CENTURY COMMUNICATIONS

# ZX SPECTRUM

# MICRO GUIDE

A Quick Reference

The authors gratefully acknowledge the permission of Sinclair Research Limited to include the copyright material from the ZX Spectrum Handbook pages 173-176 and 189-192

First published in Great Britain in 1984
This edition published in 1985 by Book Club Associates
By arrangement with Century Communications Ltd

Grateful acknowledgment is hereby given to Sinclair Research Limited for permission to use the following Trade Marks: SINCLAIR, ZX, ZX80, ZX81, ZX SPECTRUM, ZX MICRODRIVE, ZX INTERFACE, ZX NET, ZX PRINTER, ZX POWER SUPPLIES.

Printed in Great Britain

# MICROGUIDE
## FOR THE
# ZX SPECTRUM

**Professor Peter Morse**
**Brian Hancock**

**GUILD PUBLISHING**
London

## CONTENTS



*The authors are grateful to Sinclair Research for permission to include this material.

## ALPHABETICAL QUICK REFERENCE GUIDE

This table, which is in alphabetical order, will enable the user to quickly reference any character for:

the CODE of the character
the relevant SECTION in Part C: BASIC Summary

| Character | Code | Section |
|---|---|---|
| ABS | 189 | 8 |
| ACS | 182 | 7 |
| AND | 198 | 10 |
| ASN | 181 | 7 |
| AT | 172 | 6 |
| ATN | 183 | 7 |
| ATTR | 171 | 4 |
| BEEP | 215 | 5 |
| BIN | 196 | 8 |
| BORDER | 231 | 4 |
| BREAK | | 1 |
| BRIGHT | 220 | 4 |
| CAPS LOCK | | A |
| CAPS SHIFT | | A |
| CAT | 207 | |
| CHR$ | 194 | 9 |
| CIRCLE | 216 | 3 |
| CLEAR | 253 | 1 |
| CLOSE# | 212 | 6 |
| CLS | 251 | 6 |
| CODE | 175 | 9 |
| CONT | 232 | 1 |
| COPY | 255 | 6 |
| COS | 179 | 7 |
| DATA | 228 | 2 |
| DEF FN | 206 | 2 |
| DELETE | 12 | 1 |
| DIM | 233 | 2 |
| DRAW | 252 | 3 |
| EDIT | 7 | 1 |
| ENTER | 13 | 1 |
| ERASE | 210 | |
| EXP | 185 | 8 |
| FLASH | 219 | 4 |
| FN | 168 | 2 |
| FOR | 235 | 2 |
| FORMAT | 208 | |
| GOSUB | 237 | 2 |
| GOTO | 236 | 2 |
| GRAPHICS | | 1 |
| IF | 250 | 2 |
| IN | 191 | 6 |
| INK | 217 | 4 |
| INKEY $ | 166 | 6 |
| INPUT | 238 | 6 |
| INT | 186 | 8 |
| INVERSE | 221 | 4 |
| LEN | 177 | 9 |
| LET | 241 | 2 |
| LINE | 202 | 6 |
| LIST | 240 | 6 |
| LLIST | 225 | 6 |
| LN | 184 | 8 |
| LOAD | 239 | 1 |
| LPRINT | 224 | 6 |
| MERGE | 213 | 1 |
| MOVE | 209 | |
| NEW | 230 | 1 |
| NEXT | 243 | 2 |
| NOT | 195 | 10 |
| OPEN # | 211 | 6 |
| OR | 197 | 10 |
| OUT | 223 | 6 |

| Character | Code | Section |
|---|---|---|
| OVER | 222 | 4 |
| PAPER | 218 | 4 |
| PAUSE | 242 | 2 |
| PEEK | 190 | 2 |
| PI | 167 | 8 |
| PLOT | 246 | 3 |
| POINT | 169 | 3 |
| POKE | 244 | 2 |
| PRINT | 245 | 6 |
| RANDomise | 249 | 8 |
| READ | 227 | 2 |
| REM | 234 | 2 |
| RESTORE | 229 | 2 |
| RETURN | 254 | 2 |
| RND | 165 | 8 |
| RUN | 247 | 1 |
| SAVE | 248 | 1 |
| SCREEN$ | 170 | 1 |
| SGN | 188 | 8 |
| SIN | 178 | 7 |
| SPACE | 32 | |
| SQR | 187 | 8 |
| STEP | 205 | 2 |
| STOP | 226 | 1 |
| STR$ | 193 | 9 |
| SYMBOL SHIFT | | A |
| TAB | 173 | 6 |
| TAN | 180 | 7 |
| THEN | 203 | 2 |
| TO | 204 | 2 |
| USR | 192 | 2 |
| VAL | 176 | 9 |
| VAL$ | 174 | 9 |
| VERIFY | 214 | 1 |
| ! | 33 | |
| " | 34 | 6 |
| # | 35 | 6 |
| $ | 36 | 9 |
| % | 37 | |
| & | 38 | |
| ' | 39 | 6 |
| ( | 40 | 10 |
| ) | 41 | 10 |
| * | 42 | 10 |
| + | 43 | 10 |
| , | 44 | 6 |
| - | 45 | 10 |
| . | 46 | |
| / | 47 | 10 |
| : | 58 | 6 |
| ; | 59 | 6 |
| < | 60 | 10 |
| = | 61 | 10 |
| > | 62 | 10 |
| ? | 63 | |
| @ | 64 | |
| [ | 91 | |
| ] | 93 | |
| \ | 92 | |
| ↑ | 94 | 10 |
| _ | 95 | |
| £ | 96 | |
| { | 123 | |
| \| | 125 | |
| } | 124 | |
| ~ | 126 | |
| © | 127 | |
| <= | 199 | 10 |
| >= | 200 | 10 |
| <> | 201 | 10 |
| ← | 8 | 1 |
| → | 9 | 1 |
| ↓ | 10 | 1 |
| ↑ | 11 | 1 |



# PART A
# SPECIAL KEYBOARD FEATURES

## KEYBOARD MODES

When inputting (keying in) program lines the position for the next entry is indicated by a cursor on the screen. The *mode* is indicated by the flashing cursors [K] [L] [G] [C] [E]

The [K] mode *(Keywords)* and [L] *(Letters)* may be used unshifted, with CAPS SHIFT or with SYMBOLS SHIFT. Letters of the alphabet are lower case unless the CAPS SHIFT key is used or the [C] mode used. The [C] mode *(Capitals)* is obtained by pressing CAPS SHIFT and CAPS LOCK simultaneously and is identical to the L mode apart from producing capitals (upper case) instead of lower-case letters. To return to [L] mode press CAPS SHIFT and CAPS LOCK simultaneously.

