    +02               диапазоном будет '0-1'.
```

```
                JR    NC,2244,REPORT-K
                LD    A,C               Восстановим регистр A.
                LD    HL,+5C91          Это P-FLAG, который должен
                                        быть изменён.
                JR    226C,CO-CHANGE    Выход через CO-CHANGE и
                                        изменение P-FLAG с использованием
                                        регистра B как маски. T.e. 0 бит
                                        для OVER и 2 бит - для INVERSE.
```

PAPER и INK обрабатываются следующей программой. На входе флаг переноса установлен для INK.

```
2234 CO-TEMP-7  LD    A,D               Выбор параметра.
                LD    B,+07             Подготовка маски для INK.
                JR    C,223E,CO-TEMP-8  Переход вперёд, если INK.
                RLCA                    Умножение параметра для PAPER
                RLCA                    на 8.
                RLCA
                LD    B,+38             Подготовка маски для PAPER.
223E CO-TEMP-8  LD    C,A               Запись параметра в регистр C,
                                        пока проверяется диапазон
                                        параметра.
                LD    A,D               Восстановление исходного значения.
                CP    +0A               Разрешения для PAPER/INK
                JR    C,2246,CO-TEMP-9  диапазона от '0' до '9'.
```

Сообщение K - 'Неправильный цвет'.

```
2244 REPORT-K   RST   0008,ERROR-1      Вызов подпрограммы обработки
                DEFB  +13               ошибок.
```

Продолжение обработки PAPER и INK.

```
2246 CO-TEMP-9  LD    HL,+5C8F          Подготовка к изменению ATTR-T,
                                        MASK-T и P-FLAG.
                CP    +08               Переход вперёд с PAPER/INK
                JR    C,2258,CO-TEMP-B  от '0' до '7'.
                LD    A,(HL)            Выбор текущего значения
                JR    Z,2257,CO-TEMP-A  ATTR-T и при переходе вперёд
                                        используйте его неизменным с
                                        PAPER/INK='8'.
                OR    B                 Но для PAPER/INK='9' цвета
                CPL                     PAPER и INK должны быть чёрным
                AND   +24               и белым.
                JR    Z,2257,CO-TEMP-A  Переход для чёрного INK/PAPER;
                LD    A,B               но продолжение для белого
                                        INK/PAPER.
2257 CO-TEMP-A  LD    C,A               Пересылка значения в регистр C.
```

Теперь маска (B) и значение (C) используются для измерения ATTR-T.

```
2258 CO-TEMP-B  LD    A,C               Пересылка значения.
                CALL  226C,CO-CHANGE    Изменение ATTR-T в соответствии
                                        с требованиями.
```

Далее рассматривается MASK-T.

```
                LD    A,+07             Биты MASK-T устанавливаются
                CP    D                 только при использовании PAPER/INK
                SBC   A,A               '8' или '9'.
                CALL  226C,CO-CHANGE    Изменение MASK-T.
```

Далее рассматривается P-FLAG.

```
                RLCA                    Соответствующая маска задана
                RLCA                    в регистре B для изменения битов
                AND   +50               4 и 6, как это необходимо.
                LD    B,A
                LD    A,+08             Биты P-FLAG устанавливаются
                CP    D                 только для PAPER/INK=9
                SBC   A,A
                                        Продолжение в CO-CHANGE для
                                        обработки P-FLAG.
```

ПОДПРОГРАММА 'CO-CHANGE'
Эта подпрограмма используется, чтобы системная переменная отражала 'характер' битов в регистре A. Регистр B содержит маску, которая показывает, какие биты должны быть скопированы из A в (HL).

```
226C CO-CHANGE  XOR   (HL)              Биты, заданные маской в регистре B,
                AND   B                 изменяются по величине, а
                XOR   (HL)              результат представляется в форме
                LD    (HL),A            системной переменной.
                INC   HL                Переход дальше на адрес следующей
                                        системной переменной.
                LD    A,B               Возврат с маской на регистр A.
                RET
```

FLASH и BRIGHT обрабатываются следующей программой.

```
2273 CO-TEMP-C  SBC   A,A               Для BRIGHT задаётся флаг Z.
                LD    A,D               Выбор и сдвиг параметра.
                RRCA
                LD    B,+80             Подготовка маски для FLASH.
                JR    NZ,227D,CO-TEMP-D Переход вперёд с FLASH.
                RRCA                    Дополнительный сдвиг и подготовка
                LD    B,+40             маски для BRIGHT.
227D CO-TEMP-D  LD    C,A               Сохранение значения в регистре C.
                LD    A,D               Выбор параметра и проверка его
                CP    +08               диапазона. Допускаются значения
                                        '0', '1' и '8'.
                JR    Z,2287,CO-TEMP-E
                CP    +02
                JR    NC,2244,REPORT-K
```

Теперь системная переменная ATTR-T может быть изменена.

```
2287 CO-TEMP-E  LD    A,C               Получает значение.
                LD    HL,+5C8F          Это ATTR-T.
                CALL  226C,CO-CHANGE    Изменение системной переменной.
```

Теперь рассматриваются значения в MASK-T.

```
                LD    A,C               Заново выбирается
                RRCA                    значение. Разряд набора
                RRCA                    FLASH/BRIGHT '8' (третий
                RRCA                    разряд) смещен на 7-й
                                        (для FLASH) или 6-й (для
                                        BRIGHT) разряд.
                JR    226C,CO-CHANGE    Выход через CO-CHANGE.
```

Программная процедура 'BORDER' (Бордюр)
Параметр команды BORDER используется с командой OUT для фактического изменения цвета окантовки. Параметр затем записывается в системную переменную BORDCR.

```
2294 BORDER     CALL  1E94,FIND-INT1    Выбирается параметр
                CP    +08               и проверяется его
                JR    NC,2244,REPORT-K  диапазон.
                OUT   (+FE),A           Затем используется команда
                                        OUT для задания цвета
                                        окантовки.
                RLCA                    Далее параметр умно-
                RLCA                    жается на 8.
                RLCA
                BIT   5,A               Если цвет окантовки является
                JR    NZ,22A6,BORDER-1  'светлым' цветом, а цвет
                                        INK в редактируемой области
                                        черный - сделайте переход.
                XOR   +07               Изменение цвета INK.
22A6 BORDER-1   LD    (BORDCR),A        Задание требуемой
                RET                     переменной и возврат.
```

ПОДПРОГРАММА 'PIXEL ADDRESS' ('Адрес пиксела')
Эта подпрограмма вызывается подпрограммой POINT и командной процедурой PLOT. Она вводится с координатами пиксела в регистровую пару BC и возвращает HL, содержащий адрес байта дисплейного файла, который содержит этот пиксел и A, указывающий на позицию пиксела внутри байта.

```
22AA PIXEL-ADD  LD    A,+AF             Проверяется, что
                SUB   B                 координата y (в B)
                JP    C,24F9,REPORT-B   не больше, чем 175.
                LD    B,A               B содержит 175 минус y.
                AND   A                 A содержит b7b6b5b4b3b2b1b0
                RRA                     разряды B. A теперь
                                        0b7b6b5b4b3b2b1.
                SCF
                RRA                     Теперь 10b7b6b5b4b3b2.
                AND   A
                RRA                     Теперь 010b7b6b5b4b3.
                XOR   B
                AND   +F8               Окончательно 010b7b6b2b1b0,
                XOR   B                 итак, H стало
                LD    H,A               64 + 8*INT (B/64) + B (mod 8),
                LD    A,C               старший байт пиксела.
                                        C содержит X.
                RLCA                    A начинается как
                RLCA                    c7c6c5c4c3c2c1c0.
                RLCA                    A теперь c2c1c0c7c6c5c4c3.
```

```
                XOR   B
                AND   +C7
                XOR   B                 Теперь c2c1b5b4b3c5c4c3.
                RLCA
                RLCA                    Наконец b5b4b3c7c6c5c4c3,
                LD    L,A               так что L становится
                LD    A,C               32*INT (B(mod 64)/8)+INT(x/8),
                AND   +07               младший байт. A содержит
                RET                     x(mod 8): итак, пиксел
                                        это разряд (A-7) внутри
                                        байта.
```

ПОДПРОГРАММА 'POINT' ('Точка')
Эта подпрограмма вызывается функцией POINT в SCANNING. Она вводится с координатами пиксела на стек вычислителя и возвращает последнее значение как 1, если пиксел является цветом краски и 0, если он является цветом бумаги (поверхности).

```
22CB POINT-SUB  CALL  2307,STK-TO-BC    Координата Y в B, X в C.
                CALL  22AA,PIXEL-ADD    B будет просчи-
                LD    B,A               тывать циклы A+1,
                INC   B                 чтобы получить нужный
                LD    A,(HL)            разряд (HL) в нужную ячейку.
22D4 POINT-LP   RLCA                    Сдвиг.
                DJNZ  22D4,POINT-LP
                AND   +01               Разряд равен 1 для
                                        краски, 0-для бумаги.
                JP    2D28,STACK-A      Занесение в стек
                                        калькулятора.
```

КОМАНДНАЯ ПРОЦЕДУРА 'PLOT' ('Чертить')
Эта программа содержит основную подпрограмму плюс одну строку для ее вызова и одну строку для выхода из нее. Основная программа используется дважды CIRCLE, а подпрограмма вызывается с помощью DRAW. Программа вводится с координатами пиксела на стек калькулятора. Она находит адрес этого пиксела и чертит его, учитывая состояние INVERSE и OVER в P-FLAG.

```
22DC PLOT       CALL  2307,STK-TO-BC    Координата Y в B,X,C.
                CALL  22E5,PLOT-SUB     Вызов подпрограммы.
                JP    0D4D,TEMPS        Выход, задание рабочих
                                        цветов.
22E5 PLOT-SUB   LD    (COORDS),BC       Задана системная переменная.
                CALL  22AA,PIXEL-ADD    Адрес пиксела в HL.
                LD    B,A               B будет просчитывать циклы
                INC   B                 A+1, чтобы получить 0 в
                                        нужном месте в A.
                LD    A,+FE             Введен ноль.
22F0 PLOT-LOOP  RRCA                    Затем выстраивается позиция
                DJNZ  22F0,PLOT-LOOP    позиция разрядов пикселов
                                        в байте.
                LD    B,A               Копирование в B.
                LD    A,(HL)            Байт-пиксел получен в A.
                LD    C,(P-FLAG)        Получен P-FLAG и про-
                BIT   0,C               верен сигнал для OVER.
                JR    NZ,22FD,PL-TST-IN Переход, если OVER 1.
                AND   B                 OVER 0 сначала создает 0
```

```
                                          пиксела.
22FD PL-TST-IN    BIT   2,C               Проверка для INVERSE.
                  JR    NZ,2303,PLOT-END  INVERSE 1 оставляет пиксел
                                          также, как и при OVER 1 или
                                          ноль (OVER 0).
                  XOR   B                 INVERSE 0 оставляет сформи-
                  CPL                     рованный пиксел (OVER 1)
                                          или (OVER 0).
2303 PLOT-END     LD    (HL),A            Введен байт, его другие
                                          разряды не меняются во
                                          всех случаях.
                  JP    0BDB,PO-ATTR      Выход, задание байта
                                          атрибутов.
```

ПОДПРОГРАММА 'STK-TO-BC'
Эта подпрограмма загружает два числа с плавающей точкой в регистровую пару BC. Таким образом, она собирает параметры в диапазоне +00 - +FF. Также, она в DE получает значения 'диагонального смещения' (+/-1,+/-1), которые используются в подпрограмме DRAW.

```
2307 STK-TO-BC    CALL  2314,STK-TO-A     Первое число в A.
                  LD    B,A               Отсюда в B.
                  PUSH  BC                Краткая запись.
                  CALL  2314,STK-TO-A     Второе число в A.
                  LD    E,C               Его знак в E.
                  POP   BC                Восстановление первого числа.
                  LD    D,C               Его знак в D.
                  LD    C,A               Второе число в C.
                  RET                     Теперь BC и DE являются
                                          тем, что требовалось.
```

ПОДПРОГРАММА STK-TO-A
Эта подпрограмма загружает регистр A числом с плавающей точкой, находящимся на вершине стека калькулятора. Число должно быть в диапазоне 00-FF.

```
2314 STK-TO-A     CALL  2DD5,FP-TO-A      Модуль последнего округлен-
                  JP    C,24F9,REPORT-B   ного значения, если
                                          возможно, заносится в A;
                                          иначе сообщение об ошибке.
                  LD    C,+01             Единица в C для последнего
                                          положительного значения.
                  RET   Z                 Возврат, если значение
                                          положительно.
                  LD    C,+FF             Иначе, изменить C на +FF
                  RET                     (т.е. минус 1). Окончание.
```

ПРОГРАММНАЯ ПРОЦЕДУРА 'CIRCLE' ('Круг')
Эта программа чертит круг с координатами центра X и Y и радиусом Z. Перед использованием эти числа округляются до ближайшего целого. Таким образом, Z должно быть меньше, чем 87.5, даже если (X,Y) указывают на центр экрана. Этот метод использует изображение серии дуг, полученных из прямых линий. Это проиллюстрировано в BASIC-программе в приложении. Описание программы приведено ниже.

CIRCLE состоит из четырех частей:

1. Проверка радиуса. Если его модуль меньше 1, то изображается только точка X,Y;
2. Вызывается CD-PRMS-1 на 2470-24B6, которая используется для инициализации параметров для CIRCLE и DRAW;
3. Задание оставшихся параметров для CIRCLE, включая начальное смещение для первой 'дуги' (в действительности прямой линии);
4. Переход в DRAW для использования цикла изображения дуг на 2420-24FA.

Пояснения к частям 1-3.

1. 2320-23AA. Радиус Z получен из стека калькулятора. Если модуль Z меньше 1, он удаляется со стека, а точка X,Y изображается с помощью перехода на PLOT.

```
2320 CIRCLE       RST   0017,GET-CHAR       Получение текущего
                                            символа.
                  CP    +2C                 Проверка на запятую.
                  JP    NZ,1C8A,REPORT-C    Если это не так, то
                                            сообщение об ошибке.
                  RST   0020,NEXT-CHAR      Получение следующего
                                            символа (радиус).
                  CALL  1C82,EXPT-1NUM      Радиус в стек калькулятора.
                  CALL  1BEE,CHECK-END      Пересылка для рассмотрения
                                            следующего оператора, если
                                            проверяется синтаксис.
                  RST   0028,FP-CALC        Использование калькулятора:
                                            стек содержит:
                  DEFB  +2A,abs             X, Y, Z.
                  DEFB  +3D,re-stack        Z перезанесено в стек;
                  DEFB  +38,end-calc        его порядок теперь
                                            допустим.
                  LD    A,(HL)              Получение порядка радиуса.
                  CP    +81                 Проверка, меньше ли радиус 1.
                  JR    NC,233B,C-R-GRE-1   Если нет, переход.
                  RST   0028,FP-CALC        Если меньше, удаление из стека.
                  DEFB  +02,delete          Стек содержит X, Y.
                  DEFB  +38,end-calc
                  JR    22DC,PLOT           Изображение только точки X, Y.
```

2. 233B-2346 и вызов CD-PRMS1. 2*PI занесено в mem-5 и вызов CD-PRMS1. Эта подпрограмма запоминает в регистр B число дуг, требуемых для круга, а именно, A=4*INT (PI*SQR Z/4)+4, отсюда 4, 8, 12 ..., до максимального значения 32. Она также заносит в mem-0 до mem-4 величины 2*PI/A, SIN(PI/A), 0, COS (2*PI/A) и SIN (2*PI/A).

```
233B C-R-GRE-1    RST   0028,FP-CALC
                  DEFB  +A3,stk-pi/2        X, Y, Z, PI/2.
                  DEFB  +38,end-calc        Теперь увеличивается
                                            порядок до 83 шестнад-
                  LD    (HL),+83            цатеричное, изменяя PI/2
                                            на 2*PI.
                  RST   0028,FP-CALC        X, Y, Z, 2*PI.
                  DEFB  +C5,st-mem-5        (2*PI скопировано в mem-5).
                  DEFB  +02,delete          X, Y, Z
                  DEFB  +38,end-calc
                  CALL  247D,CD-PRMS1       Задание начальных параметров.
```

3. 2347-2381: остающиеся параметры и переход на DRAW. Сделана проверка, меньше ли начальная длина 'дуги', чем 1. Если это так, то переход на изображение только точки X,Y. В противном случае, задаются параметры: X+Z и X-Z*SIN (PI/A) заносятся в стек дважды как начальная и конечная точки, и копируются в COORDS; ноль и 2*Z*SIN (PI/A) хранятся в mem-1 и mem-2 как начальные параметры, задающие в качестве первой 'дуги' вертикальную прямую линию, соединяющую X+Z, Y-Z*SIN (PI/A) и X+Z, Y+Z*SIN (PI/A). Цикл изображения дуг в DRAW обеспечит, что все остальные точки останутся в круге, как и эти две, с угловым приращением 2*PI/A. Но ясно, что в действительности эти две точки стягивают этот угол в точку X+Z*(1-COS (PI/A)), Y, а не X, Y. Отсюда, конечные точки каждой дуги круга размещаются справа на величину 2*(1-COS (PI/A)), которая меньше, чем половина пиксела, и округляется до одного пиксела.

```
2347              PUSH  BC                  Запись счета дуг в B.
                  RST   0028,FP-CALC        X,Y,Z
                  DEFB  +31,duplicate       X,Y,Z,Z
                  DEFB  +E1,get-mem-1       X,Y,Z,Z,SIN (PI/A)
                  DEFB  +04,multiply        X,Y,Z,Z*SIN (PI/A)
                  DEFB  +38,end-calc        Z*SIN (PI/A) является половиной
                  LD    A,(HL)              начальной длины дуги, проверяется
                  CP    +80                 не меньше ли она значения 0.5.
                  JR    NC,235A,C-ARC-GE1   Если нет, делается переход.
                  RST   0028,FP-CALC        В противном случае, Z
                  DEFB  +02,delete          удаляется из стека,
                  DEFB  +02,delete          с половиной дуги;
                  DEFB  +38,end-calc        машинный стек очищается;
                  POP   BC                  осуществляется переход
                  JP    22DC,PLOT           для изображения точки X, Y.
235A C-ARC-GE1    RST   0028,FP-CALC        X,Y,Z,Z*SIN (PI/A).
                  DEFB  +C2,st-mem-2        (Z*SIN (PI/A) помещается в
                                            mem-2 в данный момент).
                  DEFB  +01,exchange        X,Y,Z*SIN (PI/A),Z
                  DEFB  +C0,st-mem-0        X,Y,Z*SIN (PI/A),Z
                  DEFB  +02,delete          X,Y,Z*SIN (PI/A)
                  DEFB  +03,subtract        X, Y - Z*SIN (PI/A)
                  DEFB  +01,exchange        Y - Z*SIN (PI/A), X
                  DEFB  +E0,get-mem-0       Y - Z*SIN (PI/A), X, Z
                  DEFB  +0F,addition        Y - Z*SIN (PI/A), X+Z
                  DEFB  +CO,st-mem-0        (X+Z копируется в mem-0)
                  DEFB  +01,exchange        X+Z, Y-Z*SIN (PI/A)
                  DEFB  +31,duplicate       X+Z, Y-Z*SIN (PI/A),
                                            Y-Z*SIN (PI/A)
                  DEFB  +E0,get-mem-0       sa,sb,sb,sa
                  DEFB  +01,exchange        sa,sb,sa,sb
                  DEFB  +31,duplicate       sa,sb,sa,sb,sb
                  DEFB  +E0,get-mem-0       sa,sb,sa,sb,sb,sa
                  DEFB  +A0,stk-zero        sa,sb,sa,sb,sb,sa,0
                  DEFB  +C1,st-mem-1        (в mem-1 занесен 0).
                  DEFB  +02,delete          sa,sb,sa,sb,sb,sa
                  DEFB  +38,end-calc
```

(Здесь sa обозначает X+Z, а sb обозначает Y-Z*SIN (PI/A)).

```
                  INC   (mem-2-1st)         Увеличение на 1
                                            байта порядка mem-2 задает
                                            mem-2 как 2*Z*SIN (PI/A).
```

```
                CALL  1E94,FIND-INT1      Последнее значение X+Z
                LD    L,A                 пересылается из стека в A и
                                          скопировано в L.
                PUSH  HL                  Записано в HL.
                CALL  1E94,FIND-INT1      Y+Z*SIN (PI/A) переходит из
                POP   HL                  стека в A и копируется в H.
                LD    H,A                 HL теперь содержит
                                          начальную точку.
                LD    (COORDS),HL         Скопировано в COORDS.
                POP   BC                  Восстановлен счет дуг.
                JP    2420,DRW-STEPS      Сделан переход на DRAW.
```

(Теперь стек содержит X+Z, Y-Z*SIN (PI/A), Y-Z*SIN (PI/A), X+Z).

ПРОГРАММНАЯ ПРОЦЕДУРА 'DRAW' ('Рисовать')
Эта подпрограмма вводится с координатами точки X0, Y0, скажем, в COORDS. Если заданы с командой DRAW только два параметра X, Y, го она чертит прямую линию от точки X0, Y0 в X0+X, Y0+Y. Если задан третий параметр G, то она чертит линию, приближенную к дуге из X0, Y0 в X0+X, Y0+Y, поворачиваясь против часовой стрелки на угол G радиан.

Программа состоит из 4 частей:

1. Если заданы только два параметра или диаметр обозначенного круга меньше 1, то чертится просто линия.
2. Вызов CD-PRMS1 в 247D-24B6 для задания первых параметров.
3. Задание оставшихся параметров, включая начальное местоположение для первой дуги.
4. Ввод цикла изображения дуг и изображение маленьких дуг, аппроксимированных с помощью прямых линий, вызов, если необходимо, подпрограммы изображения линий в 24B7-24FA.

Две программы, CD-PRMS1 и DRAW-LINE, следуют за главной программой. Теперь рассмотрим 4 части главной программы.

1. Если присутствуют только два параметра, осуществляется переход в LINE-DRAW на 2477. Линия также изображается, если величина Z = (ABS X + ABS Y)/ABS SIN (G/2) меньше 1. Z находится в диапазоне между 1 и 1,5 раз от диаметра указанного круга. В этой части в mem-0 задан SIN (G/2), в mem-1 - Y, и в mem-5 - G.

```
2382 DRAW         RST   0018,GET-CHAR       Прочитать текущий символ.
                  CP    +2C                 Если это запятая,
                  JR    Z,238D,DR-3-PRMS    то переход.
                  CALL  1BEE,CHECK-END      Если проверяется синтаксис,
                                            то пересылка на следующий
                                            оператор.
                  JP    2477,LINE-DRAW      Переход для изображения
                                            линии.
238D DR-3-PRMS    RST   0020,NEXT-CHAR      Получение следующего
                                            символа (угол).
                  CALL  1C82,EXPT-1NUM      Угол в стеке калькулятора.
                  CALL  1BEE,CHECK-END      Если проверяется синтаксис,
                                            пересылка на следующий
                                            оператор.
                  RST   0028,FP-CALC        X, Y, G в стеке.
                  DEFB  +C5,st-mem-5        (G скопировано в mem-5).
                  DEFB  +A2,stk-half        X, Y, G, 0.5
                  DEFB  +04,multiply        X, Y, G/2
```

```
                DEFB  +1F,sin             X, Y, SIN (G/2)
                DEFB  +31,duplicate       X, Y, SIN (G/2), SIN (G/2)
                DEFB  +30,not             X, Y, SIN (G/2), (0/1)
                DEFB  +30,not             X, Y, SIN (G/2), (1/0)
                DEFB  +00,jump-true       X, Y, SIN (G/2)
                DEFB  +06,to DR-SIN-NZ    (Если SIN (G/2)=0, т.е.
                DEFB  +02,delete          G = 2*N*PI, изображается
                                          только прямая линия).
                DEFB  +38,end-calc        X, Y
                JP    2477,LINE-DRAW      Линия от X0, Y0 до X0+X, Y0+Y.
23A3 DR-SIN-NZ  DEFB  +C0,st-mem-0        (SIN (G/2) скопировано в mem-0)
                DEFB  +02,delete          X, Y в стеке.
                DEFB  +C1,st-mem-1        (Y скопировано в mem-1).
                DEFB  +02,delete          X
                DEFB  +31,duplicate       X, X
                DEFB  +2A,abs             X, X' (X' = ABS X)
                DEFB  +E1,get-mem-1       X, X', Y
                DEFB  +01,exchange        X, Y, X'
                DEFB  +E1,get-mem-1       X, Y, X', Y
                DEFB  +2A,abs             X, Y, X', Y' (Y' = ABS Y)
                DEFB  +0F,addition        X, Y, X'+Y'
                DEFB  +E0,get-mem-0       X, Y, X'+Y', SIN (G/2)
                DEFB  +05,division        X, Y, (X'+Y')/SIN (G/2)=Z', say
                DEFB  +2A,abs             X, Y, Z (Z = ABS Z')
                DEFB  +E0,get-mem-0       X, Y, Z, SIN (G/2)
                DEFB  +01,exchange        X, Y, SIN (G/2), Z
                DEFB  +3D,re-stack        (Z перезаносится в стек,
                DEFB  +38,end-calc        чтобы убедиться, что его
                                          порядок является допустимым).
                LD    A,(HL)              Получение порядка Z.
                CP    +81                 Если Z больше или равно
                JR    NC,23C1,DR-PRMS     1, то переход.
                RST   0028,FP-CALC        X, Y, SIN (G/2), Z
                DEFB  +02,delete          X, Y, SIN (G/2)
                DEFB  +02,delete          X, Y
                DEFB  +38,end-calc        Изображение линии из X0, Y0
                JP    2477,LINE-DRAW      в X0+X, Y0+Y.
```

2. Просто вызывается CD-PRMS1. Эта подпрограмма записывает в регистр B количество минимальных дуг, необходимых для построения полной дуги, а именно A=4*INT (G'*SQR Z/8)+4, где G' = mod G, или 252 если это выражение превышает 252 (что может случиться при большой хорде и малом угле). Итак, A равно 4, 8, 12, ... , и т.д. до 252. Эта подпрограмма также хранит в памяти от mem-0 до mem-4 величины G/A, SIN (G/2*A), 0, COS (G/A), SIN (G/A).

```
23C1 DR-PRMS    CALL  247D,CD-PRMS1       Подпрограмма вызвана.
```

3. Задание остальных параметров происходит следующим образом. Стек будет содержать эти 4 элемента, считываемых к вершине: X0+X и Y0+Y, как начало последней дуги; затем X0 и Y0, как начало первой дуги. Mem-0 будет содержать X0 и mem-5 - Y0. Mem-1 и mem-2 будет содержать начальные смещения для первой дуги, U и V; а mem-3 и mem-4 будут содержать COS (G/A) и SIN (G/A) для использования в цикле изображения дуг.

Формула для U и V может быть объяснена следующим образом. Вместо пошагового движения вдоль последней хорды, длиной L, со смещениями X и Y, мы пройдем вдоль первой хорды (которая может быть длиннее) длиной L*W, где W=SIN (G/2*A)/SIN (G/2), со смещением X*W и Y*W, но повернутого

на угол (G/2 - G/2*A), отсюда с истинными смещениями:

```
U = Y*W*SIN (G/2 - G/2*A) + X*W*COS (G/2 - G/2*A)
Y = Y*W*COS (G/2 - G/2*A) - X*W*SIN (G/2 - G/2*A)
```

Эти формулы могут быть проверены с помощью диаграммы, использующей стандартное разложение COS (P - Q) и SIN (P - Q), где Q = G/2 - G/2*A

```
23C4            PUSH  BC                Запись счетчика дуг в B.
                RST   0028,FP-CALC      X,Y,SIN(G/2),Z
                DEFB  +02,delete        X,Y,SIN(G/2)
                DEFB  +E1,get-mem-1     X,Y,SIN(G/2),SIN(G/2*A)
                DEFB  +01,exchange      X,Y,SIN(G/2*A),SIN(G/2)
                DEFB  +05,division      X,Y,SIN(G/2*A)/SIN(G/2)=W
                DEFB  +C1,st-mem-1      (W скопировано в mem-1).
                DEFB  +02,delete        X,Y
                DEFB  +01,exchange      Y,X
                DEFB  +31,duplicate     Y,X,X
                DEFB  +E1,get-mem-1     Y,X,X,W
                DEFB  +04,multiply      Y,X,X*W
                DEFB  +C2,st-mem-2      (X*W скопировано в mem-2).
                DEFB  +02,delete        Y,X
                DEFB  +01,exchange      X,Y
                DEFB  +31,duplicate     X,Y,Y
                DEFB  +E1,get-mem-1     X,Y,Y,W
                DEFB  +04,multiply      X,Y,Y*W
                DEFB  +E2,get-mem-2     X,Y,Y*W,X*W
                DEFB  +E5,get-mem-5     X,Y,Y*W,X*W,G
                DEFB  +E0,get-mem-0     X,Y,Y*W,X*W,G,G/A
                DEFB  +03,subtract      X,Y,Y*W,X*W,G - G/A
                DEFB  +A2,stk-half      X,Y,Y*W,X*W,G - G/A, 1/2
                DEFB  +04,multiply      X,Y,Y*W,X*W, G/2 - G/2*A=F
                DEFB  +31,duplicate     X,Y,Y*W,X*W, F, F
                DEFB  +1F,sin           X,Y,Y*W,X*W, F, SIN F
                DEFB  +C5,st-mem-5      (SIN F скопирован в mem-5).
                DEFB  +02,delete        X,Y,Y*W,X*W,F
                DEFB  +20,cos           X,Y,Y*W,X*W, COS F
                DEFB  +C0,st-mem-0      (COS F скопирован в mem-0).
                DEFB  +02,delete        X,Y,Y*W,X*W
                DEFB  +C2,st-mem-2      (X*W скопирован в mem-2).
                DEFB  +02,delete        X,Y,Y*W
                DEFB  +C1,st-mem-1      (Y*W скопирован в mem-1).
                DEFB  +E5,get-mem-5     X,Y,Y*W,SIN F
                DEFB  +04,multiply      X,Y,Y*W*SIN F
                DEFB  +E0,get-mem-0     X,Y,Y*W*SIN F,X*W
                DEFB  +E2,get-mem-2     X,Y,Y*W*SIN F,X*W, COS F
                DEFB  +04,multiply      X,Y,Y*W*SIN F,X*W*COS F
                DEFB  +0F,addition      X,Y,Y*W*SIN F+X*W*COS F=U
                DEFB  +E1,get-mem-1     X,Y,U,Y*W
                DEFB  +01,exchange      X,Y,Y*W,U
                DEFB  +C1,st-mem-1      (U скопировано в mem-1)
                DEFB  +02,delete        X,Y,Y*W
                DEFB  +E0,get-mem-0     X,Y,Y*W, COS F
                DEFB  +04,multiply      X,Y,Y*W*COS F
                DEFB  +E2,get-mem-2     X,Y,Y*W*COS F,X*W
                DEFB  +E5,get-mem-5     X,Y,Y*W*COS F,X*W, SIN F
                DEFB  +04,multiply      X,Y,Y*W*COS F,X*W*SIN F
                DEFB  +03,subtract      X,Y,Y*W*COS F - X*W*SIN
                                        F = V
```

```
                DEFB  +C2,st-mem-2        (V скопировано в mem-2).
                DEFB  +2A,abs             X, Y, V' (V' = ABS V)
                DEFB  +E1,get-mem-1       X, Y, V', U
                DEFB  +2A,abs             X, Y, V', U' (U' = ABS U)
                DEFB  +0F,addition        X, Y, U' + V'
                DEFB  +02,delete          X, Y
                DEFB  +38,end-calc        (DE указывает теперь
                                          на U' + V').
                LD    A,(DE)              Прочитать порядок U' + V'.
                CP    +81                 Если U' + V' меньше 1,
                POP   BC                  то стек очищается и
                JP    C,2477,LINE-DRAW    чертится линия из
                                          X0, Y0 в X0+X, Y0+Y.
                PUSH  BC                  Иначе, продолжайте с параметрами:
                RST   0028,FP-CALC        X, Y, в стеке.
                DEFB  +01,exchange        Y, X
                DEFB  +38,end-calc
                LD    A,(COORDS-lo)       Получение X0 в A и
                CALL  2D28,STACK-A        далее в стек.
                RST   0028,FP-CALC        Y, X, X0
                DEFB  +C0,st-mem-0        (X0 скопировано в mem-0).
                DEFB  +0F,addition        Y, X0 + X
                DEFB  +01,exchange        X0+X, Y
                DEFB  +38,end-calc
                LD    A,(COORDS-hi)       Получение Y0 в A и
                CALL  2D28,STACK-A        далее в стек.
                RST   0028,FP-CALC        X0+X, Y, Y0
                DEFB  +C5,st-mem-5        (Y0 скопировано в mem-5).
                DEFB  +0F,addition        X0+X, Y0+Y
                DEFB  +E0,get-mem-0       X0+X, Y0+Y, X0
                DEFB  +E5,get-mem-5       X0+X, Y0+Y, X0, Y0
                DEFB  +38,end calc
                POP   BC                  Восстановление счетчика
                                          дуг в B.
```

4. Цикл изображения дуг. Вводится в 2439 с координатами начальной точки на вершине стека и начальным смещением для первой дуги в mem-1 и mem-2. Использует простую тригонометрию для обеспечения изображения всей последовательности дуг в точках, которые лежат на том же круге, что и первые две, с центральным углом. Это может быть проиллюстрировано так, что, если 2 точки X1, Y1 и X2, Y2 лежат на окружности и образуют угол N в центре, который также является началом координат, то X2 = X1*COS N - Y1*SIN N, а Y2=X1*SIN N + Y1*COS N. Но так как началом является нижний левый угол экрана, то цикл изображения дуг применяет эти соотношения пошагово, скажем, Un = Xn+1 - Xn и Vn = Yn+l - Yn, таким образом, получая желаемый результат. Стек показан ниже на (n+1)-ом проходе через цикл, так как Xn и Yn увеличиваются на Un и Vn, после того как эти приращения получены из Un-1 и Vn-1. Четыре значения на вершине стека в 2425, в DRAW, считываемые по направлению вверх, являются X0+X, Y0+Y, Xn и Yn, но чтобы сохранить место, они не показаны до 2439. Для начальных значений в CIRCLE смотрите окончание CIRCLE выше. Кроме того, в CIRCLE угол G должен быть 2*PI.

```
2420 DRW-STEPS    DEC   B                 B просчитывает проходы
                                          через цикл.
                  JR    Z,245F,ARC-END    Переход, когда B достигает 0.
                  JR    2439,ARC-START    Переход для старта в цикл.
2425 ARC-LOOP     RST   0028,FP-CALC      (См. выше текст для стека).
                  DEFB  +E1,get-mem-1     Un-1
```

```
                DEFB  +31,duplicate       Un-1,Un-1
                DEBF  +E3,get-mem-3       Un-1,Un-1,COS(G/A)
                DEFB  +04,multiply        Un-1,Un-1*COS(G/A)
                DEFB  +E2,get-mem-2       Un-1,Un-1*COS(G/A),Vn-1
                DEFB  +E4,get-mem-4       Un-1,Un-1*COS(G/A),Vn-1,
                                          SIN(G/A)
                DEFB  +04,multiply        Un-1,Un-1*COS(G/A),Vn-1*
                                          SIN(G/A)
                DEFB  +03,subtract        Un-1,Un-1*COS(G/A)-Vn-1*
                                          SIN(G/A)=Un
                DEFB  +C1,st-mem-1        (Un скопировано в mem-1).
                DEFB  +02,delete          Un-1
                DEFB  +E4,get-mem-4       Un-1,SIN(G/A)
                DEFB  +04,multiply        Un-1*SIN(G/A)
                DEFB  +E2,get-mem-2       Un-1*SIN(G/A),Vn-1
                DEFB  +E3,get-mem-3       Un-1*SIN(G/A),Vn-1,COS(G/A)
                DEFB  +04,multiply        Un-1*SIN(G/A),Vn-1*COS(G/A)
                DEFB  +0F,addition        Un-1*SIN(G/A)+Vn-1*COS
                                          (G/A)=Vn
                DEFB  +C2,st-mem-2        (Vn скопировано в mem-2).
                DEFB  +02,delete          (Как отмечено в тексте, стек
                DEFB  +38,end-calc        в действительности содержит
                                          X0+X, Y0+Y, Xn и Yn).
2439 ARC-START  PUSH  BC                  Запись счетчика дуг.
                RST   0028,FP-CALC        X0+X, Y0+Y, Xn, Yn
                DEFB  +C0,st-mem-0        (Yn скопировано в mem-0).
                DEFB  +02,delete          X0+X, Y0+Y, Xn
                DEFB  +E1,get-mem-1       X0+X, Y0+Y, Xn, Un
                DEFB  +0F,addition        X0+X, Y0+Y, Xn+Un = Xn+1
                DEFB  +31,duplicate       X0+X, Y0+Y, Xn+1, Xn+1
                DEFB  +38,end-calc        Следующее Xn', аппроксимированное
                                          значение Xn полученное
                                          подпрограммой рисования линии
                LD    A,(COORDS-lo)       копируется в A
                CALL  2D28,STACK-A        и отсюда в стек.
                RST   0028,FP-CALC        X0+X,Y0+Y,Xn+1,Xn'
                DEFB  +03,subtract        X0+X,Y0+Y,Xn+1,Xn+1,Xn'
                                          - Xn' = Un'
                DEFB  +E0,get-mem-0       X0+X,Y0+Y,Xn+1,Un',Yn
                DEFB  +E2,get-mem-2       X0+X,Y0+Y,Xn+1,Un',Yn,Vn
                DEFB  +0F,addition        X0+X,Y0+Y,Xn+1,Un',Yn +
                                          Vn = Yn+1
                DEFB  +C0,st-mem-0        (Yn+1 скопировано в mem-0).
                DEFB  +01,exchange        X0+X,Y0+Y,Xn+1,Yn+1,Un'
                DEFB  +E0,get-mem-0       X0+X,Y0+Y,Xn+1,Yn+1,
                                          Un',Yn+1
                DEFB  +38,end-calc
                LD    A,(COORDS-hi)       Yn', аппроксимированное как Xn',
                CALL  2D28,STACK-A        скопировано в A и отсюда в стек.
                RST   0028,FP-CALC        X0+X,Y0+Y,Xn+1,Yn+1,
                                          Un',Yn+1,Yn'
                DEFB  +03,subtract        X0+X,Y0+Y,Xn+1,Yn+1,
                                          Un',Vn'
                DEFB  +38,end-calc
                CALL  24B7,DRAW-LINE      Чертится следующая дуга.
                POP   BC                  Восстановлен счетчик дуг.
                DJNZ  2425,ARC-LOOP       Переход, если большинство
                                          дуг начерчено.
245F ARC-END    RST   0028,FP-CALC        Координаты конца
                DEFB  +02,delete          последней начерченной дуги
                DEFB  +02,delete          теперь удалены из стека.
```

```
                DEFB  +01,exchange        Y0+Y, X0+X
                DEFB  +38,end-calc
                LD    A,(COORDS-lo)       Координата X последней
                CALL  2D28,STACK-A        начерченной дуги, скажем
                RST   0028,FP-CALC        Xz', скопирована в стек.
                DEFB  +03,subtract        Y0+Y, X0+X - Xz'
                DEFB  +01,exchange        X0+X - Xz', Y0+Y
                DEFB  +38,end-calc
                LD    A,(COORDS-hi)       Получена координата Y.
                CALL  2D28,STACK-A
                RST   0028,FP-CALC        X0+X - Xz', Y0+Y, Yz'
                DEFB  +03,subtract        X0+X - Xz', Y0+Y - Yz'
                DEFB  +38,end-calc
2477 LINE-DRAW  CALL  24B7,DRAW-LINE      Последняя начерченная дуга
                                          достигает X0+X, Y0+Y (или
                                          замыкает круг).
                JP    0D4D,TEMPS          Выход с заданием рабочих
                                          цветов.
```

ПОДПРОГРАММА 'INITIAL PARAMETERS' ('Начальные параметры')
Эта подпрограмма вызывается и CIRCLE, и DRAW для установки их начальных параметров. При CIRCLE она вызывается с X, Y и радиусом Z на вершине стека, читая вверх. При DRAW с его собственными X, Y, SIN (G/2) и Z, как определено в DRAW выше, на вершине стека. Последовательность показана в стеке, начиная с Z вверх.
Подпрограмма возвращает в B счет дуг A, как это объясняется в CIRCLE и DRAW, и в mem-0 - mem-5, величины G/A, SIN (G/2*A), 0, COS (G/A), SIN (G/A) и G. Для круга G должно быть равно 2*PI.

```
247D CD-PRMS1   RST   0028,FP-CALC        Z
                DEFB  +31,duplicate       Z, Z
                DEFB  +28,sqr             Z, SQR Z
                DEFB  +34,stk-data        Z, SQR Z, 2

                DEFB  +32,exponent +82
                DEFB  +00,(+00,+00,+00)
                DEFB  +01,exchange        Z, 2, SQR Z
                DEFB  +05,division        Z, 2/SQR Z
                DEFB  +E5,get-mem-5       Z, 2/SQR Z, G
                DEFB  +01,exchange        Z, G, 2/SQR Z
                DEFB  +05,division        Z, G*SQR Z/2
                DEFB  +2A,abs             Z, G'*SQR Z/2 (G' = mod G)
                DEFB  +38,end-calc        Z, G'*SQR Z/2 = A1, скажем
                CALL  2DD5,FP-TO-A        A1 идет в A из стека, если
                                          это возможно.
                JR    C,2495,USE-252      Если A1 округляется до 256
                                          или более, используется 252.
                AND   +FC                 4*INT (A1/4) в A.
                ADD   A,+04               Добавьте 4, задавая счет
                                          дуг A.
                JR    NC,2497,DRAW-SAVE   Переход, если еще не дошли
                                          до 256.
2495 USE-252    LD    A,+FC               Здесь используйте только
                                          десятичное 252.
2497 DRAW-SAVE  PUSH  AF                  Теперь запишите счет дуг.
                CALL  2D28,STACK-A        Скопировать также в стек
                                          калькулятора.
                RST   0028,FP-CALC        Z, A
                DEFB  +E5,get-mem-5       Z, A, G
```

```
                DEFB  +01,exchange        Z, G, A
                DEFB  +05,division        Z, G/A
                DEFB  +31,duplicate       Z,G/A, G/A
                DEFB  +1F,sin             Z, G/A, SIN (G/A)
                DEFB  +C4,st-mem-4        (SIN (G/A) скопирован в mem-4).
                DEFB  +02,delete          Z, G/A
                DEFB  +31,duplicate       Z, G/A, G/A
                DEFB  +A2,stk-half        Z, G/A, G/A, 0.5
                DEFB  +04,multiply        Z, G/A, G/2*A
                DEFB  +1F,sin             Z, G/A, SIN (G/2*A)
                DEFB  +C1,st-mem-1        (SIN (G/2*A) скопирован в
                                          mem-1).
                DEFB  +01,exchange        Z, SIN (G/2*A), G/A
                DEFB  +C0,st-mem-0        (G/A скопирован в mem-0).
                DEFB  +02,delete          Z, SIN (G/2*A) = S
                DEFB  +31,duplicate       Z, S, S
                DEFB  +04,multiply        Z, S*S
                DEFB  +31,duplicate       Z, S*S, S*S
                DEFB  +0F,addition        Z, 2*S*S
                DEFB  +A1,stk-one         Z, 2*S*S, 1
                DEFB  +03,subtract        Z, 2*S*S - 1
                DEFB  +1B,negate          Z, 1 - 2*S*S = COS (G/A)
                DEFB  +C3,st-mem-3        (COS (G/A) скопирован в mem-4).
                DEFB  +02,delete          Z
                DEFB  +38,end-calc
                POP   BC                  Восстановление счета дуг в B.
                RET                       Окончание.
```

ПОДПРОГРАММА 'DRAW-LINE' ('Рисование линий')
Эта подпрограмма вызывается DRAW для того, чтобы начертить прямую линию из точки X0, Y0, содержащейся в COORDS, в точку X0+X, Y0+Y, где инкременты X и Y находятся на вершине стека калькулятора. Подпрограмма была изначально предназначена для ZX80 и ZX81 с ОЗУ 8K и описана в программе BASIC на странице 121 руководства по ZX81. Она проиллюстрирована здесь, в приложении в программе CIRCLE.
Метод разбивает на такое множество горизонтальных или вертикальных шагов, какое необходимо для основного набора диагональных шагов, используя алгоритм, который располагает горизонтальные и вертикальные шаги так ровно, насколько это возможно.

```
24B7 DRAW-LINE  CALL  2307,STK-TO-BC     ABS Y to B; ABS X to C;
                                         SGN Y to D; SGN X to E.
                LD    A,C                Переход, если ABS X больше
                CP    B                  или равно ABS Y, при этом
                JR    NC,24C4,DL-X-GE-Y  меньшее идет в L, а большее
                LD    L,C                (позднее) идет в H.
                PUSH  DE                 запись диагонального шага
                                         (+/-1,+/-1) в DE.
                XOR   A                  Вставка вертикального шага
                LD    E,A                (+/-1, 0) в DE.
                                         (D содержит SGN Y).
                JR    24CB,DL-LARGER     Теперь переход для
                                         задания H.
24C4 DL-X-GE-Y  OR    C                  Возврат, если ABS X и ABS Y
                RET   Z                  равны 0.
                LD    L,B                Меньшее значение (здесь ABS Y)
                                         идет в L.
                LD    B,C                ABS X идет в B здесь, для H.
```

```
                PUSH  DE                Здесь также запись
                                        диагонального шага.
                LD    D,+00             Здесь горизонтальный
                                        шаг (0, +/-1) в DE.
24CB DL-LARGER  LD    H,B               Большее ABS X, ABS Y в H.
```

Здесь начинается алгоритм. Величина, большая чем ABS X и ABS Y, скажем H, посылается в A и понижается до INT (H/2). H - L горизонтальных и вертикальных шагов и L диагональных шагов берутся следующим образом (где L меньше, чем ABS X и ABS Y): L добавляет H, оно уменьшается на H и берется диагональный шаг; в противном случае берется вертикальный или горизонтальный шаг. Это повторяется H раз (B также содержит H). Отметим, что тем временем используются регистры обмена H' и L', содержащие COORDS.

```
                LD    A,B               B в A также как и в H.
                RRA                     A начинает в INT(H/2).
24CE D-L-LOOP   ADD   A,L               L добавлено A.
                JR    C,24D4,D-L-DIAG   Если 256 или более, переход
                                        - диагональный шаг.
                CP    H                 Если A меньше H, то
                JR    C,24DB,D-L-HR-VT  переход для горизонтального
                                        или вертикального шага.
24D4 D-L-DIAG   SUB   H                 Понижение A на H.
                LD    C,A               Восстановление в C.
                EXX                     Теперь используются
                                        регистры обмена.
                POP   BC                Диагональный шаг в B'C'.
                PUSH  BC                Запись этого.
                JR    24DF,D-L-STEP     Переход, чтобы сделать шаг.
24DB D-L-HR-VT  LD    C,A               Запись A (не уменьшенного)
                                        в C.
                PUSH  DE                Шаг в стек.
                EXX                     Получение регистров обмена.
                POP   BC                Теперь шаг в B'C'.
24DF D-L-STEP   LD    HL,(COORDS)       Теперь берем шаг, сначала
                                        COORDS в H'L', как начальную
                                        точку.
                LD    A,B               Шаг Y из B' в A.
                ADD   A,H               Добавить в H'.
                LD    B,A               Результат в B'.
                LD    A,C               Теперь шаг X; он будет
                INC   A                 проверен по диапазону (Y
                                        будет проверен в PLOT).
                ADD   A,L               Добавить L' в C' A,
                JR    C,24F7,D-L-RANGE  переход, если перенос, для
                                        дальнейшей проверки.
                JR    Z,24F9,REPORT-B   Если нет переноса, ноль
                                        обозначает, что X-позиция -1,
                                        находится вне диапазона.
24EC D-L-PLOT   DEC   A                 Восстановление истинного
                                        значения в A.
                LD    C,A               Значение в C для изображения.
                CALL  22E5,PLOT-SUB     Изображение шага.
                EXX                     Восстановление основных
                                        регистров.
                LD    A,C               C возвращается в A для
                                        продолжения алгоритма.
                DJNZ  24CE,D-L-LOOP     Опять прокрутить цикл для
```

```
                                          шагов B (т.е. шагов H).
                POP   DE                  Очистка машинного стека.
                RET                       Окончание.
24F7 D-L-RANGE  JR    Z,24EC,D-L-PLOT     Ноль после переноса обоз-
                                          начает, что X-позиция-255,
                                          в диапазоне.
```

Сообщение B - 'Целое вне диапазона'.

```
24F9 REPORT-B   RST  0008,ERROR-1         Вызов программы обработки
                DEFB  +0A                 ошибок.
```

РАСЧЕТ ВЫРАЖЕНИЙ

ПОДПРОГРАММА 'SCANNING' ('Просмотр')
Эта подпрограмма используется для получения результата вычисления 'следующего выражения'.

Возвращаемый на стек калькулятора результат является 'последним значением'. Для численного результата последнее значение (будет действительным числом с плавающей точкой. Однако, для строкового результата последнее значение будет состоять из набора параметров. Первые пять байт являются непреодолимыми, вторые и третьи содержат адреса начала строки, а четвертые и пятые содержат длину строки.

6 разряд во FLAGS задается для численного результата и сбрасывается для строкового.

Когда 'следующее выражение' состоит только из одного оператора, например, ... A ..., ... RND ..., ... A$ (4, от 3 до 7) ... , то последующее значение просто является значением, которое получено после обработки оператора.

Однако, когда следующее значение содержит функцию и операнд, например, ... CHR$ A ..., ... NOT A ..., SIN 1 ..., код операции функции заносится на машинный стек, пока последнее значение посчитается. Это последнее значение затем соотносится о соответствующей операцией и дает новое последнее значение.

В случае, если выполнена арифметическая или логическая операция, например, ... A+B ..., A*B ..., ... A=B ..., то и последнее значение первого аргумента, и код операции должны передаваться до тех пор, пока не будет найдено значение второго аргумента, может также включать в себя хранение последних значений и кодов операций, пока выполняются вычисления. Можно показать, как вычисляется сложное выражение, например, ... CHR$ (T+A - 26*INT ((T+A)/26)+65)...; иерархия операций, подлежащих выполнению, строится до достижения точки, начиная с которой она должна быть разобрана для создания окончательного последнего значения.

Каждый код операции связан с соответствующим приоритетным кодом, и операции более высокого приоритета всегда выполняются раньше, чем более низкого.

Подпрограмма начинается с регистра, который содержит первый символ выражения и начальный маркер приоритета 0, который занесен в машинный стек.

```
24FB SCANNING     RST   0018,GET-CHAR       Выбран первый символ.
                  LD    B,+00               Начальный маркер приоритета.
                  PUSH  BC                  Создан стек.
24FF S-LOOP-1     LD    C,A                 Главная точка повторного
                                            входа.
                  LD    HL,+2596            Введен индекс в таблицу
                  CALL  16DC,INDEXER        функций просмотра и
                                            помещен в C.
                  LD    A,C                 Восстановление кода в A.
                  JP    NC,2684,S-ALPHNUM   Переход, если не обнаружен
                                            код в таблице.
                  LD    B,+00               Используйте вход,
                  LD    C,(HL)              обнаруженный в таблице,
                  ADD   HL,BC               чтобы встроить требуемый
                  JP    (HL)                адрес в HL и перейдите туда.
```

Далее следуют четыре подпрограммы: они вызываются программой из таблицы функций просмотра. Первая, 'подпрограмма просмотра кавычек', вызываемая как S-QUOTE, проверяет, совпадают ли строковые кавычки с другими кавычками.

```
250F S-QUOTE-S    CALL  0074,CH-ADD+1       Указывает на следующий
                                            символ.
                  INC   BC                  Увеличивает подсчет длины
                                            на 1.
                  CP    +0D                 Это возврат каретки?
                  JP    Z,1C8A,REPORT-C     Сообщение об ошибке,
                                            если это так.
                  CP    +22                 Это другие кавычки?
                  JR    NZ,250F,S-QUOTE-S   Опять работа в цикле,
                                            если это не так.
                  CALL  0074,CH-ADD+1       Указывает на следующий сим-
                  CP    +22                 вол: задает признак нуля,
                                            если символ это другие
                                            кавычки.
                  RET                       Окончание.
```

Следующая подпрограмма. Просмотр: две координаты как S-SCREEN$, S-ATTR и S-POINT, которые задают две требуемые координаты в нужном виде.

```
2522 S-2-COORD    RST   0020, NEXT-CHAR     Выбор следующего символа.
                  CP    +28                 Это '('?
                  JR    NZ,252D,S-RPORT-C   Если нет, сообщение об
                                            ошибке.
                  CALL  1C79,NEXT-2NUM      Координаты помещаются в
                                            стек.
                  RST   0018,GET-CHAR       Выбор текущего символа.
                  CP    +29                 Это ')'?
252D S-RPORT-C    JP    NZ,1C8A,REPORT-C    Если нет, сообщение об
                                            ошибке.
```

ПОДПРОГРАММА 'SYNTAX-Z'
Подпрограмма 'SYNTAX-Z' является интерполируемой. Она вызывается 32 раза, с записью только одного байта на каждом вызове. Простой тест седьмого разряда FLAGS будет сбрасывать признак нуля во время выполнения и устанавливать его во время проверки синтаксиса.
Таким образом, SYNTAX задает множество Z.

```
2530 SYNTAX-Z     BIT   7,(FLAGS)           Проверка 7 разряда FLAGS.
                  RET                       Окончание.
```

Следующая подпрограмма - это 'подпрограмма просмотра SCREEN$', которая используется S-SCREEN$ для обнаружения символа, который появляется в строке X и столбце Y экрана. Она только ищет набор символов, 'указанных' в CHARS.

Примечание. Это обычно символы от +20 (пробел) до +7F ((C)), хотя пользователь может изменить CHARS и для других символов, включая определенную пользователем графику.

```
2535 S-SCRN$-S    CALL  2307,STK-TO-BC      x в C, y в B:
                  LD    HL,(CHARS)          0<=x<=23, десятичное
                                            0<=y<=31, десятичное.
                  LD    DE,+0100            CHARS плюс 256 десятичное
                  ADD   HL,DE               дает HL, указывающее набор
                                            символов.
                  LD    A,C                 x скопировано в A.
                  RRCA                      Число 32 (дес.) * (x
                  RRCA                      mod 8) + y формируется в A
                                            и копируется в E.
```

```
                RRCA                    Это младший байт требуемого
                AND   +E0               адреса экрана.
                XOR   B
                LD    E,A
                LD    A,C               X опять копируется в A.
                AND   +18               Теперь число 64 (дес.) +
                XOR   +40               8*INT (x/8) введено в D.
                LD    D,A               DE теперь содержит адрес
                                        экрана.
                LD    B,+60             B считает 96 символов.
254F S-SCRN-LP  PUSH  BC                Запись счета.
                PUSH  DE                И указателя экрана.
                PUSH  HL                И указателя набора символов.
                LD    A,(DE)            Получение первой строки
                                        символов экрана.
                XOR   (HL)              Сравнение набора символов
                                        со строкой.
                JR    Z,255A,S-SC-MTCH  Переход, если найдено
                                        прямое совпадение.
                INC   A                 Теперь проверка для сравне-
                                        ния с противоположным симво-
                                        лом (получение +00 в A из
                                        +FF).
                JR    NZ,2573,S-SCR-NXT Переход, если не найдено
                                        совпадение.
                DEC   A                 Восстановление +FF в A.
255A S-SC-MTCH  LD    C,A               Противоположное состояние
                                        (+00 или + FF) в C.
                LD    B,+07             A считает остальные 7 строк.
255D S-SC-ROWS  INC   D                 Пересылка DE в следующую
                                        строку (добавить 256, дес.)
                INC   HL                Пересылка HL в следующую
                                        строку (т.е. следующий байт).
                LD    A,(DE)            Прочитать строку экрана.
                XOR   (HL)              Сравнить со строкой из ПЗУ.
                XOR   C                 Включение противоположного
                                        состояния.
                JR    NZ,2573,S-SCR-NXT Переход, если строка не
                                        совпадает.
                DJNZ  255D,S-SC-ROWS    Переход назад, пока
                                        обработаются все строки.
                POP   BC                Отбросить указатель набора
                                        символов.
                POP   BC                И указатель экрана.
                POP   BC                Окончательный счет в BC.
                LD    A,+80             Код последнего символа в
                                        наборе, плюс 1.
                SUB   B                 Теперь A содержит требуемый
                                        код.
                LD    BC,+0001          В рабочей области необходимо
                                        одно место.
                RST   0030,BC-SPACES    Создание места.
                LD    (DE),A            Занести в него символ.
                JR    257D,S-SCR-STO    Переход для помещения
                                        символа в стек.
2573 S-SCR-NXT  POP   HL                Восстановление указателя
                                        набора символов.
                LD    DE,+0008          Переслать его на 8 байт к
                ADD   HL,DE             следующему символу в наборе.
```

```
                  POP   DE                Восстановление указателя
                                           экрана.
                  POP   BC                И счетчика.
                  DJNZ  254F,S-SCRN-LP    Прогон цикла для 96 символов.
                  LD    C,B               Помещение в стек пустой
                                           строки (нулевой длины).
257D S-SCR-STO    JP 2AB2,STK-STO-$       Переход для помещения в
                                           стек совпавшего символа
                                           или пустой строки, если
                                           таковой не обнаружен.
```

Примечание. Выход через STK-STO-$ является ошибкой, т.к. приводит к 'двойному запоминанию' результатов строки (см. S-STRING, 25DB). Командной строкой должна быть 'RET'.
Последней из этих программ является 'подпрограмма атрибутов просмотра'. Вызывая S-ATTR для возвращения значения ATTR (x,y), которая кодирует атрибуты строки x и столбца y на телевизионном экране.

```
2580 S-ATTR-S     CALL  2307,STK-TO-BC    x в C, y в B: опять
                  LD    A,C               0<=x<=23, десятичное,
                                          0<=y<=31, десятичное.
                  RRCA                    x скопировано в A, и в A
                  RRCA                    сформировано число
                  RRCA                    32 (дес.)*x (mod 8)+y и
                                          скопировано в L.
                  LD    C,A               В C также скопирова-
                  AND   +E0               но 32*x (mod 8)+INT (x/8).
                  XOR   B
                  LD    L,A               L содержит младший
                                          байт адреса атрибута.
                  LD    A,C               В A скопировано
                                          32*x (mod 8)+INT (x/8).
                  AND   +03               В A сформировано
                  XOR   +58               88 (десятичное) + INT(X/8).
                  LD    H,A               H содержит старший разряд
                                          байта атрибута.
                  LD    A,(HL)            Байт атрибута скопирован
                                          в A.
                  JP    2D28,STACK-A      Выход, помещение в стек
                                          требуемого байта.
```

ТАБЛИЦА ФУНКЦИЙ ПАРАМЕТРА
Таблица содержит 8 функций параметра и 4 оператора. Она, таким образом, объединяет пять новых функций Spectrum и обеспечивает доступ к некоторым функциям и операторам, которые уже существуют в ZX81.

Ячейка	Код	Смещение	Имя	Адрес обрабатывающей программы
2596	22	1C	S-QUOTE	25B3
2598	28	4F	S-BRACKET	25E8
259A	2E	F2	S-DECIMAL	268D
259C	2B	12	S-U-PLUS	25AF
259E	A8	56	S-FN	25F5
25A0	A5	57	S-AND	25F8
25A2	A7	84	S-PI	2627
25A4	A6	8F	S-INKEY$	2634
25A6	C4	E6	S-BIN (EQU. S-DECIMAL)	268D

```
25A8    AA          BF      S-SCREEN$           2668
25AA    AB          C7      S-ATTR              2672
25AC    A9          CE      S-POINT             267B
25AE    00                                      Маркер конца
```

ПРОГРАММЫ ПРОСМОТРА ФУНКЦИЙ

```
25AF S-U-PLUS     RST   0020,NEXTCHAR       Для унарного плюса простая
                  JP    24FF,S-LOOP-1       пересылка на следующий сим-
                                            вол и переход назад к основ-
                                            ной точке повторного входа
                                            SCANNING.
```

'Программа просмотра QUOTE': Эта программа рассматривает кавычки в строке, или простые типа "name", или более сложные типа "a ""white"" lie", или подобно VAL$ """a""".

```
25B3 S-QUOTE      RST   0018,GET-CHAR       Выбор текущего символа.
                  INC   HL                  Указывает начало строки.
                  PUSH  HL                  Запись начального адреса.
                  LD    BC,+0000            Приведение длины в ноль.
                  CALL  250F,S-QUOTE-S      Вызов подпрограммы
                                            'сопоставления'.
                  JR    NZ,25D9,S-Q-PRMS    Переход, если сброшен
                                            0 - кавычек больше нет.
25BE S-Q-AGAIN    CALL  250F,S-QUOTE-S      Повторный вызов для
                                            третьих кавычек.
                  JR    Z,25BE,S-Q-AGAIN    И снова для пятых,
                                            седьмых и т.д.
                  CALL  2530,SYNTAX-Z       При проверке синтаксиса,
                  JR    Z,25D9,S-Q-PRMS     переход для сброса 6
                                            разряда FLAGS.
                  RST   0030,BC-SPACES      Создание места в рабочей
                                            области для строки и
                                            завершающих кавычек.
                  POP   HL                  Установка указателя на
                                            начало.
                  PUSH  DE                  Запись указателя на первое
                                            место.
25CB S-Q-COPY     LD    A,(HL)              Получить символ из строки.
                  INC   HL                  Указать следующий.
                  LD    (DE),A              Скопировать последний в
                                            рабочую область.
                  INC   DE                  Указать следующее место.
                  CP    +22                 Последний символ '"'?
                  JR    NZ,25CB,S-Q-COPY    Если нет, переход для
                                            копирования следующего.
                  LD    A,(HL)              Если да, следующий символ не
                  INC   HL                  копировать; если следующий
                  CP    +22                 '"', переход для копирова-
                                            ния символа после этого.
                  JR    Z,25CB,S-Q-COPY     В противном случае
                                            закончить копирование.
25D9 S-Q-PRMS     DEC   BC                  Получение в BC истинной
                                            длины.
```

Отметим, что первые кавычки не входят в длину; последние были сосчитаны, а теперь отброшены. Внутри строки первые, третьи, пятые и

т.д. сосчитаны, но вторые, четвертые и т.д., нет.

```
                 POP   DE                 Восстановление начала
                                          скопированной строки.
25DB S-STRING    LD    HL,+5C3B           Это FLAGS; эта точка входа
                 RES   6,(HL)             используется, когда сбрасы-
                 BIT   7,(HL)             вается 6 разряд, а выполняю-
                 CALL  NZ,2AB2,STK-STO-S  щаяся строка, помещается в
                                          стек. Это сейчас сделано.
                 JP    2712,S-CONT-2      Переход для продолжения
                                          просмотра строки.
```

Отметим, что при копировании строки в рабочую область, каждые две пары строковых кавычек внутри строки ("") преобразовываются в одну пару кавычек (").

```
25E8 S-BRACKET   RST 0020,NEXT-CHAR       'Программа просмотра
                 CALL  24FB,SCANNING      BRACKET' получает следующий
                 CP    +29                символ и рекурсивно вызывает
                                          SCANNING.
                 JP    NZ,1C8A,REPORT-C   Если нет отождествления
                 RST   0020,NEXT-CHAR     скобок, то сообщение об
                 JP    2712,S-CONT-2      ошибке, затем продолжение
                                          просмотра.
25F5 S-FN        JP    27BD,S-FN-SBRN     'Программа просмотра FN'.
```

Это программа для функций, определенных пользователем, просто переходит в 'подпрограмму просмотра FN'.

```
25F8 S-RND       CALL  2530,SYNTAX-Z      Если синтаксис не был прове-
                 JR    Z,2626,S-RND-END   рен, то переход для вычисле-
                                          ния случайного числа.
                 LD    BC,(SEED)          Выбор текущего значения
                                          SEED.
                 CALL  2D2B,STACK-BC      Засылка его в стек
                                          калькулятора.
                 RST   0028,FP-CALC       Сейчас используется
                                          калькулятор.
                 DEFB  +A1,stk-one        'Последнее значение'
                 DEFB  +0F,addition       теперь SEED+1.
                 DEFB  +34,stk-data       Засылка десятичного числа
                 DEFB  +37,exponent+87    75 в стек калькулятора.
                 DEFB  +16,(+00,+00,+00)
                 DEFB  +04,multiply       'Последнее значение'
                                          (SEED+1)*75.
                 DEFB  +34,stk-data       Посмотрите STACK LITERALS,
                 DEFB  +80,(four bytes)   чтобы увидеть, как расширены
                 DEFB  +41,exponent +91   байты для занесения деся-
                 DEFB  +00,+00,+80,(+00)  тичного числа 65537 в стек
                                          калькулятора.
                 DEFB  +32,n-mod-m        Разделить (SEED+1)*75 на
                                          65537, чтобы дать 'остаток'
                                          и 'ответ'.
                 DEFB  +02,delete         Отбросить 'ответ'.
                 DEFB  +A1,stk-one        'Последним значением' теперь
                 DEFB  +03,subtract       является 'остаток'-1.
                 DEFB  +31,duplicate      Создание копии 'последнего
                                          значения'.
                 DEFB  +38,end-calc       Вычисление закончено.
```

```
                CALL  2DA2,FP-TO-BC       Использование 'последнего
                LD    (SEED),BC           значения' для задания нового
                                          значения для SEED.
                LD    A,(HL)              Выбор порядка 'последнего
                                          значения'.
                AND   A                   Переход вперед, если
                JR    Z,2625,S-RND-END    порядок равен 0.
                SUB   +10                 Понижение порядка, т.е.
                LD    (HL),A              деление 'последнего
                                          значения' на 65536 для
                                          задания требуемого
                                          'последнего значения'.
2625 S-RND-END  JR    2630,S-PI-END       Переход за программу 'PI'.
```

'Программа просмотра PI': пока не проверен синтаксис, вычисляется значение 'PI' и формируется 'последнее значение' на стеке калькулятора.

```
2627 S-PI       CALL  2530,SYNTAX-Z       Проверка синтаксиса.
                JR    Z,2630,S-PI-END     Если требуется, переход.
                RST   0028,FP-CALC        Теперь используется
                                          калькулятор.
                DEFB  +A3,stk-pi/2        Значение PI/2 заносится на
                DEFB  +3B,end-calc        стек калькулятора как
                                          'последнее значение'.
                INC   (HL)                Порядок увеличивается, в
                                          связи с чем удваивается
                                          'последнее значение',
                                          задающее PI.
2630 S-PI-END   RST   0020,NEXT-CHAR      Переход на следующий символ,
                JP    26C3,S-NUMERIC      Переход вперед.
2634 S-INKEY$   LD    BC,+105A            Установить приоритет для
                RST   0020,NEXT-CHAR      16-теричного числа +10,
                                          операционного кода +5A для
                                          подпрограммы 'read-in'.
                CP    +23                 Если следующий символ
                                          '#', переход.
                JP    Z,270D,S-PUSH-PO    Будет числовой аргумент.
                LD    HL,+5C3B            Это FLAGS.
                RES   6,(HL)              Для строкового результата
                                          сброс 6 разряда.
                BIT   7,(HL)              Проверка синтаксиса.
                JR    Z,2665,S-INK$-EN    Если требуется, переход.
                CALL  028E,KEY-SCAN       Выбор значения клавиши в DE.
                LD    C,+00               Подготовка пустой строки:
                JR    NZ,2660,S-IK$-STK   поместите ее в стек, если
                                          нажато много клавиш.
                CALL  031E,K-TEST         Проверка клавиш: если
                JR    NC,2660,S-IK$-STK   удовлетворительна, поместите
                                          в стек пустую строку.
                DEC   D                   +FF в D для режима L
                                          (задан третий разряд).
                LD    E,A                 Значение клавиши помещается
                                          в E для декодировки.
                CALL  0333,K-DECODE       Декодирование значения
                                          клавиши.
                PUSH  AF                  Сжатая запись ASCII.
                LD    BC,+0001            В рабочей области необходимо
                                          место.
                RST   0030,BC-SPACES      Его создание.
```

```
                  POP   AF                  Восстановление значения
                                            ASCII.
                  LD    (DE),A              Подготовка для помещения
                                            в стек в виде отроки.
                  LD    C,+01               Длина равна 1.
2660 S-IK$-STK    LD    B,+00               Заполняется параметр длины.
                  CALL  2AB2,STK-STO-$      Помещение в стек требуемой
                                            строки.
2665 S-INK$-EN    JP    2712,S-CONT-2       Переход вперед.
2668 S-SCREEN$    CALL  2522,S-2-COORD      Проверить, что две коорди-
                                            наты являются заданными.
                  CALL  NZ,2535,S-SCRN$-S   Пока не проверен синтаксис,
                  RST   0020,NEXT-CHAR      вызывается подпрограмма;
                  JP    25DB,S-STRING       затем получение следующего
                                            символа и переход назад.
2672 S-ATTR       CALL  2522,5-2-COORD      Проверить, что две
                                            координаты заданы.
                  CALL  NZ,2580,S-ATTR-S    Пока не проверен синтаксис,
                  RST   0020,NEXT-CHAR      вызывается подпрограмма;
                  JR    26C3,S-NUMERIC      затем прочитать следующий
                                            символ и переход вперед.
267B S-POINT      CALL  2522,S-2-COORD      Проверка, что заданы 2
                                            координаты.
                  CALL  NZ,22CB,POINT-SUB   Вызов подпрограммы пока
                  RST   0020,NEXT-CHAR      не проверен синтаксис, затем
                  JR    26C3,S-NUMERIC      прочитать следующий символ
                                            и переход вперед.
2684 S-ALPHNUM    CALL  2C88,ALPHANUM       Символ алфавитно-цифровой.
                  JR    NC,26DF,S-NEGATE    Если нет буквы или цифры,
                                            то переход.
                  CP    +41                 Теперь переход, если это
                                            буква.
                  JR    NC,26C9,S-LETTER    В противном случае
                                            продолжать в S-DECIMAL.
```

'Программа просмотра DECIMAL', которая последует дальше, рассматривает десятичную точку или число, которое начинается с цифры. Она также работает с выражением 'BIN', которое рассматривается в подпрограмме 'десятичное выражение с плавающей точкой'.

```
268D S-DECIMAL    CALL  2530,SYNTAX-Z       Если строка выполнена, то
     (EQU. S-BIN) JR    NZ,2685,S-STK-DEC   переход вперед.
```

Предложенное действие теперь является различным для проверки синтаксиса и вычисления строки. Если синтаксис проверен, то форма с плавающей точкой должна быть посчитана и скопирована в фактическую строку BASIC. Однако, когда строка выполнена, то форма с плавающей точкой будет всегда доступна и копируется в стек калькулятора в форме 'последнего значения'.

Во время проверки синтаксиса:

```
                  CALL  2C9B,DEC-TO-FP      Найдена форма с плавающей
                                            точкой.
                  RST   0018,GET-CHAR       HL указывает на ячейку
                                            после последней цифры.
                  LD    BC,+0006            Требуется 6 ячеек.
                  CALL  1655,MAKE-ROOM      Создание места в строке
                                            BASIC.
                  INC   HL                  Указывает на первое
```

```
                                свободное место.
                LD    (HL),+0E  Ввод кода маркера числа.
                INC   HL        Указывает на вторую ячейку.
                EX    DE,HL     Этот указатель нужен в DE.
                LD    HL,(STKEND) Выбор 'старого' STKEND.
                LD    C,+05     Для пересылки есть 5 байтов.
                AND   A         Очистить признак переноса.
                SBC   HL,BC     'Новый' STKEND =
                                'старый' STKEND - 5.
                LD    (STKEND),HL Пересылка числа с плавающей
                LDIR            точкой из стека калькулятора
                                в строку.
                EX    DE,HL     Поместить в HL указатель
                                строки.
                DEC   HL        Указать последний
                                добавленный байт.
                CALL  0077,TEMP-PTR1 Задается CH-ADD.
                JR    26C3,S-NUMERIC Переход вперед.
```

Во время выполнения строки:

```
26B5 S-STK-DEC  RST   0018,GET-CHAR  Получение текущего символа.
26B6 S-SD-SKIP  INC   HL        Теперь пересылка на
                LD    A,(HL)    следующий символ,
                CP    +0E       пока не обнаружится
                JR    NZ,26B6,S-SD-SKIP маркер числа.
                INC   HL        Указывает первый байт числа.
                CALL  33B4,STACK-NUM Пересылка числа с плавающей
                                точкой.
                LD    (CH-ADD),HL Задается CH-ADD.
```

Числовой материал, поступивший из RND, PI, ATTR, POINT или десятичного числа был

```
26C3 S-NUMERIC  SET   6,(FLAGS) Задает признак числового
                                маркера.
                JR    26DD,S-CONT-1 Переход вперед.
```

ПРОГРАММА ПРОСМОТРА ПЕРЕМЕННОЙ
Когда идентифицировано имя переменной, обращается к LOOK-VARS, просматривающей те переменные, которые уже существуют в области переменных (или в программной области операторов DEF FN для определенной пользователем функции FN). Если найдено соответствующее числовое значение, то оно с использованием STACK-NUM копируется в стек калькулятора. Однако, строка или элемент массива строк должны иметь соответствующие параметры, переданные в стек калькулятора подпрограммой STK-VAR (или в случае функции, определенной пользователем, подпрограммой STK-F-ARG, вызванной из LOOK-VARS).

```
26C9 S-LETTER   CALL  28B2,LOOK-VARS Просмотр существующих пере-
                                менных для сопоставления.
                JP    C,1C2E,REPORT-2 Сообщение об ошибке, если
                                сопоставления не существует.
                CALL  Z,2996,STK-VARS Помещение в стек параметров
                                строки.
                LD    A,(FLAGS) Выбор FLAGS.
                CP    +C0       Одновременная проверка 6
                                и 7 разрядов.
```

```
                JR    C,26DD,S-CONT-1     Сброс одного или обоих
                                          разрядов.
                INC   HL                  Помещение в стек числового
                                          значения.
                CALL  33B4,STACK-NUM      Пересылка числа.
26DD S-CONT-1   JR    2712,S-CONT-2       Переход вперед.
```

Сравнение кода символа с кодом '-', что идентифицирует операцию 'унарный минус'.

Перед фактической проверкой регистр B заполняется приоритетной величиной +09, а регистр C кодом операции +D8, которые необходимы для этой операции.

```
26DF S-NEGATE   LD    BC,+09DB            Приоритетная величина +09,
                                          код операции +D8.
                CP    +2D                 Это '-'?
                JR    Z,270D,S-PUSH-PO    Если это 'унарный минус', то
                                          переход вперед.
```

Следующий символ проверяется на код для 'VAL$', с приоритетной величиной 16 десятичное и кодом операции 18 шестнадцатеричное.

```
                LD    BC,+1018            Приоритет - 16 десятичное,
                                          код операции +18 шестнадцатеричное.
                CP    +AE                 Это 'VAL$'?
                JR    Z,270D,S-PUSH-PO    Переход вперед, если это
                                          'VAL$'.
```

Текущий символ должен теперь представить одну из функций от CODE до NOT, с кодами от +AF до +C3.

```
                SUB   +AF                 Диапазон функций изменяется
                                          из диапазона от +AF до +C3
                                          в диапазон от +00 до + 14
                                          шестнадцатеричный.
                JP    C,1C8A,REPORT-C     Если вне диапазона, то
                                          сообщение об ошибке.
```

Функция 'NOT' идентифицируется и рассматривается отдельно от остальных.

```
                LD    BC,+04F0            Приоритет +04, код
                                          операции +F0.
                CP    +14                 Функция 'NOT'?
                JR    Z,270D,S-PUSH-PO    Если да, переход.
                JP    NC,1C8A,REPORT-C    Опять проверка диапазона.
```

Оставшиеся функции имеют приоритет 16 десятичное. Код операции считается теперь для этих функций. Функции, которые работают со строками, требуют сброса 6 разряда, а функции, которые выдают строковые результаты, требуют сброса 7 разряда в их кодах операций.

```
                LD    B,+10               Приоритет 16 десятичное.
                ADD   A,+DC               Диапазон функций теперь
                                          +DC - +EF.
                LD    C,A                 Передача кода операций.
                CP    +DF                 Разъединить CODE, VAL и
                JR    NC,2707,S-NO-TO-$   LEN, которые работают со
                RES   6,C                 строками, для выдачи числовых
                                          результатов.
```

```
2707 S-NO-TO-$    CP    +EE                 Разъединить STR$ и CHR$,
                  JR    C,2700,S-PUSH-PO    которые работают с числами
                                            для выдачи строкового
                                            результата.
                  RES   7,C                 Отметка кода операции.
                                            Другие коды операций
                                            задали 6 и 7 биты.
```

Рассмотренные код приоритета и код операции для функций теперь заносятся в машинный стек. Посредством этого выстраивается иерархия операций.

```
270D S-PUSH-PO    PUSH  BC                  Поместить в стек коды
                  RST   0020,NEXT-CHAR      приоритета и операции
                  JP    24FF,S-LOOP-1       перед пересылкой следующей
                                            части выражения.
```

Теперь продолжаем просмотр строки. За текущим аргументом может следовать '(' или бинарный оператор, если достигнут конец выражения, затем, например, символ возврата каретки или двоеточие, разделитель или 'THEN'.

```
2712 S-CONT-2     RST   0018,GET-CHAR       Выбор текущего оператора.
2713 S-CONT-3     CP    +28                 Переход вперед, если он не
                  JR    NZ,2723,S-OPERTR    '(', что указывает на заклю-
                                            ченное в скобки выражение.
```

Если 'последнее значение' является числовым, то выражение в круглых скобках является истинным подвыражением и должно обрабатываться самостоятельно. Однако, если 'последнее выражение' является строкой, то выражение в скобках представляет элемент массива или вырезку строки. Обращение к SLICING преобразует нужным образом параметры строки.

```
                  BIT   6,(FLAGS)           Переход вперед, если
                  JR    NZ,2734,S-LOOP      рассматривается числовое
                                            выражение в скобках.
                  CALL  2A52,SLICING        Преобразование параметров
                                            'последнего значения'.
                  RST   0020,NEXT-CHAR      Пересылка для рассмотрения
                  JR    2713,S-CONT-3       следующего символа.
```

Если текущий символ действительно является бинарным оператором, то он будет задаваться кодом операции в диапазоне +C3 - +CF (16-ричное) и соответствующим кодом приоритета.

```
2723 S-OPERTR     LD    B,+00               Исходный код в BC для внесе-
                  LD    C,A                 ния в таблицу операторов.
                  LD    HL,+2795            Указатель в таблицу.
                  CALL  16DC,INDEXER        Индекс в таблицу.
                  JR    NC,2734,SLOOP       Переход вперед, если
                                            операция не обнаружена.
                  LD    C,(HL)              Получение требуемого кода
                                            из таблицы.
                  LD    HL,+26ED            Указатель в таблицу
                                            приоритетов; т.е 26ED+C3
                                            дает 27B0, как первый адрес.
                  ADD   HL,BC               Индекс в таблицу.
                  LD    B,(HL)              Выбор соответствующего
                                            приоритета.
```

Теперь вводится основной цикл этой подпрограммы. На этой стадии:

1. 'Последнее значение' на стеке калькулятора.

2. Рынок начальных приоритетов на машинном стеке ниже иерархии неизвестных размеров, функций и кодов бинарных операций. Эта иерархия может быть пустой.

3. Рассматривая пару BC, содержащую 'текущую' операцию или приоритет, которые при достижении конца выражения будут нулем приоритета.

Изначально, 'последняя' операция и приоритет выводятся из машинного стека и сравниваются с 'текущей' операцией и приоритетом.

Если 'текущий' приоритет старше, чем 'последний', то осуществляется выход из цикла, т.к. 'текущий' приоритет необходимо связать более 'плотно', чем 'последний' приоритет.

Однако, если текущий приоритет меньше, то выполняется операция, определенная как 'последняя'. 'Текущая операция' и приоритет возвращаются на машинный стек для повторного прогона в цикле. В этом случае иерархия функций и бинарных операций, которые были выстроены по очереди, рассматриваются в правильном порядке.

```
2734 S-LOOP       POP   DE                Получение 'последней
                                          операции и приоритета'.
                  LD    A,D               Приоритет переходит на
                                          регистр A.
                  CP    B                 Сравнение 'последнего
                                          с 'текущим'.
                  JR    C,2773,S-TIGHTER  Выход для ожидания аргумента.
                  AND   A                 Оба приоритета нулевые?
                  JP    Z,0018,GET-CHAR   Выход через GET-CHAR
                                          посредством чего создается
                                          'последнее значение'
                                          требуемого результата.
```

Перед выполнением 'последней операции', функция 'USR' разделяется на 'USR число' и 'USR строка' в соответствии с тем, как 6 разряд FLAGS задается или сбрасывается при занесении аргумента функции в стек в виде 'последнего значения'.

```
                  PUSH  BC                Занесение в стек
                                          'текущего значения'.
                  LD    HL,+5C3B          Это FLAGS.
                  LD    A,E               'Последня операция' сравни-
                  CP    +ED               вается с кодом USR, который
                                          будет давать 'USR число' до
                  JR    NZ,274C,S-STK-LST изменения; если нет 'USR',
                                          то переход.
                  BIT   6,(HL)            Проверка 6 разряда FLAGS.
                  JR    NZ,274C,S-STK-LST Переход, если он задан
                                          ('USR число').
                  LD    E,+99             Изменение кода 'последней'
                                          операции: смещение 19, +80
                                          для строкового ввода и
                                          числовой результат
                                          ('USR строка').
274C S-STK-LST    PUSH  DE                Занесение в стек
                                          'последнего значения'.
                  CALL  2530,SYNTAX-Z     Если проверяется синтаксис,
                  JR    Z,275B,S-SYNTEST  фактическую операцию
                                          не выполнять.
```

```
                LD    A,E               Код 'последней' операции.
                AND   +3F               Удаление 6 и 7 разрядов,
                LD    B,A               чтобы преобразовать код опе-
                                        рации в смещение калькулятора.
                RST   0028,FP-CALC      Теперь используется
                                        калькулятор.
                DEFB  +3B,fp-calc-2     Выполнение фактических
                                        операций.
                DEFB  +38,end-calc      Это сделано.
                JR    2764,S-RUNTEST    Переход вперед.
```

Важной частью проверки синтаксиса является проверка операции для того, чтобы обеспечить правильный тип 'последнего значения' для рассматриваемой операции.

```
275B S-SYNTEST  LD    A,E               Получение кода 'последней'
                                        операции.
                XOR   (FLAGS)           Этим проверяется характер
                AND   +40               'последнего значения' в
                                        соответствии с требованиями
                                        операции. При правильном
                                        синтаксисе они должны быть
                                        одинаковыми.
2761 S-RPORT-C  JP    NZ,1C8A,REPORT-C  Если синтаксис неправильный,
                                        то переход.
```

Перед переходом назад для повторного обхода цикла характер 'последнего значения' должен быть записан во FLAGS.

```
2764 S-RUNTEST  POP   DE                Прочитать код 'последней'
                                        операции.
                LD    HL,+5C3B          Это FLAGS.
                SET   6,(HL)            Допускается числовой
                                        результат.
                BIT   7,E               Переход вперед, если
                JR    NZ,2770,S-LOOPEND 'последнее значение' числовое.
                RES   6,(HL)            Строка.
2770 S-LOOPEND  POP   BC                Получить 'текущее значение' в BC.
                JR    2734,S-LOOP       Переход назад.
```

Когда 'текущая' операция плотно связывается, то 'последнее' и 'текущее' значения возвращаются на машинный стек. Однако, если 'текущая' операция требует строку в качестве операнда, то код операции изменяется для отображения этого требования.

```
2773 S-TIGHTER  PUSH  DE                'Последние' значения идут на стек.
                LD    A,C               Получение кода 'текущей' операции.
                BIT   6,(FLAGS)         Не изменяется код операции,
                JR    NZ,2790,S-NEXT    если работаете с числовым операндом.
                AND   +3F               Очистить 6 и 7 разряды.
                ADD   A,+08             Увеличить код на +08
                                        шестнадцатеричное.
                LD    C,A               Возврат кода на регистр C.
                CP    +10               Операция 'AND'?
                JR    NZ,2788,S-NOT-AND Если нет, переход.
```

```
                SET   6,C                 'AND' требует числового операнда.
                JR    2790,S-NEXT         Переход вперед.
2788 S-NOT-AND  JR    C,2761,S-RPORT-C    Операции -, *, /, ^ и OR
                                          невозможны между строками.
                CP    +17                 Операция '+'?
                JR    Z,2790,S-NEXT       Если да, переход.
                SET   7,C                 Другие операции выдают
                                          числовой результат.
2790 S-NEXT     PUSH  BC                  'Текущее' значение
                                          перешло на машинный стек.
                RST   0020,NEXT-CHAR      Рассмотреть следующий символ.
                JP    24FF,S-LOOP-1       Повторный обход цикла.
```

ТАБЛИЦА ОПЕРАТОРОВ

Ячейка	Код	Код оператора	Оператор	Ячейка	Код	Код оператора	Оператор
2795	2B	CF	+	27A3	3C	CD	<
2797	2D	C3	-	27A5	C7	C9	<=
2799	2A	C4	*	27A7	C8	CA	>=
279B	2F	C5	/	27A9	C9	CB	<>
279D	5E	C6	^	27AB	C5	C7	OR
279F	3D	CE	=	27AD	C6	C8	AND
27A1	3E	CC	>	27AF	00	Маркер конца	

ТАБЛИЦА ПРИОРИТЕТОВ

Ячейка	Приоритет	Оператор	Ячейка	Приоритет	Оператор
27B0	06	-	27B7	05	>=
27B1	08	*	27B8	05	<>
27B2	08	/	27B9	05	>
27B3	0A	^	27BA	05	<
27B4	02	OR	27BB	05	=
27B5	03	AND	27BC	06	+
27B6	05	<=			

ПОДПРОГРАММА 'SCANNING FUNCTION' ('ФУНКЦИЯ просмотра')
Эта подпрограмма вызывается о помощью 'программы просмотра FN' для обработки определенной пользователем функции, которая располагается в строке BASIC. Подпрограмму можно рассматривать в 4 этапа:

1. Во время проверки синтаксиса проверяется синтаксис оператора FN.
2. Во время выполнения строки осуществляется поиск области программы оператора DEF FN, сравниваются имена функций, пока не обнаружится совпадение, или будет сообщение об ошибке.
3. Аргументы FN обрабатываются обращением к SCANNING.
4. Сама по себе функция обрабатывается вызовом SCANNING, которая в свою очередь вызывает LOOK-VARS и обращается к подпрограмме 'STACK FUNCTION ARGUMENT'.

```
27BD S-FN-SBRN  CALL  2530,SYNTAX-Z       Если не проверен синтаксис,
                JR    NZ,27F7,SF-RUN      осуществляется переход
                                          на SF-RUN.
```

```
                  RST   0020,NEXT-CHAR      Получение первого символа
                                            имени.
                  CALL  2C8D,ALPHA          Если он не текстовый,
                  JP    NC,1C8A,REPORT-C    то сообщение об ошибке.
                  RST   0020,NEXT-CHAR      Получение следующего символа.
                  CP    +24                 Это '$'?
                  PUSH  AF                  Запись признака нуля в стек.
                  JR    NZ,27D0,SF-BRKT-1   Если он не $, то переход.
                  RST   0020,NEXT-CHAR      Но, если это так, то получе-
                                            ние следующего символа.
27D0 SF-BRKT-1    CP    +28                 Если символ не '(',
                  JR    NZ,27E6,SF-RPRT-C   сообщение об ошибке.
                  RST   0020,NEXT-CHAR      Получение следующего символа.
                  CP    +29                 Это ')'?
                  JR    Z,27E9,SF-FLAG-6    Если да, переход;
                                            аргументов нет.
27D9 SF-ARGMTS    CALL  24FB,SCANNING       В цикле вызов SCANNING для
                                            проверки синтаксиса каждого
                                            аргумента и вставки числа с
                                            плавающей точкой.
                  RST   0018,GET-CHAR       Получение символа, который
                  CP    +2C                 следует за аргументом;
                  JR    NZ,27E4,SF-BRKT-2   если это не ',', то переход
                                            (больше аргументов нет).
                  RST   0020,NEXT-CHAR      Получение первого символа
                                            следующего аргумента.
                  JR    27D9,SF-ARGMTS      Опять прокрутить цикл и
                                            рассмотреть этот аргумент.
27E4 SF-BRKT-2    CP    +29                 Текущий символ ')'?
27E6 SF-RPRT-C    JP    NZ,1C8A,REPORT-C    Если нет, то сообщение
                                            об ошибке.
27E9 SF-FLAG-6    RST   0020,NEXT-CHAR      Указать следующий символ
                                            в строке BASIC.
                  LD    HL.+5C3B            Это FLAGS: разрешает функцию
                  RES   6,(HL)              со строковым значением и
                                            сбрасывает 6 разряд в FLAGS.
                  POP   AF                  Восстанавливает признаки
                  JR    Z,27F4,SF-SYN-EN    нуля: переход, если FN
                                            действительно имеет
                                            строковое значение.
                  SET   6,(HL)              Иначе, задается 6 разряд
                                            в FLAGS.
27F4 SF-SYN-EN    JP    2712,S-CONT-2       Переход назад для продолже-
                                            ния просмотра строки.
```

2. Во время выполнения строки, поиск должен сначала осуществляться для оператора DEF FN.

```
27F7 SF-RUN       RST   0020,NEXT-CHAR      Прочитать первый символ
                                            имени.
                  AND   +DF                 Сброс 5 разряда для
                                            верхнего регистра.
                  LD    B,A                 Скопировать имя в B.
                  RST   0020,NEXT-CHAR      Прочитать следующий символ.
                  SUB   +24                 Вычесть 24 (16-ричноеое),
                                            код для '$'.
                  LD    C,A                 Скопировать результат в C
                                            (ноль для строки, не ноль
                                            для числовой функции).
```

```
                JR    NZ,2802,SF-ARGMT1   Если не ноль, переход:
                                          числовая функция.
                RST   0020,NEXT-CHAR      Прочитать следующий
                                          символ, '('.
2802 SF-ARGMT1  RST   0020,NEXT-CHAR      Прочитать первый символ
                                          первого аргумента.
                PUSH  HL                  Записать его указатель
                                          в стек.
                LD    HL,(PROG)           Указать начало программы.
                DEC   HL                  Переход назад на одну
                                          ячейку.
2808 SF-FND-DF  LD    DE,+00CE            Поиск будет для 'DEF FN'.
                PUSH  BC                  Записать имя и 'статус
                                          строки'.
                CALL  1D86,LOOK-PROG      Теперь поиск программы.
                POP   BC                  Восстановить имя и статус.
                JR    NC,2814,SF-CP-DEF   Переход, если обнаружен
                                          оператор DEF FN.
```

Сообщение P - 'FN без DEF'.

```
2812 REPORT-P   RST   0008,ERROR-1        Вызов программы
                DEFB  +18                 обработки ошибок.
```

Когда обнаружен оператор DEF FN, то сравниваются имена и статусы двух функций: если они не сопоставляются, то поиск продолжается.

```
2814 SF-CP-DEF  PUSH  HL                  В случае возобновления
                                          поиска записать указатель
                                          символа DEF FN.
                CALL  28AB,FN-SKPOVR      Прочитать имя функции
                                          DEF FN.
                AND   +DF                 Сброс 5 разряда для
                                          верхнего регистра.
                CP    B                   Отождествляется имя FN?
                JR    NZ,2825,SF-NOT-FD   Переход, если нет.
                CALL  28AB,FN-SKPOVR      Прочитать следующий символ
                                          в DEF FN.
                SUB   +24                 Вычесть 24, (16-ричное),
                                          код для '$'.
                CP    C                   Сравнить статус
                                          со статусом FN.
                JR    Z,2831,SF-VALUES    Если полное совпадение,
                                          то переход.
2825 SF-NOT-FD  POP   HL                  Восстановить указатель
                                          'DEF FN'.
                DEC   HL                  Шагнуть назад
                                          на одну ячейку.
                LD    DE,+0200            Для обнаружения конца опера-
                PUSH  BC                  тора DEF FN использовать
                CALL  198B,EACH-STMT      программу поиска, подготовка
                POP   BC                  к следующему поиску, тем
                                          временем запись имени
                                          и статуса.
                JR    2808,SF-FND-DF      Переход назад для
                                          дальнейшего поиска.
```

3. Теперь найдем правильный оператор DEF FN. Аргументы FN будут отработаны повторным вызовом SCANNING, а их 5-байтные значения (или

параметры для строк) будут вставлены в оператор DEF FN на места, созданные при проверке синтаксиса. Регистровая пара HL будет использована для указания на оператор DBF FN (вызывая FN-SKPOVR, если это необходимо), пока CH-ADD указывает на оператор FN (вызывая RST 0020, NEXT-CHAR, если это необходимо).