The [G] mode *(Graphics)* accesses the graphics characters and may be obtained using the GRAPHICS key as an on-off switch. Press CAPS SHIFT and GRAPHICS simultaneously to enter [G] mode. Repeat to cancel.

The [E] mode *(Extended)* is obtained by pressing CAPS SHIFT and SYMBOLS SHIFT simultaneously and lasts for *one character only*. It may be used unshifted, with CAPS SHIFT or with SYMBOLS SHIFT.

TOP ROW KEYS

| Example | | Mode | Press | Effect |
|---|---|---|---|---|
| YELLOW | → | [E] | [CAPS SHIFT] + Key | Sets ink colour |
| | | [E] | Key | Sets paper colour |
| ◇ | → | [K] [L] [C] | [CAPS SHIFT] + Key | Performs function |
| | | [G] | Key | Graphics character |
| 6 ▚ & | → | [G] | [CAPS SHIFT] + Key | Reverse graphics character |
| | → | [K] [L] [C] | Key | Number |
| | → | [K] [L] [C] | [SYMBOL SHIFT] + Key | Symbol |
| MOVE | → | [E] | [SYMBOL SHIFT] + Key | Keyword |

KEYS ON LOWER 3 ROWS

| Example | | Mode | Press | Effect |
|---|---|---|---|---|
| BIN | → | [E] | Key | Keyword |
| B * BORDER | → | [K] [L] [C] | [SYMBOL SHIFT] + Key | Symbol |
| | | [L] | Key | Lowercase letter |
| | → | [C] | [CAPS SHIFT] + Key | Uppercase letter |
| | | [G] | Key | User defined graphics |
| | → | [K] | Key | Keyword |
| BRIGHT | → | [E] | [SYMBOL SHIFT] + Key | Keyword |

# PART B
## CHARACTER SET AND CODES

| Code | Character | Code | Char. | Code | Char. |
|---|---|---|---|---|---|
| 0 | not used | 51 | 3 | 102 | f |
| 1 | | 52 | 4 | 103 | g |
| 2 | | 53 | 5 | 104 | h |
| 3 | | 54 | 6 | 105 | i |
| 4 | | 55 | 7 | 106 | j |
| 5 | | 56 | 8 | 107 | k |
| 6 | **PRINT** comma | 57 | 9 | 108 | l |
| 7 | **EDIT** | 58 | : | 109 | m |
| 8 | cursor left | 59 | ; | 110 | n |
| 9 | cursor right | 60 | < | 111 | o |
| 10 | cursor down | 61 | = | 112 | p |
| 11 | cursor up | 62 | > | 113 | q |
| 12 | **DELETE** | 63 | ? | 114 | r |
| 13 | **ENTER** | 64 | @ | 115 | s |
| 14 | number | 65 | A | 116 | t |
| 15 | not used | 66 | B | 117 | u |
| 16 | **INK** control | 67 | C | 118 | v |
| 17 | **PAPER** control | 68 | D | 119 | w |
| 18 | **FLASH** control | 69 | E | 120 | x |
| 19 | **BRIGHT** control | 70 | F | 121 | y |
| 20 | **INVERSE** control | 71 | G | 122 | z |
| 21 | **OVER** control | 72 | H | 123 | { |
| 22 | **AT** control | 73 | I | 124 | \| |
| 23 | **TAB** control | 74 | J | 125 | } |
| 24 | not used | 75 | K | 126 | ~ |
| 25 | | 76 | L | 127 | © |
| 26 | | 77 | M | | |
| 27 | | 78 | N | 128 |  |
| 28 | | 79 | O | 129 |  |
| 29 | | 80 | P | 130 |  |
| 30 | | 81 | Q | 131 |  |
| 31 | | 82 | R | 132 |  |
| 32 | space | 83 | S | 133 |  |
| 33 | ! | 84 | T | 134 |  |
| 34 | " | 85 | U | 135 |  |
| 35 | # | 86 | V | 136 |  |
| 36 | $ | 87 | W | 137 |  |
| 37 | % | 88 | X | 138 |  |
| 38 | & | 89 | Y | 139 |  |
| 39 | ' | 90 | Z | 140 |  |
| 40 | ( | 91 | [ | 141 |  |
| 41 | ) | 92 | / | 142 |  |
| 42 | * | 93 | ] | 143 |  |
| 43 | + | 94 | ↑ | | |
| 44 | , | 95 | _ | | |
| 45 | – (minus sign) | 96 | £ | | |
| 46 | . | 97 | a | | |
| 47 | / | 98 | b | | |
| 48 | 0 | 99 | c | | |
| 49 | 1 | 100 | d | | |
| 50 | 2 | 101 | e | | |



| | | | |
|---|---|---|---|
| 144 | (a) | 200 | >= |
| 145 | (b) | 201 | <> |
| 146 | (c) | 202 | LINE |
| 147 | (d) | 203 | THEN |
| 148 | (e) | 204 | TO |
| 149 | (f) | 205 | STEP |
| 150 | (g) | 206 | DEF FN |
| 151 | (h) | 207 | CAT |
| 152 | (i) | 208 | FORMAT |
| 153 | (j) | 209 | MOVE |
| 154 | (k) | 210 | ERASE |
| 155 | (l) | 211 | OPEN # |
| 156 | (m) | 212 | CLOSE # |
| 157 | (n) | 213 | MERGE |
| 158 | (o) | 214 | VERIFY |
| 159 | (p) | 215 | BEEP |
| 160 | (q) | 216 | CIRCLE |
| 161 | (r) | 217 | INK |
| 162 | (s) | 218 | PAPER |
| 163 | (t) | 219 | FLASH |
| 164 | (u) | 220 | BRIGHT |
| 165 | **RND** | 221 | INVERSE |
| 166 | **INKEY$** | 222 | OVER |
| 167 | **PI** | 223 | OUT |
| 168 | **FN** | 224 | LPRINT |
| 169 | **POINT** | 225 | LLIST |
| 170 | **SCREEN$** | 226 | STOP |
| 171 | **ATTR** | 227 | READ |
| 172 | **AT** | 228 | DATA |
| 173 | **TAB** | 229 | RESTORE |
| 174 | **VAL$** | 230 | NEW |
| 175 | **CODE** | 231 | BORDER |
| 176 | **VAL** | 232 | CONTINUE |
| 177 | **LEN** | 233 | DIM |
| 178 | **SIN** | 234 | REM |
| 179 | **COS** | 235 | FOR |
| 180 | **TAN** | 236 | GO TO |
| 181 | **ASN** | 237 | GO SUB |
| 182 | **ACS** | 238 | INPUT |
| 183 | **ATN** | 239 | LOAD |
| 184 | **LN** | 240 | LIST |
| 185 | **EXP** | 241 | LET |
| 186 | **INT** | 242 | PAUSE |
| 187 | **SQR** | 243 | NEXT |
| 188 | **SGN** | 244 | POKE |
| 189 | **ABS** | 245 | PRINT |
| 190 | **PEEK** | 246 | PLOT |
| 191 | **IN** | 247 | RUN |
| 192 | **USR** | 248 | SAVE |
| 193 | **STR$** | 249 | RANDOMIZE |
| 194 | **CHR$** | 250 | IF |
| 195 | **NOT** | 251 | CLS |
| 196 | **BIN** | 252 | DRAW |
| 197 | **OR** | 253 | CLEAR |
| 198 | **AND** | 254 | RETURN |
| 199 | <= | 255 | COPY |