```
2831 SF-VALUES    AND   A                  Если HL теперь указывает на
                  CALL  Z,28AB,FN-SKPOVR   $, то перемещение к '('.
                  POP   DE                 Убрать указатель 'DEF FN'.
                  POP   DE                 Получить указатель первого
                  LD    (CH-ADD),DE        аргумента FN и скопировать
                                           его в CH-ADD.
                  CALL  28AB,FN-SKPOVR     Теперь перемещение к '('.
                  PUSH  HL                 Записать этот указатель
                                           на стек.
                  CP    +29                Указывает на ')'?
                  JR    Z,2885,SF-R-BR-2   Если да, переход:
                                           у FN нет перехода.
2843 SF-ARG-LP    INC   HL                 Указывает на следующий код.
                  LD    A,(HL)             Поместить код в A.
                  CP    +0E                Это код 'числового маркера',
                                           0E, шестнадцатеричное?
                  LD    D,+40              Задать 6 разряд D для
                                           числового аргумента.
                  JR    Z,2852,SF-ARG-VL   Переход на ноль:
                                           числовой аргумент.
                  DEC   HL                 HL указывает на символ $,
                  CALL  28AB,FN-SKPOVR     (например, не на код
                                           управления).
                  INC   HL                 HL теперь указывает на
                                           'маркер числа'.
                  LD    D,+00              Сброшен 6 разряд D:
                                           строковый аргумент.
2852 SF-ARG-VL    INC   HL                 Указать первый из 5 байтов
                                           в DEF FN.
                  PUSH  HL                 Запись этого указателя
                                           на стек.
                  PUSH  DE                 Запись 'статуса строки'
                                           аргумента.
                  CALL  24FB,SCANNING      Теперь обработка аргумента.
                  POP   AF                 Получить в A признак
                                           число/строка.
                  XOR   (FLAGS)            Проверка его 6 разряда
                  AND   +40                с результатом SCANNING.
                  JR    NZ,288B,REPORT-Q   Если они не отождествляются,
                                           выдать сообщение Q.
                  POP   HL                 Получить указатель первого
                  EX    DE,HL              из 5 мест в DEF FN в DE.
                  LD    HL,(STKEND)        Указать HL в STKEND.
                  LD    BC,+0005           BC будет считать 5
                                           перемещенных байт.
                  SBC   HL,BC              Сначала уменьшить STKEND
                  LD    (STKEND),HL        на 5, удаляется 'последнее
                                           значение' из стека.
                  LDIR                     Копируется 5 байт
                                           на места в DBF FN.
                  EX    DE,HL              Указывает HL на следующем
                                           коде.
                  DEC   HL                 HL указывает на
```

```
                CALL  28AB,FN-SKPOVR      символ после 5 байт.
                CP    +29                 Это ')'?
                JR    Z,2885,SF-R-BR-2    Если да, переход: в операторе
                                          DEF FN нет больше
                                          аргументов.
                PUSH  HL                  Это ',': записать ее
                                          указатель.
                RST   0018,GET-CHAR       Прочитать символ после
                                          последнего аргумента,
                                          который обрабатывается из FN.
                CP    +2C                 Если это не ',', то
                JR    NZ,288B,REPORT-Q    переход: несовпадение
                                          аргументов FN и DEF FN.
                RST   0020,NEXT-CHAR      Указать CH-ADD на
                                          следующий аргумент FN.
                POP   HL                  Указать HL на '.' опять
                                          в DEF FN.
                CALL  28AB,FN-SKPOVR      Переместить HL на следующий
                                          аргумент в DEF FN.
                JR    2843,SF-ARG-LP      Переход назад, чтобы
                                          рассмотреть этот аргумент.
2885 SF-R-BR-2  PUSH  HL                  Записать указатель ')'
                                          в DEF FN.
                RST   0018,GET-CHAR       Прочитать символ после
                                          последнего аргумента в FN.
                CP    +29                 Это ')'?
                JR    Z,288D,SF-VALUE     Если да, переход для
                                          вычисления функции: но если
                                          нет, то сообщение Q.
```

Сообщение Q – 'Ошибка параметра'.

```
288B REPORT-Q   RST   0008,ERROR-1        Вызов программы
                DEFB  +19                 обработки ошибки.
```

4. И, наконец, сама функция вычисляется вызовом SCANNING, после первой установки DEFADD, которое содержит адреса аргументов, появившихся в операторе DEF FN. Этим обеспечивается, что LOOK-VARS, при вызове ее SCANNING, будет искать эти аргументы для требуемых значений, до поиска области переменных.

```
288D SF-VALUE   POP   DE                 Восстановить указатель
                                          ')' в DEF FN.
                EX    DE,HL               Получить указатель в HL.
                LD    (CH-ADD),HL         Вставить его в CH-ADD.
                LD    HL,(DEFADD)         Прочитать старое значение
                                          в DEFADD.
                EX    (SP),HL             Поместить в стек и
                LD    (DEFADD),HL         получит начальный адрес
                                          области аргументов DEF FN
                                          в DEFADD.
                PUSH  DE                  Записать адрес ')' в FN.
                RST   0020,NEXT-CHAR      Переместить CH-ADD за ')',
                RST   0020,NEXT-CHAR      а '-' на начало выражения
                                          для функции в DEF FN.
                CALL  24FB,SCANNING       Теперь вычислить функцию.
                POP   HL                  Восстановить адрес
                                          ')' в FN.
                LD    (CH-ADD),HL         Запомнить его в CH-ADD.
```

```
                POP   HL                Восстановить исходное
                                        значение DEFADD.
                LD    (DEFADD),HL       Поместить его обратно
                                        в DEFADD.
                RST   0020,NEXT-CHAR    Прочитать следующий символ
                                        в строке BASIC.
                JP    2712,S-CONT-2     Переход назад для
                                        продолжения просмотра.
```

ПОДПРОГРАММА 'FUNCTION SKIPOVER' ('Прогон функции')
Эта подпрограмма используется FN и STK-F-ARG для перемещения HL по оператору DEF FN пока не готова CH-ADD, т.к. она указывает на оператор FN.

```
28AB FN-SKPOVR  INC   HL                Указывает на следующий код
                                        в операторе.
                LD    A,(HL)            Копирование кода в A.
                CP    +21               Переход назад, чтобы пропус-
                JR    C,28AB,FN-SKPOVR  тить его, если это код
                                        сравнения или пробел.
                RET                     Окончание.
```

ПОДПРОГРАММА 'LOOK-VARS'
Эта подпрограмма вызывается а том случае, когда требуется поиск области переменных или аргументов оператора DEF FN. Подпрограмма вводится с системной переменной CH-ADD, указывающей на первую букву в имени переменной, ячейка которой найдена. Имя будет в программной или рабочей области. Сначала подпрограмма создает байт дискриминатора в регистре C, который основывается на первой букве имени переменной. 5 и 6 разряды этого байта отражают тип обрабатываемой переменной.
Регистр B используется как регистр разрядов и содержит признаки.

```
28B2 LOOK-VARS  SET   6,(FLAGS)         Допустим, числовая переменная.
                RST   0018,GET-CHAR     Прочитайте в A первый символ.
                CALL  2C8D,ALPHA        Он текстовый?
                JP    NC,1C8A,REPORT-C  Если нет, то сообщение
                                        об ошибке.
                PUSH  HL                Запишите указатель первой
                                        буквы.
                AND   +1F               Передача 0-4 разрядов буквы
                LD    C,A               в регистр C; разряды 5 и 7
                                        всегда сбрасываются.
                RST   0020,NEXT-CHAR    Прочитать 2 символ в A.
                PUSH  HL                Опять запишите указатель.
                CP    +28               Второй символ ')'?
                JR    Z,28EF,V-RUN/SYN  Разделите массивы чисел.
                SET   6,C               Задать 6 разряд.
                CP    +24               Второй символ '$'?
                JR    Z,28DE,V-STR-VAR  Разделение всех строк.
                SET   5,C               Задать 5 разряд.
                CALL  2C88,ALPHANUM     Если у имени переменной
                JR    NC,28E3,V-TEST-FN только один символ,
                                        переход вперед.
```

Теперь найдем последний символ имени, которое состоит из нескольких символов.

```
28D4 V-CHAR       CALL  2C88,ALPHANUM       Символ алфавитно-цифровой?
                  JR    NC,28EF,V-RUN/SYN   Выход из цикла, если
                                            найден конец имени.
                  RES   6,C                 Отметить байт дискримина-
                                            тора.
                  RST   0020,NEXT-CHAR      Прочитать следующий символ.
                  JR    28D4,V-CHAR         Возврат для его проверки.
```

Простые строки и массивы строк требуют сброса 6 разряда FLAGS.

```
28DE V-STR-VAR    RST   0020,NEXT-CHAR      Шагнуть CH-ADD за '$'.
                  RES   6,(FLAGS)           Сброс 6 разряда для
                                            обозначения строки.
```

Если DEFADD-hi не ноль, это означает, что 'функция' ('FN') вычислена, и, если во 'время выполнения', то будет организован поиск аргументов в операторе DEF FN.

```
28E3 V-TEST-FN    LD    A,(DEFADD-hi)       Ноль ?
                  AND   A
                  JR    Z,28EF,V-RUN/SYN    Если да, переход вперед.
                  CALL  2530,SYNTAX-Z       Во 'время выполнения'?
                  JP    NZ,2951,STK-F-ARG   Бели да, то переход вперед
                                            для поиска оператора DEF FN.
```

В противном случае (или, если не обнаружена переменная в операторе DEF FN) поиска области переменных не будет, пока не проверится синтаксис.

```
28EF V-RUN/SYN    LD    B,C                 Скопировать байт
                                            дискриминатора в регистр B.
                  CALL  2530,SYNTAX-Z       Переход вперед,
                  JR    NZ,28FD,V-RUN       если во 'время выполнения'.
                  LD    A,C                 Переместить дискриминатор
                                            в A.
                  AND   +E0                 Удалить часть кода символа.
                  SET   7,A                 Показывает синтаксис,
                                            задавая 7 разряд.
                  LD    C,A                 Восстановление дискримина-
                                            тора.
                  JR    2934,V-SYNTAX       Для продолжений - переход
                                            вперед.
```

Выполнена строка BASIC для осуществления поиска области переменных.

```
28FD V-RUN        LD    HL,(VARS)           Считать указатель VARS.
```

Теперь ввод цикла для просмотра имен существующих переменных.

```
2900 V-EACH       LD    A,(HL)              Первая буква каждой
                                            существующей переменной.
                  AND   +7F                 Сопоставления на 0-6
                                            разрядах.
                  JR    Z,2932,V-80-BYTE    Переход при достижении
                                            '80 байта'.
                  CP    C                   Фактическое сравнение.
                  JR    NZ,292A,V-NEXT      Переход вперед, если не
                                            отождествлен первый символ.
                  RLA                       Сдвинуть A влево, а затем
                  ADD   A,A                 удвоить для проверки
                                            битов 5 и 6.
```

```
                JP    P,293F,V-FOUND-2    Строки и переменные
                                          массивов.
                JR    C,293F,V-FOUND-2    Простые числовые и FOR-NEXT
                                          переменные.
```

Длинные имена требуют полного отождествления.

```
                POP   DE                  Взять копию указателя
                PUSH  DE                  второго символа.
                PUSH  HL                  Записать указатель первой
                                          буквы.
2912 V-MATCHES  INC   HL                  Рассмотреть следующий символ.
2913 V-SPACES   LD    A,(DE)              По очереди выбрать каждый
                                          символ.
                INC   DE                  Указать следующий символ.
                CP    +20                 Символ 'пробел'?
                JR    Z,2913,V-SPACES     Проигнорировать пробелы.
                OR    +20                 Задать 5 разряд так, чтобы
                                          сравнить буквы верхнего и
                                          нижнего регистров.
                CP    (HL)                Сделать сравнение.
                JR    Z,2912,V-MATCHES    Переход назад для другого
                                          символа, если он
                                          сравнивается.
                OR    +80                 Будет он сравниваться
                                          с заданным 7 разрядом.
                CP    (HL)                Попытка сделать это.
                JR    NZ,2929,V-GET-PTR   Переход вперед, если
                                          'последний символ'
                                          не отождествлен.
                LD    A,(DE)              Проверить перед
                CALL  2C88,ALPHANUM       переходом вперед, что
                JR    NC,293E,V-FOUND-1   достигнут конец имени.
```

Во всех случаях, где имена не отождествляются, должно быть сделано так, чтобы регистровая пара HL указывала на следующую переменную в области переменных.

```
2929 V-GET-PTR  POP   HL                  Выбор указателя.
292A V-NEXT     PUSH  BC                  Краткая запись B и C.
                CALL  19B8,NEXT-ONE       Сделано DE для указания
                                          следующей переменной.
                EX    DE,HL               Переключение двух указателей.
                POP   BC                  Получить B и C назад.
                JR    2900,V-EACH         Повторный обход цикла.
```

Переход сюда, если не обнаружен вход с правильным именем.

```
2932 V-80-BYTE  SET   7,B                 Сигнал 'переменная
                                          не обнаружена'.
```

Переход сюда, если проверяется синтаксис.

```
2934 V-SYNTAX   POP   DE                  Убрать указатель второго
                                          символа.
                RST   0018,GET-CHAR       Выбрать текущий символ.
                CP    +28                 Это '('?
                JR    Z,2943,V-PASS       Переход вперед.
```

```
                  SET   5,B               Обозначает, что не обрабаты-
                  JR    294B,V-END        вается массив, и переход
                                          вперед.
```

Переход сюда, если обнаружен вход с правильным именем.

```
293E V-FOUND-1    POP   DE                Отбросить записанный
                                          указатель переменной.
293F V-FOUND-2    POP   DE                Отбросить указатель 2-го
                                          символа.
                  POP   DE                Отбросить указатель
                                          первой буквы.
                  PUSH  HL                Записать указатель
                                          'последней' буквы.
                  RST   0018,GET-CHAR     Выбор текущего символа.
```

Если отождествленное имя переменной имеет больше чем одну букву, то другие символы должны быть пропущены.

Примечание: Это делается в V-CHAR.

```
2943 V-PASS       CALL  2C88,ALPHANUM     Алфавитно-цифровой?
                  JR    NC,294B,V-END     Переход при обнаружении
                                          конца имени.
                  RST   0020,NEXT-CHAR    Выбор следующего символа.
                  JR    2943,V-PASS       Возврат и его проверка.
```

Теперь задаются параметры выхода.

```
294B V-END        POP   HL                HL содержит указатель буквы
                                          короткого имени или 'послед-
                                          ний' символ длинного имени.
                  RL    B                 Сдвиг всего регистра.
                  BIT   6,B               Определение состояния
                                          6 разряда.
                  RET                     Окончание.
```

Параметры выхода для подпрограммы могут быть организованы следующим образом: Системная переменная CH-ADD указывает на первую ячейку после имени переменной при ее появлении в строке BASIC.

Когда переменная не обнаружена:
1. Задается признак переноса.
2. Признак нуля задается только когда поиск осуществляется для переменной массива.
3. Регистровая пара HL указывает первую букву имени переменной после ее появления в строке BASIC.

Когда переменная обнаружена:
1. Сбрасывается признак переноса.
2. Признак нуля задается и для переменных строки, и для переменных массива.
3. Регистровая пара HL указывает букву 'короткого' имени или последующий символ 'длинного' имени, существующего сообщения, который обнаружен в области переменных.

Во всех случаях разряды 5 и 6 регистра C обозначают тип обработанной переменной. 7 разряд является дополнением признака SYNTAX/RUN. Но только когда подпрограмма используется в 'программе', разряды 0-4 будут содержать код буквы переменной.

Во время проверки синтаксиса возврат всегда происходит со сбросом признака переноса. Признак нуля задается для массивов и сбрасывается для всех других переменных, исключая случай, когда имя строки, за которым неправильно следует '$', задает признак нуля и в случае SAVE 'имя' DATA a$(), тоже пропускает синтаксис.

ПОДПРОГРАММА 'STACK FUNCTION ARGUMENT' ('Занесение в стек аргумента функций')

Эта подпрограмма вызывается LOOK-VARS, когда DEFADD-hi не ноль, для осуществления поиска области аргументов оператора DEF FN, перед поиском в области переменных. Если переменная найдена в операторе DEF FN, то параметры строковой переменной заносятся в стек и задается сигнал о том, что нет необходимости вызывать STK/VAR. Но оно остаётся в SCANNING, для занесения в стек значения числовой переменной в 26DA обычным путем.

```
2951 STK-F-ARG    LD    HL,(DEFADD)         Указать первый символ
                  LD    A,(HL)              в области аргументов
                                            и запись его в A.
                  CP    +29                 Это ')'?
                  JP    Z,28EF,V-RUN/SYN    Переход для поиска области
                                            переменных.
295A SFA-LOOP     LD    A,(HL)              Прочитать следующий
                                            аргумент в цикле.
                  OR    +60                 Задать 5 и 6 разряды,
                  LD    B,A                 подразумевающие простую
                                            числовую переменную;
                                            скопировать в B.
                  INC   HL                  Указать следующий код.
                  LD    A,(HL)              Занести его в регистр A.
                  CP    +0E                 Это код 'числового маркера',
                                            0E шестнадцатеричный?
                  JR    Z,296B,SFA-CP-VR    Если да, переход: числовая
                                            переменная.
                  DEC   HL                  HL указывает, что символ
                  CALL  28AB,FN-SKPOVR      не пробел или код
                                            управления.
                  INC   HL                  HL теперь указывает
                                            'числовой маркер'.
                  RES   5,B                 Сброс 5 разряда B:
                                            строковая переменная.
296B SFA-CP-VR    LD    A,B                 Прочитать имя переменной в A.
                  CP    C                   Это то, что мы ищем?
                  JR    Z,2981,SFA-MATCH    Если да, переход.
                  INC   HL                  Теперь пропустить
                  INC   HL                  5 байт числа с плавающей
                  INC   HL                  точкой или строковых
                  INC   HL                  параметров, чтобы прочитать
                  INC   HLследующий аргумент.
                  CALL  28AB,FN-SKPOVR      Перейти к следующему символу.
                  CP    +29                 Это ')'?
                  JP    Z,28EF,V-RUN/SYN    Если да, переход для поиска
                                            области переменных.
                  CALL  28AB,FN-SKPOVR      Указать следующий аргумент.
                  JR    295A,SFA-LOOP       Перейти назад и рассмотреть
                                            его.
```

Обнаружено совпадение. Параметры строковой переменной заносятся в стек, избегая необходимости вызвать подпрограмму STK-VAR.

```
2981 SFA-MATCH    BIT   5,C               Тест для числовой переменной.
                  JR    NZ,2991,SFA-END   Переход, если переменная
                                          числовая; SCANNING поместит
                                          ее в стек.
                  INC   HL                Указать первый из 5 байт,
                                          заносимых в стек.
                  LD    DE,(STKEND)       Указать DE на STKEND.
                  CALL  33C0,MOVE-FP      Занести в стек 5 байт.
                  EX    DE,HL             Указать HL на новую позицию
                  LD    (STKEND),HL       в STKEND и сбросить
                                          системную переменную.
2991 SFA-END      POP   DE                Отбросить указатели
                  POP   DE                LOOK-VARS (указатели
                                          1 и 2 символа).
                  XOR   A                 Возврат из поиска со
                  INC   A                 сбросом двух признаков -
                                          нуля и переноса - сигнализи-
                                          руя, STK-VAR не требуется.
                  RET                     Окончание.
```

ПОДПРОГРАММА 'STK-VAR'
Эта подпрограмма обычно используется или для обнаружения параметров, которые определяют существующую строковую входную переменную, или для возврата в регистровую пару HL базового адреса конкретного элемента или массива чисел. Когда подпрограмма вызвана из DIM, то она только проверяет синтаксис оператора BASIC.

Отметим, что параметры, которые определяют строку, могут изменяться с помощью SLICING, в случае, если это задано.

Изначально, регистры A и B очищаются, а 7 разряд регистра C проверяется для определения, проверен ли синтаксис.

```
2996 STK-VAR      XOR   A                 Сбросить признак массива.
                  LD    B,A               Очистить регистр B.
                  BIT   7,C               Если синтаксис проверен,
                  JR    NZ,29E7,SV-COUNT  переход вперед.
```

Далее 'простая' строка выделяется из переменной.

```
                  BIT   7,(HL)            Если работа с переменной
                  JR    NZ,29AE,SV-ARRAYS массива, то переход вперед.
```

Параметры для простой строки легко обнаружить.

```
                  INC   A                 Сигнал 'простая строка'.
29A1 SV-SIMPLE$   INC   HL                Перемещение по входному
                                          сообщению.
                  LD    C,(HL)            Подбор младшего счетчика
                                          длины.
                  INC   HL                Продвижение указателя.
                  LD    B,(HL)            Подбор старшего счетчика
                  INC   HL                длины.
                  EX    DE,HL             Передача указателя
                                          в реальную строку.
                  CALL  2AB2,STK-STORE    Передача этих параметров
                                          в стек калькулятора.
                  RST   0018,GET-CHAR     Выбор текущего символа
                  JP    2A49,SV-SLICE?    и переход вперед, если
                                          требуется 'вырезка'.
```

Базовый адрес элемента в массиве теперь найден. Выбирается 'число размерностей'.

```
29AE SV-ARRAYS    INC   HL                Шаг за байты длины.
                  INC   HL
                  INC   HL
                  LD    B,(HL)            Отобрать число
                                          'размерностей'.
                  BIT   6,C               Если отрабатывается массив
                  JR    Z,29C0,SV-PTR     чисел, переход вперед.
```

Если массив строк имеет свое 'число размерностей', равное 1, то такой массив можно обрабатывать как простую строку.

```
                  DEC   B                 Понизить 'число размернос-
                  JR    Z,29A1,SV-SIMPLE$ тей' и перейти вперед, если
                                          число теперь равно нулю.
```

Далее делается проверка, которая гарантирует, что в строке BASIC за переменной следует индекс.

```
                  EX    DE,HL             Записать указатель в DE.
                  RST   0018,GET-CHAR     Прочитать текущий символ.
                  CP    +28               Это '('?
                  JR    NZ,2A20,REPORT-3  Если нет, то сообщение
                                          об ошибке.
                  EX    DE,HL             Восстановить указатель.
```

И для числового массива, и для массива строк указатели переменной передаются в регистровую пару DE перед вычислением индекса.

```
29C0 SV-PTR       EX    DE,HL             Перемещение указателя к DE.
                  JR    29E7,SV-COUNT     Переход вперед.
```

Последующий цикл используется для нахождения параметров заданного элемента внутри массива. Цикл вводится на средней точке – SV-COUNT-, где обсчет элемента задается с нуля.
Обращение к циклу происходит 'B' раз, что для числового массива равно числу используемых размерностей, но для массива строк 'B' на единицу меньше числа используемых размерностей, т.к. последний импульс используется для определения 'вырезки' строки.

```
29C3 SV-COMMA     PUSH  HL                Запись счетчика.
                  RST   0018,GET-CHAR     Прочитать следующий символ.
                  POP   HL                Восстановить счетчик.
                  CP    +2C               Текущий символ ','?
                  JR    Z,29EA,SV-LOOP    Переход вперед для рассмот-
                                          рения другого индекса.
                  BIT   7,C               Если выполнена строка,
                  JR    Z,2A20,REPORT-3   в ней ошибка.
                  BIT   6,C               Если работа с массивом
                  JR    NZ,29D8,SV-CLOSE  строк, то переход вперед.
                  CP    +29               Текущий символ ')'?
                  JR    NZ,2A12,SV-RPT-C  Если нет, то сообщение
                                          об ошибке.
                  RST   0020,NEXT-CHAR    Продвинуть CD-ADD.
                  RET                     Если синтаксис верен,
                                          то возврат.
```

Для массива строк текущий индекс может представлять 'вырезку', или индекс для 'вырезки' может еще быть текущим в строке BASIC.

```
29D8 SV-CLOSE     CP    +29                Текущий символ ')'?
                  JR    Z,2A48,SV-DIM      Переход вперед и проверка:
                                           есть ли другой индекс.
                  CP    +CC                Текущий символ 'TO'?
                  JR    NZ,2A12,SV-RPT-C   Не должно быть иначе.
29E0 SV-CH-ADD    RST   0018,GET-CHAR      Прочитать текущий символ.
                  DEC   HL                 Указать предшествующий
                  LD    (CH-ADD),HL        символ и задать CH-ADD.
                  JR    2A45,SV-SLICE      Вычисление 'вырезки'.
```

Здесь вход в цикл.

```
29E7 SV-COUNT     LD    HL,+0000           Задать счетчик в 0.
29EA SV-LOOP      PUSH  HL                 Коротко записать счетчик.
                  RST   0020,NEXT-CHAR     Продвижение CH-ADD.
                  POP   HL                 Восстановить счетчик.
                  LD    A,C                Выбрать байт дискриминатора.
                  CP    +C0                Переход, пока не проверится
                  JR    NZ,29FB,SV-MULT    синтаксис для массива строк.
                  RST   0018,GET-CHAR      Прочитать текущий символ.
                  CP    +29                Это ')'?
                  JR    Z,2A48,SV-DIM      По окончании просчета
                                           элементов, переход вперед.
                  CP    +CC                Это 'TO'?
                  JR    Z,29E0,SV-CH-ADD   Переход назад, если работа
                                           с 'вырезкой'.
29FB SV-MULT      PUSH  BC                 Запись счетчика числа
                                           размерностей и байта
                                           дискриминатора.
                  PUSH  HL                 Запись счетчика элементов.
                  CALL  2AEE,DE,(DE+1)     Получить в DE величину
                                           размерностей.
                  EX    (SP),HL            Счетчик смещается к HL
                                           и указатель переменных
                                           заносится в стек.
                  EX    DE,HL              Счетчик смещается к DE, а
                                           величина размерности к HL.
                  CALL  2ACC,INT-EXP1      Вычисление следующего
                                           индекса.
                  JR    C,2A20,REPORT-3    Выдача ошибок, если вне
                                           диапазона.
                  DEC   BC                 Результат вычислений умень-
                                           шается, т.к. счетчик считает
                                           элементы, появляющиеся перед
                                           заданным элементом.
                  CALL  2AF4,GET-HL*DE     Умножение счетчика
                                           на величину размерности.
                  ADD   HL,BC              Добавить результат
                                           'INT-EXP1' к текущему
                                           счетчику.
                  POP   DE                 Выбор указателя переменной.
                  POP   BC                 Выбор числа размерностей
                                           и байта дискриминатора.
                  DJNZ  29C3,SV-COMMA      Работать в цикле, пока 'B'
                                           станет равным нулю.
```

Признак SYNTAX/RUN проверяется до отделения массива строк от массива чисел.

```
                  BIT   7,C                 Сообщение об ошибке,
2A12 SV-RPT-C     JR    NZ,2A7A,SL-RPT-C    если в этой точке идет
                                            проверка синтаксиса.
                  PUSH  HL                  Запись счетчика.
                  BIT    6,C                При обработке массива
                  JR    NZ,2A2C,SV-ELEM$    строк - переход вперед.
```

При работе с массивом чисел текущий символ должен быть ')'.

```
                  LD    B,D                 Передача указателя переменной
                  LD    C,E                 в регистровую пару BC.
                  RST   0018,GET-CHAR       Выбор текущего символа.
                  CP    +29                 Это ')'?
                  JR    Z,2A22,SV-NUMBER    Переход за сообщение об
                                            ошибке, если оно не нужно.
```

Сообщение 3 - 'Индекс вне диапазона'.

```
2A20 REPORT-3     RST   0008,ERROR-1        Вызов программы обработки
                  DEFB  +02                 ошибки.
```

Теперь можно вычислить адрес ячейки перед фактической формой с плавающей точкой.

```
2A22 SV-NUMBER    RST   0020,NEXT-CHAR      Продвижение CH-ADD.
                  POP   HL                  Выбор счетчика.
                  LD    DE,+0005            5 байт в каждом элементе
                                            массива.
                  CALL  2AF4,GET-HL*DE      Вычисление общего количества
                                            байт перед требуемым
                                            элементом.
                  ADD   HL,BC               HL указывает ячейку перед
                                            требуемым элементом.
                  RET                       Возврат с этим адресом.
```

При работе с массивом строк длина элемента задается последней 'величиной размерности'. Вычисляются соответствующие параметры, а затем передаются в стек калькулятора.

```
2A2C SV-ELEM$     CALL  2AEE,DE,(DE+1)      Выбор последней величины
                                            размерности.
                  EX    (SP),HL             Указатель переменной
                                            в стек, а счетчик в HL.
                  CALL  2AF4,GET-HL*DE      Умножение счетчика на
                                            величину размерности.
                  POP   BC                  Выбор указателя переменной.
                  ADD   HL,BC               Это задает HL, указывающее
                                            ячейку перед строкой.
                  INC   HL                  Итак, указание 'фактического
                                            начала'.
                  LD    B,D                 Передача последней величины
                  LD    C,E                 размерности в BC для
                                            формирования длины.
                  EX    DE,HL               Перемещение 'начала' в DE.
                  CALL  2AB1,STK-ST-0       Передача этих параметров
                                            в стек калькулятора. Первый
```

```
                                    параметр является 0, обозна-
                                    чающим строку из 'массива
                                    строк' и, отсюда, существую-
                                    щий вход на сообщение
                                    не восстанавливается.
```

Существуют три возможные формы последнего индекса. Первая иллюстрируется как A$(2,4 TO 8), вторая - A$(2)(4 TO 8), и третья - A$(2), которая является формой по умолчанию и обозначает, что требуется строка целиком.

```
                RST   0018,GET-CHAR       Прочитать текущий символ.
                CP    +29                 Это ')'?
                JR    Z,2A48,SV-DIM       Если да, переход.
                CP    +2C                 Это ','?
                JR    NZ,2A20,REPORT-3    Если нет, сообщение
                                          об ошибке.
2A45 SV-SLICE   CALL  2A52,SLICING        Используется SLICING для
                                          изменения набора параметров.
2A48 SV-DIM     RST   0020,NEXT-CHAR      Выбор следующего символа.
2A49 SV-SLICE?  CP    +28                 Это '('?
                JR    Z,2A45,SV-SLICE     Переход назад, если
                                          рассматривается 'вырезка'.
```

После рассмотрения последнего индекса делается возврат.

```
                RES   6,(FLAGS)           Сигнал - результат строки.
                RET                       Возврат с параметрами
                                          требуемой строки, формирую-
                                          щими последнее значение на
                                          стеке калькулятора.
```

ПОДПРОГРАММА 'SLICING' ('Вырезание')
С использованием этой подпрограммы можно разрезать строку. Подпрограмма вводится с параметрами строки, представленными не вершине стека калькулятора и в регистрах A,B,C,D и E. Изначально проверяется признак SYNTAX/RUN, а параметры строки, если строка обрабатывается, только выбираются.

```
2A52 SLICING    CALL  2530,SYNTAX-Z     Проверка признака.
                CALL  NZ,2BF1,STK-FETCH  Вынос из стека параметров
                                         во 'время выполнения'.
```

Рассматривается возможность 'вырезки' при '()'.

```
                RST   0020,NEXT-CHAR      Прочитать следующий символ.
                CP    +29                 Это ')'?
                JR    Z,2AAD,SL-STORE     Если да, переход вперед.
```

Перед продолжением с регистрами делают следующее:

```
                PUSH  DE                  'Начало' пошло в машинный
                                          стек.
                XOR   A                   Регистр A очищен
                PUSH  AF                  и записан.
                PUSH  BC                  Записана 'длина'.
                LD    DE,+0001            Предположим, что 'вырезка'
                                          начинается с первого
```

```
                                        символа.
                  RST   0018,GET-CHAR   Прочитать первый символ.
                  POP   HL              Передать 'длину' в HL.
```

Теперь вычисляется первый параметр 'вырезки'.

```
                  CP    +CC             Текущий символ 'TO'?
                  JR    Z,2A81,SL-SECOND Если переход, то первый
                                        параметр по умолчанию
                                        будет '1'.
                  POP   AF              На этой стадии A - ноль.
                  CALL  2ACD,INT-EXP2   Создано BC, чтобы содержать
                                        первый параметр. A будет
                                        содержать +FF, если была
                                        ошибка - 'вне диапазона'.
                  PUSH  AF              Запись любого значения.
                  LD    D,B             Передача первого параметра
                  LD    E,C             в DE.
                  PUSH  HL              Запись 'длины'.
                  RST   0018,GET-CHAR   Прочитать текущий символ.
                  POP   HL              Восстановить 'длину'.
                  CP    +CC             Текущий символ 'TO'?
                  JR    Z,2A81,SL-SECOND Если да, то переход вперед,
                                        чтобы рассмотреть второй
                                        параметр.
                  CP    +29             В противном случае показать,
2A7A SL-RPT-C     JP    NZ,1C8A,REPORT-C что там закрывающая скобка.
```

В этой точке идентифицирована 'вырезка' одного символа, например, A$(4).

```
                  LD    H,D             Последний символ 'вырезки'
                  LD    L,E             является также первым
                                        символом.
                  JR    2A94,SL-DEFINE  Переход вперед.
```

Теперь вычисляется второй параметр 'вырезки'.

```
2A81 SL-SECOND    PUSH  HL              Запись 'длины'.
                  RST   0020,NEXT-CHAR  Прочитать следующий символ.
                  POP   HL              Восстановить 'длину'.
                  CP    +29             Текущий символ ')'?
                  JR    Z,2A94,SL-DEFINE Переход, если это не второй
                                        параметр.
                  POP   AF              Если первый параметр в
                                        диапазоне, то A будет
                                        содержать ноль, иначе +FF.
                  CALL  2ACD,INT-EXP2   BC содержит второй параметр.
                  PUSH  AF              Запись 'регистра ошибок'.
                  RST   0018,GET-CHAR   Прочитать текущий символ.
                  LD    H,B             Передать результат, получен-
                  LD    L,C             ный из INT-EXP2 в
                                        регистровую пару HL.
                  CP    +29             Проверка, что теперь
                  JR    NZ,2A7A,SL-RPT-C есть закрывающая скобка.
```

Теперь определяется 'новый' параметр.

```
2A94 SL-DEFINE    POP   AF              Выбор 'регистра ошибок'.
                  EX    (SP),HL         Второй параметр идет в стек,
```

```
                                        а 'начало' идет в HL.
                ADD   HL,DE             Первый параметр добавляется
                                        к 'началу'.
                DEC   HL                Возврат ячейки для ее
                                        правильного чтения.
                EX    (SP),HL           'Новый старт' идет в стек,
                                        а второй параметр в HL.
                AND   A                 Вычесть первый параметр
                SBC   HL,DE             из второго, чтобы найти длину
                                        'вырезки'.
                LD    BC,+0000          Инициализировать 'новую
                                        длину'.
                JR    C,2AA8,SL-OVER    Отрицательная вырезка
                                        является 'пустой строкой',
                                        что лучше, чем ошибочное
                                        условие (см.руководство).
                INC   HL                Допускается для включения
                                        в себя байта.
                AND   A                 Проверяется только 'регистр
                                        ошибок'.
                JP    M,2A20,REPORT-3   Переход, если никакой пара-
                                        метр не был вне диапазона.
                LD    B,H               Передача 'новой
                LD    C,L               длины' в BC.
2AA8 SL-OVER    POP   DE                Прочитать 'новое начало'.
                RES   6,(FLAGS)         Обеспечивает индикацию
                                        строки.
2AAD SL-STORE   CALL  2530,SYNTAX-Z     Возврат в эту точку, если
                RET   Z                 проверяется синтаксис; иначе,
                                        продолжение в STK-STORE.
```

ПОДПРОГРАММА 'STK-STORE'
Эта подпрограмма передает значения, содержащиеся в регистрах A,B,C,D и E в стек калькулятора. Из-за этого стек увеличивается в размере на 5 байт при каждом обращении к подпрограмме.
Подпрограмма обычно используется для передачи параметров строки, но она также используется STACK-BC и LOG(2^A) для передачи 'малых целых чисел' в стек.
Отметим, что при запоминании параметров строки первое заполненное значение (приходящее из регистра A) будет нулем, если строка приходит из массива строк или является 'вырезкой' строки. Для полной строки значение будет '1'. Этот 'признак' используется в командной процедуре 'LET', когда '1' сигнализирует, что старая запись строки восстановлена.

```
2AB1 STK-ST-0   XOR   A                 Сигнал - строка из массива
                                        строк или вырезанная строка.
2AB2 STK-STO-$  RES   6,(FLAGS)         Признак обозначает результат
                                        строки.
2AB6 STK-STORE  PUSH  BC                Запись B и C.
                CALL  33A9,TEST-5-SP    Это место для 5 байт? Сюда
                                        не возвращаться, пока
                                        не станет достаточно места.
                POP   BC                Восстановление B и C.
                LD    HL,(STKEND)       Выбор адреса первой ячейки
                                        над текущим стеком.
                LD    (HL),A            Передача первого байта.
                INC   HL                Шаг дальше.
```

```
                LD    (HL),E            Передела второго и
                INC   HL                третьего байта; для
                LD    (HL),D            строки они будут 'началом'.
                INC   HL                Шаг дальше.
                LD    (HL),C            Передача четвертого
                INC   HL                и пятого байта; для
                LD    (HL),B            строки они будут 'длиной'.
                INC   HL                Шаг для указания ячейки
                                        над стеком.
                LD    (STKEND),HL       Запись этого адреса
                RET                     в STKEND и возврат.
```

ПОДПРОГРАММА 'INT-EXP'
Эта подпрограмма возвращает результат вычисления 'следующего выражения' как значение целого числа, содержащегося а регистровой паре HL. Признак переноса задается, если присутствует ошибка 'вне диапазона'.
Регистр A используется как 'регистр ошибки' и содержит +00, если нет 'предыдущей ошибки', и +FF, если есть.

```
2ACC INT-EXP1   XOR   A                 Очистить 'регистр ошибок'.
2ACD INT-EXP2   PUSH  DE                Запись и DE, и HL.
                PUSH  HL
                PUSH  AF                Запись 'регистра ошибок'.
                CALL  1C82,EXPT-1NUM    Вычисление 'следующего
                                        выражения' для выдачи
                                        'последнего значения'
                                        на стек калькулятора.
                POP   AF                Восстановление 'регистра
                                        ошибок'.
                CALL  2530,SYNTAX-Z     При проверке синтаксиса,
                JR    Z,2AEB,I-RESTORE  переход вперед.
                PUSH  AF                Снова запись регистра ошибок.
                CALL  1E99,FIND-INT2    'Последнее значение' внесено
                                        в BC.
                POP   DE                Регистр ошибок в D.
                LD    A,B               'Следующее выражение',
                OR    C                 задающее 0,
                SCF                     всегда ошибочно; если это
                JR    Z,2AE8,I-CARRY    так, то переход вперед.
                POP   HL                Копия предельного значения.
                PUSH  HL                Это будет 'величина
                                        размерности', 'DIM-limit'
                                        или 'длина строки'.
                AND   A                 Теперь сравнение результата
                SBC   HL,BC             вычисления выражения
                                        с предельным значением.
```

Состояние признака переноса и значения, содержащегося в регистре D, обрабатываются, чтобы выдать значение для 'регистра ошибок'.

```
2AE8 I-CARRY    LD    A,D               Выбор 'старого значения
                                        ошибок'.
                SBC   A,+00             Формирование 'нового значе-
                                        ния ошибки', +00, если ошиб-
                                        ки нет/+FF или меньше, если
                                        есть ошибка 'вне диапазона'
```

```
                                        на этом проходе или
                                        предыдущем.
```

Восстановление регистров перед возвратом.

```
2AEB I-RESTORE    POP   HL              Восстановить HL и DE.
                  POP   DE
                  RET                   Возврат: 'регистром ошибки'
                                        является регистр A.
```

ПОДПРОГРАММА 'DE,(DE+1)'
Эта подпрограмма выполняет конструкцию LD DE,(DE+1) - и возвращает HL, указывающее на 'DE+2'.

```
2AEE DE,(DE+1)    EX    DE,HL           Для конструкции
                                        использовать HL.
                  INC   HL              Указывает 'DE+1'.
                  LD    E,(HL)          Фактически - LD E,(DE+1).
                  INC   HL              Указывает 'DE+2'.
                  LD    D,(HL)          Фактически - LD D,(DE+2).
                  RET                   Окончание.
```

ПОДПРОГРАММА 'GET-HL*DE'
Если не проверен синтаксис, эта подпрограмма вызывает 'HL=HL*DE', которое выполняет обозначенную конструкцию.
Переполнение 16 разрядов, возможное в регистровой паре HL, выдает сообщение 'нет памяти'. Это не точно отражает ситуацию, но означает, что памяти недостаточно для задачи, предложенной программистом.

```
2AF4 GET-HL*DE    CALL  2530,SYNTAX-Z   Немедленный возврат, если
                  RET   Z               проверен синтаксис.
                  CALL  30A9,HL=HL*DE   Выполнить умножение.
                  JP    C,1F15,REPORT-4 Сообщение 'нет памяти'.
                  RET                   Окончание.
```

Программная процедура 'LET'
Эта программа является фактической программой прерывания для команд LET, READ и INPUT.
Когда переменная адреса назначения является 'вновь объявленной переменной', то DEST укажет первую букву и имени переменной в строке BASIC. Будет задан первый разряд FLAGX.
Однако, если переменная адреса назначения 'уже существует', то 1 разряд FLAGX будет сброшен, а DEST укажет для числовой переменной ячейки перед пятью байтами 'старого числа'; а для строковой переменной - первую ячейку 'старой отроки'. Таким образом, DEST используется при работе с простыми переменными и элементами массива.
0 разряд FLAGX задается, если переменная адреса назначения является 'полной' простой строковой переменной. (Сигнализируя - удалить старую запись).
Изначально текущее значение DEST отбирается, и проверяется первый разряд FLAGS.

```
2AFF LET          LD    HL,(DEST)       Выбор текущего адреса в DEST.
                  BIT   1,(FLAGX)       Переход, если обрабатывается
                  JR    Z,2B66,L-EXISTS переменная уже 'существует'.
```

Используется 'вновь объявленная переменная'. Сначала находится длина ее имени.

```
                 LD    BC,+0005            Предположим, числовая
                                           переменная - 5 байт.
```

Введите цикл для работы с символами длинного имени. Любые пробелы или коды цвета в имени игнорируются.

```
2B0B L-EACH-CH   INC   BC                  Добавить '1' в счетчик для
                                           каждого символа имени.
2B0C L-NO-SP     INC   HL                  Движение вдоль имени
                                           переменной.
                 LD    A,(HL)              Выбор 'текущего кода'.
                 CP    +20                 Переход назад, если это
                                           'пробел'.
                 JR    Z,2B0C,L-NO-SP      Таким образом, игнорируются
                                           пробелы.
                 JR    NC,2B1F,L-TEST-CH   Переход вперед, если
                                           код от +21 до +FF.
                 CP    +10                 Допустим, как конечный
                 JR    C,2B29,L-SPACES     код, в диапазоне +00 - +0F.
                 CP    +16                 Также допускаем, диапазон
                 JR    NC,2B29,L-SPACES    +16 - +1F.
                 INC   HL                  Шаг за код управления
                                           после любого INK-OVER.
                 JR    2B0C,L-NO-SP        Переход назад, пока
                                           коды управления трактуются
                                           как пробелы.
```

Разделить 'числовые' и 'строковые' имена.

```
2B1F L-TEST-CH   CALL  2C88,ALPHANUM       Код алфавитно-цифровой?
                 JR    C,2B0B,L-EACH-CH    Если да, то он символ
                                           'длинного' имени.
                 CP    +24                 Текущий код '$'?
                 JP    Z,2BC0,L-NEWS       Переход вперед, пока
                                           обрабатывается 'вновь
                                           объявленная' строка.
```

'Вновь объявленная числовая переменная', обработанная в настоящий момент, потребует 'BC' байт в области переменных для ее имени и значения. Место создается и имя переменной копируется с требуемыми 'отмеченными' символами.

```
2B29 L-SPACES    LD    A,C                 Скопировать 'длину' в A.
                 LD    HL,(E-LINE)         HL указывает '80-байт'
                 DEC   HL                  в конце области переменных.
                 CALL  1655,MAKE-ROOM      Теперь раскроем область
                                           переменных.
                                           Примечание: В действительности
                                           места 'BC' созданы до размещения
                                           '80+байта'.
                 INC   HL                  Указывает первый 'новый'
                                           байт.
                 INC   HL                  DE указывает второй
                 EX    DE,HL               'новый' байт.
                 PUSH  DE                  Запись указателя.
```

```
                LD    HL,(DEST)          Выбор указателя начала
                                         имени.
                DEC   DE                 DE указывает первый 'новый
                                         файл'.
                SUB   +06                В содержит 'число лишних
                LD    B,A                букв', обнаруженных
                                         в 'длинном имени'.
                JR    Z,2B4F,L-SINGLE    Переход вперед, если работа
                                         с переменной с 'коротким
                                         именем'.
```

'Лишние' коды длинного имени передаются в область переменных.

```
2B3E L-CHAR     INC   HL                 Указывает каждый 'лишний
                                         код'.
                LD    A,(HL)             Выбор кода.
                CP    +21                Разделить коды от +21 до
                JR    C,2B3E,L-CHAR      +FF; игнорировать коды
                                         от +00 до +20.
                OR    +20                Задать 5 разряд, что касается
                                         букв нижнего регистра.
                INC   DE                 Передача по очереди кодов
                LD    (DE),A             вперед ко второму 'новому'
                                         байту.
                DJNZ  2B3E,L-CHAR        Прогнать цикл для всех
                                         'лишних' кодов.
```

С последним кодом 'длинного' имени должна быть выполнена операция OR с +80.

```
                OR    +80                Отмечает требуемый код
                LD    (DE),A             и перезаписывает последний
                                         код.
```

Теперь рассматривается первая буква имени обрабатываемой переменной.

```
                LD    A,+C0              Подготовить отметку буквы
                                         'длинного' имени.
2B4F L-SINGLE   LD    HL,(DEST)          Выбор указателя буквы.
                XOR   (HL)               A содержит +00 для
                                         'короткого' имени и +C0
                                         для 'длинного'.
                OR    +20                Задается 5 разряд, что касается
                                         букв нижнего регистра.
                POP   HL                 Теперь убрать указатель.
```

Теперь вызывается подпрограмма L-FIRST для ввода 'букв' в соответствующую ячейку.

```
                CALL  2BEA,L-FIRST       Ввод буквы и возврат с HL,
                                         указывающим 'новый' 80 байт.
```

Теперь можно передать 'последнее значение' в область переменных. Отметим, что в этой точке HL всегда указывает ячейку после пяти ячеек заранее выбранных для числа.

Команда 'RST 0028' используется для вызова CALCULATOR, а 'последнее значение' удаляется. Однако это значение не перезаписывается.

```
2B59 L-NUMERIC  PUSH  HL                 Запись указателя 'адреса
                                         назначения'.
```

```
                RST   0028,FP-CALC        Использование калькулятора.
                DEFB  +02,delete          Перемещение STKEND
                DEFB  +38,end-calc        назад за 5 байтов.
                POP   HL                  Восстановление указателя.
                LD    BC,+0005            Задает 'длину' 5 байт
                AND   A                   HL указывает первую из
                SBC   HL,BC               5 ячеек; и переход вперед,
                JR    2BA6,L-ENTER        чтобы передача была
                                          реальной.
```

Если рассматривается переменная, которая 'уже существует', то переход сюда. Сначала проверяется 6 разряд FLAGS, чтобы разделить числовые переменные из строки или массив строковых переменных.

```
2B66 L-EXISTS   BIT  6,(FLAGS)           Переход вперед, если
                JR    Z,2B72,L-DELETES    обрабатывается любой тип
                                          переменных.
```

Для числовых переменных 'новое' число накладывается на 'старое'. Итак, сначала HL должно быть создано, чтобы указывать ячейку после 5 байт существующей записи. В данное время HL указывает ячейку перед 5 байтами.

```
                LD    DE,+0006            5 байт числа +'1'.
                ADD   HL,DE               HL теперь указывает 'после'.
                JR    2B59,L-NUMERIC      Переход назад для создания
                                          фактической передачи.
```

Выбраны параметры строковой переменной и полные простые строки выбираются из 'вырезанных' строк и строк массивов.