(144–164): user graphics

PART C

## BASIC SUMMARY

**CONVENTIONS**

**n, m or p**
numeric expression

**s**
string expression

**e**
expression (string or numeric)

**V**
variable name

**‹s›**
statement

**[]**
indicates an optional item

Numeric variables are first character a letter then any alpha-numeric characters. String variables are a letter followed close by $.

SECTION 1
**OPERATING COMMANDS**

**BREAK**
Interrupts operation e.g. execution, printer

**CLEAR**
Clears variables

**CLEAR n**
Changes position of RAMTOP

**CONT**
Continues execution after BREAK or STOP

**DELETE**
Allows deletion of character

**EDIT**
Allows editing of current line indicated by > cursor. Copies line to bottom of screen. ↑ ↓ keys control > cursor movement in program. →← keys move mode cursors along the line.

**ENTER**
Line entered into program

**GRAPHICS**
Puts into graphics mode



**LOAD s**
Clears program and existing variables and loads program specified from tape. (string may be "" in which case the first program is loaded)

**LOAD s CODE n,m**
Loads m bytes into memory starting at address n

**LOAD s DATA V()**
Loads specified array (string or numeric) into memory

**MERGE s**
Merges program s with the one already in memory

**NEW**
Clears program and variables

**RUN [n]**
Runs program [starting at line n]

**SAVE s**
Saves program and variables on tape

**SAVE s LINE n**
Saves program so that a LOAD is automatically followed by a GOTO n

**SAVE s CODE n,m**
Saves m bytes starting at address n

**SAVE s SCREEN $**
Saves the picture on tape

**SAVE s DATA V()**
Save specified array (string or numeric) on tape

**STOP**
Stops program execution

**VERIFY s**
Verifies program specified has been saved on tape

**VERIFY s CODE n,m**
Verifies bytes have been saved on tape

**VERIFY s DATA V()**
Verifies array specified has been saved on tape

SECTION 2
**GENERAL INSTRUCTIONS**

**:**
(Colon) separates multiple statements in line.
Example: 10 PRINT : PRINT : INPUT A$.

**DATA e1,e2,....**
Gives data items within a program (see READ).

**DEF FN**
User-defined function definition. It must be followed by the name (single letter) of the string or numeric function and the definition.
Example: FNa(x,y,z)=x ↑ 3+y ↑ 4+z ↑ 5.

**FN**
Calls up the user-defined function. Arguments enclosed in brackets.
Example: (see above) FNa(3,5,7) would give result of $3^3+5^4+7^5$.

**DIM V [$](n[,m])**
Reserves storage space for an array V. Numeric array of n rows [and m columns]. String array of n strings each of length m characters. Multi-dimensional arrays possible.
Examples: DIM A$(3,5) reserves storage for 3 strings, each of length 5 characters. DIM B(4,6) reserves storage for a 4 row and 6 column numeric array.

**FOR V = n TO m [STEP p]**
V a single letter, initiates a loop.

**NEXT V**
Completes the loop.
Example: FOR A = 3 TO 9 STEP 2
(body of loop)
NEXT A
(A will take values 3,5,7 and 9)

**GOSUB n**
Go to subroutine at line n.

**RETURN**
Return from subroutine to main program (control returns to the line immediately following the GOSUB call). Must not enter subroutine except from a GOSUB call.

**GOTO n**
Transfers control to line n.
Example: GOTO 100

**IF e THEN ‹s›**
Executes statement when the condition is met. (There may be several numeric and logical conditions). If the condition is true the command following the THEN is executed, if the condition is not true control passes to the next line.
Examples: IF X = 0 THEN PRINT "ZERO"
IF X > 5 OR X < 10 THEN GOTO 500

**LET V [$] = e [$]**
Assigns value e to variable V.
Examples: LET RADIUS = 200
LET A$ = "JONES"



**PAUSE n**
Makes program wait a specified time (n = 0 waits for ever, n = 1 to 65535 waits n/50 seconds in UK and n/60 seconds in US). Pressing any key cancels PAUSE.

**PEEK n**
Returns the value stored in the memory location n.

**POKE n,m**
Stores the value m in memory location n.

**READ V1 [$], V2 [$],....**
Allocates variables the values specified in DATA statements.
Example: 10 READ A, B, C
100 DATA 100, 200, 300
Will assign A = 100, B = 200, C = 300

**REM**
Allows remarks to be inserted, anything following REM is ignored by the computer.
Example: REM**Draw Picture**

**RESTORE n**
Makes subsequent READ statements obtain data from DATA statements after line n.

**USR n**
Calls the machine code routine starting at line n.

SECTION 3
**GRAPHICS**

22 lines with 32 columns available.
Each character cell consists of 8 by 8 pixels.
256 horizontal points and 176 vertical points.