```
2B72 L-DELETE$  LD    HL,(DEST)           Выбрано 'начало'. Эта строка
                                          резервная.
                LD    BC,(STRLEN)         Выбор 'длины'.
                BIT   0,(FLAGX)           Если работа с полной простой
                JR    NZ,2BAF,L-ADD$      строкой, переход: старую
                                          строку будет необходимо
                                          удалить.
```

Работа с 'вырезкой' существующей простой строки, 'вырезкой' строки из массива строк или полной строки из массива строк включает в себя два разных этапа. Сначала строится 'новая' строка в рабочей области, в зависимости от требований, длинная или короткая. Второй этап нужен для копирования 'новой' строки в предназначенное ей место области переменных.
Однако, ничего не делайте, если строка не имеет 'длину'.

```
                LD    A,B                 Возврат, если строка
                OR    C                   является пустой строкой.
                RET   Z
```

Затем создается требуемое количество доступных в рабочей области мест.

```
                PUSH  HL                  Записать 'начало' (DEST).
                RST   0030,BC-SPACES      Создание необходимого
                                          количества мест в рабочей
                                          области.
                PUSH  DE                  Записать указатель первой
                                          ячейки.
```

```
                PUSH  BC                Записать 'длину' для
                                        дальнейшего использования.
                LD    D,H               DE указывает последнюю
                LD    E,L               ячейку.
                INC   HL                HL указывает ячейку 'на 1
                                        после' новой ячейки.
                LD    (HL),+20          Ввод символа 'пробел'.
                LDDR                    Скопировать этот символ во
                                        все новые ячейки. Закончить
                                        с HL, указывающим первую
                                        новую ячейку.
```

Параметры обработанной строки теперь выбираются из стека калькулятора.

```
                PUSH  HL                Записать указатель.
                CALL  2BF1,STK-FETCH    Выбор 'нового' параметра.
                POP   HL                Восстановление указателя.
```

Примечание: В этой точке создается требуемое число доступных в рабочей области мест, например, для 'переменной присваивания'. Для оператора - LET A$(от 4 до 8)="abcdefg" - созданы 5 ячеек. Выбранные вами как 'последнее значение' параметры, представляют строку, которая копируется в новые ячейки, с Прокрустовым требованием длины. Длина 'новой строки' сравнивается с длиной созданного для нее места.

```
                EX    (SP),HL           'Длину' новой области в HL.
                                        'Указатель' новой области
                                        в стек.
                AND   A                 Сравнивание двух
                SBC   HL,BC             'длин' и переход
                ADD   HL,BC             вперед, если 'новая' строка
                                        войдет на место.
                JR    NC,2B9B,L-LENGTH  Т.е. без укорачивания.
                LD    B,H               Однако, измените 'новую'
                LD    C,L               длину, если она слишком
                                        длинная.
2B9B L-LENGTH   EX    (SP),HL           'Длина' новой области
                                        заносится в стек. Указатель
                                        новой области в HL.
```

Пока длинная новая строка не является пустой строкой, она копируется в рабочую область. Прокрустова установка длины достигается автоматически, если 'новая' строка короче, чем предназначенное ей место.

```
                EX    DE,HL             'Начало' новой строки
                                        в HL. 'Указатель' новой
                                        области в DE.
                LD    A,B               Переход вперед, если
                OR    C                 'новая' строка является
                JR    Z,2BA3,L-IN-W/S   'пустой' строкой.
                LDIR                    В противном случае перемещение
                                        'новой' строки в рабочую
                                        область.
```

Восстанавливаются значения, которые были записаны в машинном стеке.

```
2BA3 L-IN-W/S   POP   BC                'Длина' новой области.
                POP   DE                'Указатель' новой области.
```

```
                  POP   HL                  'Начало-указатель' перемен-
                                            ной присваивания, которая
                                            изначально была в DEST.
                                            Теперь L-ENTER используется
                                            для передачи 'новой' строки
                                            в область переменных.
```

ПОДПРОГРАММА 'L-ENTER'
Эта короткая подпрограмма предназначена для передачи или числового значения из стека калькулятора, или строки из рабочей области на их соответствующие позиции в области переменных.
Поэтому подпрограмма используется для 'вновь объявленных' строк и 'полных и существующих' простых строк.

```
2BA6 L-ENTER      EX    DE,HL               Изменить указатели.
                  LD    A,B                 Повторная проверка того,
                  OR    C                   что длина не равна 0.
                  RET   Z
                  PUSH  DE                  Запись указателя адреса
                                            назначения.
                  LDIR                      Перемещение числового
                                            значения или строки.
                  POP   HL                  Возврат с регистровой парой
                  RET                       HL, указывающей первый байт
                                            числового значения или
                                            строки.
```

ПОДПРОГРАММА 'LET' (продолжение)
При обработке 'полной и существующей' простой строки вводится новая строка как 'вновь объявленная' простая строка перед 'восстановленной' существующей версией.

```
2BAF L-ADD$       DEC   HL                  HL указывает букву
                  DEC   HL                  в имени переменной,
                  DEC   HL                  т.е. DEST - 3.
                  LD    A,(HL)              Подбор буквы.
                  PUSH  HL                  Запись указателя
                                            'существующей версии'.
                  PUSH  BC                  Запись длины 'существующей
                                            строки'.
                  CALL  2BC6,L-STRING       Использование L-STRING для
                                            добавления новой строки
                                            в область переменных.
                  POP   BC                  Восстановление 'длины'.
                  POP   HL                  Восстановление указателя.
                  INC   BC                  Допускается один
                  INC   BC                  байт для буквы и два
                  INC   BCбайта для длины.
                  JP    19E8,RECLAIM-2      Выход переходом в RECLAIM-2,
                                            которая восстанавливает всю
                                            существующую версию.
```

'Вновь объявленные' простые строки обрабатываются следующим образом.

```
2BC0 L-NEW$       LD    A,+DF               Подготовиться отметить
                                            букву переменной.
                  LD    HL,(DEST)           Выбор указателя буквы.
```

```
                AND   (HL)              Отметить требуемую букву;
                                        L-STRING теперь используется
                                        для добавки новой отроки
                                        в область переменных.
```

ПОДПРОГРАММА 'L-STRING'
Выбираются параметры 'новой' строки, создается достаточное количество места для нее и затем передача строки.

```
2BC6 L-STRING   PUSH  AF                Запись буквы переменной.
                CALL  2BF1,STK-FETCH    Выбор 'начала' и 'длины'
                                        'новой' строки.
                EX    DE,HL             Пересылка 'начала' в HL.
                ADD   HL,BC             HL указывает на 'страницу
                                        после' строки.
                PUSH  BC                Запись 'длины'.
                DEC   HL                HL указывает конец строки.
                LD    (DEST),HL         Запись указателя.
                INC   BC                Допускается один байт
                INC   BC                для буквы и два байта
                INC   BCдля длины.
                LD    HL,(E-LINE)       HL указывает '80 байт'
                DEC   HL                в конце области переменной.
                CALL  1655,MAKE-ROOM    Теперь раскроем область
                                        переменной. Примечание:
                                        В действительности места 'BC'
                                        созданы перед размещением
                                        '80 байт'.
                LD    HL,(DEST)         Восстановить указатель
                                        конца 'новой' строки.
                POP   BC                Создание копии длины
                PUSH  BC                'новой' строки.
                INC   BC                Добавить 1 к длине в случае,
                                        если 'новая' отрока является
                                        'пустой строкой'.
                LDDR                    Теперь копируем 'новую'
                                        строку плюс 1 байт.
                EX    DE,HL             HL указывает байт, который
                INC   HL                содержит верхний уровень
                                        длины.
                POP   BC                Выбор 'длины'.
                LD    (HL),B            Ввод верхнего уровня длины.
                DEC   HL                На 1 назад.
                LD    (HL),C            Ввод лишнего уровня длины.
                POP   AF                Выбор буквы переменной.
```

ПОДПРОГРАММА 'L-FIRST'
Эта подпрограмма вводится с буквой переменной, соответствующим образом отмеченной, в регистре A. Буква накладывается на 'старые 80 байт' в области переменных. Подпрограмма возвращается с регистровой парой HL, указывающей на 'новые 80 байт'.

```
2BEA L-FIRST    DEC   HL                HL указывает 'старый
                                        80 байт'.
                LD    (HL),A            На него накладывается
                                        буква переменной.
                LD    HL,(E-LINE)       HL указывает 'новый
```

```
                DEC   HL                  80 байт'.
                RET                       Окончание со всеми 'вновь
                                          объявленными переменными'.
```

ПОДПРОГРАММА 'STK-FETCH'
Эта важная подпрограмма отбирает 'последнее значение' из стека калькулятора. Пять байт могут быть или числом с плавающей точкой в 'короткой' или 'длинной' форме, или набором параметров, которые определяют строку.

```
2BF1 STK-FETCH  LD    HL,(STKEND)         Прочитать STKEND.
                DEC   HL                  На 1 назад.
                LD    B,(HL)              Пятое значение.
                DEC   HL                  На 1 назад.
                LD    C,(HL)              Четвертое значение.
                DEC   HL                  На 1 назад.
                LD    D,(HL)              Третье.
                DEC   HL                  На 1 назад.
                LD    E,(HL)              Второе.
                DEC   HL                  На 1 назад.
                LD    A,(HL)              Первое.
                LD    (STKEND),HL         Сброс STKEND на его
                                          новой позиции.
                RET                       Окончание.
```

КОМАНДНАЯ ПРОЦЕДУРА 'DIM'
Эта процедура устанавливает новые массивы о области переменных. Программа начинается с поиска существующей области переменной, чтобы определить, существует ли массив с таким же именем, он 'восстанавливается' перед установкой нового массива.
У нового массива все элементы нули, если это числовой массив, или 'пробелы', если это массив строк.

```
2C02 DIM        CALL  28B2,LOOK-VARS      Поиск области переменных.
2C05 D-RPORT-C  JP    NZ,1C8A,REPORT-C    Выдает сообщение C, т.к.
                                          была ошибка.
                CALL  2530,SYNTAX-Z       Переход вперед, если
                JR    NZ,2C15,D-RUN       во 'время выполнения'.
                RES   6,C                 Проверка синтаксиса строко-
                                          вых массивов, как числовых.
                CALL  2996,STK-VAR        Проверка синтаксиса
                                          выражения в круглых скобках.
                CALL  1BEE,CHECK-END      Перемещение для рассмотрения
                                          следующего оператора, т.к.
                                          синтаксис правилен.
```

Восстановлен 'существующий массив'.

```
2C15 D-RUN      JR    C,2C1F,D-LETTER     Переход вперед, если нет
                                          'существующего массива'.
                PUSH  BC                  Запись байта дискриминатора.
                CALL  19B8,NEXT-ONE       Найти начало следующей
                                          переменной.
                CALL  19E8,RECLAIM-2      Восстановление 'сущест-
                                          вующего массива'.
                POP   BC                  Восстановление байта
                                          дискриминатора.
```

Обнаружены начальные параметры нового массива.

```
2C1F D-LETTER     SET   7,C               Задан 7 разряд в байте
                                          дискриминатора.
                  LD    B,+00             Установка 0 счетчика
                                          размерности.
                  PUSH  BC                Запись счетчика и байта
                                          дискриминатора.
                  LD    HL,+0001          Регистровая пара HL
                  BIT   6,C               содержит размеры
                  JR    NZ,2C2D,D-SIZE    в массиве, '1' для
                  LD    L,+05             строкового массива/
                                          '5' для числового.
2C2D D-SIZE       EX    DE,HL             Размеры элемента в DE.
```

Следующий цикл годится для каждой размерности, которая задана в выражении в скобках оператором DIM. Общее число байт, требуемых для хранения элементов массива, формируется в регистровой паре DE.

```
2C2E D-NO-LOOP    RST   0020,NEXT-CHAR    Вызов CH-ADD на каждом
                                          проходе.
                  LD    H,+FF             Здесь 'предельное значение'.
                  CALL  2ACC,INT-EXP1     Вычисление параметра.
                  JP    C,2A20,REPORT-3   Выдать ошибку, если
                                          'вне диапазона'.
                  POP   HL                Выбор счетчика размерности
                                          и байта дискриминатора.
                  PUSH  BC                Запись параметра на каждом
                                          проходе через цикл.
                  INC   H                 Увеличение счетчика размер-
                                          ности на каждом проходе.
                  PUSH  HL                Перенесение в стек счетчика
                                          размерностей и байта
                                          дискриминатора.
                  LD    H,B               Перемещение параметра
                  LD    L,C               в регистровую пару HL.
                  CALL  2AF4,GET-HL*DE    Байтовая сумма в HL,
                  EX    DE,HL             а затем передача в DE.
                  RST   0018,GET-CHAR     Прочитать следующий
                  CP    +2C               параметр и проверить
                  JR    Z,2C2E,D-NO-LOOP  опять цикл, если есть
                                          другая размерность.
```

Примечание: В этой точке регистровая пара DE показывает, что число байт, требуемых для элементов нового массива, и величины каждой размерности занесены в стек.
Теперь проверим, что действительно имеется закрываемая скобка у выражения в круглых скобках.

```
                  CP    +29               Это ')'?
                  JR    NZ,2C05,D-REPORT-C Если нет, переход назад.
                  RST   0020,NEXT-CHAR    Перевести CH-ADD за нее.
```

Теперь делаем допущение по величине размерности.

```
                  POP   BC                Выбор счетчика размерности
                                          и байта дискриминатора.
                  LD    A,C               Передать, байт дискри-
                                          минатора в регистр A для
                                          дальнейшего использования.
```

```
                LD    L,B               Переместить счетчик в L.
                LD    H,+00             Очистить регистр H.
                INC   HL                Увеличить, счетчик
                INC   HL                размерности на 2 и
                ADD   HL,HL             удвоить результат, и
                ADD   HL,DE             сформировать правильную
                                        общую длину для переменной,
                                        добавив байтовую сумму
                                        элемента.
                JP    C,1F15,REPORT-4   Если требуется, выдача
                                        сообщения 'Нет памяти'.
                PUSH  DE                Запись байтовой суммы
                                        элемента.
                PUSH  BC                Запись счетчика размерности
                                        и байта дискриминатора.
                PUSH  HL                Запись общей длины.
                LD    B,H               Перемещение общей
                LD    C,L               длины в BC.
```

Создано требуемое место для нового массива в конце области переменных.

```
                LD    HL,(E-LINE)       Регистровая пара HL
                DEC   HL                указывает '80 байт'.
                CALL  1655,MAKE-ROOM    Создано необходимое место.
                INC   HL                Создано HL для указания
                                        первой новой ячейки.
```

Теперь вводятся параметры.

```
                LD    (HL),A            Сначала вводятся соответ-
                                        ствующим образом отмеченная
                                        буква.
                POP   BC                Выбирается общая
                DEC   BC                длина и уменьшается
                DEC   BC                на '3'.
                DEC   BC
                INC   HL                Продвижение HL.
                LD    (HL),C            Ввод верхнего уровня длины.
                INC   HL                Продвижение HL.
                LD    (HL),B            Ввод верхнего уровня длины.
                POP   BC                Выбор счетчика размерности.
                LD    A,B               Перемещение его в регистр A.
                INC   HL                Продвижение HL.
                LD    (HL),A            Ввод счетчика размерности.
```

Теперь 'очищаются' элементы нового массива.

```
                LD    H,D               Создано HL для указания
                LD    L,E               последней ячейки массива и
                DEC   DE                DE к ячейке перед ней.
                LD    (HL),+00          Ввести поле в последнюю
                BIT   6,C               ячейку, но перезаписать ее
                JR    Z,2C7C,DIM-CLEAR  'пробелом', если идёт работа
                LD    (HL),+20          с массивом отрок.
2C7C DIM-CLEAR  POP   BC                Выбор байтовой суммы
                                        элемента.
                LDDR                    Очистить массив + одна
                                        ячейка сверх.
```

Теперь ввод величин размерности.

```
2C7F DIM-SIZES    POP   BC                  Прочитать величину
                                            размерности.
                  LD    (HL),B              Ввод старшего байта.
                  DEC   HL                  На единицу назад.
                  LD    (HL),C              Ввод младшего байта.
                  DEC   HL                  На единицу назад.
                  DEC   A                   Уменьшение счетчика
                                            размерности.
                  JR    NZ,2C7F,DIM-SIZES   Повторить операцию пока не
                  RET                       будут рассмотрены все
                                            размерности; затем переход.
```

ПОДПРОГРАММА 'ALPHANUM'
Эта подпрограмма возвращается с установленным признаком переноса, если текущее значение регистра A обозначает правильную цифру или букву.

```
2C88 ALPHANUM     CALL  2D1B,NUMERIC        Тест для цифры; для цифры
                                            признак будет сброшен.
                  CCF                       Дополнить признак перекоса.
                  RET   C                   Если цифра, то возврат, в
                                            противном случае, продолжение.
```

ПОДПРОГРАММА 'ALPHA'
Эта подпрограмма возвращается с установленным признаком переноса, если текущее, значение регистра A обозначает правильную букву алфавита.

```
2C8D ALPHA        CP    +41                 Проверка шестнадцатеричного
                                            числа 41, код для 'A'.
                  CCF                       Дополнить признак переноса.
                  RET   NC                  Возврат, если код символа
                                            неверен.
                  CP    +5B                 Проверка шестнадцатеричного
                                            числа 58, на 1 больше, чем
                                            код 'Z'.
                  RET   C                   Возврат, если буква верхнего
                                            регистра.
                  CP    +61                 Проверка шестнадцатеричного
                                            числа 61, код для 'a'.
                  CCF                       Дополнить признак переноса.
                  RET   NC                  Возврат, если код символа
                                            неверен.
                  CP    +7B                 Проверка шестнадцатеричного
                                            числа 7B, на 1 больше, чем
                                            код 'z'.
                  RET                       Окончание.
```

ПОДПРОГРАММА 'DECIMAL TO FLOATING-POINT' ('Десятичное в форму с плавающей точкой')
Часть синтаксиса, работающая с десятичными числами, которые появляются в строке BASIC и преобразуют в формы с плавающей точкой. Эта подпрограмма читает цифра за цифрой десятичное число и выдает результат как 'последнее значение' на стек вычислителя. Но сначала она работает с альтернативной записью BIN, которая представляет

последовательность 0 или 1, задающих двоичное представление требуемых чисел.

```
2C9B DEC-TO-FP    CP    +C4                 Этот символ 'BIN'?
                  JR    NZ,2CB8,NOT-BIN     Переход, если не 'BIN'.
                  LD    DE,+0000            Инициализировать результат
                                            в DE нулем.
2CA2 BIN-DIGIT    RST   0020,NEXT-CHAR      Прочитать следующий символ.
                  SUB   +31                 Вычесть символьный код '1'.

                  ADC   A,+00               0 теперь выдает 0 с установ-
                                            ленным признаком переноса;
                                            1 выдает 0 со сброшенным
                                            признакам переноса.
                  JR    NZ,2CB3,BIN-END     Любой другой символ
                                            вынуждает перейти на BIN-END
                                            и будет проверяться на синтаксис
                                            во время или после сканирования.
                  EX    DE,HL               Теперь результат в HL.
                  CCF                       Дополнить признак переноса.
                  ADC   HL,HL               Переместить результат влево
                                            с признаком, переходящим
                                            в 0 разряд.
                  JP    C,31AD,REPORT-6     Сообщение о переполнении,
                                            если больше 65535.
                  EX    DE,HL               Пока результат возвратить
                                            в DE.
                  JR    2CA2,BIN-DIGIT      Переход назад для следующего
                                            0 или 1.
2CB3 BIN-END      LD    B,D                 Скопировать результат в
                  LD    C,E                 BC для занесения в стек.
                  JP    2D2B,STACK-BC       Переход вперед для занесения
                                            результата в стек.
```

Для других чисел, сначала преобразуется целая часть; если следующий символ является десятичным числом, то далее рассматривается дробная часть десятичного числа.

```
2CB8 NOT-BIN      CP    +2E                 Первый символ '.'?
                  JR    Z,2CCB,DECIMAL      Бели да, переход вперед.
                  CALL  2D3B,INT-TO-FP      В противном случае сформиро-
                                            вать 'последнее значение'
                                            целого числа.
                  CP    +2E                 Следующий символ '.'?
                  JR    NZ,2CEB,E-FORMAT    Переход вперед, если это 'E'.
                  RST   0020,NEXT-CHAR      Прочитать следующий символ.
                  CALL  2D1B,NUMERIC        Это цифра?
                  JR    C,2CEB,E-FORMAT     Переход, если нет
                                            (например 1.E4).
                  JR    2CD5,DEC-STO-1      Переход вперед для работы с
                                            цифрами после десятичной за-
                                            пятой.
2CCB DECIMAL      RST   0020,NEXT-CHAR      Если число начинается с
                  CALL  2D1B,NUMERIC        точки, рассмотрите его,
                                            если следующее число яв-
                                            ляется цифрой.
2CCF DEC-RPT-C    JP    C,1C8A,REPORT-C     Если нет, сообщение об
                                            ошибке.
```

```
                    RST   0028,FP-CALC        Используйте калькулятор
                    DEFB  +A0,stk-zero        для занесения в стек 0 как
                    DEFB  +38,end-calc        целой части таких чисел.
2CD5 DEC-STO-1      RST   0028,FP-CALC        Опять используйте каль-
                    DEFB  +A1,stk-one         кулятор. Найдите
                    DEFB  +C0,st-mem-0        форму с плавающей
                    DEFB +02,delete           точкой десятичного числа
                    DEFB  +38,end-calc        '1' и запишите в память.
2CDA NXT-DGT-1      RST   0018,GET-CHAR       Прочитать следующий символ.
                    CALL  2D22,STK-DIGIT      Если цифра, занести
                                              ее в стек.
                    JR    C,2CEB,E-FORMAT     Если нет, переход вперед.
                    RST   0028,FP-CALC        Теперь используйте
                                              вычислитель.
                    DEFB  +E0,get-mem-0       Для каждого прогона
                    DEFB  +A4,stk-ten         цикла выбирается число,
                    DEFB  +05,division        записанное в памяти, делен-
                    DEFB  +C0,st-mem-0        ное на 10 и восстановленное:
                                              т.е. от .1 до .01, до .001 и
                                              т.д.
                    DEFN  +04,multiply        Текущая цифра умножается
                    DEFB  +0F,addition        на 'записанное число' и
                    DEFB  +38,end-calc        добавляется к 'последнему
                                              значению'.
                    RST   0020,NEXT-CHAR      Прочитать следующий символ.
                    JR    2CDA,NXT-DGT-1      Переход назад (на 1 байт
                                              больше, чем это необходимо),
                                              чтобы рассмотреть его.
```

Далее рассматривается любая 'запись E', т.е. форма xEm или xem, где m это положительное или отрицательное целое число.

```
2CEB E-FORMAT       CP    +45                 Текущий, символ 'E'?
                    JR    Z,2CF2,SIGN-FLAG    Если да, переход вперед.
                    CP    +65                 Это 'e'?
                    RET   NZ                  Окончание, если это не так.
2CF2 SIGN-FLAG      LD    B,+FF               Используйте B как знак
                                              признака, +FF для '+'.
                    RST   0020,NEXT-CHAR      Прочитать следующий символ.
                    CP    +2B                 Это '+'?
                    JR    Z,2CFE,SIGN-DONE    Переход вперед.
                    CP    +2D                 Это '-'?
                    JR    NZ,2CFF,ST-E-PART   Переход, если не '+', не '-'.
                    INC   B                   Изменение знака признака.
2CFE SIGN-DONE      RST   0020,NEXT-CHAR      Указать первую цифру.
2CFF ST-E-PART      CALL  2D1B,NUMERIC        Действительно это цифра?
                    JR    C,2CCF,DEC-RPT-C    Если нет, сообщение
                                              об ошибке.
                    PUSH  BC                  Запись признака в B.
                    CALL  2D3B,INT-TO-FP      Занесение в стек ABS m, где
                                              m это порядок.
                    CALL  2DD5,FP-TO-A        Передача ABS m в A.
                    POP   BC                  Восстановление признака
                                              знака в B.
                    JP    C,31AD,REPORT-6     Сообщение о переполнении
                    AND   A                   если ABS m больше, чем 255
                    JP    M,31AD,REPORT-6     или больше, чем 127 (другие
                                              значения больше, чем 39,
                                              будут рассмотрены позднее).
```

```
                INC   B                 Проверка признака знака в B;
                                        '+' (т.е. +FF) теперь будет
                                        задаваться признаком нуля.
                JR    Z,2D18,E-FP-JUMP  Переход, если знак m '+'.
                NEG                     Инвертировать m, если
                                        знак '-'.
2D18 E-FP-JUMP  JP    2D4F,E-TOO-FP     Переход, чтобы присвоить
                                        'последнему значению'
                                        результат выражения X*10^m.
```

ПОДПРОГРАММА 'NUMERIC'
Эта подпрограмма возвращается со сброшенным признаком переноса, если текущее значение регистра A является правильной цифрой.

```
2D1B NUMERIC    CP    +30               Проверка шестнадцатеричного
                                        числа 30, код для '0'.
                RET   C                 Возврат, если код символа
                                        неверен.
                CP    +3A               Проверка верхнего предела.
                CCF                     Добавить признак переноса.
                RET                     Окончание.
```

ПОДПРОГРАММА 'STK-DIGIT'
Эта подпрограмма просто возвращается, если текущее значение, содержащееся в регистре A, не является цифрой, но если является, то форма с плавающей точкой для цифры становится 'последним значением' на стеке калькулятора.

```
2D22 STK-DIGIT  CALL  2D1B,NUMERIC      Символ цифра?
                RET   C                 Если нет, возврат.
                SUB   +30               Замена кода реальной цифрой.
```

ПОДПРОГРАММА 'STACK-A'
Эта подпрограмма задает форму с плавающей точкой для полной двоичной величины, находящейся в регистре A.

```
2D28 STACK-A    LD    C,A               Значение в регистр C.
                LD    B,+00             Очистка регистра B.
```

ПОДПРОГРАММА 'STACK-BC'
Эта подпрограмма задает форму с плавающей точкой для полной двоичной величины, находящейся в регистровой паре BC.
Используемая форма, также как и в двух предыдущих программах, является зарезервированной в SPECTRUM для малых целых чисел n, где −65535 <= n <= 65535. Первый и пятый байты 0; третий и четвертый байты являются именем и более значимыми для 16 разрядного целого числа n в двух дополнительных формах (если n отрицательно, эти байты содержат 65536+n); второй байт является байтом знака, 00 для '+' и FF для '-'.

```
2D2B STACK-BC   LD    IY,+5C3A          Реинициализация IY в ERR-NR.
                XOR   A                 Очистить регистр A.
                LD    E,A               И регистр E, чтобы
                                        обозначить '+'.
```

```
                LD    D,C               Скопировать менее
                                        значимый байт в D.
                LD    C,B               A более значимый в C.
                LD    B,A               Очистить регистр B.
                CALL  2AB6,STK-STORE    Теперь занести в стек число.
                RST   0028,FP-CALC      HL указывает на
                DEFB  +38,end-calc      STKEND-5.
                AND   A                 Очистить признак переноса.
                RET                     Окончание.
```

ПОДПРОГРАММА 'INTEGER TO FLOATING-POINT'
('Целое число в форму с плавающей точкой')
Эта подпрограмма возвращает 'последнее значение' на стек калькулятора которое является результатом преобразования целого числа в строке BASIC, т.е. целая часть десятичного числа или номера строки преобразуется в форму с плавающей точкой.

Повторное обращение к CH-ADD+1 выбирает по очереди каждую цифру целого числа. Выход осуществляется, когда выбирается код, не представляющий цифру.

```
2D3B INT-TO-FP  PUSH  AF                Запись первой цифры в A.
                RST   0028,FP-CALC      Использование вычислителя.
                DEFB  +A0,stk-zero      Теперь 'последнее
                DEFB  +38,end-calc      значение' 0.
                POP   AF                Восстановить первую цифру.
```

Теперь задается цикл. Он работает, пока коды представляют цифры, затем находится форма с плавающей точкой и помещается в стек как 'последнее значение'. 'Последнее значение' затем умножается на десятичное 10 и добавляется к 'цифре', чтобы сформировать новое 'последнее значение', которое возвращается на начло цикла.

```
2D40 NXT-DGT-2  CALL  2D22,STK-DIGIT    Если код представляет цифру,
                RET   C                 то в стек заносится форма
                                        с плавающей точкой.
                RST   0028,FP-CALC      Использование калькулятора.
                DEFB  +01,exchange      'Цифра' идет как 'последнее
                                        значение'.
                DEFB  +A4,stk-ten       Определение десятичного 10.
                DEFB  +04,multiply      'Последнее значение' =
                                        'последнее значение' * 10.
                DEFB  +0F,addition      'Последнее значение' =
                                        'последнее значение' +
                                        'цифра'.
                DEFB    +38,end-calc
                CALL  0074,CH-ADD+1     В A поступает следующий код.
                JR    2D40,NXT-DGT-2    Прогон цикла с этим кодом.
```

АРИФМЕТИЧЕСКИЕ ПРОЦЕДУРЫ

ПОДПРОГРАММА 'E-FORMAT TO FLOATING-POINT'
(Смещение 3C - см. CALCULATE: 'e-to-fp')
('Число в форме E в форму с плавающей точкой')

Эта подпрограмма задает 'последнее значение' на вершине стека калькулятора, как результат преобразования числа, заданного в форме xEm, где m является положительным или отрицательным целым числом. Подпрограмма вводится с X на вершине стека калькулятора и m в регистре A.

Используемый метод применяется для нахождения полного значения m, скажем p, и умножения или деления x на 10^p в соответствии с тем, каким является m - положительным или отрицательным.

Чтобы получить это, p смещается вправо, пока не станет 0, а x умножается или делится на 10^(2^n) для каждого разряда набора b(n) p. Так как p никогда не становится больше, чем десятичное 39, 6 и 7 разряды p не будут установлены.

```
2D4F E-TO-FP      RLCA                    Проверка знака m, с
                  RRCA                    помощью циклического сдвига
                                          7 разряда A без изменения A.
                  JR    NC,2D55,E-SAVE    Переход, если m положительно.
                  CPL                     Инвертировать m в A без
                  INC   A                 разрушения признака переноса.
2D55 E-SAVE       PUSH  AF                Записать m в A.
                  LD    HL,+5C92          Это MEMBOT: признак знака
                  CALL  350B,FP-0/1       теперь занесен первый байт
                                          mem-0, т.е. 0 для '+' и 1
                                          для '-'.
                  RST   0028,FP-CALC      Стек содержит x.
                  DEFB  +A4,stk-ten       x,10 (десятичное).
                  DEFB  +38,end-calc      x,10.
                  POP   AF                Восстановить m в A.
2D60 E-LOOP       SRL   A                 В цикле сдвинуть без сохра-
                                          нения следующий разряд m
                                          изменением признака переноса
                                          и нулем соответственно.
                  JR    NC,2D71,E-TST-END Переход, если перенос сброшен.
                  PUSH  AF                Запись остатка m и признаков.
                  RST   0028,FP-CALC      Стек содержит x' и 10^(2^n),
                                          где x' это промежуточное
                                          значение при умножении x на
                                          10^m, и n=0,1,2,3,4 или 5.
                  DEFB  +C1,st-mem-1      (10^(2^n) скопировано
                                          в mem-1).
                  DEFB  +E0,get-mem-0     x', 10^(2^n), (1/0).
                  DEFB  +00,jump-true     x', 10^(2^n)
                  DEFB  +04,to E-DIVSN    x', 10^(2^n)
                  DEFB  +04,multiply      x'*10^(2^n)=x''
                  DEFB  +33,jump          x''
                  DEFB  +02,to E-FETCH    x''
2D6D E-DIVSN      DEFB  +05,division      x'/10^(2^n)=x'' (x'' это
                                          N'*10^(2^n) или x'/10^(2^n)
                                          в зависимости от m '+' или
                                          '-'.
2D6E E-FETCH      DEFB  +E1,get-mem-1     x'', 10^(2^n)
                  DEFB  +38,end-calc      x'', 10^(2^n)
                  POP   AF                Восстановить m в A, и
```

```
                                          признаки.
2D71 E-TST-END    JR    Z,2D7B,E-END      Переход, если m было
                                          уменьшено до 0.
                  PUSH  AF                Записать остаток m в A.
                  RST   0028,FP-CALC      x'', 10^(2^n)
                  DEFB  +31,duplicate     x'', 10^(2^n), 10^(2^n)
                  DEFB  +04,multiply      x'', 10^(2^(n+1))
                  DEFB  +38,end-calc      x'', 10^(2^(n+1))
                  POP   AF                Восстановить остаток m в A.
                  JR    2D60,E-LOOP       Переход назад для всех
                                          разрядов m.
2D7B E-END        RST   0028,FP-CALC      Использование калькулятора
                  DEFB  +02,delete        для удаления достигнутого
                  DEFB  +28,end-calc      конечного показателя степени
                  RET                     10, оставляя последнее
                                          значение x*10^m в стеке.
```

ПОДПРОГРАММА 'INT-FETCH'
Эта подпрограмма получает в DE малое целое число n (-65535 <= n <= 65535) из ячейки адресованной HL, т.е. n обычно является первым (или вторым) числом на вершине стека калькулятора; но HL может также получить (изменив DE) число, которое было удалено из стека.
Подпрограмма сама не удаляет числа из стека или памяти: она возвращает HL, указывающую четвертую часть числа на его исходной позиции.

```
2D7F INT-FETCH    INC   HL                Указать байт знака.
                  LD    C,(HL)            Скопировать байт знака в C.
```

Следующий механизм будет осуществлять дополнение до двух числа, если оно отрицательное (C это FF), но оставит без изменений, если оно положительно (C это 00).

```
                  INC   HL                Указывает наименее значимый
                                          байт.
                  LD    A,(HL)            Получение байта в A.
                  XOR   C                 Первое изменение, если
                                          значение отрицательно.
                  SUB   C                 Добавляется 1 для отрица-
                                          тельных чисел; если байт не
                                          был 0, задается перенос.
                  LD    E,A               Теперь в E менее значимый
                                          байт.
                  INC   HL                Указать более значимый байт.
                  LD    A,(HL)            Получить его в A.
                  ADC   A,C               В случае отрицательного
                  XOR   C                 числа, закончить оба
                                          изменения; отметим, что
                                          перенос всегда остается
                                          сброшенным.
                  LD    D,A               Теперь более значимый байт
                                          в D.
                  RET                     Окончание.
```

ПОДПРОГРАММА 'INT-STORE'
Эта подпрограмма заносит малое целое число (-65535 <= n <= 65535) в ячейку адресованную HL и последующие четыре ячейки: т.е. n заменяет

первое (или второе) число на вершине стека калькулятора. Подпрограмма возвращает HL, указывающее на первый байт n на стеке.

```
2D8C P-INT-STO    LD    C,+00              Эта точка входа должна
                                           заносить число, известное
                                           как положительное.
2D8E INT-STORE    PUSH  HL                 Записать указатель первой
                                           ячейки.
                  LD    (HL),+00           Первый байт задан 0.
                  INC   HL                 Указать вторую ячейку.
                  LD    (HL),C             Ввести байт знака.
```

Используется тот же механизм, что и в 'INT-FETCH', т.е. дополнение до двух отрицательны чисел.
Это необходимо, например, до и после умножения малых целых чисел.

```
                  INC   HL                 Указать третью ячейку.
                  LD    A,E                Получить менее значимый байт.
                  XOR   C                  Дополнение до двух,
                  SUB   C                  если число отрицательное.
                  LD    (HL),A             Заполнение байта.
                  INC   HL                 Указать четвертую ячейку.
                  LD    A,D                Получить более значимый байт.
                  ADC   A,C                Дополнение до двух,
                  XOR   C                  если число отрицательное.
                  LD    (HL),A             Запомнить байт.
                  INC   HL                 Указать пятую ячейку.
                  LD    (HL),+00           Пятый байт задан 0.
                  POP   HL                 Возврат с HL, указывающий
                  RET                      первый байт на стек.
```

ПОДПРОГРАММА 'FLOATING-POINT TO BC' ('Форма с плавающей точкой в BC')
Эта подпрограмма вызывается из четырех различных мест, для разных целей и используется для помещения 'последнего значения' с плавающей точкой в регистровую пару BC.
Если результат очень большой, т.е. больше, чем десятичное 65536, то подпрограмма возвращается с установленным признаком переноса. Если 'последнее значение' отрицательно, сбрасывается признак нуля.
Также копируется в регистр A младший байт результата.

```
2DA2 FP-TO-BC     RST   0028,FP-CALC       Использование калькулятора
                  DEFB  +38,end-calc       для указания с помощью HL
                                           STKEND-5.
                  LD    A,(HL)             Получение байта порядка
                  AND   A                  'последнего значения';
                  JR    Z,2DAD,FP-DELETE   если ноль, переход с обозна-
                                           чением 'малое целое число'.
                  RST   0028,FP-CALC       Теперь использование
                  DEFB  +A2,stk-half       калькулятора для округления
                  DEFB  +0F,addition       'последнего значения' до
                  DEFB  +27,int            ближайшего целого, который
                  DEFB  +38,end-calc       также изменяет его до формы
                                           'малое целое число' на стеке
                                           калькулятора, если это
                                           возможно, т.е. если
                                           -65535.5 <= x < 65535.5.
2DAD FP-DELETE    RST   0028,FP-CALC       Использование калькулятора
```

```
                DEFB  +02,delete          для удаления целого числа из
                DEFB  +38,end-calc        стека; DE все еще указывает
                                          на разряд в памяти (STKEND).
                PUSH  HL                  Запись обоих указа-
                PUSH  DE                  телей стека.
                EX    DE,HL               Теперь HL указывает на число.
                LD    B,(HL)              Копирование первого байта
                                          в B.
                CALL  2D7F,INT-FETCH      Копирование 2, 3 и 4 байта
                                          в C, E и D.
                XOR   A                   Очистить регистр A.
                SUB   B                   Если B не 0, задается
                                          перенос.
                BIT   7,C                 Если число положительное
                                          (NZ обозначает отрицатель-
                                          ное), задается признак 0.
                LD    B,D                 Скопировать старший
                LD    C,E                 байт в B, а младший в C.
                LD    A,E                 Также в A скопировать
                                          младший байт.
                POP   DE                  Восстановить указатель
                POP   HL                  стека.
                RET                       Окончание.
```

ПОДПРОГРАММА 'LOG (2^A)'
Эта подпрограмма вызывается подпрограммой 'PRINT-FP' для вычисления количества цифр перед десятичной точкой в X, или, если нет цифр до точки, то количество головных нулей после нее. Она вводится с регистром A, содержащим e', истинный порядок x, или e'-2 и вычисляет z, равное логарифму с основанием 10 величины (2^A). Затем A задается равным ABS INT (z + 0.5), используя для этого FP-TO-A.

```
2DC1 LOG (2^A)  LD    D,A                 Целое число A заносится в
                RLA                       стек, или как 00 00 A 00 00
                SBC   A,A                 (для положительного A), или
                                          00 FF A FF 00 (для отрица-
                                          тельного A).
                LD    E,A                 Эти байты сначала загружают-
                LD    C,A                 ся в A, E, D, C, B, а затем
                XOR   A                   вызывается STK-STORE, чтобы
                LD    B,A                 занести число в стек
                                          калькулятора.
                CALL  2AB6,STK-STORE
                RST   0028,FP-CALC        Используется калькулятор.
                DEFB  +34,stk-data        Теперь заносится в стек
                                          log 2 с основанием 10.
                DEFB  +EF,exponent +7F    Стек содержит A, log 2.
                DEFB  +1A,+20,+9A,+85
                DEFB  +04,multiply        A*log 2, т.е. log (2^A)
                DEFB  +27,int             INT log (2^A).
                DEFB  +38,end-calc
```

Подпрограмма для полного завершения вычислений продолжается в FP-TO-A.

ПОДПРОГРАММА 'FLOATING-POINT TO A' ('Число с плавающей точкой в A)
Эта короткая, но важная программа вызывается, по крайней мере,

для различных целей 8 раз. Она использует подпрограмму FP-TO-BC для получения 'последнего значения' в регистре A. Она проверяет, округляется ли модуль числа до числа, большего чем 255, и, если да, то возврат с установленным признаком переноса. В противном случае возврат с модулем числа, округленного до ближайшего целого, в регистре A, и с установленным признаком нуля, если число было положительным, или сброшенным, если оно отрицательно.

```
2DD5 FP-TO-A      CALL  2DA2,FP-TO-BC       Помещение 'последнего
                                            значения' в BC.
                  RET   C                   Возврат, если уже вне
                                            диапазона.
                  PUSH  AF                  Запись результата и
                                            признаков.
                  DEC   B                   Если регистр B не содержит
                  INC   B                   0, снова будет вне
                                            диапазона.
                  JR    Z,2DE1,FP-A-END     Бели в диапазоне, переход.
                  POP   AF                  Выбор результата и признаков.
                  SCF                       Сигнал - результат вне
                                            диапазона.
                  RET                       Окончание - неудачное.
2DE1 FP-A-END     POP   AF                  Выбор результата и признаков.
                  RET                       Успешное завершение.
```

ПОДПРОГРАММА 'PRINT A FLOATING-POINT NUMBER' ('Печать числа с плавающей точкой')
Эта подпрограмма, вызываемая командной процедурой PRINT в 2039 и STR$ в 3630, преобразует в строку числа, которые необходимо напечатать. Подпрограмма печатает x, 'последнее значение' на стеке калькулятора. Формат печати никогда не занимает более чем 14 символов.
8 наиболее значащих цифр x, правильно округлённых, запоминаются в специальном буфере печати в mem-3 и в mem-4. Малые числа, меньшие 1, и большие числа, большие 2 ^ 27, обрабатываются раздельно. Формирователь умножается на 10 ^ n, где n это соответствующее число головных нулей после десятичной точки, в то время как остаток делится на 10 ^ (n-7), где n число цифр до десятичной точки. Таким образом, все числа приводятся к среднему диапазону, а число цифр, требуемое до десятичной точки, вносится во второй байт mem-5. Печать производится с использованием E-формата, если перед десятичной течкой больше 8 цифр, или, для малых чисел, если после десятичной точки находится более 4 головных нулей.
Следующая программа показывает диапазон форматов печати:

```
10 FOR a=-11 TO 12: PRINT SGN a*9^a,: NEXT a
```

1. Сначала обрабатывается знак x:
   Если x отрицательный, подпрограмма переходит на PF-NEGATIVE, берет ABS x и печатает знак минус.
   Если x - ноль, x удаляется из стека калькулятора, '0' печатается и осуществляется возврат из подпрограммы.
   Если x положительно, подпрограмма продолжается.

```
2DE3 PRINT-FP     RST   0028,FP-CALC        Использование калькулятора.
                  DEFB  +31,duplicate       x,x
                  DEFB  +36,less-0          x, (1/0) Логическое
                                            значение x.
                  DEFB  +00,jump-true       x
                  DEFB  +0B,to PF-NEGTVE    x
                  DEFB  +31,duplicate       x,x
```

```
                DEFB  +37,greater-0       x,(1/0) Логическое
                                          значение x.
                DEFB  +00,jump-true       x
                DEFB  +0D,to PF-POSTVE    x Затем x'=ABS x
                DEFB  +02,delete          -
                DEFB  +38,end-calc        -
                LD    A,+30               Ввести код символа для '0'.
                RST   0010,PRINT-A-1      Печать '0'.
                RET                       Окончание, т.к. 'последнее
                                          значение' = 0.
2DF2 PF-NEGTVE  DEFB  +2A,abs             x' x'=ABS x.
                DEFB  +38,end-calc        x'
                LD    A,+2D               Ввести код символа для '-'.
                RST   0010,PRINT-A-1      Печать '-'.
                RST   0028,FP-CALC        Опять использование
                                          калькулятора.
2DF8 PF-POSTVE  DEFB  +A0,stk-zero        15 байт mem-3, mem-4 и mem-5
                DEFB  +C3,st-mem-3        теперь изначально заданы
                DEFB  +C4,st-mem-4        нулевыми для использования в
                DEFB  +C5,st-mem-5        буфере печати и двух
                                          счетчиках.
                DEFB  +02,delete          Очищен стек, кроме x'.
                DEFB  +38,end-calc        x'
                EXX                       H'L', который используется,
                PUSH  HL                  чтобы содержать смещения
                EXX                       калькулятора (например, для 'STR$'),
                                          записан в машинном стеке.
```

2. Это начало цикла, который работает с большими числами. Однако, каждое число x сначала разделяется на целую часть i и дробную f. Если i является малым целым числом, т.е. если -65535 <= i <= 65535, оно запоминается в D'E' для занесения в буфер печати.

```
2E01 PF-LOOP    RST   0028,FP-CALC        Опять использование
                                          калькулятора.
                DEFB  +31,duplicate       x', x'
                DEFB  +27,int             x', INT (x')=i
                DEFB  +C2,st-mem-2        (i запоминается в mem-2)
                DEFB  +03,subtract        x'-i=f
                DEFB  +E2,get-mem-2       f,i
                DEFB  +01,exchange        i,f
                DEFB  +C2,st-mem-2        (f запоминается в mem-2)
                DEFB  +03,delete          i
                DEFB  +38,end-calc        i
                LD    A,(HL)              i малое целое число
                AND   A                   (первый байт - ноль),
                                          т.е. ABS i <= 65535?
                JR    NZ,2E56,PF-LARGE    Переход, если нет.
                CALL  2D7F,INT-FETCH      i копируется в DE
                                          (i, подобно x', >=0).
                LD    B,+10               B устанавливается для
                                          отсчета 16 бит.
                LD    A,D                 D копируется в A для
                AND   A                   проверки: оно ноль?
                JR    NZ,2E1E,PF-SAVE     Переход, если не ноль.
                OR    E                   Теперь проверка E.
                JR    Z,2E24,PF-SMALL     Переход, если DE ноль:
                                          x состоит только из
                                          дробной части.
```

```
                LD    D,E               Перемещение E в D и
                LD    B,+08             установка B для 8 разрядов:
                                        D было ноль, E - не ноль.
2E1E PF-SAVE    PUSH  DE                Передать DE в D'E',
                EXX                     через машинный стек
                POP   DE                для пересылки в бу-
                EXX                     фер печати на PF-BITS
                JR    2E78,PF-BITS      Переход вперед.
```

3. Чистые дробные части умножается на 10^n, где n соответствующее число головных нулей после десятичной точки; -n добавляется ко второму байту mem-5, который содержит количество необходимым цифр перед десятичной точкой; отрицательное число обозначает головные нули после десятичной точки.

```
2E24 PF-SMALL   RST   0028,FP-CALC      i (i=0 здесь),
                DEFB  +E2,get-mem-2     i, f
                DEFB  +38,end-calc      i, f
```

Отметим, что стек теперь неуравновешен. Лишний байт 'DBFB+02, delete' необходим в 2E25, сразу же за RST 0028. Теперь выражение типа '2' +STR$ 0.5 вычисляется неправильно как 0.5; ноль, оставленный на стеке, заменяет '2' и обрабатывается как пустая строка. Аналогично, все сравнения строк могут вырабатывать неправильные значения, если вторая строка берет форму STR$ x, где x численно меньше 1; например, выражение '50'<STR$ 0.1 вырабатывает логическое значение 'истина'; '' еще раз используется вместо '50'.