**CIRCLE n,m,p**
Draws a circle centre (n,m) and radius p

**DRAW n,m[,p]**
Draws line [arc] from previous specified point to a point *relative* n horizontal and m vertical [turning through angle p radians (anticlockwise if p positive)]

**PLOT n,m**
Plots a pixel
0<=n<=255 horizontal
0<=m<=175 vertical

**POINT (n,m)**
Returns 0 (paper colour) or 1 (ink colour) of the pixel (n,m)

SECTION 4
## COLOUR

The picture is divided into 768 (24 lines of 32 columns) character cells.

| | | | |
|---|---|---|---|
| 0 black | 2 red | 4 green | 6 yellow |
| 1 blue | 3 magenta | 5 cyan | 7 white |

**ATTR (n,m)**
Gives colour attributes of the character cell (n,m)
0<=n<=23 (lines)
0<=m<=31 (columns)

**BORDER n**
Makes border specified colour (n = 0 to 7)

**BRIGHT n**
Controls brightness (n = 0 normal, n = 1 bright, n = 8 transparent)

**FLASH n**
Controls flashing (n = 0 normal, n = 1 flash, n = 8 no change)

**INK n**
Makes ink (foreground) specified colour (n = 0 to 7, n = 8 transparent, n = 9 contrast)

**INVERSE n**
Controls dot pattern (n = 0 normal, n = 1 inverse)

**OVER n**
Controls overprinting (n = o normal, n = 1 mixing)

**PAPER n**
Makes paper (background) specified colour (n = 0 to 7, n = 8 transparent, n = 9 contrast)

**Direct Colour**
Coloured flashing programs and characters printed between quotes can be obtained by keying in programs using the E mode to directly control the attributes of the characters entered. Line numbers are unaffected.

Mode E keys 0 – 7 gives PAPER Colour
Mode E CAPS SHIFT keys 0 – 7 give INK Colour
Mode E CAPS SHIFT key 9 gives FLASH on
Mode E CAPS SHIFT key 8 gives FLASH off
Mode E key 9 gives BRIGHT on
Mode E key 8 gives BRIGHT off

Remember to cancel all effects.

SECTION 5
## SOUND

**BEEP n,m**
Produces sound of duration n seconds and pitch m semitones above (or below) Middle C. (m = 0).



SECTION 6
## INPUT/OUTPUT INSTRUCTIONS

**CLS**
Clears the screen

**COPY**
Prints out copy of screen on the printer

**IN n**
Returns the byte read from I/O port n.

**OUT n,m**
Writes value m to I/O port n.

**INKEY $**
Reads current input character. Does not wait for key to be pressed. Example: 100 IF INKEY $ = "" THEN GOTO 100 waits for you to press a key

**INPUT V [$]**
Input numeric [or string] variable from the keyboard.

**INPUT LINE V$**
Allows string variable to be input without quotes.

**LIST [n]**
Displays program [starting from line n]

**LLIST [N]**
Lists program on printer [starting from line n]

**LPRINT [e] [,e] [;e] [TAB n]**
Prints out on printer (see PRINT for details).

**PRINT #**
Allocates a stream number to I/O devices.
#0 = keyboard
#1 = lines 22, 23 on screen (keyboard)
#2 = lines 0-21 on screen
#3 = printer
Example: PRINT #1; "Spectrum"

**OPEN # n, "device type" CLOSE # n**
Used to route or re-route output to specified device types.
n = stream/device NUMBER
device types:- K = keyboard
s = screen
p = printer
Example: OPEN # 2, "p". Make device 2 (screen) a printer device ie output to screen is re-routed to printer. CLOSE # 2 resets it.

**PRINT [e] [,e] [;e] [AT p,m] [TAB m].**
Prints on screen.

22 lines 0<=p<=21
32 columns 0<=m<=31

**PRINT A**
Prints out value of numeric variable A.

**PRINT B$**
Prints out value of string variable B$.

**PRINT "YOUR NAME?"**
Prints out whatever is within the quotes (inverted commas).

**A semicolon (;)**
Between two items causes the printing of the second item immediately after the first.
Example: PRINT A$;B

**A comma (,)**
Between two items causes the print position to be shifted on (at least one place) to either column 16 or to the next line column 0.
Example: PRINT A,B

**An apostrophe (')**
Causes print position to shift to the next line.

**TAB C;**
Moves the print position to column C. If this would involve back-spacing it moves on to the next line.
Example: PRINT TAB 5; "NAME"; TAB 15; A$

**PRINT AT L , C**
Moves the print position to line L and column C.

**PRINT**
Leaves a blank line.

N.B.
For the LPRINT instruction, TAB works exactly as PRINT TAB. LPRINT AT L,C is converted to LPRINT TAB C, and the number of the line is ignored.

SECTION 7
**TRIGONOMETRICAL FUNCTIONS**
*(n evaluated in radians)*

**ACS n**
Arc cosine n.

**ASN n**
Arc sine n.

**ATN n**
Arc tangent n.

} radians = degrees * PI/180
eg PRINT TAN (45 * PI/180)

**COS n**
Cosine n.

**SIN n**
Sine n.

**TAN n**
Tangent n.



SECTION 8
**NUMERIC FUNCTIONS**

**ABS n**
Absolute value of n.

**BIN n**
Puts binary number n into decimal.

**EXP n**
Exponential n (i.e. $e^n$).

**INT n**
Integer of n (rounds down).
Examples: INT(2.7) = 2, INT (–2.7) = – 3

**LN n**
Natural logarithm of n (i.e. $\log_e n$ or ln n).

**PI**
π, 3.1415927

**RAND [n]**
Random number seed.
n between 1 and 65535 gives a set sequence of random numbers (n determines start position) n omitted (or zero) gives a different set of random numbers each time.

**RND**
Returns a random number between 0 and 1.

**SGN n**
Returns 1 if n is positive.
0 if n is zero
–1 if n is negative.

**SQR n**
Square root of n.

SECTION 9
**STRING FUNCTIONS**

**String**
Is a set of characters in quotes.
Examples: "SMITH", "PROG01", "*?!"
The null string "" has no characters.

**String variable**
Is used to store strings. It consists of a single letter followed by the $ sign.
Example: Z$

**String array variable**
Is an array of strings. A string variable must be dimensioned.
Examples: DIM N$ (6,4) saves storage for 6 strings each 4 characters long
DIM A$ (3,4,5) saves storage for a string array of 3 rows and 4 columns, each string having up to 5 characters

**Substring**
Is any set of consecutive characters taken in sequence from the parent string. Also called a string slice.
Example: A$ = "COMPUTER", A$ (3 TO 6) = "PUT", A$ (7 TO 8) = "ER"

**Concatenation**
Is the joining together of strings.
Example: A$ = "SPEC", B$ = "TRUM", A$ + B$ = "SPECTRUM"

**String comparison**
May be done using any of relational operators (see 10).

**CHR$ n**
Character of code n.
Example: CHR$ 96 gives "£"

**CODE s**
Code of first character of string s.
Example: CODE "LAST" = CODE "L" = 76

**LEN s**
Returns length of string s as the number of characters in the string.
Example: LEN "SPECTRUM" gives 8

**STR$ n**
Converts a numeric expression n into a string
Example: STR$ (3.4) gives "3.4"

**VAL s**
Converts string expression into numeric.
Example: VAL "SQR16" gives 4

**VAL$ s**
Converts s to a string expression (strips off quotes).
Example: VAL$ """ME""" gives ME

## SECTION 10
## ARITHMETIC AND LOGIC

### ARITHMETIC OPERATIONS

\+ addition
* multiplication
↑ exponentiation
— subtraction
/ division
Example: 2 ↑ 3 = $2^3$ = 2 × 2 x 2 = 8

### RELATIONAL OPERATORS

| | |
|---|---|
| = equal to | >< not equal to |
| < less than | > greater than |
| <= less than or equal | >= greater than or equal |



NUMBERS
Are stored to an accuracy of 9 digits and returned to an accuracy of 8 digits.