```
                LD    A,(HL)            В A скопирован байт
                                        порядка e величины f.
                SUB   +7E               A становится e - 126
                                        десятичное, т.е. e'+2, где
                                        e' - истинный порядок f.
                CALL  2DC1,LOG (2^A)    Конструкция A=ABS INT
                                        (LOG (2^A)) выполнена
                                        (LOG с основанием 10), т.е
                LD    D,A               A=n, n скопировано из A в D.
                LD    A,(mem-5-2nd)     Текущее значение счетчика,
                SUB   D                 полученное из
                LD    (mem-5-2nd),A     второго байта mem-5,
                                        a n вычитается из него.
                LD    A,D               n копируется из D в A.
                CALL  2D4F,E-TO-FP      y=f*10^n формируется и
                                        помещается на стек.
                RST   0028,FP-CALC      i, y
                DEFB  +31,duplicate     i, y, y1,У
                DEFB  +27,int           i, y, (INT (y) = i21.У.У
                DEFB  +C1,st-mem-1      (i2 копируется в mem-1).
                DEFB  +03,subtract      i, y - i2
                DEFB  +E1,get-mem-1     i, y - i2, i2
                DEFB  +38,end-calc      i, f2, i2 (f2 = y - i2)
                CALL  2DD5,FP-TO-A      i2 передано из стека в A.
                PUSH  HL                Запись указателя в f2.
                LD    (mem-3-1st),A     i2 записывается в первом
                                        байте mem-3: цифра для
                                        печати.
                DEC   A                 i2 не будет подсчитываться
                RLA                     как цифра для пеяати, если
                SBC   A,A               оно ноль; A обработано так,
```

```
                INC   A                 что ноль будет создавать
                                        ноль, но ненулевая цифра
                                        будет создавать 1.
                LD    HL,+5CAB          О или 1 вставляются в первый
                LD    (HL),A            байт mem-5 (число цифр для
                INC   HL                печати) и добавляется ко
                ADD   A,(HL)            второму байту mem-5 (число
                LD    (HL),A            цифр перед десятичной
                                        точкой).
                POP   HL                Запоминается указатель f2.
                JP    2ECF,PF-FRACTN    Переход, чтобы запомнить f2
                                        в буфере (HL теперь
                                        указывает на f2, DE в i2.
```

4. Числа, большие, чем 2 ^ 27 умножаются на 2 ^ (-n+7), уменьшая число цифр перед десятичной точкой до 8, а цикл переводится в PF-LOQP.

```
2E56 PF-LARGE   SUB   +80              e - 80 (16-ричное) = e',
                                       истинный порядок i.
                CP    +1C              e' меньше, чем 28
                                       десятичное?
                JR    C,2E6F,PF-MEDIUM Переход, если меньше.
                CALL  2DC1,LOG (2^A)   n сформировано в A.
                SUB   +07              И уменьшено до n - 7.
                LD    B,A              Затем скопировано в B.
                LD    HL,+5CAC         n - 7 добавлено во вто-
                ADD   A,(HL)           рой байт mem-5, число
                LD    (HL),A           цифр, требуемое перед
                                       десятичной точкой в x.
                LD    A,B              Затем i умножается на
                                       10^(-n+7).
                NEG                    Это приводит его к среднему
                CALL  2D4F,E-TO-FP     диапазону для печати.
                JR    2E01,PF-LOOP     Пустить цикл опять, чтобы
                                       обработать числа средних
                                       размеров.
```

5. Целая часть x теперь запоминается в буфере печати в mem-3 и mem-4.

```
2E6F PF-MEDIUM  EX    DE,HL            DE теперь указывает на i,
                                       HL на f.
                CALL  2FBA,FETCH-TWO   Мантисса i теперь в
                                       D',E',D,E.
                EXX                    Прочитать регистры обмена.
                SET   7,D              Истинный числовой разряд
                                       7 в D'.
                LD    A,L              Байт порядка e зна-
                                       чения i в A.
                EXX                    Назад на основные регистры.
                SUB   +80              Истинный порядок e'=e - 80,
                                       шестнадцатеричное, в A.
                LD    B,A              Задается требуемый счет
                                       разрядов.
```

Отметим, что в случае, если i является малым целым числом (меньше 65536), то снова вход здесь.

```
2E7B PF-BITS    SLA   E                Мантисса i теперь прокручивается
                RL    D                влево и все разряды i
```

```
                EXX                     тогда будут смещены в mem-4,
                RL    E                 а каждый байт mem-4
                                        является десятичным,
                                        установленным на каждое
                                        смещение.
                RL    D                 Все 4 байта i.
                EXX                     Назад на основной регистр.
                LD    HL,+5CAA          Адрес 5 байта mem-4
                LD    C,+05             в HL; счет 5 байт в C.
2E8A PF-BYTES   LD    A,(HL)            Прочитать байт в mem-4.
                ADC   A,A               Сдвинуть его влево, взяв
                                        в новом разряде.
                DAA                     Десятичная коррекция байта.
                LD    (HL),A            Восстановить его в mem-4.
                DEC   HL                Указать следующий байт в
                                        mem-4.
                DEC   C                 Понизить число байт на 1.
                JR    NZ,2E8A,PF-BYTES  Переход для каждого байта
                                        mem-4.
                DJNZ  2E7B,PF-BITS      Переход для каждого разряда
                                        INT(x).
```

Десятичная коррекция каждого байта mem-4 дала две десятичные цифры на байт, самое большее возможно 9 цифр. Цифры теперь будут перепакованы, одна на байт, в mem-3 и mem-4, используя команду RLD.

```
                XOR   A                 Для получения цифр очищено A.
                LD    HL,+5CA6          Исходный адрес: первый байт
                                        mem-4.
                LD    DE,+5CA1          Адрес назначения: первый
                                        байт mem-3.
                LD    B,+09             Самое большее 9 цифр.
                RLD                     Отброшен левый полубайт
                                        mem-4.
                LD    C,+FF             FF в C будет обозначать
                                        головные кули,
                                        00 - неголовные нули.
2EA1 PF-DIGITS  RLD                     Левый полубайт (HL) в A,
                                        правый - налево.
                JR    NZ,2EA9,PF-INSERT Переход, если цифра в A не
                                        ноль.
                DEC   C                 Проверка для головных нулей.
                INC   C                 Задается сброс нуля.
                JR    NZ,2EB3,PF-TEST-2 Если головной ноль, переход.
2EA9 PF-INSERT  LD    (DE),A            Теперь вставить цифру.
                INC   DE                Указывает следующий адрес
                                        назначения.
                INC   (mem-5-1st)       Еще одна цифра для печати и
                INC   (mem-5-2nd)       одна перед десятичной точкой.
                LD    C,+00             Изменение признака из голов-
                                        ного нуля в другой ноль.
2EB3 PF-TEST-2  BIT   0,B               Когда B является нечетным,
                JR    Z,2EB8,PF,ALL-9   исходный счетчик нуждается в
                INC   HL                уменьшении на каждом втором
                                        проходе через цикл.
2EB8 PF-ALL-9   DJNZ  2EA1,PF-DIGITS    Переход назад для всех 9
                                        цифр.
                LD    A,(mem-5-1st)     Прочитать счетчик:
                SUB   +09               были 9 цифр, исключая
```

```
                                          головные нули.
                JR    C,2ECB,PF-MORE      Если нет, переход, чтобы
                                          прочитать больше цифр.
                DEC   (mem-5-1st)         Подготовка округления:
                                          понизить до 8.
                LD    A,+04               Сравнить 9-ю цифру, 4-й
                                          байт mem-4, с 4,
                CP    (mem-4-4th)         чтобы задать переход для
                                          округления в большую
                                          сторону.
                JR    2F0C,PF-ROUND       Переход вперед для округле-
                                          ния в большую сторону.
2ECB PF-MORE    RST   0028,FP-CALC        Опять использовать
                                          калькулятор.
                DEFB  +02,delete          -(i удалено).
                DEFB  +E2,get-mem-2       f
                DEFB  +38,end-calc        f
```

6. Дробная часть x запоминается в буфере печати.

```
2ECF PF-FRACTN  EX    DE,HL               Теперь DE указывает на f.
                CALL  2FBA,FETCH-TWO      Мантисса f в D',E',D,E.
                EXX                       Получить регистры обмена.
                LD    A,+80               Порядок f понижается
                SUB   L                   до нуля с помощью
                LD    L,+00               сдвига разрядов f вправо на
                                          80 (16-ричное) - e позиций,
                                          где L' содержит e.
                SET   7,D                 Истинный числовой разряд в 7
                                          разряд D'.
                EXX                       Восстановление основных
                                          регистров.
                CALL  2FDD,SHIFT-FP       Создание сдвига.
2EDF PF-FRN-LP  LP    A,(mem-5-1st)       Прочитать число цифр.
                CP    +08                 Уже 8 цифр?
                JR    C,2EEC,PR-FR-DGT    Если нет, переход вперед.
                EXX                       Если 8 цифр, то используйте
                RL    D                   f, чтобы округлить i в боль-
                                          шую сторону, сдвигая влево
                                          D' для установки переноса.
                EXX                       Восстановление основных
                JR    2F0C,PF-ROUND       регистров и переход вперед
                                          для округления в большую
                                          сторону.
2EEC PF-FR-DGT  LD    BC,+0200            Инициализировать ноль
                                          в C, число 2 в B.
2EEF PF-FR-EXX  LD    A,E                 D'E'DE умножаются на
                CALL  2F8B,CA=10*A+C      10 в два этапа, сначала
                LD    E,A                 DE, затем D'E',
                LD    A,D                 байт за байтом за 2
                                          шага, а целая часть
                CALL  2F8B,CA=10*A+C      результата, получен-
                LD    D,A                 ная в C, передается в
                                          буфер печати.
                PUSH  BC                  Число и результат пе-
                EXX                       реставляется между
                POP   BC                  BC и B'C'.
                DJNZ  2EEF,PF-FR-EXX      Просмотр регистров обмена.
                LD    HL,+5CA1            Начало - первый байт mem-3.
```

```
                LD    A,C               Результат для запоминания
                                        в A.
                LD    C,(mem-5-1st)     Подсчет цифр в C.
                ADD   HL,BC             Адресовать первый пустой
                                        байт.
                LD    (HL),A            Запомнить следующую цифру.
                INC   (mem-5-1st)       Провести счет цифр.
                JR    2EDF,PF-FRN-LP    Прогнать цикл для 8 цифр.
```

7. Запомненные в буфере печати цифры округляются до 8 цифр и подлежат печати.

```
2F0C PF-ROUND   PUSH  AF                Запись признака переноса для
                                        округления.
                LD    HL,+5CA1          Запись адреса числа: mem-3,
                                        байт 1.
                LD    C,(mem-5-1st)     Смещение(номер цифры
                LD    B,+00             в числе) в BC.
                ADD   HL,BC             Адрес последнего байта числа.
                LD    B,C               Скопировать C в качестве
                                        счетчика в B.
                POP   AF                Восстановление признака
                                        переноса.
2F18 PF-RND-LP  DEC   HL                Это последний байт числа.
                LD    A,(HL)            Прочитать байт в A.
                ADC   A,+00             Добавить перенос, т.е.
                                        округлить в большую
                                        сторону.
                LD    (HL),A            Запомнить округленный байт
                                        в буфере.
                AND   A                 Если байт =0 или 10,
                JR    Z,2F25,PF-R-BACK  B будет уменьшено и
                CP    +0A               конечный 0 (или 10) не будет
                                        отсчитываться для печати.
                CCF                     Сброс переноса для
                                        правильной цифры.
                JR    NC,2F2D,PF-COUNT  Переход, если сброшен
                                        перенос.
2F25 PF-R-BACK  DJNZ  2F18,PF-RND-LP    Переход назад для дополни-
                                        тельного округления и допол-
                                        нительных конечный нулей.
                LD    (HL),+01          Переполнение влево,
                INC   B                 необходима дополнительная 1.
                INC   (mem-5-2nd)       Дополнительная цифра перед
                                        десятичной точкой.
2F2D PF-COUNT   LD    (mem-5-1st),B     B теперь отсчитывает цифры
                                        для печатания (конечные нули
                                        печататься не будут).
                RST   0028,FP-CALC      f подлежит удалению.
                DEFB  +02,delete        -
                DEFB  +38,end-calc      -
                EXX                     Смещение калькулятора,
                POP   HL                записанное в стеке,
                EXX                     восстанавливается в H'L'.
```

8. Теперь можно напечатать число. Сначала будет установлено C, чтобы содержать число цифр, подлежащее печати, не считая конечные нули, в то время как в B будет находиться число цифр, требуемых перед десятичной точкой.

```
                LD    BC,(mem-5-1st)      Заданы счетчики.
                LD    HL,+5CA1            Начало цифр.
                LD    A,B                 Если цифр, требуемых перед
                CP    +09                 десятичной тачкой больше 9,
                JR    C,2F46,PF-NOT-E     или меньше, чем -4,
                                          потребуется формат E.
                CP    +FC                 Меньше, чем 4, означает, что
                JR    C,2F6C,PF-E-FRMT    за десятичной точкой более
                                          4 головных нулей.
2F46 PF-NOT-E   AND   A                   Перед десятичной точкой нет
                CALL  Z,15EF,OUT-CODE     цифр? Если так, то
                                          напечатать начальные нули.
```

Следующая точка входа используется для печати цифр, нуждающихся в E-формате.

```
2F4A PF-E-SBRN  XOR   A                   Сначала A задается нулем.
                SUB   B                   Вычесть B: минус будет
                JR    M,2F52,PF-OUT-LP    означать, что цифры перед
                                          десятичной точкой; переход
                                          вперед для их печати.
                LD    B,A                 A требуется как счетчик.
                JR    2F5E,PF-DC-OUT      Переход вперед для печати
                                          десятичной части.
2F52 PF-OUT-LP  LD    A,C                 Скопировать количество цифр,
                AND   A                   подлежащих печати, в A.
                JR    Z,2F59,PF-OUT-DT    Если A это 0, то печатаются
                                          еще конечные нули (B не
                                          ноль); переход.
                LD    A,(HL)              Прочитать цифру на буфере
                                          печати.
                INC   HL                  Указать следующую цифру.
                DEC   C                   Уменьшить отсчет на 1.
2F59 PF-OUT-DT  CALL  15EF,OUT-CODE       Печать соответствующей цифры.
                DJNZ  2F52,PF-OUT-LP      Прогон цикла, пока B не
                                          станет нулем.
2F5E PF-DC-OUT  LD    A,C                 Пора печатать десятичную
                AND   A                   точку, если C не ноль;
                RET   Z                   иначе возврат-окончание.
                INC   B                   Добавить 1 к B - включить
                                          десятичную точку.
                LD    A,+2E               Занести в A код для '.'.
2F64 PF-DEC-0S  RST   0010,PRINT-A-1      Печать '.'.
                LD    A,+30               Ввести код символа для '0'.
                DJNZ  2F64,PF-DEC-0S      Прогон цикла, чтобы напеча-
                                          тать все необходимые нули.
                LD    B,C                 Задать отсчет всех
                                          оставшихся цифр.
                JR    2F52,PF-OUT-LP      Переход назад для их печати.
2F6C PF-E-FRMT  LD    D,B                 Счет цифр, скопированных в D.
                DEC   D                   Понижение для задания
                                          порядка.
                LD    B,+01               Перед десятичной точкой в
                                          формате E требуется одна
                                          цифра.
                CALL  2F4A,PF-E-SBRN      Все части числа перед 'E'
                                          напечатаны.
                LD    A,+45               Ввод кода символа для 'E'.
                RST   0010,PRINT-A-1      Печать 'E'.
```

```
                  LD    C,D               Порядок для печати заносится
                                          в C.
                  LD    A,C               И в A для проверки.
                  AND   A                 Проверка знака.
                  JP    P,2F83,PF-E-POS   Если он положителен, переход.
                  NEG                     В противном случае,
                                          инвертировать его в A.
                  LD    C,A               Затем повторное копирование
                                          в C для печати.
                  LD    A,+2D             Ввод кода символа для '-'.
                  JR    2F85,PF-E-SIGN    Переход для печати знака.
2F83 PF-E-POS     LD    A,+2B             Ввод кода символа '+'.
2F85 PF-E-SIGN    RST   0010,PRINT-A-1    Теперь печать знака:
                                          '+' или '—'.
                  LD    B,+00             В BC находится порядок для
                                          печати.
                  JP    1A1B,OUT-NUM      Переход назад, чтобы
                                          напечатать его, и окончание.
```

ПОДПРОГРАММА 'CA=10*A+C'
Эта подпрограмма вызывается подпрограммой PRINT-FP для умножения каждого байта D'E'DE на 10 и возврата целой части результата в регистр C. На входе регистр A содержит байт, умноженный на 10, а регистр C содержит перенос из предыдущего байта. На выходе регистр A содержит результирующий байт, а регистр C - переход вперед к следующему байту.

```
2F8B CA=10*A+C    PUSH  DE                Запись какой угодно пары DE
                                          для использования.
                  LD    L,A               Скопировать множимое
                  LD    H,+00             из A в HL.
                  LD    E,L               Также скопировать
                  LD    D,H               его в DE.
                  ADD   HL,HL             Сдублировать HL.
                  ADD   HL,HL             Еще раз.
                  ADD   HL,DE             Добавить в DE, чтобы задать
                                          HL=5*A.
                  ADD   HL,HL             Сдублировать еще раз теперь
                                          HL=10*A.
                  LD    E,C               Скопировать C в DE (D-ноль)
                                          для суммирования.
                  ADD   HL,DE             Теперь HL=10*A+C.
                  LD    C,H               H скопировано в C .
                  LD    A,L               L копируется в A, завершая
                                          задачу.
                  POP   DE                Восстанавливается регистро-
                                          вая пара в DE.
                  RET                     Окончание.
```

ПОДПРОГРАММА 'PREPARE TO ADD' ('Подготовка к сложению')
Эта подпрограмма является первой из четырех подпрограмм, которые используются программами 4-х арифметических действий - SUBTRACTION, ADDITION, MULTIPLICATION и DIVISION.
Эта конкретная подпрограмма готовит число с плавающей тачкой к сложению, главным образом, заменяя знаковый разряд первым истинным числовым разрядом и инвертируя (дополняя до 2) число, если оно отрицательно. Порядок возвращается в регистре A, первый байт задается 00 (шестнадцатеричное) для положительного числа и FF (шестнадцатеричное) для отрицательного.

```
2F9B PREP-ADD     LD    A,(HL)              Передать порядок в A.
                  LD    (HL),+00            Допустим, что число
                                            положительно.
                  AND   A                   Если число равно 0,
                  RET   Z                   то подготовка закончена.
                  INC   HL                  Теперь указать знаковый байт.
                  BIT   7,(HL)              Задать признак нуля
                                            для положительного числа.
                  SET   7,(HL)              Восстановить истинный
                                            числовой разряд.
                  DEC   HL                  Указать опять первый байт.
                  RET   Z                   Положительные числа подго-
                                            товлены, но отрицательные
                                            необходимо дополнить до 2.
                  PUSH  BC                  Запись любого более раннего
                                            порядка.
                  LD    BC,+0005            Подлежат обработке пять
                                            байтов.
                  ADD   HL,BC               Указать на единицу после
                                            последнего байта.
                  LD    B,C                 Передать '5' в B.
                  LD    C,A                 Запись порядка в C.
                  SCF                       Задать признак переноса
                                            для инвертирования.
2FAF NEG-BYTE     DEC   HL                  Указать по очереди каждый
                                            байт.
                  LD    A,(HL)              Прочесть каждый байт.
                  CPL                       Дополнить байт до единицы.
                  ADC   A,+00               Добавить перенос для
                                            инвертирования.
                  LD    (HL),A              Восстановить байт.
                  DJNZ  2FAF,NEG-BYTE       Прогнать цикл '5' раз.
                  LD    A,C                 Восстановить порядок в A.
                  POP   BC                  Восстановить любой более
                                            ранний порядок.
                  RET                       Окончание.
```

ПОДПРОГРАММА 'FETCH TWO NUMBERS' ('Выбор двух чисел')
Эта подпрограмма вызывается ADDITION, MULTIPLICATION и DIVISION, чтобы получить два числа из стека калькулятора, а затем поместить их в регистр, включая регистры обмена.
На входе в подпрограмму регистровая пара HL указывает первый байт первого числа, регистровая пара DE указывает первый байт второго числа.
Когда подпрограмма вызывается MULTIPLICATION или DIVISION знак результата записывается во второй байт первого числа.

```
2FBA FETCH-TWO    PUSH  HL                  HL сохранен.
                  PUSH  AF                  AF сохранен.

Вызвать пять байт первого числа - M1,M2,M3,M4 и M5.
И второго числа                - N1,N2,N3,N4 и N5.

                  LD    C,(HL)              M1 в C.
                  INC   HL                  Следующий.
                  LD    B,(HL)              M2 в B.
                  LD    (HL),A              Скопировать знак результата
```

```
                                в (HL).
                INC   HL        Следующий.
                LD    A,C       M1 в A.
                LD    C,(HL)    M3 в C.
                PUSH  BC        Записать M2 и M3 на машинный
                                стек.
                INC   HL        Следующий.
                LD    C,(HL)    M4 в C.
                INC   HL        Следующий.
                LD    B,(HL)    M5 в B.
                EX    DE,HL     HL указывает на N1.
                LD    D,A       M1 в D.
                LD    E,(HL)    N1 в E.
                PUSH  DE        Записать M1 и N1 на машинный
                                стек.
                INC   HL        Следующий.
                LD    D,(HL)    N2 в D.
                INC   HL        Следующий.
                LD    E,(HL)    N3 в E.
                PUSH  DE        Записать N2 и N3 на машинный
                                стек.
                EXX             Получить регистры обмена.
                POP   DE        N2 в D' и N3 во E'.
                POP   HL        M1 в H' и N1 в L'.
                POP   BC        M2 в B' и M3 в C'.
                EXX             Получить исходный набор регистров.
                INC   HL        Следующий.
                LD    D,(HL)    N4 в D.
                INC   HL        Следующий.
                LD    E,(HL)    N5 в E.
                POP   AF        Восстановить исходный AF.
                POP   HL        Восстановить исходный HL.
                RET             Окончание.

Итого:          M1 - M5 в H', B', C', C, B.
     N1 - N5 в: L', D', E', D, E.
     HL указывает на первый байт первого числа.
```

ПОДПРОГРАММА 'SHIFT ADDEND'
Подпрограмма сдвигает число с плавающей точкой до 32 (десятичное), 20 (шестнадцатеричное) позиции вправо, чтобы выровнять его для суммирования. Число с меньшим порядком было занесено на позицию второго слагаемого перед вызовом подпрограммы. Любое переполнение вправо к переносу добавляется. Если разность порядков больше 32 (десятичное) или перенос сдвинулся назад к началу числа, то число задается нулем и сложение не изменяет другого числа.

```
2FDD SHIFT-FP   AND   A                 Если разность порядков равна
                RET   Z                 0, то подпрограмма сразу же
                CP    +21               возвращается, если разность
                JR    NC,2FF9,ADDEND-0  больше 20 (16-ричное), то
                                        переход вперед.
                PUSH  BC                Запись BC.
                LD    B,A               Передать разность порядков в
                                        B, чтобы подсчитать сдвиг
                                        вправо.
```

```
2FE5 ONE-SHIFT    EXX                    Арифметический сдвиг
                  SRA   L                вправо для 'L', предохра-
                                         няющий разряды знака.
                  RR    D                Сдвиг вправо с переносом
                  RR    E                D', E', D и E.
                  EXX                    Перемещение всех пяти
                  RR    D                байт числа вправо столько
                  RR    E                раз, сколько считает B.
                  DJNZ  2FE5,ONE-SHIFT   Прогон цикла, пока B не
                                         достигнет нуля.
                  POP   BC               Восстановление исходного BC.
                  RET   NC               Сделано, если не найден
                                         перенос.
                  CALL  3004,ADD-BACK    Отыскать перенос.
                  RET   NZ               Возврат, если перенос не
                                         распространяется обратно
                                         (в этом случае нечего
                                         складывать).
2FF9 ADDEND-0     EXX                    Выбор L', D' и E'.
                  XOR   A                Очистить регистр A.
2FFB ZEROS-4/5    LD    L,+00            Задать нулем в D',E' второе
                                         слагаемое.
                  LD    D,A              D и E вместе с их маркерным
                  LD    E,L              байтом L' (индикатор знака),
                  EXX                    который 00 (16-ричное) для
                  LD    DE,+0000         положительного числа и FF
                                         (16-ричное) для отрицательного.
                                         ZEROS-4/5 создает только 4
                                         нулевых байта при вызове при
                                         почти полной потере значи-
                                         мости в 3160.
                  RET                    Окончание.
```

ПОДПРОГРАММА 'ADD-BACK' ('Добавление обратно')
Эта подпрограмма добавляет обратно к числу любой перенос, который был с переполнением вправо. В крайнем случае, перенос распространяется назад влево от числа. Когда подпрограмма вызывается во время сложения, это распространение означает, что мантисса числа 0.5 сдвинута на полные 32 позиции вправо, а второе слагаемое теперь будет нулем; когда она вызывается при MULTIPLICATION, это означает, что порядок должен быть изменен с приращением, а это может привести к переполнению.

```
3004 ADD-BACK     INC   E                Добавить перенос к крайнему
                                         правому байту.
                  RET   NZ               Возврат, если слева нет
                                         переполнения.
                  INC   D                Продолжать со следующим
                                         байтом.
                  RET   NZ               Возврат, если слева нет
                                         переполнения.
                  EXX                    Прочитать следующий байт.
                  INC   E                Дать ему тоже приращение.
                  JR    NZ,300D,ALL-ADDED Переход, если нет
                                         переполнения.
                  INC   D                Изменить приращением
                                         последний байт.
300D ALL-ADDED    EXX                    Восстановить исходные
```

```
                                        регистры.
                RET                     Окончание.
```

ОПЕРАЦИЯ 'SUBTRACTION' ('Вычитание')
(Смещение 03 - см. CALCULATE ниже: 'вычитание')
Эта подпрограмма просто изменяет знак вычитаемого и переходит в ADDITION.
Отметим, что HL указывает уменьшаемое, а DE вычитаемое (подробнее см. ADDITION).

```
300F SUBTRACT   EX    DE,HL             Перестановка указателей.
                CALL  346E,NEGATE       Изменить знак вычитаемого.
                EX    DE,HL             Перестановка указателей
                                        обратно и продолжение в
                                        ADDITION.
```

ОПЕРАЦИЯ 'ADDITION' ('Сложение')
(Смещение 0F - см. CALCULATE ниже: 'сложение')
Первая из трех важнейших арифметических подпрограмм; эта подпрограмма выдает сложение в форме с плавающей точкой двух чисел, каждое с 4-байтовой мантиссой и 1-байтным порядком. В этих трех подпрограммах два числа на вершине стека калькулятора складываются / умножаются / делятся, чтобы выдать одно число на вершине стека как 'последнее значение'.
HL показывает второе число из вершины, первое слагаемое / множитель / делимое. DE указывает на число на вершине стека калькулятора, второе слагаемое / множимое / делитель. После чего HL указывает на результирующее 'последнее значение', чей адрес может быть также рассмотрен для STKEND - 5.
Но подпрограмма сложения сначала проверит, являются ли 2 числа, подлежащих сложению, 'малыми целыми числами'. Если это так, то они просто складываются в HL и BC, а результат заносится в стек. Перед или после сложения требуется дополнение до двух, хотя такие числа содержатся на стеке, готовые для сложения в форме дополнения до двух.

```
3014 addition   LD    A,(DE)            Проверка, являются ли первые
                OR    (HL)              байты обоих чисел нулями.
                JR    NZ,303E,FULL-ADDN Если нет, переход для
                                        полного сложения.
                PUSH  DE                Запись указателя второго
                                        числа.
                INC   HL                Указывает второй байт
                PUSH  HL                первого числа и запись этого
                                        указателя.
                INC   HL                Указать менее значимый байт.
                LD    E,(HL)            Выбор его в E.
                INC   HL                Указать более значимый байт.
                LD    D,(HL)            Выбор его в D.
                INC   HL                Сместиться ко второму
                INC   HL                байту второго
                INC   HL                числа.
                LD    A,(HL)            Выбрать его в A (это
                                        знаковый байт).
                INC   HL                Указать менее значимый байт.
                LD    C,(HL)            Выбор его в C.
                INC   HL                Указать более значимый байт.
                LD    B,(HL)            Выбор его в B.
                POP   HL                Выбрать указатель знакового
                EX    DE,HL             байта первого числа; занести
                                        его в DE, а число в HL.
```

```
                ADD   HL,BC             Выполнить сложение:
                                        результат в HL.
                EX    DE,HL             Результат в DE, знаковый
                                        байт в HL.
                ADC   A,(HL)            Добавить знаковые байты и
                RRCA                    перенос в A; этим будет
                                        выявляться переполнение.
                ADC   A,+00             Теперь ненулевое A
                                        показывает переполнение.
                JR    NZ,303C,ADDN-OFLW Переход, чтобы сбросить
                                        указатели и сделать полное
                                        сложение.
                SBC   A,A               Для результата определить
3032            LD    (HL),A            правильный знаковый байт.
                INC   HL                Указать следующую ячейку.
                LD    (HL),E            Запомнить младший байт
                                        результата.
                INC   HL                Указать следующую ячейку.
                LD    (HL),D            Запомнить старший байт
                                        результата.
                DEC   HL                Сместить указатель
                DEC   HL                назад к адресу первого
                DEC   HL                байта результата.
                POP   DE                Восстановить STKEND в DE.
                RET                     Окончание.
```

Отметим, что десятичное число –65536 может появиться здесь в форме 00 FF 00 00 00, как результат сложения двух малых отрицательных целых чисел, например –65000 и –536. В этой форме оно просто заносится в стек. Это ошибка. Spectrum не сможет обработатьэто число.

Большинство функций рассматривает его как ноль и распечатывается оно как –1E–38, рассмотренное как 'минус ноль' в неправильном формате. Единственная возможность исправить это – проверить на это число где-то рядом с байтом 3032 и, если оно существует, сделать второй байт 80 (16-ричное) и первый байт 91 (16-ричное), создав полную пятибайтную форму числа с плавающей точкой, т.е. 91 80 00 00 00, которое не вызывает проблем. Смотрите также замечание в 'truncate' ('усечение') ниже, перед байтом 3225, и Приложении.

```
303C ADDN-OFLW  DEC   HL                Восстановить указатель
                                        первого числа.
                POP   DE                Восстановить указатель
                                        первого числа.
303E FULL-ADDN  CALL  3293,RE-ST-TWO     Перенести в стек оба числа
                                        в полной пятибайтной форме с
                                        плавающей точкой.
```

Полная подпрограмма ADDITION сначала вызывает PREP-ADD для каждого числа, затем получает два числа из стека калькулятора и заносит одно с малым порядком на позицию второго слагаемого. Затем вызывает SHIFT-FP, чтобы сдвинуть второе слагаемое на 32 (десятичное) позиции вправо, чтобы выровнять его для сложения. Фактическое сложение сделано с несколькими байтами, а единичный сдвиг осуществлен, если это необходимо, для переноса (переполнение влево), результат дополнен до двух, если он отрицателен, и сообщено о любом арифметическом переполнении: в противнем случае, подпрограмма переходит к TEST-NORM, чтобы нормализовать результат и вернуть его на стек с правильным знаковым разрядом, вставленным ее второй байт.

```
                EXX                     Обмен набора регистров.
```

```
                PUSH  HL                  Запись следующего адреса
                                          литерала.
                EXX                       Замена регистров.
                PUSH  DE                  Запись указателя второго
                                          слагаемого.
                PUSH  HL                  Запись указателя первого
                                          слагаемого.
                CALL  2F9B,PREP-ADD       Подготовить первое
                                          слагаемое.
                LD    B,A                 Запись его порядка в B.
                EX    DE,HL               Замена регистров.
                CALL  2F9B,PREP-ADD       Подготовка второго
                                          слагаемого.
                LD    C,A                 Запись его порядка в C.
                CP    B                   Если первый порядок
                JR    NC,3055,SHIFT-LEN   меньше, держать пер-
                LD    A,B                 вое число на позиции
                LD    B,C                 второго слагаемого;
                EX    DE,HL               в противном случае изменить
                                          порядок и указатели
                                          возвратить назад.
3055 SHIFT-LEN  PUSH  AF                  Запись большего порядка в A.
                SUB   B                   Разница между порядками
                                          является длиной сдвига
                                          вправо.
                CALL  2FBA,FETCH-TWO      Получить два числа из стека.
                CALL  2FDD,SHIFT-FP       Сдвиг вправо второго
                                          слагаемого.
                POP   AF                  Восстановить больший порядок.
                POP   HL                  HL указывает на результат.
                LD    (HL),A              Запомнить порядок результата.
                PUSH  HL                  Опять запись указателя.
                LD    L,B                 M4 в H и M5 в L.
                LD    H,C                 (см. FETCH-TWO).
                ADD   HL,DE               Добавить два правых байта.
                EXX                       N2 в H' и N3 в L'.
                EX    DE,HL               (см. FETCH-TWO).
                ADC   HL,BC               Добавить левые байты с
                                          переносом.
                EX    DE,HL               Результат обратно в D'E'.
                LD    A,H                 Добавить H'L' и перенос;
                ADC   A,L                 механизмы образования
                LD    L,A                 результата обеспечат то,
                RRA                       что единичный сдвиг
                XOR   L                   вправо вызывается,
                EXX                       если сумма двух чисел
                                          вызывает переполнение влево,
                                          или, если сумма двух
                                          отрицательных чисел вызывает
                                          переполнение влево.
                EX    DE,HL               Теперь результат в DED'E'.
                POP   HL                  Прочитать указатель порядка.
                RRA                       Проверка сдвига (H', L' были
                JR    NC,307C,TEST-NEG    для положительного числа
                                          00 (16-ричное) и FF (16-ричное)
                                          для отрицательного.
                LD    A,+01               A обеспечивает единичный
                                          сдвиг вправо.
                CALL  2FDD,SHIFT-FP       Вызван сдвиг.
```

```
                    INC   (HL)                К порядку добавляется 1;
                    JR    Z,309F,ADD-REP-6    это может привести к ариф-
                                              метическому переполнению.
307C TEST-NEG       EXX                       Тест для отрицательного
                    LD    A,L                 результата; получить в A
                    AND   +80                 знаковый разряд L' (теперь
                    EXX                       правильно отрабатывается
                                              знак результата).
                    INC   HL                  Запомнить это на позиции
                    LD    (HL),A              второго байта результата на
                    DEC   HL                  стеке калькулятора.
                    JR    Z,30A5,GO-NC-MLT    Если ноль, то результат не
                                              дополнять до двух.
                    LD    A,E                 Прочитать первый байт.
                    NEG                       Проинвертировать его.
                    CCF                       Дополнить перенос для
                    LD    E,A                 продолжения инвертирования и
                                              запомнить байт.
                    LD    A,D                 Прочитать следующий байт.
                    CPL                       Дополнить его до 1.
                    ADC   A,+00               Добавить перенос для
                                              инвертирования.
                    LD    D,A                 Запомнить байт.
                    EXX                       Продолжить, чтобы
                    LD    A,E                 получить следующий байт в
                                              регистре A.
                    CPL                       Дополнить его до 1.
                    ADC   A,+00               Добавить перенос для
                                              инвертирования.
                    LD    E,A                 Запомнить байт.
                    LD    A,D                 Получить последний байт.
                    CPL                       Дополнить его до 1.
                    ADC   A,+00               Добавить перенос для
                                              инвертирования.
                    JR    NC,30A3,END-COMPL   Сделано, если нет переноса.
                    RRA                       Иначе, получить 0.5
                                              к мантиссе и добавить 1 к
                    EXX                       порядку; это будет необходи-
                    INC   (HL)                мо, когда добавляется два
                                              отрицательных числа, чтобы
                                              задать точный показатель
                                              степени значения 2, а это
                                              может привести к ариф-
                                              метическому переполнению.
309F ADD-REP-6      JP    Z,31AD,REPORT-6     Если требуется, выдается
                    EXX                       ошибка.
30A3 END-COMPL      LD    D,A
                    EXX                       Запомнить последний байт.
30A5 GO-NC-MLT      XOR   A                   Очистить признак переноса.
                    JP    3155,TEST-NORM      Выход через TEST-NORM
```

ПОДПРОГРАММА 'HL=HL*DE'
Эта подпрограмма вызывается 'GET-HL*DE' и 'MULTIPLICATION' для выполнения 16-разрядного умножения.
Любое возможное переполнение 16 разрядов дает возврат из подпрограммы.

```
30A9 HL=HL*DE     PUSH  BC                Записано BC.
                  LD    B,+10             Этим представляется 16
                                          разрядное умножение.
                  LD    A,H               A содержит старший байт.
                  LD    C,L               C содержит младший байт.
                  LD    HL,+0000          Результат инициализирован
                                          нулем.
30B1 HL-LOOP      ADD   HL,HL             Удвоить результат.
                  JR    C,30BE,HL-END     Если переполнение, переход.
                  RL    C                 Сдвинуть 7 разряд C на
                                          перенос.
                  RLA                     Сдвинуть разряд переноса в
                                          0 разряд, а разряд C на
                                          признак переноса.
                  JR    NC,30BC,HL-AGAIN  Переход, если сброшен
                                          признак переноса.
                  ADD   HL,DE             В противном случае добавить
                                          DE единожды.
                  JR    C,30BE,HL-END     Если переполнение, переход.
30BC HL-AGAIN     DJNZ  30B1,HL-LOOP      Делать, пока не будут
                                          осуществлены 16 проходов.
30BE HL-END       POP   BC                Восстановить BC.
                  RET                     Окончание.
```

ПОДПРОГРАММА 'PREPARE TO MULTIPLY OR DIVIDE' ('Подготовка к умножению или делению')
Эта подпрограмма готовит число с плавающей точкой для умножения или деления, возвращаясь с установленным переносом, если число равно нулю, получая в регистре A знак результата и заменяя знаковый разряд в числе истинным числовым разрядом, 1.

```
30C0 PREP-M/D     CALL  34E9,TEST-ZERO    Если число ноль, возврат с
                  RET   C                 установленным признаком переноса.
                                          Указывает знаковый байт.
                  INC   HL                Получить в A знак для
                  XOR   (HL)              результата (возможные знаки
                                          плюс, невозможные минус);
                                          также сброс признака
                                          переноса.

                  SET   7,(HL)            Установка истинного
                                          числового разряда.
                  DEC   HL                Опять указывает порядок.
                  RET                     Возврат со сброшенным
                                          признаком переноса.
```

ОПЕРАЦИЯ 'MULTIPLICATION' ('Умножение')
(Смещение 04 - см. CALCULATE ниже: 'multiply' умножение)
Эта подпрограмма сначала проверяет, являются ли два числа, подлежащих умножению, 'малыми целыми числами'. Если да, она использует INT-FETCH, чтобы получить их из стека, HL=HL*DE, чтобы вернуть результат на стек. Любое переполнение этого 'короткого умножения' (т.е. если результат сам не 'малое целое число') принуждает к переходу на умножение в полной пятибайтной форме с плавающей точкой (см.ниже).

```
30CA multiply     LD    A,(DE)            Проверка, являются ли пять
                  OR    (HL)              байт обоих чисел нулями.
```

```
                JR    NZ,30F0,MULT-LONG   Если нет, переход для
                                          'длинного' умножения.
                PUSH  DE                  Запись указателей второго
                                          числа.
                PUSH  HL                  И первого числа.
                PUSH  DE                  И еще раз второго.
                CALL  2D7F,INT-FETCH      Выбор числа в C, числа в DE.
                EX    DE,HL               Теперь число в HL.
                EX    (SP),HL             Число в стек, второй
                                          указатель в HL.
                LD    B,C                 Запись первого знака в B.
                CALL  2D7F,INT-FETCH      Выбор второго знака в C,
                                          числа в DE.
                LD    A,B                 Сформировать знак результата
                XOR   C                   в A: возможные знаки дают
                                          плюс (00), невозможные минус
                                          (FF).
                LD    C,A                 Запомнить знак результата
                                          в C.
                POP   HL                  Восстановить первое число
                                          в HL.
                CALL  30A9,HL=HL*DE       Выполнить фактическое
                                          умножение.
                EX    DE,HL               Запомнить результат в DE.
                POP   HL                  Восстановить указатель
                                          первого числа.
                JR    C,30EF,MULT-OFLW    Переход при переполнении
                                          к 'полному' умножению.
30E5            LD    A,D                 Эти пять байт обеспечивают
                OR    E                   то, что 00 FF 00 00 00 заме-
                JR    NZ,30EA,MULT-RSLT   няются нулями; если это
                LD    C,A                 число исключено из системы,
                                          то они (байты) не являются
                                          необходимыми (см. выше
                                          после 303B).
30EA MULT-RSLT  CALL  2D8E,INT-STORE      Теперь запоминаем в стеке
                                          результат.
                POP   DE                  Восстановить STKEND в DE.
                RET                       Окончание.
30EF MULT-OFLW  POP   DE                  Восстановить указатель
                                          второго числа.
30F0 MULT-LONG  CALL  3293,RE-ST-TWO      Перенести в стек числа
                                          в полной пятибайтной форме
                                          с плавающей точкой.
```

Полная подпрограмма MULTIPLICATION готовит первое число для умножения, с помощью обращения к PREP-M/D, возвращаясь, если оно равно нулю; в противном случае, второе число подготовлено опять вызовом PREP-M/D, и, если оно равно нулю, подпрограмма задает результат нулем. Далее она выбирает два числа на стеке калькулятора и обычным способом перемножает их мантиссы, сдвигая первое число (трактуемое, как множитель) вправо и добавляется второе число (множимое) к результату, где бы не задавался разряд множителя. Порядки складываются вместе и выполняются проверки на переполнение и потерю значимости (задавая нулевой результат). В конце концов результат нормализуется и возвращается на стек калькулятора с правильным знаковым разрядом во вторам байте.

```
                XOR   A                   A задано 00 (16-ричное), так
                                          что знак первого числа будет
```

```
                                        идти в A.
                CALL  30C0,PREP-M/D     Подготовка первого числа,
                RET   C                 возврат, если ноль
                                        (результат уже ноль).
                EXX                     Замена регистров.
                PUSH  HL                Запись следующего адреса
                                        литерала.
                EXX                     Замена регистров.
                PUSH  DE                Запись указателя множимого.
                EX    DE,HL             Замена указателей.
                CALL  30C0,PREP-M/D     Подготовка второго числа.
                EX    DE,HL             Опять замена указателей.
                JR    C,315D,ZERO-RSLT  Переход вперед, если второе
                                        число равно нулю.
                PUSH  HL                Запись указателя результата.
                CALL  2FBA,FETCH-TWO    Получить из стека два числа.
                LD    A,B               M5 в A (см. FETCH-TWO).
                AND   A                 Подготовка для вычитания.
                SBC   HL,HL             Инициализировать HL нулем
                                        для результата.
                EXX                     Замена регистров.
                PUSH  HL                Запись M1 и N1 (см.
                                        FETCN-TWO) для результата.
                SBC   HL,HL             Инициализировать H'L' для
                                        результата.
                EXX                     Замена регистров.
                LD    B,+21             B считает 33 (десятичное),
                                        21 (16-ричное) сдвига,
                JR    3125,STRT-MLT     Переход вперед в цикл.

Теперь ввод цикла множителя.

3114 MLT-LOOP   JR    NC,311B,NO-ADD    Переход вперед в NO-ADD,
                                        если нет переноса, т.е. раз-
                                        ряд множителей был сброшен.
                ADD   HL,DE             Иначе, добавить мно-
                EXX                     жимое в D'E'DE (см.
                ADC   HL,DE             FETCH-TWO) в результат,
                EXX                     встроенный в H'L'HL.
311B NO-ADD     EXX                     Было ли добавлено
                RR    H                 множимое или нет,
                RR    L                 сдвинуть результат
                EXX                     вправо в H'L'HL, т.е. сдвиг
                                        осуществлен сдвигом каждого
                                        байта с переносом, так что
                                        любой разряд, брошенный в
                RR    H                 перенос, подвигается
                RR    L                 следующим байтом, а сдвиг
                                        продолжается в B'C'CA.
3125 STRT-MLT   EXX                     Сдвиг вправо множителя в
                RR    B                 B'C'CA (см. FETCH-TWO и
                                        выше).
                RR    C                 Конечный разряд, свя-
                EXX                     занный с переносом,
                RR    C                 еще раз прибавит мно-
                RRA                     жимое к результату.
                DJNZ  3114,MLT-LOOP     Прогнать цикл 33 раза для
                                        получения всех разрядов.
                EX    DE,HL             Переместить резуль-
```

```
                EXX                     тат из:
                EX    DE,HL             H'L'HL в D'E'DE.
                EXX
```

Теперь порядки сложим вместе.

```
                POP   BC                Восстановить порядки -
                                        M1 и N1.
                POP   HL                Восстановить указатель
                                        байта порядка.
                LD    A,B               Получить сумму двух
                ADD   A,C               байт порядков в A; и
                                        правильный перенос.
                JR    NZ,313B,MAKE-EXPT Если сумма равна нулю, то
                AND   A                 очистить перенос, иначе,
                                        оставить неизменным.
313B MAKE-EXPT  DEC   A                 Подготовка для увеличения
                CCF                     порядка на 80 (16-ричное).
```

Остальная часть подпрограммы является общей для MULTIPLICATION и DIVISION.

```
313D DIVN-EXPT  RLA                     Эти несколько байт очень
                CCF                     хорошо создают правильный
                                        байт порядка.
                RRA                     Сдвигая влево, затем вправо
                                        получим байт (истинный
                                        порядок плюс 80 (16-ричное))
                                        в A.
                JP    P,3146,OFLW1-CLR  Если признак знака сброшен,
                                        не нужно сообщение об
                                        арифметическом переполнении.
                JR    NC,31AD,REPORT-6  Если сброшен перенос,
                                        сообщение об ошибке.
                AND   A                 Теперь очистить перенос.
3146 OFLW1-CLR  INC   A                 Байт порядка теперь
                JR    NZ,3151,OFLW2-CLR завершен; но если A ноль,
                JR    C,3151,OFLW2-CLR  дальнейшая проверка
                                        переполнения необходима.
                EXX                     Если перенос не задан,
                BIT   7,D               результат уже в обычной
                EXX                     форме (7 разряд D' задан),
                JR    NZ,31AD,REPORT-6  то сообщение о переполне-
                                        нии; но, если 7 разряд D'
                                        сброшен, результат только в
                                        диапазоне, т.е. только до
                                        2**127.
3151 OFLW2-CLR  LD    (HL),A            Запомнить байт порядка.
                EXX                     Передать пятый байт резуль-
                LD    A,B               тата в A для нормализованной
                EXX                     последовательности, т.е.
                                        из L в B'.
```

Остаток подпрограммы занимается нормализацией и является общим для всех арифметических программ.

```
3155 TEST-NORM  JR    NC,316C,NORMALISE Если нет переноса, то
                                        нормализация.
                LD    A,(HL)            Иначе, потеря значимости
```

```
                AND   A                 (нулевой результат) или
3159 NEAR-ZERO  LD    A,+80             нахождение около нее (ре-
                                        зультат 2**-128).
                JR    Z,315E,SKIP-ZERO  Возврат порядка в A,
315D ZERO-RSLT  XOR   A                 проверяет ноль ли A
315E SKIP-ZERO  EXX                     (случай 2**-128), и,
                AND   D                 также, создается ли 2**-128,
                CALL  2FFB,ZEROS-4/5    нормальное ли число; иначе,
                RLCA                    создается ноль.
                LD    (HL),A            Восстановить байт порядка.
                JR    C,3195,OFLOW-CLR  Переход, если случай
                                        2**-128.
                INC   HL                В противном случае,
                LD    (HL),A            занести ноль во второй байт
                DEC   HL                результата на стеке
                                        калькулятора.
                JR    3195,OFLOW-CLR    Переход вперед для передачи
                                        результата.
```

Фактическая операция нормализации.