Largest number $10^{38}$. Smallest number $4 \times 10^{-39}$ (anything smaller taken as zero).

Scientific (or E) notation.
Examples: 1.73E5 = $1.73 \times 10^5$ = 173000, 2.56E – 7 = $2.56 \times 10^{-7}$ = .000000256.

### LOGICAL EXPRESSIONS

**AND**
Combines relations so that (condition 1) AND (condition 2) is only TRUE when both conditions are true.
Example: IF x > = 1 AND x < = 10 THEN PRINT "BETWEEN 1 AND 10"

In numeric operations

$$a \text{ and } b = \begin{cases} a \text{ if } b <> 0 \\ 0 \text{ if } b = 0 \end{cases}$$

In string operations

$$a\$ \text{ AND } b = \begin{cases} a\$ \text{ if } b><0 \\ "" \text{ (null string)} \\ \text{if } b = 0 \end{cases}$$

**NOT**
Logically gives inverse of an expression.
Example: IF NOT (A = B) THEN PRINT "NOT EQUAL"

In numeric operations

$$\text{NOT } a = \begin{cases} 0 \text{ if } a <>0 \\ 1 \text{ if } a = 0 \end{cases}$$

**OR**
Combines relations so that (condition 1) OR (condition 2) is TRUE when either (condition 1) or (condition 2) is true (or both true).
Example: IF x < 13 OR x > 19 THEN PRINT "NOT TEENAGER"

In numeric operations

$$a \text{ OR } b = \begin{cases} 1 \text{ if } b <>0 \\ a \text{ if } b = 0 \end{cases}$$

### EXPRESSIONS PRIORITY

| | | |
|---|---|---|
| 12 | () | bracketed expressions |
| 11 | any function | functions |
| 10 | ↑ | exponentiation |
| 9 | — s | unary minus |
| 8 | * | multiplication |
| 7 | / | division |
| 6 | + – | addition and subtraction |
| 5 | =, <>, <, >, <=, >= | equality and inequalities |
| 4 | NOT | logical inversion |
| 3 | AND | logical AND |
| 2 | OR | logical OR |

PART D

## ERROR CODES

The report has a code number or letter (so that you can refer to the following table), a brief message explaining what happened and the line number and statement number within that line where it stopped. (A command is shown as line 0. Within a line, statement 1 is at the beginning, statement 2 comes after the first colon or THEN, and so on.)

The behaviour of CONTINUE depends very much on the reports. Normally CONTINUE goes to the line and statement specified in the last report, but there are exceptions with reports 0, 9 and D.

Here is a table showing all the reports. It also tells you in what circumstances the report can occur.

| Code | Meaning | Situations |
|---|---|---|
| 0 | **OK**<br>Successful completion, or jump to a line number bigger than any existing. This report does not change the line and statement jumped to by CONTINUE. | Any |
| 1 | **NEXT without FOR**<br>This control variable does not exist (it has not been set up by a FOR statement), but there is an ordinary variable with the same name. | NEXT<br>Jumping *into* a loop is a common cause. |
| 2 | **Variable not found**<br>For a simple variable this will happen if the variable is used before it has been assigned to in a LET, READ or INPUT statement, loaded from tape or set up in a FOR statement. For a subscripted variable it will happen if the variable is used before it has been dimensioned in a DIM statement or loaded from tape. | Any |
| 3 | **Subscript wrong**<br>A subscript is beyond the dimension of the array, or there are the wrong number of subscripts. If the subscript is negative or bigger than 65535, then error B will result. | Subscripted variables (arrays). Substrings |
| 4 | **Out of memory**<br>There is not enough room in the computer for what you are trying to do. If the computer really seems to be stuck in this state, you may have to | LET, INPUT, FOR, DIM, GO SUB, LOAD, MERGE. Sometimes |



| Code | Meaning | Situations |
|---|---|---|
| | clear out the command line using DELETE and then delete a program line or two (with the intention of putting them back afterwards) to give yourself room to manoeuvre with — say — CLEAR. | during expression evaluation. |
| 5 | **Out of screen**<br>An INPUT statement has tried to generate more than 23 lines in the lower half of the screen. Also occurs with PRINT AT 22.... | INPUT, PRINT AT |
| 6 | **Number too big**<br>Calculations have led to a number greater than about $10^{38}$ | Any arithmetic. Division by zero is common cause. |
| 7 | **RETURN without GO SUB**<br>There has been one more RETURN than there were GO SUBs. | RETURN. No STOP statement before a subroutine is common. |
| 8 | **End of file** | Microdrive, etc, operations only. |
| 9 | **STOP statement**<br>After this, CONTINUE will not repeat the STOP, but carries on with the statement after, or next line after, STOP | STOP |
| A | **Invalid argument**<br>The argument for a function is no good for some reason. | SQR, LN, ASN, ACS, USR (with string argument). |
| B | **Integer out of range**<br>When an integer is required, the floating point argument is rounded to the nearest integer. If this is outside a suitable range then error B results. | RUN, RANDOMIZE, POKE, DIM, GO TO, GO SUB, LIST, LLIST, PAUSE, PLOT, CHR$, PEEK, USR (with numeric argument) |
| C | **Nonsense in BASIC**<br>The text of the (string) argument does not form a valid expression. | VAL, VAL$ |