```
316C NORMALISE  LD    B,+20             Нормализовать сдвигами влево
316E SHIFT-ONE  EXX                     D'E'DE до 32 (десятичное), 20
                BIT   7,D               (16-ричное), с примкнувшим
                EXX                     A), пока не установится 7
                JR    NZ,3186,NORML-NOW разряд D'. A содержит ноль
                RLCA                    после сложения, точность не
                RL    E                 увеличивается и не умень-
                RL    D                 шается; A содержит пятый
                EXX                     байт из B' после умножения
                RL    E                 или деления; но как только
                RL    D                 32 разряда будут правильны-
                EXX                     ми, точность не теряется.
                                        Отметим, что A циклично
                                        сдвигается случайным
                                        образом на перенос.
                DEC   (HL)              На каждом сдвиге уменьшается
                                        порядок.
                JR    Z,3159,NEAR-ZERO  Если порядок становится
                                        нулем, то число 2**-129
                                        округляется до 2**-128.
                DJNZ  316E,SHIFT-ONE    Прогон цикла до 32 раз.
                JR    315D,ZERO-RSLT    Если 7 разряд никогда не
                                        станет 1, то весь результат
                                        нулевой.
```

Окончить нормализацию рассмотрев 'перенос'.

```
3186 NORML-NOW  RLA                     После нормализации добавить
                JR    NC,3195,OFLW-CLR  назад любой конечный
                                        перенос, который пришел в A.
                CALL  3004,ADD-BACK     Переход вперед, если пере-
                JR    NZ,3195,OFLW-CLR  нос не смещается обратно.
                EXX                     Если он должен сместиться
                LD    D,+80             обратно, то мантиссу задать
                EXX                     0.5 и изменить порядок.
                INC   (HL)              Это может привести к
                JR    Z,31AD,REPORT-6   арифметическому переполнению.
```

Конечная часть подпрограммы содержит переход результата к байтам, зарезервированным для него на стеке калькулятора и сброс указателей.

```
3195 OFLOW-CLR    PUSH  HL                  Запись указателя результата.
                  INC   HL                  Указать знаковый байт в
                                            результате.
                  EXX                       Результат перемещен
                  PUSH  DE                  из его текущих реги-
                  EXX                       стров D'E'DE, в
                  POP   BCBCDE; а затем в ACDE.
                  LD    A,B                 Знаковый разряд из временной
                  RLA                       памяти передан на его пра-
                  RL    (HL)                вильную позицию 7 разряда
                  RRA                       первого байта мантиссы.
                  LD    (HL),A              Запомнен первый байт
                                            мантиссы.
                  INC   HL                  Следующий.
                  LD    (HL),C              Запомнен второй байт
                                            мантиссы.
                  INC   HL                  Следующий.
                  LD    (HL),D              Запомнен третий байт
                                            мантиссы.
                  INC   HL                  Следующий.
                  LD    (HL),E              Запомнен четвертый бейт
                                            мантиссы.
                  POP   HL                  Восстановить указатель
                                            результата.
                  POP   DE                  Восстановить указатель
                                            второго числа.
                  EXX                       Замена регистра.
                  POP   HL                  Восстановить следующий
                                            адрес литерала.
                  EXX                       Замена регистров.
                  RET                       Окончание.
```

Сообщение 6 - 'Арифметическое переполнение'.

```
31AD REPORT-6     RST   0008,ERROR-1        Вызов подпрограммы
                  DEFB  +05                 обработки ошибок.
```

ОПЕРАЦИЯ 'DIVISION' ('Деление')
(Смещение 05 - см. CALCULATE: 'division' деление)
Эта подпрограмма сначала готовит делитель, вызвав PREP-M/D, которое сообщает об арифметическом переполнении, если делитель равен нулю; затем готовит делимое, опять вызвав PREP-M/D, возвращаясь, если оно равно нулю. Далее, выбирает два числа из стека калькулятора и делит их мантиссы посредством обычного восстановления деления, с пробным вычитанием делителя из делимого и восстановления, если есть перенос, иначе, добавляя 1 к частному. Максимальная точность получается для 4-байтного деления, а после вычитания порядков подпрограмма выходит, объединяясь с последней частью MULTIPLICATION.

```
31AF division     CALL  3293,RE-ST-TWO      Использование полной формы
                                            с плавающей точкой.
                  EX    DE,HL               Замена указателей.
                  XOR   A                   A задано 00 (16-ричное),
                                            поэтому знак первого числа
                                            пойдет в A.
                  CALL  30C0,PREP-M/D       Подготовка делителя и, если
```

```
                JR    C,31AD,REPORT-6     он равен нулю, выдача
                                          сообщения об арифметическом
                                          переполнении.
                EX    DE,HL               Замена указателей.
                CALL  30C0,PREP-M/D       Подготовка делимого и
                RET   C                   возврат, если оно равно нулю
                                          (результат уже нулевой).
                EXX                       Замена регистров.
                PUSH  HL                  Запись следующего адреса
                                          литерала.
                EXX                       Замена регистров.
                PUSH  DE                  Запись указателя делителя.
                PUSH  HL                  Запись указателя делимого.
                CALL  2FBA,FETCH-TWO      Получить два числа из стека.
                EXX                       Замена регистров.
                PUSH  HL                  Запись M1 и N1 на машинный
                                          стек.
                LD    H,B                 Скопировать четыре
                LD    L,C                 байта делимого из регистров
                EXX                       B'C'CB (т.е. M2, M3, M4 и M5;
                LD    H,C                 см. FETCH-TWO) в регистры
                LD    L,BH'L'HL.
                XOR   A                   Очистить A и сбросить
                                          признак переноса.
                LD    B,+DF               B будет просчитывать от -33
                                          до -1 с дополнением до
                                          двух, от DF (16-ричное) до
                                          FF, организовывая цикл по
                                          минусу, и будет опять
                                          переходить на ноль для
                                          дополнительной точности.
                JR    31E2,DIV-START      Переход вперед в цикл
                                          деления для первого пробного
                                          вычитания.
```

Теперь ввод цикла деления.

```
31D2 DIV-LOOP   RLA                       Сдвинуть результат влево в
                RL    C                   B'C'CA, сдвигая без
                EXX                       сокращения выдвигаемых
                RL    C                   разрядов разряды, которые
                RL    B                   уже там есть, считывая 1 из
                EXX                       переноса каждый раз, когда
                                          он задается, и, сдвигая
                                          влево каждый байт с пере-
                                          носом, чтобы получить сдвиг
                                          32 разряда.
31DB DIV-34TH   ADD   HL,HL               Переместить остатки делимого
                EXX                       влево в H'L'HL перед пробным
                ADC   HL,HL               вычитанием; если разряд по-
                EXX                       падает на перенос, то форси-
                                          руйте не восстановление, а
                                          разряд для частного, таким
                                          образом находя потерянный
                                          разряд и допуская полный 32
                JR    C,31F2,SUBN-ONLY    разрядный делитель.
31E2 DIV-START  SBC   HL,DE               Осуществить пробное
                EXX                       вычитание делителя в D'E'DE
                SBC   HL,DE               из остатка делимого в
                EXX                       H'L'HL: нет начального пе-
```

```
                                          реноса (см. предыдущий шаг).
                JR    NC,31F9,NO-RSTORE   Если нет переноса, переход
                                          вперед.
                ADD   HL,DE               В противном случае,
                EXX                       восстановление, т.е.
                ADC   HL,DE               добавление назад делителя.
                EXX                       Затем очищение переноса,
                AND   A                   с тем, чтобы не было разряда
                                          для чистого (делитель
                                          'не прошел').
                JR    31FA,COUNT-ONE      Переход вперед на счетчик.
31F2 SUBN-ONLY  AND   A                   Только вычитание без
                SBC   HL,DE               восстановления и установка
                EXX                       признака переноса, потому,
                SBC   HL,DE               что ищется потерянный
                EXX                       разряд делимого и исполь-
                                          зуется для частного.
31F9 NO-RSTORE  SCF                       1 для частного в B'C'CA.
31FA COUNT-ONE  INC   B                   Шаг цикла на 1 счета.
                JP    M,31D2,DIV-LOOP     Прочитать цикл 32 раза для
                                          всех разрядов.
                PUSH  AF                  Записать любой 33 разряд для
                                          дополнительной точности
                                          (текущий перенос).
                JR    Z,31E2,DIV-START    Выполнить пробное еще раз
                                          для 34 разряда; PUSH AF
                                          запишет дополнительно этот
                                          разряд тоже.
```

Примечание: Этот переход осуществляется на неправильное место. 34 разряд не будет получен без первого сдвига делимого. Отсюда важные результаты типа 1/10 и 1/1000 не округляются. Округление никогда не производится, если оно зависит от 34 разряда. Переход должен быть на 31DB DIV-34TH: т.е. байт 3200 (16-ричное) в ПЗУ следует читать DA (16-ричное) (218 десятичное) вместо 81 16-ричного (225 десятичное).

```
                LD    E,A                 Теперь переместим
                LD    D,C                 четыре байта, кото-
                EXX                       рые формируют байты
                LD    E,C                 мантиссы результата
                LD    D,B                 из B'C'CA в D'E'DE.
                POP   AF                  Затем внесем 34 и 33
                RR    B                   разряды в B' для нор-
                POP   AF                  мализации.
                RR    B
                EXX                       Восстановить байты
                POP   BC                  порядков M1 и N1.
                                          Восстановить указатель
                POP   HL                  результата.
                                          Получить в A разность между
                LD    A,B                 двумя байтами порядков и
                SUB   C                   задать, если требуется, при-
                                          знак переноса.
                JP    313D,DIVN-EXPT      Выход через DIVN-EXPT.
```

ПОДПРОГРАММА 'INTEGER TRUNCATION TOWARDS ZERO'
(Смещение 3A - см. CALCULATE 'truncate')
Эта подпрограмма (скажем I(x)) возвращает результат, полученный

усечением числа x до целой части, 'последнее значение', по направлению к 0. Таким образом, I(2.4) это 2, а I(-2.4) это -2. Эта программа сразу же возвращается, если x представлено в форме 'короткое целое'. Она возвращает ноль, если байт порядка x меньше 81 (16-ричное) (ABS x меньше 1). Если I(x) является 'коротким целым', подпрограмма возвращает его в этой форме. Она возвращает x, если байт порядка равен A0 (16-ричное) или больше (x не имеет значащей нецелой части). В противном случае, правильное число байт x задается нулем и, если необходимо, еще один байт разделяется с маской.

```
3214 truncate     LD    A,(HL)              Получить байт перехода x в A.
                  AND   A                   Если A ноль, возврат,
                  RET   Z                   хотя x уже малое число.
                  CP    +81                 Сравнить e, порядок,
                                            с 81 (16-ричным).
                  JR    NC,3221,T-GR-ZERO   Переход, если e больше,
                                            чем 80 (16-ричное).
                  LD    (HL),+00            Иначе, задать переход нулем:
                  LD    A,+20               ввести 32 (десятичное), 20
                  JR    3272,NIL-BYTES      (16-ричное) в A и перейти
                                            вперед на NIL BYTES, чтобы
                                            сделать все разряды x
                                            ненулевыми.
3221 T-GR-ZERO    CP    +91                 Сравнить e с 91 (16-
                                            ричным) 145 (десятичным).
3223              JR    NZ,323F,T-SMALL     Переход, если, не 91
                                            (16-ричное).
```

Следующие 26 байт кажутся созданными для проверки x: действительно если x=-65536 десятичному, т.е. 91 80 00 00 00 и, если это так, то задать его 00 FF 00 00 00. Это ошибка. Как уже было выше на 303B, система Spectrum не может отработать это число. Результат здесь для создания INT (-65536) и возврата значения -1. Жалко, хотя число само совершенно правильно. Следует просто пренебречь 28 байтами от 3223 до 323E.

```
3225              INC   HL                  HL указано в четвертом байте
                  INC   HL                  x, где 17 разрядов целой
                  INC   HL                  части X заканчиваются
                                            после первого разряда.
                  LD    A,+80               Первый разряд получен в A,
                  AND   (HL)                используя как маску 80
                                            (16-ричное).
                  DEC   HL                  Этот разряд и предыдущие 8
                  OR    (HL)                разрядов вместе проверяются
                                            на ноль.
                  DEC   HL                  HL указано во втором байте x.
                  JR    NZ,3233,T-FIRST     Если уже не ноль, тест можно
                                            заканчивать.
                  LD    A,+80               В противном случае, тест для
                  XOR   (HL)                -65536 завершен: 91 80 00
                                            00 00 будет оставлять
                                            заданным признак нуля.
3233 T-FIRST      DEC   HL                  HL указано во втором байте x.
                  JR    NZ,326C,T-EXPNENT   Если ноль сброшен, то
                                            делается переход.
                  LD    (HL),A              Первый байт задан нулем.
                  INC   HL                  HL указывает второй байт.
                  LD    (HL),+FF            Второй байт задан FF
                  DEC   HL                  HL снова указывает первый
```

```
                                          байт.
                LD    A,+18               Последние 24 разряда
                                          - нулевые.
                JR    3272,NIL-BYTES      Переход на NIL-BYTES завер-
                                          шает число 00 FF 00 00 00.
```

Если байт порядка X лежит между 81 и 90 (16-ричные) (129 и 144 десятичные), I(x) является 'малым целым' и будет помещено в один или два байта. Но сначала выполняется тест, проверяющий, является ли X после всего, большим.

```
323F T-SMALL    JR    NC,326D,X-LARGE     Переход с байтом порядка
                                          92 или более (было бы лучше
                                          перейти с 91).
                PUSH  DE                  Запись с STKEND в DE.
                CPL                       Диапазон 129 <= A <= 144
                                          становится 126 >= A >= 111.
                ADD   A,+91               Диапазон теперь
                                          15 (десятичное) >= A >= 0.
                INC   HL                  Указать HL во втором байте.
                LD    D,(HL)              Второй байт в D.
                INC   HL                  Указать HL в третьем байте.
                LD    E,(HL)              Третий байт в E.
                DEC   HL                  Указать HL опять на
                DEC   HLпервом байте.
                LD    C,+00               Предложить положительное
                                          число.
                BIT   7,D                 Теперь проверка для отрица-
                                          тельного числа (задан 7
                                          разряд).
                JR    Z,3252,T-NUMERIC    Переход, если оно
                                          положительно.
                DEC   C                   Изменение знака.
3252 T-NUMERIC  SET   7,D                 Вставка истинного числового
                                          разряда, 1, в D.
                LD    B,+08               Проверить, является ли A >= 8
                SUB   B                   (только один байт) или
                                          необходимо два байта.
                ADD   A,B                 Оставить A неизменным.
                JR    C,325E,T-TEST       Переход, если необходимо
                                          два байта.
                LD    E,D                 Поместить один байт в E.
                LD    D,+00               И задать D нулем.
                SUB   B                   Теперь 1<=A<=7, для счете
                                          необходимых сдвигов.
325E T-TEST     JR    Z,3267,T-STORE      Переход, если сдвиги не
                                          нужны.
                LD    B,A                 В будет считать сдвиг.
3261 T-SHIFT    SRL   D                   Сдвинуть D и E вправо
                RR    E                   В раз для создания
                                          правильного числа.
                DJNZ  3261,T-SHIFT        Прогон цикла, пока B не
                                          станет нулем.
3267 T-STORE    CALL  2D8E,INT-STORE      Запомнить результат в стеке.
                POP   DE                  Восстановить STKEND в DE.
                RET                       Окончание.
```

Рассматриваются большие значения x.

```
326C T-EXPNENT    LD    A,(HL)              Получить байт порядка X в A.
326D X-LARGE      SUB   +A0                 Вычесть 160 десятичное, A0
                                            16-ричное из e.
                  RET   P                   Возврат на плюс - X не имеет
                                            значащей нецелой части.
                                            (Если истинный порядок был
                                            понижен до нуля, то
                                            'двоичная запятая' будет
                                            появляться до или после 4
                                            байт мантиссы).
                  NEG                       Иначе, инвертировать поря-
                                            док; это дает число нулевых
                                            разрядов (число разрядов
                                            после 'двоичной запятой').
```

Теперь могут быть очищены разряды мантиссы.

```
3272 NIL-BYTES    PUSH  DE                  Записать текущее значение
                                            DE (STKEND).
                  EX    DE,HL               Создать точку HL на 1 после
                                            пятого байта.
                  DEC   HL                  Теперь HL указывает на
                                            пятый байт x.
                  LD    B,A                 Прочитать число раз-
                  SRL   B                   рядов и задать их
                  SRL   B                   нулем в B, разделить
                  SRL   B                   на B, чтобы в число
                                            входили все байты.
                  JR    Z,3283,BITS-ZERO    Если результат нулевой,
                                            переход вперед.
327E BYTE-ZERO    LD    (HL),+00            Иначе задать байты
                  DEC   HL                  нулем; B их считает.
                  DJNZ  327E,BYTE-ZERO
3283 BITS-ZERO    AND   +07                 Прочитать A (mod 8): это
                                            число разрядов, которые еще
                                            нужно задать нулями.
                  JR    Z,3290,IX-END       Переход в конец, если
                                            больше нечего делать.
                  LD    B,A                 B теперь будет считать
                                            разряды.
                  LD    A,+FF               Подготовка маски.
328A LESS-MASK    SLA   A                   На каждом цикле ноль
                  DJNZ  328A,LESS-MASK      вводит справа маску и, таким
                                            образом, создается маска
                                            правильной длины.
                  AND   (HL)                Нежелательные разряды (HL)
                  LD    (HL),A              теряется при выполнении
                                            маскирования.
3290 IX-END       EX    DE,HL               Возврат указателя HL.
                  POP   DE                  Возврат STKEND в DE.
                  RET                       Окончание.
```

ПОДПРОГРАММА 'RE-STACK TWO'
Эта подпрограмма вызывается для перенесения в стек двух 'малых целых чисел' в полной пятибайтной форме с плавающей точкой для двоичных операций сложения, умножения и деления. Это осуществляется с помощью двойного вызова подпрограммы.

```
3293 RE-ST-TWO    CALL  3296,RESTK-SUB      Вызов подпрограммы и затем
                                            работа с ней для второго
                                            обращения.
3296 RESTK-SUB    EX    DE,HL               Замена указателей на каждом
                                            обращении.
```

ПОДПРОГРАММА 'RE-STACK'
(Смещение 3D - см. CALCULATE ниже: 're-stack')
Эта подпрограмма вызывается для перезаписи в стек одного числа (каждое может быть 'малым целым' в полной пятибайтной форме с плавающей точкой). Она используется ARCTAN для одного числа, а также, через смещение калькулятора EXP, LN и 'get-argt'.

```
3297 RE-STACK     LD    A,(HL)              Если первый байт не ноль,
                  AND   A                   то возврат - число не может
                  RET   NZ                  быть 'малым числом'.
                  PUSH  DE                  Запись 'другого' указателя
                                            в DE.
                  CALL  2D7F,INT-FETCH      Выбор знака в C и числа в DE.
                  XOR   A                   Очистить регистр A.
                  INC   HL                  Указать пятую ячейку.
                  LD    (HL),A              Задать пятый байт нулем.
                  DEC   HL                  Указать четвертую ячейку.
                  LD    (HL),A              Задать четвертый байт нулем:
                                            2 и 3 байт содержат мантиссу.
                  LD    B,+91               Задать B значением 145
                                            (десятичное) для порядка,
                                            т.е. для доведения до 16
                                            разрядов в целом числе.
                  LD    A,D                 Проверка, является ли D
                  AND   A                   нулем, с тем, что бы самое
                                            большее 8 разрядов были бы
                                            необходимы.
                  JR    NZ,32B1,RS-NRMLSE   Переход, если необходимо
                                            больше 8 разрядов.
                  OR    E                   Также, проверка E.
                  LD    B,D                 Запись 0 в B (будет задан
                                            нулевой порядок, если E
                                            также 0).
                  JR    Z,32BD,RS-STORE     Переход, если E
                                            действительно 0.
                  LD    D,E                 Пересылка E в D (D было 0,
                                            E нет).
                  LD    E,B                 Теперь E задать 0.
                  LD    B,+89               Задать B для порядка 137
                                            (десятичное) - теперь не
                                            больше 8 разрядов.
32B1 RS-NRMLSE    EX    DE,HL               Указатель в DE, число в HL.
32B2 RSTK-LOOP    DEC   B                   Уменьшить порядок на каждом
                                            сдвиге.
                  ADD   HL,HL               Сдвинуть число вправо на
                                            одну позицию.
                  JR    NC,32B2,RSTK-LOOP   Пока не будет задан перенос.
                  RRC   C                   Знаковый разряд теперь к
                                            признаку переноса.
```

```
                RR    H               Вставьте его на место, т.к.
                RR    L               число сдвинуто на одно
                                      место назад - теперь так,
                                      как надо.
                EX    DE,HL           Указатель 4 байта назад в HL.
32BD RS-STORE   DEC   HL              Указать третью ячейку.
                LD    (HL),E          Запомнить третий байт.
                DEC   HL              Указать вторую ячейку.
                LD    (HL),D          Запомнить второй байт
                DEC   HL              Указать первую ячейку.
                LD    (HL),B          Запомнить байт порядка.
                POP   DE              Восстановить 'другие'
                                      указатели в DE.
                RET                   Окончание.
```

КАЛЬКУЛЯТОР С ПЛАВАЮЩЕЙ ТОЧКОЙ

ТАБЛИЦА КОНСТАНТ
Эта первая таблица содержит пять необходимых и часто используемых чисел: ноль, единица, половина PI и десять. Числа содержатся в уплотненной форме, которая расширяется с помощью подпрограммы STACK LITERALS, см. ниже, чтобы задать требуемую форму с плавающей точкой.

	данные	константа	при расширении дает: расшир. мантисса: (16-ричная)
32C5 stk-zero	DEFB +00 DEFB +B0 DEFB +00	zero	00 00 00 00 00
32C8 stk-one	DEFB +40 DEFB +B0 DEFB +00 DEFB +01	one	00 00 01 00 00
32CC stk-half	DEFB +30 DEFB +00	a half	80 00 00 00 00
32CE stk-pi/2	DEFB +F1 DEFB +49 DEFB +0F DEFB +DA DEFB +A2	a half of pi	81 49 0F DA A2
32D3 stk-ten	DEFB +40 DEFB +B0 DEFB +00 DEFB +0A	ten	00 00 0A 00 00

ТАБЛИЦА АДРЕСОВ
Вторая таблица представляет адреса 66 операционных подпрограмм калькулятора. Используемые в таблице смещения получены или из кодов операций, используемых в SCANNING, см. 2734 и т.д., или из литералов, которые следуют за командой RST 0028.

	смещ.	метка	адрес		смещ.	метка	адрес
32D7	00	jump-true	8F 36	3319	21	tan	DA 37
32D9	01	exchange	3C 34	331B	22	asn	33 38
32DB	02	delete	A1 33	331D	23	acs	43 38
32DD	03	subtract	0F 30	331F	24	atn	E2 37
32DF	04	multiply	CA 30	3321	25	ln	13 37
32E1	05	division	AF 31	3323	26	exp	C4 36
32E3	06	to-power	51 38	3325	27	int	AF 36
32E5	07	or	1B 35	3327	28	sqr	4A 38
32E7	08	no-&-no	24	3329	29	sgn	92

		35			34
32E9 09	no-l-eql	3B	332B 2A	abs	6A
		35			34
32EB 0A	no-gr-eq	3B	332D 2B	peek	AC
		35			34
32ED 0B	nos-neql	3B	332F 2C	in	A5
		35			34
32EF 0C	no-grtr	3B	3331 2D	usr-no	B3
		35			34
32F1 0D	no-less	3B	3333 2E	str$	1F
		35			36
32F3 0E	nos-eql	3B	3335 2F	chr$	C9
		35			35
32F5 0F	addition	14	3337 30	not	01
		30			35
32F7 10	str-&-no	2D	3339 31	duplicate	C0
		35			33
32F9 11	str-l-eql	3B	333B 32	n-mod-m	A0
		35			36
32FB 12	str-gr-eq	3B	333D 33	jump	86
		35			36
32FD 13	strs-neql	3B	333F 34	stk-data	C6
		35			33
32FF 14	str-grtr	3B	3341 35	dec-jr-nz	7A
		35			36
3301 15	str-less	3B	3343 36	less-0	06
		35			35
3303 16	strs-eql	3B	3345 37	greater-0	F9
		35			34
3305 17	strs-add	9C	3347 38	end-calc	9B
		35			36
3307 18	val$ DE	33	3349 39	get-argt	83
		35			37
3309 19	usr-$ BC	33	334B 3A	truncate	14
		34			32
330B 1A	read-in	45	334D 3B	fp-calc-2	A2
		36			33
330D 1B	negate	6E	334F 3C	e-to-fp	4F
		34			2D
330F 1C	code	69	3351 3D	re-stack	97
		36			32
3311 1D	val	DE	3353 3E	series-06	49
		45		etc.	34
3313 1E	len	74	3355 3F	stk-zero	1B
		36		etc.	34
3315 1F	sin	B5	3357 40	st-mem-0	2D
		37		etc.	34
3317 20	cos	AA	3359 41	get-mem-0	0F
		37		etc.	34

Примечание: Последние четыре подпрограммы являются многоцелевыми подпрограммами и вводятся с параметром, который является копией правых пяти разрядов исходного литерала.
Полный набор следующий:
Смещение 3E: series-06, series-08 & series-0C; литералы 86, 88 & 8C.
Смещение 3F: stk-zero, stk-one, stk-half, stk-pi/2 & stk-ten; литералы от A0 до A4.
Смещение 40: st-mem-0, st-mem-1, st-mem-2, st-mem-3, st-mem-4 & st-mem-5; литералы от C0 до C5.

Смещение 41: get-mem-0, get-mem-1, get-mem-2, get-mem-3, get-mem-4 & get-mem-5; литералы от E0 до E5.

ПОДПРОГРАММА 'CALCULATE'
Эта подпрограмма выполняет вычисления с плавающей точкой. Можно рассмотреть три типа:

1. Двоичные операции, например, сложение, когда два числа с плавающей точкой складываются вместе и выдают одно 'последнее значение'.

2. Унарная операция, например, SIN, где 'последнее значение' изменяется для выдачи соответствующего результата функции, как нового 'последнего значения'.

3. Операции обработки, например, st-mem-0, где 'последнее значение' копируется в первые пять байт области памяти калькулятора.

Для вычисления операции задаются как последовательности байтов данных, литералов, которые следуют за командой RST 0028, вызывающей эту подпрограмму. Последний литерал в списке всегда '38', что ведет к завершению всей операции.

В случае, если надо выполнить одну операцию, номер операции передается в CALCULATOR в регистре B, и выполняется операция '3B', SINGLE CALCULATION.

Также, эта программа вызывается рекурсивно, и в этом случае возможно использовать системную переменную BREG как счетчик, который управляет выполнением операций перед возвратом.

Первая часть этой подпрограммы сложна, но по существу, она выполняет две задачи: задает регистры с требуемыми значениями и создает смещение и, возможно, параметр, из рассматриваемых литералов.

Смещение используется для обозначения адресов в таблице калькулятора, см. выше, чтобы найти требуемый адрес подпрограммы.

Параметры используются при вызове многоцелевых подпрограмм.

Примечание: Число с плавающей точкой в действительности может быть набором строковых параметров.

```
335B CALCULATE    CALL  35BF,STK-PNTRS      Предположим унарную операцию
                                            и поэтому зададим HL указы-
                                            вать на начало 'последнего
                                            значения' на стеке калькуля-
                                            тора и DE на 1 после этого
                                            числа с плавающей точкой
                                            (STKEND).
335E GEN-ENT-1    LD    A,B                 Или передача смещения
                  LD    (BREG),A            единичной операции временно
                                            в BREG, или при рекурсивном
                                            использовании подпрограммы
                                            передача параметров в BREG
                                            для использования в качестве
                                            счетчика.
3362 GEN-ENT-2    EXX                       Адрес возврата подпрограммы
                  EX    (SP),HL             помещается в H'L'. Записы-
                  EXX                       вается указатель первого
                                            литерала. Используется вход
                                            в CALCULATOR через GEN-ENT-2
                                            при использовании BREG в
                                            качестве счетчика и не
                                            подлежит разрушению.
3365 RE-ENTRY     LD    (STKEND),DE         Теперь вводится цикл для
                                            обработки каждого литерала в
                                            списке который следует за
```

```
                                        командой вызова: итак
                                        поначалу всегда задается
                                        STKEND.
                EXX                     Переход на заданный альтер-
                LD    A,(HL)            нативный регистр и выбор
                                        литерала для этого цикла.
                INC   HL                H'L' указывает следующий
                                        литерал.
336C SCAN-ENT   PUSH  HL                Этот указатель записан на
                                        машинный стек SCAN-ENT,
                                        используется подпрограммой
                                        SINGLE CALCULATION для
                                        обнаружения требуемой
                                        подпрограммы.
                AND   A                 Проверка регистра A.
                JP    P,3380,FIRST-3D   Отделение простых литералов
                                        от многоцелевых. Переход с
                                        литералами 00-3D.
                LD    D,A               Запись литерала в D.
                AND   +60               Продолжение только с
                                        разрядами 5 и 6.
                RRCA                    Четыре сдвига вправо
                RRCA                    сделаем их 1 и 2
                RRCA                    разрядами.
                RRCA
                ADD   A,+7C             Требуются смещения 3E-41,
                LD    L,A               а L теперь будет содержать
                                        требуемое двойное смещение.
                LD    A,D               Теперь создается параметр с
                AND   +1F               помощью 0,1,2,3,4 разрядов
                                        литерала; держите параметры
                                        в A.
                JR    338E,ENT-TABLE    Переход вперед, чтобы найти
                                        адрес требуемой подпрограммы
3380 FIRST-3D   CP    +18               Переход вперед, если
                JR    NC,338C,DOUBLE-A  выполняется унарная операция.
                EXX                     Все подпрограммы, которые
                                        выполняют двоичные операции,
                                        требуют, чтобы HL
                LD    BC,+FFFB          указывала на первый
                LD    D,H               операнд, а DE на второй опе-
                LD    E,L               ранд ('последнее значение'),
                ADD   HL,BC             так как они появляются на
                EXX                     стек калькулятора.
338C DOUBLE-A   RLCA                    Т.к. каждый элемент в
                LD    L,A               таблице адресов занимает
                                        два байта, созданное
                                        смещение удваивается.
338E ENT-TABLE  LD    DE,+32D7          Базовый адрес таблицы.
                LD    H,+00             Адрес требуемого эле-
                ADD   HL,DE             мента таблицы форми-
                LD    E,(HL)            руется в HL, а адрес
                INC   HL                требуемой подпрограммы
                LD    D,(HL)            загружается в регистровую
                                        пару DE.
                LD    HL,+3365          RE-ENTRY адрес 3365
                EX    (SP),HL           занесен на машинный стек
                PUSH  DE                ниже адреса подпрограммы.
                EXX                     Возврат к основному набору
```

```
                                          регистров.
                LD    BC,(STKEND-hi)      Текущее значение BREG
                                          передается в регистр B,
                                          таким образом, возвращая
                                          единичный номер операции.
                                          (см. COMPARISON в 353B).
33A1 delete     RET                       Непрямой переход на
                                          требуемую подпрограмму.
```

ПОДПРОГРАММА 'DELETE' ('Удаление')
(Смещение 02: 'delete')

Эта подпрограмма содержит только одну команду RET в 33A1, см. выше. Литерал '02' завершает рассматриваемую подпрограмму, как двоичную операцию, которая должна вводиться с первым числом, адресованным регистровой парой HL, и вторым числом, адресованным регистровой парой DE, а созданный результат вновь адресуется регистровой парой HL.

Единственная команда RET, таким образом, рассматривает первое число как 'последнее значение', второе удаляет. Конечно, второе число из памяти не удаляется, оно лишь остается пассивным и на него будет наложена другая запись.

ПОДПРОГРАММА 'SINGLE OPERATION' ('Единственная операция')
(Смещение 3B: 'fp-calc-2')

Эта подпрограмма вызывается только из SCANNING на 2757 (16-ричное) и используется для выполнения одной операции. Смещение, задающее подлежащую выполнению операцию, заносится в калькулятор на регистр B и частично передается в системную переменную BREG.

Эффект от вызова этой подпрограммы является существенным для осуществления перехода на соответствующую подпрограмму для одной операции.
THE 'SINGLE OPERATION' SUBROUTINE
(Offset 3B: 'fp-calc-2')

```
33A2 fp-calc-2  POP   AF                Отбросить адрес RE-ENTRY.
                LD    A,(BREG)          Передать смещение в A.
                EXX                     Второй набор регистров.
                JR    336C,SCAN-ENT     Переход назад, чтобы найти
                                        требуемый адрес; занести в
                                        стек адрес RE-ENTRY и пе-
                                        реход на подпрограмму для
                                        операции.
```

ПОДПРОГРАММА 'TEST 5-SPACES' ('Проверка места')
Эта подпрограмма проверяет, достаточно ли места в памяти для другого 5 байтного числа с плавающей точкой, которые нужно добавить на стек калькулятора.

```
33A9 TEST-5-SP  PUSH  DE                Записать DE.
                PUSH  HL                Записать HL.
                LD    BC,+0005          Определить тест для 5 байт.
                CALL  1F05,TEST-ROOM    Осуществить тест.
                POP   HL                Восстановить HL.
                POP   DE                Восстановить DE.
                RET                     Окончание.
```

ПОДПРОГРАММА 'STACK NUMBER' ('Поместить число в стек')
Эта подпрограмма вызывается BEEP и дважды SCANNING для

копирования STKEND в DE, пересылает числа с плававшей точкой в стек калькулятора и сбрасывает STKEND из DE. Для фактической пересылки она вызывает 'MOVE-FP'.

```
33B4 STACK-NUM    LD    DE,(STKEND)         Скопировать STKEND в DE
                                            как адрес назначения.
                  CALL  33C0,MOVE-FP        Переслать число.
                  LD    (STKEND),DE         Сбросить STKEND из DE.
                  RET                       Окончание.
```

ПОДПРОГРАММА 'MOVE A FLOATING-POINT NUMBER' ('Пересылка числа с плавающей точкой')
(Смещение 31: 'duplicate')

Эта подпрограмма пересылает число с плавающей точкой на вершину стека калькулятора (3 случая) или с вершины стека в область памяти калькулятора (1 случай). Также вызывается через калькулятор, когда он дублирует число на вершине стека, 'последнее значение', расширяя, таким образом, стек на 5 байт.

```
33C0 MOVE-FP      CALL  33A9,TEST-5-SP      Сделан тест для места на
                                            стеке.
                  LDIR                      Переместить 5 байт.
                  RET                       Окончание.
```

ПОДПРОГРАММА 'STACK LITERALS' ('Поместить в стек литерал')
(Смещение 34: 'stk-data')

Эта подпрограмма размещает на стеке калькулятора в качестве 'последнего значения' число с плавающей точкой, занесенное туда как 2, 3, 4 или 5 литералов.
При вызове с использованием смещения '34' литералы следует за '34' в списке литералов; при вызове с помощью SERIES GENERATOR, см. ниже, литералы дополняются подпрограммой, которая вызвана для подлежащего создания ряда; а когда вызывается с использованием SKIP CONSTANTS и STACK A CONSTANT литералы получают из таблицы констант калькулятора (32C5-32D6).
В каждом случае дополнительный литерал делится на 40 (16-ричное), и целочисленное частное плюс 1 определяют, будут ли 1, 2, 3 или 4 дальнейшие литералы браться из источника для формирования мантиссы числа. Любые незаполненные байты из 5 байт, которые поступают для формирования 5-байтного числа с плавающей точкой, задаются нулями. Первый литерал также используется для определения порядка после понижения в 40 раз (16-ричное), если остаток не является нулем, при этом используется второй литерал, какой он есть без уменьшения. В любом случае к литералу добавляется 50 (16-ричное), задавая увеличенный байт порядка, e (истинный порядок e' плюс 80 16-ричное).
Остаток 5 байт заносится в стек, включая любые необходимые нули, и программа возвращается.

```
33C6 STK-DATA     LD    H,D                 Эта подпрограмма добавляет
                  LD    L,E                 'последнее значение' на стек
                                            калькулятора; отсюда HL
                                            устанавливается, чтобы
                                            указывать 1 после текущего
                                            'последнего значения' и
                                            соответственно результат.
33C8 STK-CONST    CALL  33A9,TEST-5-SP      Теперь проверим, что
                                            действительно ли есть место.
                  EXX                       Идти на альтернативный набор
```

```
                PUSH  HL                регистров и занести а стек
                EXX                     указатель следующего
                                        литерала.
                EX    (SP),HL           Переключить указатель
                                        результата на указатель
                                        следующего литерала.
                PUSH  BC                Запись BC.
                LD    A,(HL)            Первый литерал заносится в A
                AND   +C0               и делится на 40 (16-ричное),
                RLCA                    чтобы задать целые значения
                RLCA                    0, 1, 2, или 3.
                LD    C,A               Значение целого числа пере-
                INC   C                 дается в C и получает
                                        приращение, задавая таким
                                        образом диапа-
                                        зон 1, 2, 3 или 4 для
                                        необходимых литералов.
                LD    A,(HL)            Литерал выбирается
                AND   +3F               вновь, понижен в 40
                JR    NZ,33DE,FORM-EXP  раз и отброшен как
                INC   HL                несоответствующий,
                LD    A,(HL)            если остаток равен 0;
                                        в этом случае выбирается
                                        следующий литерал, который
                                        не понижается.
33DE FORM-EXP   ADD   A,+50             Порядок e формируется с
                LD    (DE),A            помощью добавления 50
                                        (16-ричное) и передается на
                                        стек калькулятора как пер-
                                        вый из 5 байт результата.
                LD    A,+05             Число литералов, за-
                SUB   C                 данных в C, берется
                INC   HL                из источника и вво-
                INC   DE                дится в байты ре-
                LD    B,+00             зультата.
                LDIR
                POP   BC                Восстановить BC.
                EX    (SP),HL           Возврат указателя
                EXX                     результата в HL и
                POP   HL                следующий указатель
                EXX                     литерала на его обычную
                                        позицию в H' и L'.
                LD    B,A               Число требуемых нулевых
                XOR   A                 байт на этом этапе задано с
                                        помощью 5-C-1: и это
33F1 STK-ZEROS  DEC   B                 число нулей склады-
                RET   Z                 вается с результатом
                LD    (DE),A            для создания требуемых
                INC   DE                5 байт.
                JR    33F1,STK-ZEROS
```

ПОДПРОГРАММА 'SKIP CONSTANTS' ('Пропустить константы')
Эта подпрограмма вводится с регистровой парой HL, содержащей базовый адрес таблицы констант калькулятора и регистром A, содержащим параметр, который показывает какая из пяти констант является затребованной.
Подпрограмма выполняет пустые операции загрузок 5 байт каждой ненужной константы в ячейки 0000, 0001, 0003 и 0004 с начала ПЗУ, пока не достигнет требуемых констант.

```
33F7 SKIP-CONS    AND   A                 Подпрограмма возвращается,
33F8 SKIP-NEXT    RET   Z                 если параметр равен нулю,
                                          или когда достигает
                                          требуемая константа.
                  PUSH  AF                Запись параметра.
                  PUSH  DE                Запись указателя результата.
                  LD    DE,+0000          Фиктивный адрес.
                  CALL  33C8,STK-CONST    Выполнить мнимое занесение в
                                          стек расширенной константы.
                  POP   DE                Восстановить указатель
                                          результата.
                  POP   AF                Восстановить параметр.
                  DEC   A                 Посчитать циклы.
                  JR    33F8,SKIP-NEXT    Переход назад, чтобы рас-
                                          смотреть значение счетчика.
```

ПОДПРОГРАММА 'MEMORY LOCATION' ('Определение места в памяти')
Подпрограмма находит базовый адрес для каждой порции из 5 байт области памяти калькулятора в или из которой пересылается число с плавающей точкой из или в стек вычислителя. Она выполняет эту операцию, складывая 5 раз параметр с базовым адресом для области, которая содержится в регистровой паре HL.
Отметим, что когда переменная FOR-NEXT обработана, то указатели изменяются так, что переменная трактуется как область памяти калькулятора (см. адрес 1D20).

```
3406 LOC-MEM      LD    C,A               Скопировать параметр в C.
                  RLCA                    Удвоить параметр.
                  RLCA                    Удвоить этот результат.
                  ADD   A,C               Добавить значение па-
                                          раметра, чтобы задать 5
                                          раз исходное значение.
                  LD    C,A               Этот результат нужен
                  LD    B,+00             в регистровой паре BC
                  ADD   HL,BC             Создание нового адреса.
                  RET                     Окончание.
```

ПОДПРОГРАММА 'GET FROM MEMORY AREA' ('Получить из области памяти')
(Смещение от E0 до E5: от 'get-mem-0' до 'get-mem-5')

Эта подпрограмма вызывается с использованием литералов E0-E5, а параметр, получаемый из этих литералов, содержится в регистре A. Подпрограмма вызывает MEMORY LOCATION, чтобы занести требуемый исходный адрес в регистровую пару HL и MOVE A FLOATING POINT NUMBER, чтобы скопировать 5 байт из области памяти калькулятора на вершину стека калькулятора для формирования нового 'последнего значения'.

```
340F get-mem-0    PUSH  DE                Запись указателя результата.
     etc.         LD    HL,(MEM)          Выбор указателя текущей
                                          области памяти (см. выше).
                  CALL  3406,LOC-MEM      Обнаружен базовый адрес.
                  CALL  33C0,MOVE-FP      Пересылка 3 байт.
                  POP   HL                Установить указатель
                                          результата.
                  RET                     Окончание.
```

ПОДПРОГРАММА 'STACK A CONSTANT' ('Занести константу в стек')
(Смещение от A0 до A4: 'stk-zero', 'stk-one', 'stk-half', 'stk-pi/2' и 'stk-ten')

Эта подпрограмма использует SKIP CONSTANTS для обнаружения базового адреса запрошенной константы из таблицы констант калькулятора и затем вызывает STACK LITERALS, входящий в STK-CONST для создания расширенной формы константы 'последнего значения' на стеке калькулятора.

```
341B stk-zero     LD    H,D                Установить HL, чтобы содер-
     etc.         LD    L,E                жать указатель результата.
                  EXX                      Переход на набор переменных
                  PUSH  HL                 регистров и запись следующе-
                                           го указателя литерала.
                  LD    HL,+32C5           Базовый адрес таблицы
                                           констант калькулятора.
                  EXX                      Назад на основной набор
                                           регистров.
                  CALL  33F7,SKIP-CONS     Найти затребованный базовый
                                           адрес.
                  CALL  33C8,STK-CONST     Расширить константу.
                  EXX
                  POP   HL                 Восстановить следующий
                                           указатель литерала.
                  EXX
                  RET                      Окончание.
```

ПОДПРОГРАММА 'STORE IN MEMORY AREA' ('Хранить в области памяти')
(Смещение от C0 до C5: 'st-mem-0' до 'st-mem-5')

Эта подпрограмма вызывается, используя литералы C0-C5 и параметр, полученный из этих литералов, содержащийся в регистре A. Эта подпрограмма очень похожа на подпрограмму GET FROM MEMORY, в ней лишь заменяется исходный указатель и указатель адреса назначения.

```
342D st-mem-0     PUSH  HL                 Запись указателя результата.
     etc.         EX    DE,HL              Источник в DE.
                  LD    HL,(MEM)           Выбор указателя текущей
                                           области памяти.
                  CALL  3406,LOC-MEM       Найден базовый адрес.
                  EX    DE,HL              Замена исходного указателя
                                           и указателя адресата.
                  CALL  33C0,MOVE-FP       Пересылка 5 байт.
                  EX    DE,HL              'Последнее значение'
                                           +5, т.е. STKEND в DE.
                  POP   HL                 Указатель результата в HL.
                  RET                      Окончание.
```

Отметим, что указатели HL и DE остаются такими же, как были; указывая на STKEND-5 и STKEND соответственно, так что 'последнее значение' остается на стеке калькулятора. Если надо, его можно перазаслать, используя 'delete' ('удалить').

ПОДПРОГРАММА 'EXCHANGE' ('Замена')
(Смещение 01: 'exchange')

Эта двоичная операция заменяет первое число вторым числом, т.е. меняются два числа на вершине стека калькулятора.

```
343C EXCHANGE     LD    B,+05              Участвуют 5 байт.
343E SWAP-BYTE    LD    A,(DE)             Каждый байт второго числа.
                  LD    C,(HL)             Каждый байт первого числа.
                  EX    DE,HL              Переключая источник и
                                           адресат.
                  LD    (DE),A             Теперь в первое число.
                  LD    (HL),C             Теперь во второе число.
                  INC   HL                 Переслать для рассмотрения
                  INC   DE                 следующую пару байт.
                  DJNZ  343E,SWAP-BYTE     Заменить 5 байт.
                  EX    DE,HL              Получить откорректированные,
                                           т.е. число 5 является
                                           нечетным числом.
                  RET                      Окончание.
```

ПОДПРОГРАММА 'SERIES GENERATOR' ('Генератор рядов')
(Смещение 87, 88 и 8C: 'series-06', 'series-08' и 'series-0C')

Эта важная подпрограмма генерирует ряды полиномов Чебышева, которые используются для аппроксимации SIN, ATN, LN и EXP и отсюда для получения других арифметических функций, которые зависят от них (COS, TAN, ASN, ACS, ** и SQR).
Полиномы создаются для n=1,2,3,4... с помощью рекурентного соотношения:

Tn+1(z) = 2zTn(z) - Tn-1(z), где Tn(z) n-й полином Чебышева в z.
Ряды в действительности генерируют:
T0, 2T1, 2T2,.... ,2Tn-1, где n равно 6 для SIN, 8 для EXP и 12 десятичное, для LN и ATN.
Показатель Z в полиномах может быть найден в "Справочнике по математическим функциям", М.Абрамович и И.А.Стегун (Дувр 1965), стр. 795.
Программы BASIC, создающие каждую из четырех функций, приведены в Приложении. Попросту говоря, эта подпрограмма вызывается на стек калькулятора с 'последним значением', скажем Z, с аргументом X и нужно вычислить функцию, например, SIN X. Вызывающая подпрограмма также обеспечивает список требуемых констант для SIN (их 6). SERIES GENERATOR затем обрабатывает эти данные и возвращает в вызывающую программу 'последнее значение', которое образует простое соотношение с затребованной функцией, например, SIN X.
Эту подпрограмму можно разбить на четыре основные части.

1. Установка счетчика циклов:
Вызывающая подпрограмма передает его параметры в регистр A для использования в качестве счетчика. Чтобы задать счетчик, калькулятор вводится на GEN-ENT-1.

```
3449 series-06    LD    B,A                Пересылка параметра в B.
     etc.         CALL  335E,GEN-ENT-1     В действительности ко-
                                           манда RST 0028, но за-
                                           дается счетчик.
```

2. Обработка 'последнего значения', Z:
Цикл генератора требует, чтобы Z*Z было размещено в mem-0, ноль в mem-2, а 'последнее значение' должно быть нулем.