| Code | Meaning | Situations |
|---|---|---|
| D | **BREAK – CONT repeats**<br>BREAK was pressed during some peripheral operation.<br>The behaviour of CONTINUE after this report is normal in that it repeats the statement. Compare with report L. | LOAD, SAVE, VERIFY, MERGE, LPRINT, LLIST, COPY. Also when the computer asks scroll? and you type N, SPACE or STOP |
| E | **Out of DATA**<br>You have tried to READ past the end of the DATA list. | READ |
| F | **Invalid file name**<br>SAVE with name empty or longer than 10 characters. | SAVE |
| G | **No room for line**<br>There is not enough room left in memory to accommodate the new program line. | Entering a line into the program |
| H | **STOP in INPUT**<br>Some INPUT data started with STOP, or — for INPUT LINE — BREAK was pressed.<br>Unlike the case with report 9, after report H CONTINUE will behave normally, by repeating the INPUT statement. | INPUT |
| I | **FOR without NEXT**<br>There was a FOR loop to be executed no times (e.g. FOR n = 1 TO 0) and the corresponding NEXT statement could not be found. | FOR |
| J | **Invalid I/O device** | Microdrive, etc., operations only |
| K | **Invalid colour**<br>The number specified is not an appropriate value. | INK, PAPER, BORDER, FLASH, BRIGHT, INVERSE, OVER; also after one of the corresponding control characters |



| Code | Meaning | Situations |
|---|---|---|
| L | **BREAK into program**<br>BREAK pressed, this is detected between two statements. The line and statement number in the report refer to the statement before BREAK was pressed, but CONTINUE goes to the statement after (allowing for any jumps to be done), so it does not repeat any statements. | Any |
| M | **RAMTOP no good**<br>The number specified for RAMTOP is either too big or too small. | CLEAR; possibly in RUN |
| N | **Statement lost**<br>Jump to a statement that no longer exist. | RETURN, NEXT, CONTINUE |
| O | **Invalid stream** | Microdrive, etc. operations only |
| P | **FN without DEF**<br>User-defined function | FN |
| Q | **Parameter error**<br>Wrong number of arguments, or one of them is the wrong type (string instead of number or vice versa). | FN |
| R | **Tape loading error**<br>A file on tape was found but for some reason could not be read in, or would not verify. | VERIFY, LOAD or MERGE |

# PART E
## HINTS AND TIPS

**Ram Size**
PRINT "48K" AND PEEK 23733 = 255

**ROM Test**
Unplug the printer, microdrive and tape recorder. Running this program gives a correct total of 1926175
10 LET c = 0: FOR b = 0 TO 16383: LET c = c + PEEK b: NEXT b: PRINT c.

**Key Repeat**
For a faster repeat POKE 23562, 3

**Time Before Repeat**
POKE 23561,n
n is in 50ths of a second
Try n = 10

**Keyboard Beep**
POKE 23609, n

n = 1 to 15 louder click
n = 15 to 255 louder beep
n = 0 disables to normal click

**Tape Contents**
Execute the command VERIFY "CATALOG" and run the tape. Programs and files will be listed.

**Auto-Save**
Make the last program line n SAVE "program name". Program is saved when run

**Auto-run**
Use SAVE "program name" LINE start line.
Example: SAVE "autostart" LINE 20
Program will run automatically when loaded from given line

**Separating Program Modules**
Spaces in listed programs are obtained by keying in

line number SPACE ENTER

Only the line number will appear in the listing.

**Avoiding SCROLL? when editing**
Instead of using LIST n, enter a virtual line number one less than the required line and press EDIT to bring down the line.
Example: n − 1 ENTER EDIT

**Multi-statement lines**
Never start with REM. Care with loops containing conditions, multi-statements after can be missed.

**Clearing a Complete Input Line**
Use EDIT ENTER



**Auto-Scroll**
Include 10 POKE 23692, 0

**PRINT**
PRINT AT l,c;, clears the screen from column c to column 16 line l.

PRINT AT l,c;,, clears to the end of the line
PRINT AT l,c1;TAB c2 clears columns c1 to c2 of line l
PRINT AT l,c; "SPACE" deletes the character at l,c
PRINT CHR$ 8 moves the print position back one place
PRINT CHR$ 13 moves the print position to the start of the next line (same as delimiter)

**Device numbers # n**
Use PRINT #n, LIST #n, INPUT # n and select n for the Device.

Where
n = 0 is keyboard
n = 1 is bottom two lines of screen (22,23) (also called keyboard!)
n = 2 is screen lines 0-21
n = 3 is Printer

Example: 10 INPUT "PRINTER?" ; LINE A$
20 LET n = 2 + (A$(1) = "Y")
30 PRINT # n;-----

**Re-routing output**
OPEN # n; "S/K/P"
Output in the form of PRINT (;) in a program can be re-routed from screen (#2) to printer or lines 22,23 by naming the device.
Example: OPEN #2; "P" naming P for printer
OPEN #2; "K" naming K for lines 22,23
OPEN #2; "S" gets back to screen

**Changing the PRINT SCREEN**
POKE 23659., n 2<n<25 will stop any printing on the bottom n lines

**Printing on 23 or 24 lines**
POKE 23659, 1 prints on 23 lines. You must POKE the value 2 back before the next INPUT, CLS, STOP or end by nnn POKE 23659, 2.

POKE 23659, 0 prints on 24 lines. Remember to add PAUSE 0 or a report overwrite will occur. To print on the 24th line directly; TAB from the 23rd.
Example: 10 POKE 23659, 0 : FOR f = 1 TO 24 : PRINT "P" : NEXT f : PAUSE 0 : POKE 23659, 2

**Current Print position**
Coordinates (line, column) are (24 − PEEK 23689, 33 − PEEK 23688). To print them, assign to variables l, c first.

**Last Plot position**
Pixel coordinates are (x,y) :− (PEEK 23677, PEEK 23678).

**Bit Patterns**
Of any character at (l,c) on the print screen in decimal.

10 FOR n = 0 TO 7 : PRINT PEEK (16384 + 32 * (l + 56 * INT(l/8)) + c + n * 256)) : NEXT n
Use binary-decimal conversion for pattern in binary.

**Resetting Colour**
When developing programs, colours remain in force during listings. Include a line

9999 BORDER 7 : PAPER 7 : INK 0 : FLASH 0 : BRIGHT 0 : CLS : STOP
USE GO TO 9999

to reset the screen before listing.

**Changing Colour**
100 PRINT AT l,c; OVER 1; INK a; PAPER b; "SPACE"
will change the ink and paper colours (a,b) of any cell (l,c) without affecting the display.

100 DIM S$ (22,32) : PRINT AT 0,0; OVER 1 ; INK a ; PAPER b ; S$
will change ink and paper colour over the whole screen without affecting the display. Use a or b = 8 for transparency.

100 INK a : INVERSE 1 ; OVER 1 ; PLOT/DRAW/CIRCLE PAPER b ; x,y will change ink/paper colours of cells plotted or drawn without affecting shape.

100 INK a : INVERSE 1 : OVER 1 : PLOT p,q : DRAW PAPER b ; x,y,z changes the ink and paper colour of a row or column of cells along a line from point (p,q).

**Data Count**
Make Z the number of items (I), in DATA statements. You can tell the program how many items to read using

10 READ Z : DIM I(Z) : FOR n = 1 to Z READ I(n) : NEXT n
20 DATA Z
30 DATA --- ITEMS(I).

**Array Size**
Programs using loaded arrays have no 'no idea of their size'. So let the first element in a data or string array be its size.
Example: DIM a (200) : LET a (1) = 200
DIM a$ (200, 10) : LET a$ (1) = CHR$ 200

When loading the arrays the sizes are determined by

FOR n = 2 to a (1)
and
FOR n = 2 to CODE a$ (1)

**Protecting Programs**
10 REM © MYSELF 1983 is protected from change by POKE (PEEK 23635 + 256 * PEEK 23636 + 1), 0 which changes the lines number to 0. Then enter the program.

POKE (PEEK 23635 + 256 * PEEK 23636), n
If n = 40 to 63, when entered, changes previous lines to a symbol



followed by 3 digits and puts them at the end of the program.

If n > 63 previous lines will disappear from the listing completely.

**Programs in Memory**
Programs are held between PROG (23635/6) and VARS (23627/8). To look at them byte by byte enter

10 FOR P = PEEK 23635 + 256 * PEEK 23636 TO
PEEK 23627 + 256 * PEEK 23628 :
PRINT P ; TAB 8 ;PEEK P; TAB 13;
CHR$ PEEK P; NEXT P

**Program length: in bytes**
PRINT (PEEK 23627 + 256 * PEEK 23628 − PEEK 23635 − 256 * PEEK 23636).

**Spare memory in bytes**
PRINT (PEEK 23730 + 256 * PEEK 23731 − PEEK 23653 − 256 * PEEK 23654).

**Timing**
Uses the T.V. frame counter, incremented 50 times/sec. Set the clock (FRAMES) to zero by
POKE 23672, 0 : POKE 23673, 0 : POKE 23674, 0

Read the clock after a time t by
LET t = PEEK 23672 + 256 * PEEK 23673 + 66536 * PEEK 23674
(last term to be included if times › 20 mins needed.)

Read it again LET t1 = PEEK 23672 + 256 * PEEK 23673 + 66536 * PEEK 23674 and take the higher value (it sometimes misreads!).
IF t < t1 THEN t = t1
PRINT t/50 ; "seconds"

**Conversion**
Decimal to Binary
10 INPUT n : FOR Z = 1 TO 8 : LET d = INT (n/2) :
LET b = n − 2 * d : LET n = d : PRINT AT 11, 19-Z, d : NEXT Z

Binary to Decimal
(1) PRINT BIN b (calculator mode. b is up to 16 binary digits).
(2) READ X: DATA BIN b : PRINT X.
(3) INPUT X: PRINT X (key in X as BIN b)
(4) INPUT LINE b$ : PRINT VAL ("BIN"+b$)
(the binary digits are handled as a string variable).

Multiple Condition Testing
Use IF a = c1 THEN IF b = c2 THEN IF c = c3 which jumps out as soon as a condition is false instead of IF a = c1 AND b = c2 AND c = c3 which tests all.

## PART F
## Memory Map

| Address | Contents |
|---|---|
| 32767 (16k)<br>65535 (48k)<br>RAMTOP | |
| | USER DEFINED GRAPHICS |
| | GOSUB STACK |
| | MACHINE STACK |
| | SPARE MEMORY |
| | CALCULATOR STACK |
| | WORKSPACE (TEMP STORAGE) |
| | VARIABLES STORAGE AREA |
| | PROGRAM AREA |
| | CHANNEL INFORMATION |
| | Microdrive Maps Area |
| 23734 | |
| 23652 | SYSTEM VARIABLES |
| 23296 | PRINTER BUFFER |
| 22528 | ATTRIBUTES FILE |
| 16384 | DISPLAY FILE |
| | ROM AREA, BASIC INTERPRETER |
| | AND OPERATING SYSTEM PROGRAM |
| 0000 | |

RAM

## MEMORY MAP WITH SYSTEM VARIABLES

| Memory Area | System Variable | Addresses | Contents returned by: |
|---|---|---|---|
| User Defined Graphics | P-RAMT | 23730 | PEEK 23730 + 256*PEEK 23731 |
| Byte with Code 60 | UDG | 23675 | PEEK 23675 + 256*PEEK 23676 |
| Byte with Code 0 | RAMTOP | 23730 | PEEK 23730 + 256*23731 |
| GOSUB STACK | | | |
| MACHINE STACK | Not accessible with BASIC | | |
| SPARE MEMORY | Not accessible with BASIC | | |
| CALCULATOR STACK | STKEND | 23653 | PEEK 23653 + 256*PEEK 23654 |
| TEMP WORK SPACE | STKBOT | 23651 | PEEK 23651 + 256*PEEK 23652 |
| Byte with Code 13 | | | |
| INPUT DATA | WORKSP | 23649 | PEEK 23649 + 256*PEEK 23650 |
| Byte with Code 128 | | | |
| Byte with Code 13 | E_LINE | 23641 | PEEK 23641 + 256*PEEK 23642 |
| Current Line Keyed In | | | |
| Byte with Code 128 | VARS | 23627 | PEEK 23627 + 256*PEEK 23628 |
| VARIABLES STORE | | | |
| BASIC PROGRAM STORAGE | PROG | 23635 | PEEK 23635 + 256*PEEK 23636 |
| Byte with Code 128 | | | |
| CHANNEL INFO. | CHANS | 23631 | PEEK 23631 + 256*PEEK 23632 |
| (Microdrive Maps) | Fixed | 23734 | |



## PART G
## System Variables

**Notes:**
X The system may crash if the variable is poked.
N Poking the variable will have no lasting effect.

The number in column 1 is the number of bytes in the variable. For two bytes, the first one is the less significant byte. To poke a value M to a two-byte variable at address N use