```
                                стек калькулятора.
                DEFB  +31,duplicate     Z,Z
                DEFB  +0F,addition      2*Z
                DEFB  +C0,st-mem-0      2*Z mem-0 содержит 2*Z
                DEFB  +02,delete        -
                DEFB  +A0,stk-zero      0
                DEFB  +C2,st-mem-2      0 mem-2 содержит 0
```

3. Основной цикл.
Ряды создаются в цикле с использованием BREG в качестве счетчика; константы в вызывающей подпрограмме по очереди заносятся в стек с помощью STK-DATA; калькулятор переводится на GEN-ENT-2 с тем, чтобы не испортить значение BREG; ряд создается в виде:

B(R) = 2*Z*B(R-1) - B(R-2) + A(R), для R = 1,2,...,N, где A(1), A(2),..., A(N) являются константами, выдаваемыми вызывающей подпрограммой (SIN, ATN, LN и EXP) и B(0) = 0 = B(-1).
(R+1)-й цикл начинается с B(R) на стеке и с 2*Z, B(R-2) и B(R-1) в mem-0, mem-1 и mem-2 соответственно.

```
3453 G-LOOP     DEFB  +31,duplicate     B(R),B(R)
                DEFB  +E0,get-mem-0     B(R),B(R),2*Z
                DEFB  +04,multiply      B(R),2*B(R)*Z
                DEFB  +E2,get-mem-2     B(R),2*B(R)*Z,B(R-1)
                DEFB  +C1,st-mem-1      mem-1 содержит B(R-1)
DEFB +38,end-calc DEFB  +03,subtract    B(R),2*B(R)*Z-B(R-1)
```

Следующая константа размещается на стеке калькулятора.

```
                CALL  33C6,STK-DATA     B(R),2*B(R)*Z-B(R-1),A(R+1)
```

Снова вход в Калькулятор без разрешения BREG.

```
                CALL  3362,GEN-ENT-2
                DEFB  +0F,addition      B(R),2*B(R)*Z-B(R-1)+A(R+1)
                DEFB  +01,exchange      2*B(R)*Z-B(R-1)+A(R+1),B(R)
                DEFB  +C2,st-mem-2      mem-2 содержит B(R)
                DEFB  +02,delete        2*B(R)*Z-B(R-1)+A(R!1) =
                                        B(R!1)
                DEFB  +35,dec-jr-nz     B(R+1)
                DEFB  +EE,to 3453,G-LOOP
```

4. Вычитание B(N-2):
Цикл, представленный выше, оставляет B(N) на стеке, а требуемый результат задается B(N) - B(N-2)

```
                DEFB  +E1,get-mem-1     B(N),B(N-2)
                DEFB  +03,subtract      B(N)-B(N-2)
                DEFB  +38,end-calc
                RET                     Окончание.
```

ФУНКЦИЯ 'ABSOLUTE MAGNITUDE' ('Абсолютная величина')
(Смещение 2A: 'abs')
Эта подпрограмма выполняет свою унарную операцию с помощью сброса знакового разряда числа с плавающей точкой.
'Малые целые' должны обрабатываться отдельно. Большинство действий вычисляется с операцией 'унарный минус'.

```
346A abs        LD    B,+FF              В устанавливается в
                                         FF (16-ричное).
                JR    3474,NEG-TEST      Сделан переход на
                                         'унарный минус'.
```

ОПЕРАЦИЯ 'UNARY MINUS' ('Унарный минус')
(Смещение 1B: 'negate').

Эта подпрограмма выполняет свою унарную операцию с помощью изменения знака 'последнего значения' на стеке калькулятора.
Ноль возвращается неизменным. Полные пятибайтные числа с плавающей точкой имеют знаковый разряд, обработанный так, что они заканчиваются сбросом (для 'abs') или изменяются (для 'инвертирования'). 'Малые целые числа' имеют знаковый разряд, заданный нулем (для 'abs') или измененным (для 'инвертирования').

```
346E NEGATE     CALL  34E9,TEST-ZERO     Если число является 0,
                RET   C                  подпрограмма возвращает
                                         остающиеся 00 00 00 00 00
                                         неизмененными.
                LD    B,+00              Для 'инвертирования'
                                         В задается +00 (16-ричнов).
```

'ABS' вводится здесь.

```
3474 NEG-TEST   LD    A,(HL)             Если первый байт ноль,
                AND   A                  выполняется переход для
                JR    Z,3483,INT-CASE    работы с 'малым целым'.
                INC   HL                 Указывает второй байт.
                LD    A,B                Получить +FF для 'abs', +00
                                         для 'инвертирования'.
                AND   +80                Теперь +80 для 'abs', +00
                                         для 'инвертирования'.
                OR    (HL)               Этим задается 7 разряд для
                                         'abs', но ничего не изме-
                                         няется для 'инвертирования'.
                RLA                      Теперь 7 разряд изменен,
                CCF                      что ведет для 'abs' сброс 7
                RRA                      разряда второго байта и
                                         просто изменяется для
                                         'инвертирования'.
                LD    (HL),A             Запоминается новый второй
                                         байт.
                DEC   HL                 HL указывает опять первый
                                         байт.
                RET                      Окончание.
```

В 'случае целого числа' выполняется аналогичная операция со знаковым байтом.

```
3483 INT-CASE   PUSH  DE                 Запись STKEND в DE.
                PUSH  HL                 Запись указателя числа в HL.
                CALL  2D7F,INT-FETCH     Выбрать знак в C, число в DE.
                POP   HL                 Восстановить указатель
                                         числа в HL.
                LD    A,B                Теперь +FF для 'ABS' +00 для
                                         'инвертирования'.
                OR    C                  Теперь +FF для 'abs' для
                                         'инвертирования' не менять.
```

```
                CPL                     Теперь +00 для 'abs' и
                LD    C,A               измененный байт для 'инвер-
                                        тирования' запомнить в C.
                CALL  2D8E,INT-STORE    Результат запомнить в стеке.
                POP   DE                Возвратить STKEND в DE.
                RET                     Окончание.
```

ФУНКЦИЯ 'SIGNUM'
(Смещение 29: 'sgn')

Эта подпрограмма обрабатывает функцию SGN X и поэтому возвращает 'последнее значение' как 1,, воли X положительно, ноль, если X-ноль, -1, если X отрицательно.

```
3492 sgn        CALL  34E9,TEST-ZERO    Если X 0, только возврат с
                RET   C                 нулем в качестве 'последнего
                                        значения'.
                PUSH  DE                Запись указателя STKEND.
                LD    DE,+0001          Запомнить 1 в DE.
                INC   HL                Указать второй байт X.
                RL    (HL)              Сдвиг 7 разряда и признак
                                        переноса.
                DEC   HL                Указать опять адресат.
                SBC   A,A               Задать с 0-м для положитель-
                LD    C,A               ного X и FF для отрицатель-
                                        ного.
                CALL  2D8E,INT-STORE    Если требуется, занести в
                                        стек 1 или -1.
                POP   DE                Восстановить указатель
                                        STKEND.
                RET                     Окончание.
```

ФУНКЦИЯ 'IN'
(Смещение 2C: 'in')

```
34A5 in         CALL  1E99,FIND-INT2    'Последнее значение' X
                                        помещается в BC.
                IN    A,(C)             Получен сигнал.
                JR    34B0,IN-PK-STK    Переход и занесение
                                        результата в стек.
```

ФУНКЦИЯ 'PEEK'
(Смещение 2B: 'peek')

Эта подпрограмма обрабатывает функцию PEEK X. 'Последнее значение' не заносится в стек с вызовом FIND-INT2 и заменяется значением, которое находится в требуемой ячейке.

```
34AC peek       CALL  1E99,FIND-INT2    Вычислить 'последнее значе-
                                        ние', округленное до ближай-
                                        шего целого, проверить, что
                                        оно в диапазоне и возврат его
                                        в BC.
                LD    A,(BC)            Выбор требуемого байта.
34B0 IN-PK-STK  JP    2D28,STACK-A      Выход с помощью перехода
                                        на STACK-A.
```

ФУНКЦИЯ 'USR'
(Смещение 2D: 'usr-no')

Эта подпрограмма ('USR-ЧИСЛО' В отличие от 'USR-строки') отрабатывает функцию 'USR X', где X является числом. Значение X получается в BC, адрес возврата заносится в стек, а машинный код выполняется из ячейки X.

```
34B3 usr-no       CALL  1E99,FIND-INT2      Вычислить 'последнее'
                                            значение, округленное до
                                            ближайшего целого,
                                            проверить, что оно в
                                            диапазоне и возвратить в BC.
                  LD    HL,+2D2B            Создание адреса возврата,
                  PUSH  HL                  отрабатываемого
                                            подпрограммой STACK-BC.
                  PUSH  BC                  Сделать непрямой переход
                  RET                       на требуемую ячейку.
```

Примечание: Интересно, что регистровая пара IY заново инициализируется при возврате в STACK-BC, но H'L', который содержит следующий указатель литерала, не запоминается. Для успешного возврата в BASIC, H'L' должен на выходе из машинного кода содержать адрес в SCANNING команды 'end-calc', 2758 (16-ричное) (10072 десятичное).

ФУНКЦИЯ 'USR STRING' ('USR-строка')
(Смещение 19: 'usr-$')

Эта подпрограмма обрабатывает функцию USR X$, где X$ является строкой. Подпрограмма возвращает на BC адрес набора разрядов для определенного пользователем графического символа, соответствующего X$. Она выдает сообщение об ошибке A, если X$ не является буквой между a и u или определенным пользователем графическим символом.

```
34BC usr-$        CALL  2BF1,STK-FETCH      Выбор параметров строки X$.
                  DEC   BC                  Для ее проверки уменьшить
                                            длину на 1.
                  LD    A,B                 Если длина не 1, то выдача
                  OR    C                   сообщения об ошибке A.
                  JR    NZ,34E7,REPORT-A
                  LD    A,(DE)              Выбор единичного кода строки.
                  CALL  2C8D,ALPHA          Она обозначает букву?
                  JR    C,34D3,USR-RANGE    Если да, переход, чтобы
                                            получить ее адрес.
                  SUB   +90                 Уменьшить диапазон для
                                            фактического определенного
                                            пользователем графического
                                            символа 0-20 десятичное.
                  JR    C,34E7,REPORT-A     Если вне диапазона, то
                                            сообщение A.
                  CP    +15                 Опять проверка диапазона.
                  JR    NC,34E7,REPORT-A    Если вне диапазона, то
                                            сообщение A.
                  INC   A                   Создание диапазона опреде-
                                            ленного пользователем
                                            графического символа, 1-21
                                            десятичное, как для a-u.
34D3 USR-RANGE    DEC   A                   Теперь создание диапазона
```

```
                                        0-20 десятичное для
                                        каждого случая.
                ADD   A,A               Умножение на 8, чтобы
                ADD   A,A               получить смещение
                ADD   A,A               для адреса.
                CP    +A8               Проверка диапазона смещения.
                JR    NC,34E7,REPORT-A  Если вне диапазона, то
                                        сообщение A.
                LD    BC,(UDG)          Выбор адреса первого опреде-
                                        ленного пользователем графи-
                                        ческого символа в BC.
                ADD   A,C               Добавить C к смещению.
                LD    C,A               Опять в C запомнить
                                        результат.
                JR    NC,34E4,USR-STACK Если нет переноса, переход.
                INC   B                 Приращение B для завершения
                                        адреса.
34E4 USR-STACK  JP    2D2B,STACK-BC     Переход для занесения адреса
                                        в стек.
```

Сообщение A - 'Неправильный аргумент'.

```
34E7 REPORT-A   RST   0008,ERROR-1      Вызов подпрограммы
                DEFB  +09               обработки ошибки.
```

ПОДПРОГРАММА 'TEST-ZERO' ('Проверка на ноль')
Эта подпрограмма вызывается самое меньшее 9 раз, чтобы проверить, является ли число с плавающей точкой нулем. Этот тест требует, чтобы первые 4 байта числа были нулями. Если число в действительности является нулем, то подпрограмма вызывается с заданным признаком переноса.

```
34E9 TEST-ZERO  PUSH  HL                Запись HL на стек.
                PUSH  BC                Запись BC на стек.
                LD    B,A               Записать A в B.
                LD    A,(HL)            Прочитать первый байт.
                INC   HL                Указать второй байт.
                OR    (HL)              OR (или) первый байт
                                        со вторым.
                INC   HL                Указать третий байт.
                OR    (HL)              OR результат с третьим
                                        байтом.
                INC   HL                Указать четвертый байт.
                OR    (HL)              OR результат с четвертым
                                        байтом.
                LD    A,B               Восстановить исходное
                                        значение A.
                POP   BC                И BC.
                POP   HL                Восстановить указатель
                                        числа в HL.
                RET   NZ                Возврат со сброшенным
                                        переносом, если любой из 4
                                        байт не ноль.
                SCF                     Задать признак переноса,
                                        чтобы показать что число
                RET                     было нулем, и возврат.
```

ОПЕРАЦИЯ 'GREATER THAN ZERO' ('Больше нуля')
(Смещение 37: 'greater-0')

Эта подпрограмма возвращает как 'последнее значение' 1, если текущее 'последнее значение' больше нуля и 0, если иначе. Она также используется другими подпрограммами для 'перехода на плюс'.

```
34F9 GREATER-0    CALL  34E9,TEST-ZERO      'Последнее значение' ноль?
                  RET   C                   Если так, возврат.
                  LD    A,+FF               Переход вперед на LESS THAN
                  JR    3507,SIGN-TO-C      ZERO лишь сигнализирует о
                                            необходимости противополож-
                                            ного действия.
```

ФУНКЦИЯ 'NOT' ('Не')
(Смещение 30: 'not')

Эта подпрограмма возвращает как 'последнее значение' 1, если текущее 'последнее значение' является нулем и 0, если иначе. Также используется другими подпрограммами для 'перехода на ноль'.

```
3501 NOT          CALL  34E9,TEST-ZERO      Признак переноса будет
                                            задан, если 'последнее
                                            значение' = 0; этим выдается
                                            правильный результат.
                  JR    350B,FP-0/1         Переход вперед.
```

ОПЕРАЦИЯ 'LESS THAN ZERO' ('Меньше нуля')
(Смещение 36: 'less-0')

Эта подпрограмма возвращает как 'последнее значение' 1, если текущее 'последнее значение' меньше нуля и 0, если иначе. Также используется другими подпрограммами для 'перехода на минус'.

```
3506 less-0       XOR   A                   Очистить регистр A.
3507 SIGN-TO-C    INC   HL                  Указать знаковый байт.
                  XOR   (HL)                Сброшен перенос для положи-
                  DEC   HL                  тельного числа и задан для
                  RLCA                      отрицательного; когда ввод
                                            из GREATER-0, противополож-
                                            ный знак идет на перенос.
```

ПОДПРОГРАММА 'ZERO OR ONE' ('Ноль или единица')
Эта подпрограмма задает 'последнее значение' нулем, если признак переноса сброшен, и единица, если задан. Если вызов из 'E-TO-FP', то она создает 0 или 1 не на стеке, а в mem-0.

```
350B FP-0/1       PUSH  HL                  Запись указателя результата.
                  LD    A,+00               Очистить A без разрушения
                                            переноса.
                  LD    (HL),A              Задать первый байт нулем.
                  INC   HL                  Указать второй байт.
                  LD    (HL),A              Задать второй байт нулем.
                  INC   HL                  Указать третий байт.
                  RLA                       Сдвиг переноса в A, создание
                                            A как 1, если перенос был
                                            задан и 0, если перенос
                                            был сброшен.
```

```
                LD    (HL),A            Задать третий байт 1 или 0.
                RRA                     Обеспечивает A как 0.
                INC   HL                Указать четвертый байт.
                LD    (HL),A            Задать четвертый байт нулем.
                INC   HL                Указать пятый байт.
                LD    (HL),A            Задать пятый байт нулем.
                POP   HL                Восстановить указатель
                                        результата.
                RET                     Окончание.
```

ОПЕРАЦИЯ 'OR' ('Или')
(Смещение 07: 'or')
Эта подпрограмма выполняет двоичную операцию 'X OR Y' и возвращает X, если Y ноль, и 1, иначе.

```
351B or         EX    DE,HL             Указать HL на Y, второе
                                        число.
                CALL  34E9,TEST-ZERO    Проверить, является ли Y
                                        нулем.
                EX    DE,HL             Восстановить указатели.
                RET   C                 Возврат, если Y был 0; X
                                        теперь 'последнее значение'.
                SCF                     Задать признак переноса и
                JR    350B,FP-0/1       сделать переход назад, чтобы
                                        задать 'последнее
                                        значение' 1.
```

ОПЕРАЦИЯ 'NUMBER AND NUMBER' ('Число и число')
(Смещение 08: 'no-&-no')

Эта подпрограмма выполняет двоичную операцию 'X AND Y' и возвращает X, если Y не ноль, и 0, иначе.

```
3524 no-&-no    EX    DE,HL             Указывает HL на Y, DE на X.
                CALL  34E9,TEST-ZERO    Проверка, является ли Y
                                        нулем.
                EX    DE,HL             Переставить указатели назад.
                RET   NC                Возврат с X, как 'последним
                                        значением'.
                                        Если Y не ноль.
                AND   A                 Сброс признака переноса и
                JR    350B,FP-0/1       переход назад, чтобы задать
                                        'последнее значение 0'.
```

ОПЕРАЦИЯ 'STRING AND NUMBER' ('Строка и число')
(Смещение 10: 'str-&-no')

Эта подпрограмма выполняет двоичную операцию 'X$ AND Y' и возвращает X$, если Y не ноль, и пустую строку, иначе.

```
352D str-&-no   EX    DE,HL             Указывает HL на Y, DE на X$.
                CALL  34E9,TEST-ZERO    Проверка, является ли Y
                                        нулем.
                EX    DE,HL             Переставить указатели назад.
                RET   NC                Возврат с X$, как 'последним
                                        значением', если Y не ноль.
                PUSH  DE                Запись указателя числа.
                DEC   DE                Указать 5 байт параметров
```

```
                                        строки, т.е. наибольшую
                                        длину.
                  XOR   A               Очистить регистр A.
                  LD    (DE),A          Теперь наибольшая длина
                                        задана нулем.
                  DEC   DE              Указать наименьшую длину.
                  LD    (DE),A          Теперь наименьшая длина
                                        задана нулем.
                  POP   DE              Восстановить указатель.
                  RET                   Возврат с параметрами
                                        строки как с 'последним
                                        значением'.
```

ОПЕРАЦИИ 'COMPARISON' ('Сравнение')
(Смещения 09-0E и 11-16: 'no-l-eql', 'no-gr-eq', 'nos-neql', 'no-grtr', 'no-less', 'nos-eql', 'str-l-eql', 'str-gr-eq', 'strs-neql', 'str-grtr', 'str-less' и 'strs-eql')
Эта подпрограмма используется для сравнения. Смещение одной операции находится в регистре B в начале подпрограммы.

```
353B no-l-eql     LD    A,B             Единичное смещение поступает
     etc.                               в регистр A.
                  SUB   +08             Теперь диапазон 01-06
                                        и 09-0E.
                  BIT   2,A             Диапазон изменен: 00-02,
                  JR    NZ,3543,EX-OR-NOT  04-06, 08-0A,
                  DEC   A               0C-0E.
3543 EX-OR-NOT    RRCA                  Затем понижен до 00-07
                                        с заданным переносом для
                                        'больше или равно' и
                                        'меньше'; операции с за-
                  JR    NC,354E,NU-OR-STR  данным переносом затем
                  PUSH  AF              обрабатываются как
                  PUSH  HL              дополняющие операции,
                  CALL  343C,EXCHANGE   значения обмениваются.
                  POP   DE
                  EX    DE,HL
                  POP   AF
354E NU-OR-STR    BIT   2,A             Числовые сравнения теперь
                  JR    NZ,3559,STRINGS отделены от строковых
                                        сравнений тестирующим 2
                                        разрядом.
                  RRCA                  Числовые операции теперь
                                        имеют диапазон 00-01 с
                                        заданным переносом для
                                        'равно' и 'не равно'.
                  PUSH  AF              Запись смещения.
                  CALL  300F,SUBTRACT   Числа вычитаются для
                  JR    358C,END-TESTS  конечных тестов.
3559 STRINGS      RRCA                  Сравнения строк имеют теперь
                                        диапазон 02-03 с переносом,
                                        заданным для 'равно' и 'не
                                        равно'.
                  PUSH  AF              Запись смещения.
                  CALL  2BF1,STK-FETCH  Длины и адреса строк
                  PUSH  DE              выбираются из стека
                  PUSH  BC              калькулятора.
                  CALL  2BF1,STK-FETCH
                  POP   HL              Длина второй строки.
3564 BYTE-COMP    LD    A,H
                  OR    L
```

```
                EX    (SP),HL
                LD    A,B
                JR    NZ,3575,SEC-PLUS    Переход, если вторая
                OR    C                   строка не пустая.
356B SECND-LOW  POP   BC                  Здесь вторая строка или
                                          пустая, или меньше первой.
                JR    Z,3572,BOTH-NULL
                POP   AF
                CCF                       Завершен перенос для выдачи
                JR    3588,STR-TEST       правильных результатов
                                          проверок.
3572 BOTH-NULL  POP   AF                  Здесь перенос используется
                JR    3588,STR-TEST       так, как будто он стоит.
3575 SEC-PLUS   OR    C
                JR    Z,3585,FRST-LESS    Первая строка теперь пустая.
                                          вторая - нет.
                LD    A,(DE)              Никакая строка не пустая,
                SUB   (HL)                т.к. сравниваются их
                                          следующие байты.
                JR    C,3585,FRST-LESS    Первый байт меньше,
                JR    NZ,356B,SECND-LOW   второй байт меньше.
                DEC   BC                  Байты равны; итак,
                INC   DE                  длины уменьшены и
                INC   HL                  сделан переход на BYTE-COMP
                EX    (SP),HL             для сравнения следующих
                DEC   HL                  байт сокращенной строки.
                JR    3564,BYTE-COMP
3585 FRST-LESS  POP   BC
                POP   AF
                AND   A                   Здесь очищен перенос для
                                          правильных результатов
                                          проверок.
3588 STR-TEST   PUSH  AF                  Для строковых тестов ноль
                RST   0028,FP-CALC        помещен на стек калькулятора.
                DEFB  +A0,stk-zero
                DEFB  +38,end-calc
358C END-TESTS  POP   AF                  Эти три теста, вызы-
                PUSH  AF                  ваемые по необходи-
                CALL  C,3501,NOT          мости, выдают коррек-
                POP   AF                  тные результаты для
                PUSH  AF                  всех 12 сравнений.
                CALL  NC,34F9,GREATER-0   Изначальный перенос
                POP   AF                  задается для 'не равно' и
                RRCA                      'равно', конечный перенос
                CALL  NC,3501,NOT         задан для 'больше, чем',
                                          для 'меньше, чем', и 'равно'.
                RET                       Окончание.
```

ОПЕРАЦИЯ 'STRING CONCATENATION' ('Сцепление строк')
(Смещение 17: 'strs-add')

Эта подпрограмма выполняет двоичную операцию 'A$+B$'. Для этих строк выбираются параметры и находится общая длина. Создается место для строк в рабочей области и строки копируются. Результатом подпрограммы является создание временной переменной 'A$+B$', которая постоянно находится в рабочей области.

```
359C strs-add   CALL  2BF1,STK-FETCH      Выбираются и записы-
                PUSH  DE                  ваются параметры.
```

```
                  PUSH  BC                  второй строки.
                  CALL  2BF1,STK-FETCH      Выбираются параметры
                                            первой отроки.
                  POP   HL
                  PUSH  HL                  Длины теперь в HL и BC.
                  PUSH  DE                  Записаны параметры
                  PUSH  BC                  первой строки.
                  ADD   HL,BC               Вычислена и передана
                  LD    B,H                 в BC общая длина
                  LD    C,L                 двух строк.
                  RST   0030,BC-SPACES      Создано необходимое место.
                  CALL  2AB2,STK-STORE      Параметры новой строки пере-
                                            даны на стек калькулятора.
                  POP   BC                  Найдены параметры
                  POP   HL                  первой строки, и
                  LD    A,B                 строка скопирована в
                  OR    C                   рабочую область с
                  JR    Z,35B7,OTHER-STR    длиной не пустой
                  LDIR                      строки.
35B7 OTHER-STR    POP   BC                  Точно такая же про-
                  POP   HL                  цедура следует для
                  LD    A,B                 второй строки для
                  OR    C                   выдачи 'A$+B$'.
                  JR    Z,35BF,STK-PNTRS
                  LDIR
```

ПОДПРОГРАММА 'STK-PNTRS'
Эта подпрограмма сбрасывает регистровую пару HL, чтобы указать первый байт 'последнего значения', т.е. STKEND-5, и регистровую пару DE, чтобы указать 'единицу после' 'последнего значения', т.е. STKEND.

```
35BF STK-PNTRS    LD    HL,(STKEND)         Выбор текущего значении
                                            STKEND.
                  LD    DE,+FFFB            Установить DE в -5 с
                                            дополнением до двух.
                  PUSH  HL                  Занести в стек значение для
                                            STKEND.
                  ADD   HL,DE               Вычислить STKEND-5.
                  POP   DE                  Теперь DE содержит STKEND,
                                            а HL содержит STKEND-5.
                  RET
```

ФУНКЦИЯ 'CHR$'
(Смещение 2F: 'chrs')

Эта подпрограмма обрабатывает функцию CHR$ X и создает единичную строку символов в рабочей области.

```
35C9 chrs         CALL  2DD5,FP-TO-A        'Последнее значение'
                                            помещено в регистр A.
                  JR    C,35DC,REPORT-B     Выдает сообщение об ошибке,
                                            если X больше, чем 256
                                            десятичное.
                  JR    NZ,35DC,REPORT-B    Или отрицательное.
                  PUSH  AF                  Запись помещенного значения
                                            X.
                  LD    BC,+0001            Создание одного места
                  RST   0030,BC-CPACES      в рабочей области.
```

```
                POP   AF                Выбор значения.
                LD    (DE),A            Скопировать значение в
                                        рабочую область.
                CALL  2AB2,STK-STORE    Передать параметры новой
                                        строки на стек калькулятора.
                EX    DE,HL             Сброс указателей.
                RET                     Окончание.
```

Сообщение B - 'Целое дне диапазона'.

```
35DC REPORT-B   RST   0008,ERROR-1      Вызов подпрограммы
                DEFB  +0A               обработки ошибок.
```

ФУНКЦИЯ 'VAL' и 'VAL$'
(Смешение 1D: 'val' и 18: 'val$')

Эта подпрограмма обрабатывает функции VAL X$ и VAL$ X$. При обработке VAL X$ она возвращает 'последнее значение', которое является результатом вычисления строки (без ограничивающих кавычек), как числовые выражения. При обработке VAL$ X$ вычисляется X$ (без ограничивающих кавычек) как строковое выражение, и на стек калькулятора возвращаются периметры строкового выражения как 'последнего значения'.

```
35DE val        LD    HL,(CH-ADD)       Текущее значение CH-ADD
     (также val$) PUSH  HL                сохранено на машинном стеке.
                LD    A,B               'Смещение' для 'val' или
                                        'val$' должно быть в
                                        регистре B; теперь оно
                                        копируется в A.
                ADD   A,+E3             Создать +00 и перенос, за-
                                        данный для 'val', +FB и пере-
                                        нос, сброшенный для 'val$'.
                SBC   A,A               Создать +FF (поэтому задан 6
                                        разряд для 'val', но +00 (6
                                        разряд сброшен для 'val$').
                PUSH  AF                Записать этот 'признак' на
                                        машинный стек.
                CALL  2BF1,STK-FETCH    Выбраны параметры
                PUSH  DE                строки; записан начальный
                INC   BC                адрес; к длине добавляется
                RST   0030,BC-SPACES    один байт и создается место
                                        для строки (+1) в рабочей
                                        области.
                POP   HL                Начальный адрес строки идет
                                        в HL, как исходный адрес.
                LD    (CH-ADD),DE       Указатель первый нового мес-
                PUSH  DE                та идет в CH-ADD и машинный
                                        стек.
                LDIR                    Строка копируется в рабочую
                                        область вместе с дополни-
                                        тельным байтом.
                EX    DE,HL             Переключение указателей.
                DEC   HL                Дополнительный байт
                LD    (HL),+0D          заменяется символом 'возврат
                                        каретки'.
                RES   7,(FLAGS)         Сброшен признак синтаксиса и
                CALL  24FB,SCANNING     строка просмотрена на
                                        правильность синтаксиса.
```

```
                RST   0018,GET-CHAR       Выбирается символ после
                                          строки.
                CP    +0D                 Проверка, достигнут ли конец
                                          выражения.
                JR    NZ,360C,V-RPORT-C   Если нет, сообщение об
                                          ошибке.
                POP   HL                  Выбирается начальный адрес
                                          строки.
                POP   AF                  Выбирается 'признак' для
                XOR   (FLAGS)             'val/val$' и 6 разряд срав-
                AND   +40                 нивается с 6 разрядом
                                          результата просмотра син-
                                          таксиса.
360C V-RPORT-C  JP    NZ,1C8A,REPORT-C    Бели они не сравниваются,
                                          то сообщение об ошибке.
                LD    (CH-ADD),HL         Начальный адрес опять в
                                          CH-ADD.
                SET   7,(FLAGS)           Для выполнения строки задан
                                          признак.
                CALL  24FB,SCANNING       Строка обрабатывается как
                                          'следующее выражение' и
                                          создается 'последнее
                                          выражение'.
                POP   HL                  Восстанавливается ис-
                LD    (CH-ADD),HL         ходное значение CH-ADD.
                JR    35BF,STK-PNTRS      Подпрограмма выходит через
                                          STK-PNTRS, которая
                                          сбрасывает указатель.
```

ФУНКЦИЯ 'STR$'
(Смещение 2E: 'str$')

Подпрограмма обрабатывает функцию STR$ X и возвращает 'последнее значение', которое является набором параметров, которые определяют строку, содержащую информацию для экрана, если X отображается командой PRINT.

```
361F str$       LD    BC,+0001            В рабочей области со-
                RST   0030,BC-SPACES      здается одно место, и
                LD    (K-CUR),HL          его адрес копируется
                                          в K-CUR, адрес курсора.
                PUSH  HL                  Этот адрес также записы-
                                          вается на стек.
                LD    HL,(CURCHL)         На машинный стек записывает-
                PUSH  HL                  ся текущий адрес канала.
                LD    A,+FF               Открыт канал 'R', позволяю-
                CALL  1601,CHAN-OPEN      щий строке 'отпечататься' в
                                          рабочей области.
                CALL  2DE3,PRINT-FP       'Последнее значение' X
                                          отпечатывается теперь в
                                          рабочей области и рабочая
                                          область с каждым сим-
                                          волом расширяется.
                POP   HL                  Восстановить CURCHL в HL и
                CALL  1615,CHAN-FLAG      восстановить признаки,
                                          которые этому соответствуют.
                POP   DE                  Восстановить начальный
                                          адрес строки.
                LD    HL,(K-CUR)          Теперь адрес курсора
```

```
                                находится после
                AND   A         конца строки и
                SBC   HL,DE     разность отсюда яв-
                                ляется длиной.
                LD    B,H       Передача длины в BC.
                LD    C,L
                CALL  2AB2,STK-STO-$  Передать параметры новой
                                строки на стек калькулятора.
                EX    DE,HL     Сброс указателей.
                RET             Окончание.
```

Примечание: Смотрите PRINT-FP для объяснения ошибки 'PRINT "A"+STR$ 0.1'.

ПОДПРОГРАММА 'READ-IN' ('Считывание')
(Смещение 1A: 'read-in')

Эта подпрограмма вызывается через смещение калькулятора, через первую строку подпрограммы S-INKEY$ в SCANNING. Она появляется, чтобы обеспечить передачу длины через различные возможные на стандартном Spectrum потоки. Подобно INKEY$ подпрограмма возвращает строку.

```
3645 read-in    CALL  1E94,FIND-INT1    В регистре A помещен
                                        числовой параметр.
                CP    +10               Он меньше, чем 16
                                        десятичное?
                JP    NC,1E9F,REPORT-B  Если нет, сообщение об
                                        ошибке.
                LD    HL,(CURCHL)       Текущий адрес канала
                PUSH  HL                записывается на машинный
                                        стек.
                CALL  1601,CHAN-OPEN    Открыт канал, определенный
                                        параметром.
                CALL  15E6,INPUT-AD     Получен сигнал типа
                                        'значение клавиши'.
                LD    BC,+0000          По умолчанию длина
                                        результирующей строки = 0.
                JR    NC,365F,R-I-STORE Переход, если сигнала нет.
                INC   C                 Задать длину 1.
                RST   0030,BC-SPACES    Создать место в рабочей
                                        области.
                LD    (DE),A            Поместить туда строку.
365F R-I-STORE  CALL  2AB2,STK-STO-$    Передать параметры строки на
                                        стек калькулятора.
                POP   HL                Восстановить CURCHL и
                CALL  1615,CHAN-FLAG    соответствующий признак.
                JP    35BF,STK-PNTRS    Выход с установкой
                                        указателей.
```

ФУНКЦИЯ 'CODE' ('Код')
(Смещение 1C: 'code')

Эта подпрограмма обрабатывает функцию CODE A$ и возвращает код Spectrum для первого символа в A$, или, если A$ должно быть пустым, ноль.

```
3669 code       CALL  2BF1,STK-FETCH    Выбраны параметры строки.
                LD    A,B               Строка проверяется, а
                OR    C                 A, содержащий 0, переносится
```

```
                JR    Z,3671,STK-CODE     вперед, если A$ является
                                          пустой строкой.
                LD    A,(DE)              В противном случае, в A за-
                                          носится код первого символа.
3671 STK-CODE   JP    2D28,STACK-A        Выход черед STACK-A, которое
                                          выдает правильное 'последнее
                                          значение'.
```

ФУНКЦИЯ 'LEN'
(Смещение 1E: 'len')

Эта подпрограмма обрабатывает функцию LEN A$ и возвращает 'последнее значение', которое равно длине строки.

```
3674 len        CALL  2BF1,STK-FETCH      Выбираются параметры строки.
                JP    2D2B,STACK-BC       Выход через STACK-BC, которое
                                          выдает правильное 'последнее
                                          значение'.
```

ПОДПРОГРАММА 'DECREASE THE COUNTER' ('Уменьшить счетчик')
(Смещение 35: 'dec-jr-nz')

Эта подпрограмма вызывается только подпрограммой SERIES GENERATOR и в действительности является операцией 'DJNZ', но счетчик является системной переменной BREG, а не регистр B.

```
367A dec-jr-nz  EXX                       Переход на заданный альтер-
                PUSH  HL                  нативный набор регистров и
                                          запись следующего указателя
                                          литерала на машинный стек.
                LD    HL,+5C67            HL должно указывать BREG.
                DEC   (HL)                Уменьшить BREG.
                POP   HL                  Восстановить следующий
                                          указатель литерала.
                JR    NZ,3687,JUMP-2      Выполнен переход на не ноль.
                INC   HL                  Передан следующий литерал.
                EXX                       Возврат на основной набор,
                                          регистров.
                RET                       Окончание.
```

ПОДПРОГРАММА 'JUMP' ('Переход')
(Смещение 33: 'jump')

Эта подпрограмма выполняет безусловный переход при вызове ее литералом '33'. Также используется подпрограммами DECREASE THE COUNTER и JUMP ON TRUE.

```
3686 JUMP       EXX                       Переход на альтернативный
                                          набор регистров.
3687 JUMP-2     LD    E,(HL)              Следующий литерал (длина
                                          перехода) занесен в
                                          регистр E'.
                LD    A,E                 В A формируются числа 00
                RLA                       00 (16-ричное) или FF
                SBC   A,A                 (16-ричное), в соответствии,
                LD    D,A                 какой E' - положительный или
                                          отрицательный, и затем
                                          копируется в D'.
                ADD   HL,DE               Теперь регистры H' и L'
```

```
                  EXX                     содержат следующий
                                          указатель литерала.
                  RET                     Окончание.
```

ПОДПРОГРАММА 'JUMP ON TRUE' ('Переход при истине')
(Смещение 00: 'jump-true')

Эта подпрограмма выполняет условный переход, если 'последнее значение' на стеке калькулятора или, более того, число, адресованное в некоторый момент регистровой парой DE, является истинным.

```
368F jump-true    INC   DE               Указывает на третий
                  INC   DE               байт, который 0 или 1.
                  LD    A,(DE)           Выбрать байт в регистре A.
                  DEC   DE               Указать еще раз на
                  DEC   DE               первый байт.
                  AND   A                Проверка третьего байта:
                                         он 0?
                  JR    NZ,3686,JUMP     Выполнить переход, если байт
                                         не 0, т.е. если число не
                                         ложно.
                  EXX                    Переход на альтернативный
                                         набор регистров.
                  INC   HL               Передать длину перехода.
                  EXX                    Назад на основной набор
                                         регистров.
                  RET                    Окончание.
```

ПОДПРОГРАММА 'END-CALC'.
(Смещение 38: 'end-calc')

Эта подпрограмма заканчивает операцию RST 0028.

```
369B end-calc     POP   AF               Отброшен адрес возврата к
                                         калькулятору ('RE-ENTRY').
                  EXX                    Вместо этого на машинный
                  EX    (SP).HL          стек помещен адрес в H'L' и
                  EXX                    выполнен туда непрямой пе-
                                         реход. Теперь H'L' будет
                                         содержать любой более ранний
                                         адрес в последовательности
                                         адресов калькулятора.
                  RET                    Окончание.
```

ПОДПРОГРАММА 'MODULUS' ('Модуль')
(Смещение 32: 'n-mod-m')

Эта подпрограмма вычисляет M (mod M), где M является положительным целым числом, содержащимся на вершине стека калькулятора, 'последним значением', а N является целым числом, содержащимся на стеке ниже M.
Подпрограмма возвращает целое частное INT (N/M) на вершину стека калькулятора, 'последнее значение', а остаток N-INT (N/M) на второе место на стеке.
Эта подпрограмма вызывается во время вычисления случайного числа, чтобы понизить N mod 65537 десятичное.

```
36A0 n-mod-m      RST   0028,FP-CALC      N,M
                  DEFB  +C0,st-mem-0      N,M mem-0 содержит M
```

```
                DEFB  +02,delete          N
                DEFB  +31,duplicate       N, N
                DEFB  +E0,get-mem-0       N, N, M
                DEFB  +05,division        N, N/M
                DEFB  +27,int             N, INT (N/M)
                DEFB  +E0,get-mem-0       N, INT (N/M),M
                DEFB  +01,exchange        N, M, INT (N/M)
                DEFB  +C0,st-mem-0        N, M, INT (N/M) mem-0 содержит
                                          INT (N/M)
                DEFB  +04,multiply        N, M*INT (N/M)
                DEFB  +03,subtract        n-M*INT (N/M)
                DEFB  +E0,get-mem-0       n-M*INT (N/M), INT (N/M)
                DEFB  +38,end-calc
                RET                       Окончание.
```

ФУНКЦИЯ 'INT'
(Смещение 27: 'int')

Эта подпрограмма обрабатывает функцию INT X и возвращает 'последнее значение', которое является 'целой частью' данной величины. Таким образом INT 2.4 выдает 2, но т.к. подпрограмма всегда округляет результат в меньшую сторону, то INT -2.4 будет -3. Подпрограмма использует подпрограмму на 3214 INTEGER TRUNCATION TOWARDS ZERO, чтобы создавать I (X), для того, чтобы I (2.4) давало 2, а I (-2.4) давало -2. Таким образом, INT X задается I (X) для значений X, которые больше или равны 0, и I (X)-1 для отрицательных значений X, которые уже не целые числа, при результате, конечно, I (X).

```
36AF int        RST   0028,FP-CALC        X
                DEFB  +31,duplicate       X, X
                DEFB  +36,less-0          X, (1/0)
                DEFB  +00,jump-true       X
                DEFB  +04, to 36B7,X-NEG  X
```

Для значений X, которые больше или равны 0, перехода нет, и I (X) находится быстро.

```
                DEFB  +3A,truncate        I (X)
                DEFB  +38,end-calc
                RET                       Окончание.
```

Когда X является отрицательным целым числом, возвращается I (X), иначе, возвращается I (X)-1.

```
36B7 X-NEG      DEFB  +31,duplicate       X, X
                DEFB  +3A,truncate        X, I (X)
                DEFB  +C0,st-mem-0        X, I (X) mem-0 сожержит I (X)
                DEFB  +03,subtract        X-I (X)
                DEFB  +E0,get-mem-0       X-I (X), I (X)
                DEFB  +01,exchange        I (X), X-I (X)
                DEFB  +30,not             I (X), (1/0)
                DEFB  +00,jump-true       I (X)
                DEFB  +03,to 36C2,EXIT    I (X)
```

Для значений X, являющихся отрицательными целыми числами, выполняется переход, иначе перехода нет, и вычисляется I (X)-1.

```
                DEFB  +A1,stk-one         I (X), 1
                DEFB  +03,subtract        I (X)-1
```

В любом случае подпрограмма заканчивается так:

```
36C2 EXIT         DEFB  +38,end-calc        I (X) или I (X)-1
                  RET
```

ФУНКЦИЯ 'EXPONENTIAL' ('Экспонента')
(Смещение 26: 'exp')

Эта подпрограмма обрабатывает функцию EXP X и является первой из четырех подпрограмм, которые использует SERIES GENERATOR для создания полиномов Чебышева.
Аппроксимация функции EXP X производится следующим образом:

1. X делится на LN2, чтобы выдать Y, так что 2 в степени Y является требуемым результатом.

2. Находится точное N, что N=INT Y.

3. Находится значение W=Y-N, где 0 <= W <= 1, для того, чтобы ряд сходился.

4. Если сформирован аргумент Z, то он такой как Z=2*W-1.

5. Для возврата 2**W используется SERIES GENERATOR.

6. Наконец N добавляется к порядку, выдавая 2**(N+W), что является 2**Y и является поэтому требуемым ответом для EXP X.

В Приложении иллюстрируется этот метод с использованием программы BASIC.

```
36C4 EXP          RST   0028,FP-CALC        X
```

Выполнить 1 шаг.

```
                  DEFB  +3D,re-stack        X (в полной форме с
                                            плавающей точкой).
                  DEFB  +34,stk-data        X, 1/LN 2
                  DEFB  +F1,exponent+81
                  DEFB  +38,+AA,+3B,+29
                  DEFB  +04,multiply        X/LN 2 = Y
```

Выполнить 2 шаг.

```
                  DEFB  +31,duplicate       Y, Y
                  DEFB  +27,int,1C46        Y, INT Y = N
                  DEFB  +C3,st-mem-3        Y, N mem-3 содержит N
```

Выполнить 3 шаг.

```
                  DEFB  +03,subtract        Y-N = W
```

Выполнить 4 шаг.

```
                  DEFB  +31,duplicate       W, W
                  DEFB  +0F,addition        2*W
                  DEFB  +A1,stk-one         2*W, 1
                  DEFB  +03,subtract        2*W-1 = Z
```

Выполнить 5 шаг, передавая в SERIES GENERATOR параметр '8' и 8 требуемых констант.

```
                  DEFB  +88,series-08       Z
     1.           DEFB  +13,exponent+63
```

```
                DEFB  +36,(+00,+00,+00)
     2.         DEFB  +58,exponent+68
                DEFB  +65,+66,(+00,+00)
     3.         DEFB  +9D,exponent+6D
                DEFB  +78,+65,+40,(+00)
     4.         DEFB  +A2,exponent+72
                DEFB  +60,+32,+C9,(+00)
     5.         DEFB  +E7,exponent+77
                DEFB  +21,+F7,+AF,+24
     6.         DEFB  +EB,exponent+7B
                DEFB  +2F,+B0,+B0,+14
     7.         DEFB  +EE,exponent +7E
                DEFB  +7E,+BB,+94,+58
     8.         DEFB  +F1,exponent+81
                DEFB  +3A,+7E,+F8,+CF
```

В конце последнего цикла 'последнее значение' равно 2**W.

Выполнить 6 шаг.

```
                DEFB  +E3,get-mem-3       2**W, N
                DEFB  +38,end-calc
                CALL  2DD5,FP-TO-A        В регистр A заносится
                                          абсолютная величина N mod
                                          256 десятичная.
                JR    NZ,3705,N-NEGTV     Если N отрицательно, то
                                          переход вперед.
                JR    C,3703,REPORT-6     Если ABS N больше, чем 255
                                          десятичное, то ошибка.
                ADD   A,(HL)              Теперь к порядку добавим
                                          ABS N.
                JR    NC,370C,RESULT-OK   Переход, если e не больше
                                          255 десятичного.
```

Сообщение 6 - 'Слишком большое число'.

```
3703 REPORT-6   RST   0008,ERROR-1        Вызов подпрограммы
                DEFB  +05                 обработки ошибок.

3705 N-NEGTV    JR    C,370E,RSLT-ZERO    Если N меньше -255 десятич-
                                          ного, то результат должен
                                          быть = 0.
                SUB   (HL)                Вычесть ABS N из порядка,
                                          поскольку N отрицательно.
                JR    NC,370E,RSLT-ZERO   Если e меньше 0, то
                                          нулевой результат.
                NEG                       Минус e заменяется на e.
370C RESULT-OK  LD    (HL),A              Вводится порядок e.
                RET                       Окончание: 'последнее
                                          значение' это EXP X.
370E RSLT-ZERO  RST   0028,FP-CALC        Используйте кальку-
                DEFB  +02,delete          лятор для создания 0
                DEFB  +A0,stk-zero        как 'последнего зна-
                DEFB  +38,end-calc        чения'.
                RET                       Окончание с EXP X=0.
```

ФУНКЦИЯ 'NATURAL LOGARITHM' ('Натуральный логарифм')
(Смещение 25: 'ln')

Эта подпрограмма обрабатывает функцию LN X и является второй из четырех подпрограмм, которые используют SERIES GENERATOR для создания

полиномов Чебышева.

Аппроксимация LN X производится следующим образом:

1. Проверяется X, и если X не положительно, выдается сообщение A.
2. X затем разбивается на истинный показатель e и мантиссу X' = X/(2**e'), где X' больше или равно 0.5, но меньше 1.
3. Формируются требуемые значения Y1 или Y2. Если X' больше 0,8, то Y1 = e'*LN 2, иначе Y2 = (e'-1)*LN 2.
4. Если X' больше 0.8, то величина X'-1 заносится в стек; иначе, в стек заносится 2*X'-1.
5. Теперь формируется аргумент Z: если X' больше 0.8, то Z = 2.5*X'-3, иначе, Z = 5*X-3. В любом случае -1 <= Z <- 1, является условием сходимости ряда.
6. Для создания требуемой функции используется SERIES GENERATOR.
7. В конечном счете к LN X, возвращенному как 'последнее значение', приводит простое умножение или сложение.

```
3713 ln          RST   0028,FP-CALC        X

Выполнить шаг 1.