**POKE N,(M − 256* INT(M/256))**
**POKE N + 1, INT M/256**

and to peek its value, use the expression

**POKE N + 256*PEEK (N + 1)**

| Notes | Address | Name | Contents |
|---|---|---|---|
| N8 | 23552 | KSTATE | Used in reading the keyboard. |
| N1 | 23560 | LAST K | Stores newly pressed key. |
| 1 | 23561 | REPDEL | Time (in 50ths of a second — in 60ths of a second in N. America) that a key must be held down before it repeats. This starts off at 35, but you can **POKE** in other values. |
| 1 | 23562 | REPPER | Delay (in 50ths of a second — in 60ths in N. America) between successive repeats of a key held down: initially 5. |
| N2 | 23563 | DEFADD | Address of arguments of user-defined function if one is being evaluated; otherwise 0. |
| N1 | 23565 | K DATA | Stores 2nd byte of colour controls entered from keyboard. |
| N2 | 23566 | TVDATA | Stores bytes of colour, **AT** and **TAB** controls going to television. |
| X38 | 23568 | STRMS | Addresses of channels attached to streams. |
| 2 | 23606 | CHARS | 256 less than address of character set (which starts with space and carries on to the copyright symbol). Normally in ROM, but you can set up your own in RAM and make CHARS point to it. |
| 1 | 23608 | RASP | Length of warning buzz. |
| 1 | 23609 | PIP | Length of keyboard click. |
| 1 | 23610 | ERR NR | 1 less than the report code. Starts off at 255 (for − 1) so **PEEK 23610** gives 255. |
| X1 | 23611 | FLAGS | Various flags to control the BASIC system. |
| X1 | 23612 | TV FLAG | Flags associated with the television. |
| X2 | 23613 | ERR SP | Address of item on machine stack to be used as error return. |
| N2 | 23615 | LIST SP | Address of return address from automatic listing. |
| N1 | 23617 | MODE | Specifies K, L, C, E or G cursor. |
| 2 | 23618 | NEWPPC | Line to be jumped to. |
| 1 | 23620 | NSPPC | Statement number in line to be jumped to. Poking first NEWPPC and then NSPPC forces a jump to a specified statement in a line. |
| 2 | 23621 | PCC | Line number of statement currently being executed. |

| Notes | Address | Name | Contents |
|---|---|---|---|
| 1 | 23623 | SUBPPC | Number within line of statement being executed. |
| 1 | 23624 | BORDCR | Border colour* 8; also contains the attributes normally used for the lower half of the screen. |
| 2 | 23625 | E PPC | Number of current line (with program cursor). |
| X2 | 23627 | VARS | Address of variables. |
| N2 | 23629 | DEST | Address of variable in assignment. |
| X2 | 23631 | CHANS | Address of channel data. |
| X2 | 23633 | CURCHL | Address of information currently being used for input and output. |
| X2 | 23635 | PROG | Address of BASIC program. |
| X2 | 23637 | NXTLIN | Address of next line of program. |
| X2 | 23639 | DATADD | Address of terminator of last DATA item. |
| X2 | 23641 | E LINE | Address of command being typed in. |
| 2 | 23643 | K CUR | Address of cursor. |
| X2 | 23645 | CH ADD | Address of the next character to be interpreted: the character after the argument of **PEEK**, or the **NEWLINE** (ENTER) at the end of a **POKE** statement. |
| 2 | 23647 | X PTR | Address of the character after the Syntax error marker. |
| X2 | 23649 | WORKSP | Address of temporary work space. |
| X2 | 23651 | STKBOT | Address of bottom of calculator stack. |
| X2 | 23653 | STKEND | Address of start of spare space. |
| N1 | 23655 | BREG | Calculator's b register. |
| N2 | 23656 | MEM | Address of area used for calculator's memory. (Usually MEMBOT, but not always.) |
| 1 | 23658 | FLAGS2 | More flags. |
| X1 | 23659 | DF SZ | The number of lines (including one blank line) in the lower part of the screen. |
| 2 | 23660 | S TOP | The number of the top program line in automatic listings. |
| 2 | 23662 | OLDPPC | Line number to which **CONTINUE** jumps. |
| 1 | 23664 | OSPPC | Number within line of statement to which **CONTINUE** jumps. |
| N1 | 23665 | FLAGX | Various flags. |
| N2 | 23666 | STRLEN | Length of string type destination in assignment. |
| N2 | 23668 | T ADDR | Address of next item in syntax table (very unlikely to be useful). |
| 2 | 23670 | SEED | The seed for **RND**. This is the variable that is set by **RANDOMIZE**. |
| 3 | 23672 | FRAMES | 3 bytes (least significant first), frame counter. Incremented every 1/50th second (U.K.) or 1/60th second (U.S.). |
| 2 | 23675 | UDG | Address of 1st user-defined graphic. |
| 1 | 23677 | COORDS | x-coordinate of last point plotted. |
| 1 | 23678 | | y-coordinate of last point plotted. |
| 1 | 23679 | P POSN | 33-column number of printer position. |
| 1 | 23680 | PR CC | Less significant byte of address of next position for **LPRINT** to print at (in printer buffer). |
| 1 | 23681 | | Not used. |



| Notes | Address | Name | Contents |
|---|---|---|---|
| 2 | 23682 | ECHO E | 33-column number and 24-line number (in lower half) of end of input buffer. |
| 2 | 23684 | DF CC | Address in display file of **PRINT** position. |
| 2 | 23686 | DFCCL | Like DF CC for lower part of screen. |
| X1 | 23688 | S POSN | 33-column number for **PRINT** position. |
| X1 | 23689 | | 24-line number for **PRINT** position. |
| X2 | 23690 | SPONSNL | Like S POSN for lower part. |
| 1 | 23692 | SCR CT | Counts scrolls: it is always 1 more than the number of scrolls that will be done before stopping with scroll? |
| 1 | 23693 | ATTR P | Permanent current colours, etc. (as set up by colour statements. |
| 1 | 23694 | MASK P | Used for transparent colours, etc. Any bit that is 1 shows that the corresponding attribute bit is taken not from ATTR P, but from what is already on the screen. |
| N1 | 23695 | ATTR T | Temporary current colours, etc (as set up by colour items). |
| N1 | 23696 | MASK T | Like MASK P, but temporary. |
| 1 | 23697 | P FLAG | More flags. |
| N30 | 23698 | MEMBOT | Calculator's memory area; used to store numbers that cannot conveniently be put on the calculator stack. |
| 2 | 23728 | | Not used. |
| 2 | 23730 | RAMTOP | Address of last byte of BASIC system area. |
| 2 | 23732 | P-RAMT | Address of last byte of physical RAM. |