                 DEFB  +3D,re-stack        X (в полной форме с
                                           плавающей точкой).
                 DEFB  +31,duplicate       X, X
                 DEFB  +37,greater-0       X, (1/0)
                 DEFB  +00,jump-true       X
                 DEFB  +04,to 371C, VALID  X
                 DEFB  +38,end-calc        X

Сообщение A - 'Неправильный аргумент'.

371A REPORT-A    RST   0008,ERROR-1        Вызов подпрограммы
                 DEFB  +09                 обработки ошибок.

Выполнить 2 шаг.

371C VALID       DEFB  +A0,stk-zero        X,0 На удаленную 1 на-
                                           кладывается 0.
                 DEFB  +02,delete          X
                 DEFB  +38,end-calc        X
                 LD    A,(HL)              Показатель e переходит в A.
                 LD    (HL),+80            X уменьшено до X'.
                 CALL  2D28,STACK-A        Стек содержит: X', e.
                 RST   0028,FP-CALC        X', e
                 DEFB  +34,stk-data        X', e, 128 (десятичное)
                 DEFB  +38,exponent+88

                 DEFB  +00,(+00,+00,+00)
                 DEFB  +03,subtract        X', e'

Выполнить 3 шаг.

                 DEFB  +01,exchange        e', X'
                 DEFB  +31,duplicate       e', X', X'
                 DEFB  +34,stk-data        e', X', X',0.8 (десятичное)
                 DEFB  +F0,exponent+80
                 DEFB  +4C,+CC,+CC,+CD
                 DEFB  +03,subtract        e', X', X'-0.8
                 DEFB  +37,greater-0       e', X', (1/0)
                 DEFB  +00,jump-true       e', X'
```

```
                DEFB  +08,to 373D, GRE.8  e', X'
                DEFB  +01,exchange        X', e'
                DEFB  +A1,stk-one         X', e', 1
                DEFB  +03,subtract        X', e'-1
                DEFB  +01,exchange        e'-1, X'
                DEFB  +38,end-calc        e'-1, X'
                INC   (HL)                2*X'
                RST   0028,FP-CALC        e'-1,2*X'
373D GRE.8      DEFB  +01,exchange        X',e' - X' большое.
                                          2*X',e'-1 - X' малое.
                DEFB  +34,stk-data        X',e',LN 2
                DEFB  +F0,exponent+80     2*X',e'-1, LN 2
                DEFB  +31,+72,+17,+F8
                DEFB  +04,multiply        X',e'*LN 2 = Y1
                                          2*X', (e'-1)*LN 2 = Y2
```

Выполнить 4 шаг.

```
                DEFB  +01,exchange        Y1, X' - X' большое.
                                          Y2, 2*X' - X' малое.
                DEFB  +A2,stk-half        Y1, X', .5 (десятичное)
                                          Y2, 2*X', .5
                DEFB  +03,subtract        Y1, X'-.5
                                          Y2, 2*X'-.5
                DEFB  +A2,stk-half        Y1, X'-.5, .5
                                          Y2, 2*X'-.5, .5
                DEFB  +03,subtract        Y1, X'-1
                                          Y2, 2*X'-1
```

Выполнить 5 шаг.

```
                DEFB  +31,duplicate       Y, X'-1, X'-1
                                          Y2, 2*X'-1, 2*X'-1
                DEFB  +34,stk-data        Y1, X'-1, X'-1, 2.5 (десятичное)
                                          Y2, 2*X'-1, 2*X'-1, 2.5
                DEFB  +32,exponent+82
                DEFB  +20,(+00,+00,+00)
                DEFB  +04,multiply        Y1, X'-1,2.5*X'-2.5
                                          Y2, 2*X'-1, 5*X'-2.5
                DEFB  +A2,stk-half        Y1, X'-1,2.5*X'-2.5, .5
                                          Y2, 2*X'-1, 5*X'-2.5, .5
                DEFB  +04,subtract        Y1, X'-1,2.5*X'-3 = Z
                                          Y2, 2*X'-1, 5*X'-3 = Z
```

Выполнить 6 шаг, передавая в SERIES GENERATOR десятичный параметр '12' и двенадцать требуемых констант.

```
                DEFB  +8C,series-0C       Y1, X'-1, Z or Y2, 2*X'-1, Z
     1.         DEFB  +11,exponent+61
                DEFB  +AC,(+00,+00,+00)
     2.         DEFB  +14,exponent+64
                DEFB  +09,(+00,+00,+00)
     3.         DEFB  +56,exponent+66
                DEFB  +DA,+A5,(+00,+00)
     4.         DEFB  +59,exponent+69
                DEFB  +30,+C5,(+00,+00)
     5.         DEFB  +5C,exponent+6C
                DEFB  +90,+AA,(+00,+00)
     6.         DEFB  +9E,exponent+6E
                DEFB  +70,+6F,+61,(+00)
```

```
     7.           DEFB  +A1,exponent+71
                  DEFB  +CB,+DA,+96,(+00)
     8.           DEFB  +A4,exponent+74
                  DEFB  +31,+9F,+B4,(+00)
     9.           DEFB  +E7,exponent+77
                  DEFB  +A0,+FE,+5C,+FC
     10.          DEFB  +EA,exponent+7A
                  DEFB  +1B,+43,+CA,+36
     11.          DEFB  +ED,exponent+7D
                  DEFB  +A7,+9C,+7E,+5E
     12.          DEFB  +F0,exponent+80
                  DEFB  +6E,+23,+80,+93
```

В конце цикла 'последним значением' является:

```
     или          LN X'/(X'-1) для больших значений X'
     или          LN (2*X')/(2*X'-1) для малых значений X'.
```

Выполнить 7 шаг.

```
                  DEFB  +04,multiply        Y1=LN (2**e'), LN X'
                                            Y2=LN (2**(e'-1)), LN (2*X')
                  DEFB  +0F,addition        LN (2**e')*X') = LN X
                                            LN (2**(e'-1)*2*X')  = LN X
                  DEFB  +38,end-calc        LN X
                  RET                       Окончание. 'Последним
                                            значением' является LN X.
```

ПОДПРОГРАММА 'REDUCE ARGUMENT' ('Уменьшение аргумента')
(Смещение 39: 'get-argt')

Эта подпрограмма преобразует аргумент X функции SIN X или COS X в значение V.
Сначала находится значение Y как: Y = X/(2*PI) - INT(X/2*PI) + 0.5), где Y больше или равен -.5, но меньше +.5.
Подпрограмма возвращает:

```
     V = 4*Y       if -1 <=4*Y <=1           - случай 1.
или, V = 2-4*Y     if 1 <4*Y <2              - случай 2.
или, V = -4*Y-2    if -2 <=4*Y < -1.         - случай 3.
```

В каждом случае, -1 < =V <=1 and SIN (PI*V/2) = SIN X

```
3783 get-argt     RST   0028,FP-CALC        X
                  DEFB  +3D,re-stack        X(в полной форме с
                                            плавающей точкой).
                  DEFB  +34,stk-data        X, 1/(2*PI)
                  DEFB  +EE,exponent+7E
                  DEFB  +22,+F9,+83,+6E
                  DEFB  +04,multiply        X/(2*PI)
                  DEFB  +31,duplicate       X/(2*PI), X/(2*PI)
                  DEFB  +A2,stk-half        X/(2*PI), X/(2*PI), 0.5
                  DEFB  +0F,addition        X/(2*PI), X/(2*PI)+0.5
                  DEFB  +27,int,1C46        X/(2*PI), INT (X/(2*PI)+0.5)
                  DEFB  +03,subtract,174C   X/(2*PI)-INT (X/(2*PI)+0.5)=Y
```

Примечание: Взяв INT и добавив 0.5, округляем результат до ближайшего целого.

```
                DEFB  +31,duplicate       Y, Y
                DEFB  +0F,addition        2*Y
                DEFB  +31,duplicate       2*Y, 2*Y
                DEFB  +0F,addition        4*Y
                DEFB  +31,duplicate       4*Y, 4*Y
                DEFB  +2A,abs             4*Y, ABS (4*Y)
                DEFB  +A1,stk-one         4*Y, ABS (4*Y), 1
                DEFB  +03,subtract        4*Y, ABS (4*Y)-1 = Z
                DEFB  +31,duplicate       4*Y, Z, Z
                DEFB  +37,greater-0       4*Y, Z, (1/0)
                DEFB  +C0,st-mem-0        mem-0 содержит результат
                                          проверки.
                DEFB  +00,jump-true       4*Y, Z
                DEFB  +04, to 37A1,ZPLUS  4*Y, Z
                DEFB  +02,delete          4*Y
                DEFB  +38,end-calc        4*Y = V - случай 1.
                RET                       Окончание.
```

Если был выполнен переход, то продолжение.

```
37A1 ZPLUS      DEFB  +A1,stk-one         4*Y, Z, 1
                DEFB  +03,subtract        4*Y, Z-1
                DEFB  +01,exchange        Z-1,4*Y
                DEFB  +36,less-0          Z-1,(1/0)
                DEFB  +00,jump-true       Z-1
                DEFB  +02,to 37A8,YNEG    Z-1
                DEFB  +1B,negate          1-Z
37A8 YNEG       DEFB  +38,end-calc        1-Z = V - случай2.
                                          Z-1 = V - случай 3.
                RET                       Окончание.
```

ФУНКЦИЯ 'COSINE'
(Смещение 20: 'cos')

Эта подпрограмма обрабатывает функцию COS X и возвращает последнее значение', которое является приближенным к COS X.
Подпрограмма использует выражение:
COS X = SIN (PI*W/2), где -1 <=W <=1.
Для вывода W для X подпрограмма использует результат проверки, полученный в предыдущей подпрограмме, и запоминает для этой цели в mem-0. Затем, она переходит в SINE, подпрограмму, вводимую на C-ENT, для создания 'последнего значения' COS X.

```
37AA cos        RST   0028,FP-CALC.       X
                DEFB  +39,get-argt        V
                DEFB  +2A,abs             ABS V
                DEFB  +A1,stk-one         ABS V, 1
                DEFB  +03,subtract        ABS V-1
                DEFB  +E0,get-mem-0       ABS V-1, (1/0)
                DEFB  +00,jump-true       ABS V-1
                DEFB  +06, to 37B7,C-ENT  ABS V-1 = W
```

Если не выполнен переход, продолжение.

```
                DEFB  +1B,negate          1-ABS V
                DEFB  +33,jump            1-ABS V
                DEFB  +03, to 37B7,C-ENT  1-ABS V = W
```

ФУНКЦИЯ 'SINE'
(Offset 1F: 'sin')

Эта подпрограмма обрабатывает функцию SIN X и является третьей из четырех подпрограмм, которые использует SERIES GENERATOR для создания полиномов Чебышева.
Аппроксимация SIN X осуществляется следующим образом:

1. Уменьшается аргумент X и в этом случае W прямо равно V.

Заметим, что -1 <= W < -1, что и требуется для сходимости ряда.

2. Сформирован аргумент Z, как Z=2*W*W-1.

3. Для возвращения (SIN (PI*W/2))/W используется SERIES GENERATOR.

4. В конце простое умножение дает SIN X.

```
37B5 sin          RST   0028 FP-CALC        X

Выполнить 1 шаг.

                  DEFB  +39,get-argt        W

Выполнить 2 шаг. Теперь подпрограмма общая для SINE и COSINE.

37B7 C-ENT        DEFB  +31,duplicate       W, W
                  DEFB  +31,duplicate       W, W, W
                  DEFB  +04,multiply        W, W*W
                  DEFB  +31,duplicate       W, W*W, W*W
                  DEFB  +0F,addition        W, 2*W*W
                  DEFB  +A1,stk-one         W, 2*W*W, 1
                  DEFB  +03,subtract        W, 2*W*W-1 = Z

Выполнить 3 шаг, передав в SERIES GENERATOR параметр '6' и 6 требуемых констант.

                  DEFB  +86,series-06       W, Z
     1.           DEFB  +14,exponent+64
                  DEFB  +E6,(+00,+00,+00)
     2.           DEFB  +5C,exponent+6C
                  DEFB  +1F,+0B,(+00,+00)
     3.           DEFB  +A3,exponent+73
                  DEFB  +8F,+38,+EE,(+00)
     4.           DEFB  +E9,exponent+79
                  DEFB  +15,+63,+BB,+23
     5.           DEFB  +EE,exponent+7E
                  DEFB  +92,+0D,+CD,+ED
     6.           DEFB  +F1,exponent+81
                  DEFB  +23,+5D,+1B,+EA

В конце последнего цикла 'последним значением' является (SIN (PI*W/2))/W.

Выполнить 5 шаг.

                  DEFB  +04,multiply        SIN (PI*W/2) = SIN X (или =
                                            COS X)
                  DEFB  +38,end-calc
                  RET                       Окончание: 'последнее
                                            значение' = SIN X или
                                             ('последнее значение' =
                                            COS X).
```

ФУНКЦИЯ 'TAN'
(Смещение 21: 'tan')

Эта подпрограмма обрабатывает функцию TAN X. Она просто возвращает SIN X/COS X, с арифметическим переполнением, если COS X = 0.

```
37DA tan          RST   0028,FP-CALC       X
                  DEFB  +31,duplicate      X, X
                  DEFB  +1F,sin            X, SIN X
                  DEFB  +01,exchange       SIN X, X
                  DEFB  +20,cos            SIN X,COS X
                  DEFB  +05,division       SIN X/COS X = TAN X. Если
                                           необходимо, сообщение об
                                           арифметическом переполнении.
                  DEFB  +38,end-calc       TAN X
                  RET                      Окончание: 'последнее
                                           значение' = TAN X.
```

ФУНКЦИЯ 'ARCTAN'
(Смещение 24: 'atn')

Этой подпрограмма обрабатывает функцию ATN X, и она является последней из четырех подпрограмм, которые используют SERIES GENERATOR для создания полиномов Чебышева. Она возвращает реальное число в промежутке -PI/2 и PI/2, которое равно в радианах значению угла, тангенс которого равен X.
ATN X находится следующим образом:

1. Значение W и Y находятся для трех случаев X:
если -1 < X < 1 то W = 0 и Y = X - случай 1.
если -1 < =X то W = PI/2 и Y = -1/X - случай 2.
если X < =-1 то W = -PI/2 и Y = -1/X - случай 3.

В каждом случае, -1 < =Y < =1, что требуется для сходимости ряда.

2. Аргумент Z формируется так:
если -1 < X < 1 то Z = 2*Y*Y-1 = 2*X*X-1 - случай 1.
если 1 < X то Z = 2*Y*Y-1 = 2/(X*X)-1 - случай 2.
если X < =-1 то Z = 2*Y*Y-1 = 2/(X*X)-1 - случай 3.

3. Для выдачи требуемой функции используется SERIES GENERATOR.

4. В конце с помощью простых умножения и сложения выдается ATN X.

Выполняется 1 шаг.

```
37E2 atn          CALL  3297,RE-STACK      Используется X в виде полной
                                           формы с плавающей точкой.
                  LD    A,(HL)             Выбор порядка X.
                  CP    +81
                  JR    C,37F8,SMALL       Для случая 1 переход вперед:
                                           Y = X.
                  RST   0028,FP-CALC       X
                  DEFB  +A1,stk-one        X, 1
                  DEFB  +1B,negate         X,-1
                  DEFB  +01,exchange       -1, X
                  DEFB  +05,division       -1/X
                  DEFB  +31,duplicate      -1/X, -1/X
                  DEFB  +36,less-0         -1/X, (1/0)
                  DEFB  +A3,stk-pi/2       -1/X, (1/0), PI/2
                  DEFB  +01,exchange       -1/X, PI/2, (1/0)
                  DEFB  +00,jump-true      -1/X, PI/2
                  DEFB  +06, to 37FA,CASES Переход вперед для случая 2:
```

```
                                        Y = -1/X  W = PI/2
                DEFB  +1B,negate        -1/X, -PI/2
                DEFB  +33,jump          -1/X, -PI/2
                DEFB  +03,to 37FA,CASES Переход вперед для случая 3:
                                        Y = -1/X  W = -PI/2
37F8 SMALL      RST   0028,FP-CALC      Y
                DEFB  +A0,stk-zero      Y, 0
                                        Продолжение для случая 1: W = 0
```

Выполнить шаг 2.

```
37FA CASES      DEFB  +01,exchange      W, Y
                DEFB  +31,duplicate     W, Y, Y
                DEFB  +31,duplicate     W, Y, Y, Y
                DEFB  +04,multiply      W, Y, Y*Y
                DEFB  +31,duplicate     W, Y, Y*Y, Y*Y
                DEFB  +0F,addition      W, Y, 2*Y*Y
                DEFB  +A1,stk-one       W, Y, 2*Y*Y, 1
                DEFB  +03,subtract      W, Y, 2*Y*Y-1 = Z
```

Выполнить 3 шаг, передав в SERIES GENERATOR параметр '12' десятичное и 12 требуемых параметров.

```
                DEFB  +8C,series-0C     W, Y, Z
     1.         DEFB  +10,exponent+60
                DEFB  +B2,(+00,+00,+00)

     2.         DEFB  +13,exponent+63
                DEFB  +0E,(+00,+00,+00)
     3.         DEFB  +55,exponent+65
                DEFB  +E4,+8D,(+00,+00)
     4.         DEFB  +58,exponent+68
                DEFB  +39,+BC,(+00,+00)
     5.         DEFB  +5B,exponent+6B
                DEFB  +98,+FD,(+00,+00)
     6.         DEFB  +9E,exponent+6E
                DEFB  +00,+36,+75,(+00)
     7.         DEFB  +A0,exponent+70
                DEFB  +DB,+E8,+B4,(+00)
     8.         DEFB  +63,exponent+73
                DEFB  +42,+C4,(+00,+00)
     9.         DEFB  +E6,exponent+76
                DEFB  +B5,+09,+36,+BE
     10.        DEFB  +E9,exponent+79
                DEFB  +36,+73,+1B,+5D
     11.        DEFB  +EC,exponent+7C
                DEFB  +D8,+DE,+63,+BE
     12.        DEFB  +F0,exponent+80
                DEFB  +61,+A1,+B3,+0C
```

В конце последнего цикла 'последним значением' является:

```
     ATN X/X      - случай 1.
     ATN (-1/X)/(-1/X)  - случай 2.
     ATN (-1/X)/(-1/X)  - случай 3.
```

Выполнить 4 шаг.

```
                DEFB  +04,multiply      W, ATN X - случай 1.
                                        W, ATN (-1/X) - случай 2.
                                        W, ATN (-1/X) - случай 3.
```

```
                DEFB  +0F,addition        ATN X - Теперь все случаи.
                DEFB  +38,end-calc
                RET                       Окончание: 'последнее
                                          значение' = ATN X.
```

ФУНКЦИЯ 'ARCSIN'
(Смещение 22: 'asn')

Эта подпрограмма обрабатывает функцию ASN X и возвращает число в диапазоне от –PI/2 до PI/2, которое равно значению угла в радианах, синус которого X. Отсюда, если Y = ASN X, то X = SIN Y.
Эта подпрограмма использует тригонометрическое тождество:
TAN (Y/2) = SIN Y/1(1+COS Y).
Отсюда получим (используя ATN) Y/2 и наконец Y.

```
3833 asn        RST   0028,FP-CALC        X
                DEFB  +31,duplicate       X, X
                DEFB  +31,duplicate       X, X, X
                DEFB  +04,multiply        X, X*X
                DEFB  +A1,stk-one         X, X*X, 1
                DEFB  +03,subtract        X, X*X-1
                DEFB  +1B,negate          X,1-X*X
                DEFB  +28,sqr             X,SQR (1-X*X)
                DEFB  +A1,stk-one         X,SQR (1-X*X), 1
                DEFB  +0F,addition        X, 1+SQR (1-X*X)
                DEFB  +05,division        X/(1+SQR (1-X*X)) = TAN
                                          (Y/2)
                DEFB  +24,atn             Y/2
                DEFB  +31,duplicate       Y/2, Y/2
                DEFB  +0F,addition        Y = ASN X
                DEFB  +38,end-calc
                RET                       Окончание: 'последнее
                                          значение' = ASN X.
```

ФУНКЦИЯ 'ARCCOS'
(Смещение 23: 'acs')

Эта подпрограмма обрабатывает функцию ACS X и возвращает реальное число в диапазоне от 0 до PI, которое равно значению угла в радианах, косинус которого X.
Эта подпрограмма использует соотношение:
ACS X = PI/2 - ASN X

```
3843 acs        RST   0028,FP-CALC        X
                DEFB  +22,asn             ASN X
                DEFB  +A3,stk-pi/2        ASN X,PI/2
                DEFN  +03,subtract        ASN X-PI/2
                DEFB  +1B,negate          PI/2-ASN X = ACS X
                DEFB  +38,end-calc
                RET                       Окончание: 'последнее
                                          значение' = ACS X.
```

ФУНКЦИЯ 'SQUARE ROOT' ('Квадратный корень')
(Смещение 28: 'sqr')

Эта подпрограмма обрабатывает функцию SQR X и возвращает положительный квадратный корень реального числа X, если X положителен, и 0, если X равен 0. Отрицательное значение X выдает сообщение A - неправильный аргумент (через ln в подпрограмме EXPONENTIATION).

Эта подпрограмма обрабатывает операцию извлечения квадратного корня, как X**.5 и поэтому заносит в стек значение .5 и возобновляется прямо в программе EXPONENTIATION.

```
384A sqr          RST   0028,FP-CALC        X
                  DEFB  +31,duplicate       X,X
                  DEFB  +30,not             X,(1/0)
                  DEFB  +00,jump-true       X
                  DEFB  +1E,to 386C,LAST    X
```

Если X = 0, делается переход, иначе, продолжение с:

```
                  DEFB  +A2,stk-half        X,.5
                  DEFB  +38,end-calc
```

и затем находится результат X**.5.

ОПЕРАЦИЯ 'EXPONENTIATION' ('Возведение в степень')
(Смещение 06: 'to-power')

Эта подпрограмма выполняет двоичную операцию возведения первого числа X в степень, определяемую вторым числом Y.

Подпрограмма обрабатывает результат X**Y в виде эквивалента EXP (Y*LN X). Она возвращает это значение, если X не ноль, в случае нуля и, если Y тоже ноль, возвращает 1 (0**0=1), возвращает ноль, если Y положителен; сообщает об арифметическом переполнении, если Y отрицателен.

```
3851 to-power     RST  0028,FP-CALC         X,Y
                  DEFB  +01,exchange        Y,X
                  DEFB  +31,duplicate       Y,X,X
                  DEFB  +30,not             Y,X,(1/0)
                  DEFB  +00,jump-true       Y,X
                  DEFB  +07,to 385D,XIS0    Y,X
```

Если X = 0, выполняется переход, иначе, формируется EXP (Y*LN X).

```
                  DEFB  +25,ln              Y,LN X
```

Если X отрицателен, выдается сообщение A.

```
                  DEFB  +04,multiply        Y*LN X
                  DEFB  +38,end-calc
                  JP    36C4,EXP            Выход через EXP для
                                            формирования EXP(Y*LN X).
```

Рассмотреть три возможных случая для X = 0.

```
385D XIS0         DEFB  +02,delete          Y
                  DEFB  +31,duplicate       Y,Y
                  DEFB  +30,not             Y,(1/0)
                  DEFB  +00,jump-true       Y
                  DEFB  +09,to 386A,ONE     Y
```

Если X = 0 и Y = 0, выполняется переход, иначе, продолжение.

```
                  DEFB  +A0,stk-zero        Y,0
                  DEFB  +01,exchange        0,Y
                  DEFB  +37,greater-0       0,(1/0)
                  DEFB  +00,jump-true       0
                  DEFB  +06,to 386C,LAST    0
```

```
Если X = 0, а Y положителен, выполняется переход, иначе, продолжение.
                  DEFB  +A1,stk-one         0,1
                  DEFB  +01,exchange        1,0
                  DEFB  +05,division        Выход через 'делитель',
                                            т.к. деление на ноль выдает
                                            'арифметическое переполнение'.

Для операции результат должен быть 1.

386A ONE          DEFB +02,delete           -
                  DEFB +A1,stk-one          1

Теперь возврат 'последнего значения' на стек 0**Y.

386C LAST         DEFB +38,end-calc         (1/0)
                  RET                       Окончание: 'последнее
                                            значение' 0 или 1.

386E - 3CFF Эти ячейки являются 'резервными'. Они все содержат +FF.

3D00 - 3FFF Эти ячейки содержат 'набор символов'. В них 8-байтные представления всех
символов с кодами от +20 (пробел) до +7F ((c)).
Например, буква 'A' имеет представление 00 3C 42 42 7E 42 42 00 и форму:

00000000
00111100
01000010
01000010
01111110
01000010
01000010
00000000
```

ПРИЛОЖЕНИЕ

BASIC ПРОГРАММЫ ДЛЯ ОСНОВНЫХ РЯДОВ
Включенные в описание BASIC программы, хорошо иллюстрируют, как полиномы Чебышева используются для аппроксимации функций SIN, EXP, LN и ATN.

Генератор рядов
Эта подпрограмма вызывается всеми программами 'функций'.

```
500          REM SERIES GENERATOR, ENTER
510          REM USING THE COUNTER BREG
520          REM AND ARRAY-A HOLDING THE
530          REM CONSTANTS.
540          REM FIRST VALUE IN Z.
550          LET M0=2*Z
560          LET M2=0
570          LET T=0
580          FOR I=BREG TO 1 STEP -1
590          LET M1=M2
600          LET U=T*M0-M2+A(BREG+1-I)
610          LET M2=T
620          LET T=U
630          NEXT I
640          LET T=T-M1
650          RETURN
660          REM LAST VALUE IN T.
```

В подпрограмме, представленной выше, переменные являются:

```
Z  -         значение входа.
T  -         значение выхода.
M0 -         mem-0
M1 -         mem-1
M2 -         mem-2
I  -         счетчик для BREG.
U  -         временная переменная для T.
A(1) в
A(BREG) -    константы.
BREG -       число используемых констант.
```

Чтобы увидеть, как создаются полиномы Чебышева, запишите на бумагу U, M1, M2 и T через строки 550-630, проходя, скажем, 6 раз через цикл, держа алгебраические выражения для A(1) - A(6) без подстановки численных значений.

Затем запишите T-M1. Множители констант A(1) - A(6) будут затем затребованы полиномами Чебышева. Точнее, множитель A(1) будет 2*T5(Z), для A(2) будет 2*T4(Z) и т.д. до 2*T1(Z) для A(5) и, наконец, T0(Z) для A(6).

Отметим, что T0(Z)=1, T1(Z)=Z и, для n>=2, Tn(Z)=2*Z*Tn-1(Z)-Tn-2(Z).

SIN X

```
10           REM DEMONSTRATION FOR SIN X
20           REM USING THE 'SERIES GENERATOR'.
30           DIM A(6)
40           LET A(1)=-.000000003
```

```
 50          LET A(2)=0.000000592
 60          LET A(3)=-.000068294
 70          LET A(4)=0.004559008
 80          LET A(5)=-.142630785
 90          LET A(6)=1.276278962
100          PRINT
110          PRINT "ENTER START VALUE IN DEGREES"
120          INPUT C
130          CLS
140          LET C=C-10
150          PRINT "BASIC PROGRAM","ROM PROGRAM"
160          PRINT "-------------","-----------"
170          PRINT
180          FOR J=1 TO 4
190          LET C=C+10
200          LET Y=C/360-INT (C/360+.5)
210          LET W=4*Y
220          IF W > 1 THEN LET W=2-W
230          IF W < -1 THEN LET W=-W-2
240          LET Z=2*W*W-1
250          LET BREG=6
260          REM USE 'SERIES GENERATOR'
270          GO SUB 550
280          PRINT TAB 6; "SIN ";C;" DEGREES"
290          PRINT
300          PRINT T*W,SIN (PI*C/180)
310          PRINT
320          NEXT J
330          GO TO 100
```

ПРИМЕЧАНИЯ:

1. При вводе C эта подпрограмма вычисляет и печатает SIN C, SIN (C+10), SIN (C+20) и SIN (C+30), где C измеряется в градусах.
Также, она печатает значения, полученные с помощью программы ПЗУ.
Для образца попытайтесь ввести эти значения в градусах:
0; 5; 100; -80; -260; 3600; -7200.
2. Константы A(1)-A(6) в строках 40-90 задаются в Абрамович и Стегун, "Справочник по математическим функциям" (Дувр, 1965), страница 76.
Их можно проверить с помощью интегрирования (SIN (PI*X/2))/X по интервалу U от 0 до PI, после первого умножения на COS (N*U) для каждой константы (т.е. N=1,2,...,6) и подстыковки COS U=2*X*X-1. Каждый результат необходимо затем разделить на PI. (Это интегрирование можно выполнить методами приближений, например, используя правило Симпсона, если имеется приемлемый компьютер или программируемый калькулятор).

EXP X

```
 10          REM DEMONSTRATION FOR EXP X
 20          REM USING THE 'SERIES GENERATOR'
 30          LET T=0                    (Создается T - первая переменная)
 40          DIM A(8)
 50          LET A(1)=0.000000001
 60          LET A(2)=0.000000053
 70          LET A(3)=0.000001851
 80          LET A(4)=0.000053453
 90          LET A(5)=0.001235714
100          LET A(6)=0.021446556
110          LET A(7)=0.248762434
```

```
120         LET A(8)=1.456999875
130         PRINT
140         PRINT "ENTER START VALUE"
150         INPUT C
160         CLS
170         LET C=C-10
180         PRINT "BASIC PROGRAM", "ROM PROGRAM"
190         PRINT "-------------", "------------"
200         PRINT
210         FOR J=1 TO 4
220         LET C=C+10
230         LET D=C*1.442695041       (D=C*(1/LN 2);EXP C=2**D).
240         LET N=INT D
250         LET Z=D-N                 (Теперь требуется 2**(N+Z)).
260         LET Z=2*Z-1
270         LET BREG=8
280         REM USE "SERIES GENERATOR"
290         GO SUB 550
300         LET V=PEEK 23627+256*PEEK 23628+1  (V=(VARS)+1)
310         LET N=N+PEEK V
320         IF N > 255 THEN STOP      (STOP с арифметическим переполнением).
330         IF N < 0 THEN GO TO 360
340         POKE V,N
350         GO TO 370
360         LET T=0
370         PRINT TAB 11;"EXP ";C
380         PRINT
390         PRINT T,EXP C
400         PRINT
410         NEXT J
420         GO TO 130
```

ПРИМЕЧАНИЯ:

1. При вводе C эта программа вычисляет и печатает EXP C, EXP (C+10), EXP (C+20) и EXP (C+30). Также печатаются значения, полученные при использовании программы ПЗУ. Для образца попытайтесь ввести эти значения: 0; 15; 65 (с переполнением в конце); -100; -40.
2. В строках 320 и 330 показатель проверяется на переполнение и ноль. Эти проверки в BASIC проще, чем в машинных кодах, так как переменная N, в отличие от регистра A, не ограничена одним байтом.
3. Константы A(1) - A(8) в строках 50-120 можно получить с помощью интегрирования 2**X в интервале U = 0 - PI, после первого умножения COS (N*U) для каждой константы (т.е. для N=1,2,...,8) и подстановки COS U = 2*X-1. Каждый результат затем должен делиться на PI.

LN X

```
 10         REM DEMONSTRATION FOR LN X
 20         REM USING THE 'SERIES GENERATOR'
 30         LET D=0                   (Создается D - первая переменная).
 40         DIM A(12)
 50         LET A(1)= -.0000000003
 60         LET A(2)=0.0000000020
 70         LET A(3)= -.0000000127
 80         LET A(4)=-0.0000000823
 90         LET A(5)= -.0000005389
100         LET A(6)=0.0000035828
110         LET A(7)= -.0000243013
120         LET A(8)=0.0001693953
```

```
130         LET A(9)= -.0012282837
140         LET A(10)=0.0094766116
150         LET A(11)= -.0818414567
160         LET A(12)=0.9302292213
170         PRINT
180         PRINT "ENTER START VALUE"
190         INPUT C
200         CLS
210         PRINT "BASIC PROGRAM", "ROM PROGRAM"
220         PRINT "-------------", "-----------"
230         PRINT
240         LET C=SQR C
250         FOR J=1 TO 4
260         LET C=C*C
270         IF C=0 THEN STOP          (STOP с 'неправильным аргументом'.)
280         LET D=C
290         LET V=PEEK 23627+256*PEEK 23628+1
300         LET N=PEEK V-128          (N содержит e').
310         POKE V,128
320         IF D<=0.8 THEN GO TO 360  (D содержит X').
330         LET S=D-1
340         LET Z=2.5*D-3
350         GO TO 390
360         LET N=N-1
370         LET S=2*D-1
380         LET Z=5*D-3
390         LET R=N*0.6931471806      (R содержит N*LN 2).
400         LET BREG=12
410         REM USE 'SERIES GENERATOR'
420         GO SUB 550
430         PRINT TAB 8;"LN ";C
440         PRINT
450         PRINT S*T+R,LN C
460         PRINT
470         NEXT J
480         GO TO 170
```

ПРИМЕЧАНИЯ:

1. При вводе C, программа вычисляет и печатает LN C, LN (C**2), LN (C**4) и LN (C**8). Также печатает значения, полученные с помощью программы ПЗУ. Для образца попытайтесь ввести значения: 1.1; 0.9; 300; 0.004; 1E5 (для переполнения) и 1E-5 (STOP как 'неправильный аргумент').
2. Константы A(1) - A(12) в строках 50-160 можно получить интегрируя 5*LN (4*(X+1)/5)/(4*X-1) на интервале U=0 - PI после первого умножения на COS (N*U) для каждой константы (т.е. для N=1,2,...,12) и подстановки COS U=2*X-1. Каждый результат должен быть разделен на PI.

ATN X

```
10          REM DEMONSTRATION FOR ATN X
20          REM USING THE 'SERIES GENERATOR'
30          DIM A(12)
40          LET A(1)= -.0000000002
50          LET A(2)=0.0000000010
60          LET A(3)= -.0000000066
70          LET A(4)=0.0000000432
80          LET A(5)= -.0000002850
```

```
 90          LET A(6)=0.0000019105
100          LET A(7)= -.0000131076
110          LET A(8)=0.0000928715
120          LET A(9)= -.0006905975
130          LET A(10)=0.0055679210
140          LET A(11)= -.0529464623
150          LET A(12)=0.8813735870
160          PRINT
170          PRINT "ENTER START VALUE"
180          INPUT C
190          CLS
200          PRINT "BASIC PROGRAM", "ROM PROGRAM"
210          PRINT "-------------", "-----------"
220          PRINT
230          FOR J=1 TO 4
240          LET B=J*C
250          LET D=B
260          IF ABS B>=1 THEN LET D= -1/B
270          LET Z=2*D*D-1
280          LET BREG=12
290          REM USE "SERIES GENERATOR"
300          GO SUB 550
310          LET T=D*T
320          IF B > =1 THEN LET T=T+PI/2
330          IF B < =-1 THEN LET T=T-PI/2
340          PRINT TAB 8;"ATN ";B
350          PRINT
360          PRINT T,ATN B         (или PRINT T*180/PI,ATN B*180/PI
370          PRINT                  для получения ответа в градусах)
380          NEXT J
390          GO TO 160
```

ПРИМЕЧАНИЯ:

1. При вводе С программа вычисляет и печатает ATN C, ATN (C*2), ATN (C*3) и ATN (C*4). Для образца попытайтесь ввести значения: 0.2; -1; 10 и -100. Результаты будут интереснее, если их перевести в градусы, умножив в строке 360 на 180/PI.
2. Константы A(1) - A(12) в сроках 40-150 заданы в Абрамович и Стегун, страница 82. Их можно проверить, интегрируя ATN X/X на интервале U=0 - PI, после первого умножения на COS (N*U) для каждого параметра (т.е. для n=1,2,...,12) и подстановки COS U=2*X*X-1. Каждый результат необходимо разделить на PI.

Альтернативная подпрограмма для SIN X

Создается полное разложение полиномов Чебышева, записываемых следующим образом:

```
550          LET T =(32*Z*Z*Z*Z*Z-40*Z*Z*Z+10*Z)*A(1)
                   +(16*Z*Z*Z*Z-16*Z*Z+2)*A(2)
                   +(8*Z*Z*Z-6*Z)*A(3)
                   +(4*Z*Z-2)*A(4)
                   +2*Z *A(5)
                   +A(6)
560          RETURN
```

Эта подпрограмма вызывается вместо SERIES GENERATOR и обладает такой же точностью.

Альтернативная подпрограмма для EXP X
Полное разложение для EXP X:

```
550          LET T =(128*Z*Z*Z*Z*Z*Z*Z*Z-224*Z*Z*Z*Z*Z*Z+112*Z*Z*Z*Z-14*Z)*A(1)
                   +(64*Z*Z*Z*Z*Z*Z*Z-96*Z*Z*Z*Z*Z+36*Z*Z-2)*A(2)
                   +(32*Z*Z*Z*Z*Z*Z-40*Z*Z*Z*Z+10*Z)*A(3)
                   +(16*Z*Z*Z*Z*Z-16*Z*Z*Z+2)*A(4)
                   +(8*Z*Z*Z*Z-6*Z)*A(5)
                   +(4*Z*Z*Z-2)*A(6)
                   +2*Z*Z*A(7)
                   +A(8)
560          RETURN
```

Разложение для LN X и ATN X, заданное алгебраически, будет:

```
                     (2048z11-5632z9+5632z7-2464z5+440z3-22z) * A (1)
+                    (1024z10-2560z8+2240z6-800z4+100z2-2) * A(2)
+                    (512z9-1152z7+864z5-240z3+18z) * A(3)
+                    (256z8-512z6+320z4-64z2+2) * A(4)
+                    (128z7-224z5+112z3-14z) * A(5)
+                    (64z6-96z4+36z2-2) * A(6)
+                    (32z5-40z3+10z) * A(7)
+                    (16z4-16z2+2) * A(8)
+                    (8z3-6z) * A(9)
+                    (4z2-2) * A(10)
+                    (2z) * A(11)
+                    A(12)
```

Алгоритм 'DRAW' ('Рисовать')
Данная программа BASIC иллюстрирует основную часть операции DRAW при использовании ее для изображения линий. Данная форма программы позволяет изображать линии при условии, что X > Y.

```
 10          REM DRAW 255,175 PROGRAM
 20          REM SET ORIGIN
 30          LET PLOTx=0: LET PLOTy=0
 40          REM SET LIMITS
 50          LET X=255: LET Y=175
 60          REM SET INCREMENT,i
 70          LET i=X/2
 80          REM ENTER LOOP
 90          FOR B=X TO 1 STEP -1
100          LET A=Y+i
110          IF X> A THEN GO TO 160
120          REM UP A PIXEL ON THIS PASS
130          LET A=A-X
140          LET PLOTy=PLOTy+1
150          REM RESET INCREMENT,i
160          LET i=A
170          REM ALWAYS ALONG ONE PIXEL
180          LET PLOTx=PLOTx+1
190          REM NOW MAKE A PLOT
200          PLOT PLOTx,PLOTy
210          NEXT B
```

Полный алгоритм находится в следующей программе как подпрограмма, которая будет чертить линию ('DRAW A LINE') от последней позиции в позицию X,Y.

Алгоритм 'CIRCLE' ('Круг')
Данная подпрограмма BASIC иллюстрирует, как команда CIRCLE создает круг.
  Сначала подсчитывается количество требуемых дуг. Затем готовится набор параметров в 'области памяти' и на 'стеке калькулятора'.
  Дуги чертятся с помощью повторяющихся обращений к подпрограмме, строящей линию, которая на каждом вызове чертит одну линию от 'последней позиции' до позиции 'X,Y'.

Примечание: В программе ПЗУ имеется конечная 'замыкающая' линия, но здесь она отсутствует.

```
 10          REM A CIRCLE PROGRAM
 20          LET X=127: LET Y=87: LET Z=87
 30          REM How many arcs?
 40          LET Arcs=4*INT (INT (ABS (PI*SQR Z)+0.5)/4)+4
 50          REM Задать область памяти; M0-M5
 60          LET M0=X+Z
 70          LET M1=0
 80          LET M2=2*Z*SIN (PI/Arcs)
 90          LET M3=1-2*(SIN (PI/Arcs)) ^ 2
100          LET M4=SIN (2*PI/Arcs)
110          LET M5=2*PI
120          REM Задать стек; Sa-Sd
130          LET Sa=X+Z
140          LET Sb=Y-Z*SIN (PI/Arcs)
15)          LET Sc=Sa
160          LET Sd-Sb
170          REM Инициализация COORDS
180          POKE 23677,Sa: POKE 23678,Sb
190          LET M0=Sd
200          REM 'DRAW THE ARCS'
210          LET M0=M0+M2
220          LET Sc=Sc+M1
230          LET X=Sc-PEEK 23677
240          LET Y=M0-PEEK 23678
250          GO SUB 510
260          LET Arcs=Arcs-1: IF Arcs=0 THEN STOP
270          LET MM1=M1
280          LET M1=M1*M3-M2*M4
290          LET M2=MM1*M4+M2*M3
300          GO TO 210

500          REM 'DRAW A LINE' от последней позиции до X,Y
510          LET PLOTx=PEEK 23677: LET PLOTy=PEEK 23678
520          LET dx=SGN X: LET dy=SGN Y
530          LET X=ABS X: LET Y=ABS Y
540          IF X> =Y THEN GO TO 580
550          LET L=X: LET B=Y
560          LET ddx=0: LET ddy=dy
570          GO TO 610
580          IF X+Y=0 THEN STOP
590          LET L=Y: LET B=X
600          LET ddx=dx: LET ddy=0
610          LET H=B
620          LET i=INT (B/2)
630          FOR N=B TO 1 STEP -1
640          LET i=i+L
650          IF i < H THEN GO TO 690
```

```
660          LET i=i-H
670          LET ix=dx: LET iy=dy
680          GO TO 700
690          LET ix=ddx: LET iy=ddy
700          LET PLOTy=PLOTy+iy
710          IF PLOTy <0 OR PLOTy > 175 THEN STOP
720          LET PLOTx=PLOTx+ix
730          IF PLOTx <0 OR PLOTx > 255 THEN STOP
740          PLOT PLOTx,PLOTy
750          NEXT N
760          RETURN
```

Примечания к малым целым и -65536

1. Малые целые n это числа, лежащие в диапазоне -65535 - 65535. В 'STACK-BC' описана форма их представления. Отметим, что руководство является неточным, когда говорится, что третий или четвертый байты содержат n плюс 131072, если n отрицательно. Т.к. диапазон n это -1 - -65535, два байта могут содержать только n плюс 131072, если он будет как mod 65536; т.е. они содержат n плюс 65536. Руководство подгоняет выход. Действительно, это не является истинной формой дополнения до двух (как может быть в других обстоятельствах форма плюс 131072). Здесь одно и то же число может представлять два различных числа в соответствии со знаковым байтом: например, 00 01 для 1, если знаковый байт 00 и для -65535, если знаковый байт FF; аналогично FF FF для 65535, если знаковый байт 00 и для -1, если знаковый байт FF.

2. Допустим, что отрицательные числа заданы в специальной форме 'дополнения до двух', основное свойство этого метода хранения чисел это то, что они готовы для 'короткого сложения' без любого дальнейшего дополнения до двух. Они просто выбираются и запоминаются непосредствен подпрограммой сложения. Но для умножения их нужно выбрать с помощью INT FETCH и запомнить позже с помощью INT-STORE. Эти подпрограммы дополняют до двух число, когда осуществлены выбор и запоминание. Обращение к INT-STORE осуществляется из 'умножения' (после 'короткого умножения'), из 'отбрасывания' (после формирования 'малого целого', между -65535 и 65535), из 'инвертиравания/abs' для 'случая целых чисел' и из 'sgn', чтобы запомнить 1 или -1. Обращение к INT-FETCH осуществляется из PRINT-FP для выбора целой части числа, когда оно 'мало', дважды из 'умножения' для выбора двух 'малых чисел', из 'RE-STACK' для выбора 'малого- целого' для перезанесения в стек, из 'инвертирования'/'abs' для выбора 'малого целого' для обработки и из FP-TO-BC для выбора целого с целью передачи его в BC.

Число -65536.

3. Число -65536 можно преобразовать в формат 'малого целого' как 00 FF 00 00 00. Тогда это будет 'ограниченное число', которое при дополнении до двух переполняется (80 шестнадцатеричное в простой системе из одного байта или 7 разрядов, т.е. -128 десятичное, которое при дополнении до двух еще выдает 80 шестнадцатеричное, т.е. -128 десятичное, т.к. положительное десятичное число 128 не подходит системе).

4. Некоторая осведомленность в этом может вдохновить на неудачную попытку создать 00 FF 00 00 00 в 'отбрасывании'. Это будет неудачно, т.к. даже не сохраняется работоспособность подпрограммы INT, частью которой является 'отбрасывание'. Это только приводит к ошибке INT(-65535) равно -1.

5. Но основная ошибка это то, что это число увеличивается от 'короткого сложения' до двух малых отрицательных целых, а затем просто заносится в стек как 00 FF 00 00 00. Система не может справиться с этим числом. Решение, предложенное в 'addition', сразу же формирует

полное пятибайтное число с плавающей точкой, т.е. тест для числа на 3032 следующий:

```
3032              PUSH  AF                Запись знакового байта в A.
3033              INC   A                 Создание любого FF в A и 00.
3034              OR    E                 Проверить, что все
                                          3 байта теперь для 0.
3035              OR    D
3036              JR    NZ,3040,ADD-STORE Переход, если не -65536.
3038              POP   AF                Очистить стек.
3039              LD    (HL),+80          Ввести 80 (16-ричное во 2
                                          байт).
303B              DEC   HL                Указать первый байт.
303C              LD    (HL),+91          Ввести 91 (16-ричное в 1
                                          байт).
303E              JR    3049,ADD-RSTOR    Переход, чтобы задать
                                          указатель и выход.
3040 ADD-STORE    POP   AF                Восстановить знаковый байт
                                          в A.
3041              LD    (HL),A            Занести его в стек.
3042              INC   HL                Указать следующую ячейку.
3043              LD    (HL),E            Запомнить младший байт
                                          результата.
3044              INC   HL                Указать следующую ячейку.
3045              LD    (HL),D            Запомнить старший байт
                                          результата.
3046              DEC   HL                Переместить указатель
3047              DEC   HL                назад к адресу первого
3048              DEC   HL                байта результата.
3049 ADD-RSTOR    POP   DE                Восстановить STKEND в DE.
304A              RET                     Окончание.
```

6. Вышеуказанная поправка (т.е. 15 дополнительных байт) с вычеркиванием байт от 3223 до 323E включительно из 'отбрасывания' решит проблемы. Она хороша тем, что ее можно проверить. Обращение к INT-STORE не приведет к занесению в стек 00 FF 00 00 00. При умножении, если число появится, будет переполнение, тогда как 65536 будет задавать признак переноса; будет использоваться 'длинное умножение'. Как отмечено в 30E5, 5 байтами, начинающихся там, можно пренебречь, если сделана вышеуказанная поправка. 'Инвертирование' исключает занесение в стек 00 FF 00 00 00 с помощью отдельной обработки нуля и возвращения его неизмененным. Отбрасывание работает с -65536 отдельно, как показано выше. SGN запоминает только 1 и -1.

С О Д Е Р Ж А Н И Е

Примечание: Указанные в оглавлении точки входа могут быть не единственными для данной подпрограммы. Для корректного использования подпрограмм внимательно разберитесь с ее структурой и способами использования.