ZX Spectrum Next

Notes on the Sinclair ZX Spectrum Next Retro Gaming, ZX Spectrum, Z80 Assembly Language

Next BASIC

Editor

EXTEND MODE, CAPSLOCK - Start of program
EXTEND MODE, GRAPHICS - End of program

```
ERASE 100,200          ;Erase lines 100 to 200
LINE 100,2             ;Renumbers lines from 100 in steps of 2, 100, 102, 104, 106...
LINE MERGE 20,40       ;Moves lines 20-40 all onto line 20
BANK n LINE first,last ;Copy lines to Bank
```

To disable the BREAK key (and Scroll? breaking) and prevent the user accessing the editor, set bit 1 of NextBASIC's %CODE variable:

```
%CODE = %CODE | 2      ;Disables BREAK key, add to the start of your program
```

Variables

Only 26 Integer variables %a to %z

```
%x,%y = 16,12          ;unsigned 16bit
result2 = 1.75         ;40bit (4 bytes mantissa, 1 byte exponent)
name$ = "string var"
n = 1.75               ;This is not %n
bonus = 1000+(%w*50)   ;Cast int to float ()
%s = %INT {bonus}
```

Arrays

```
DIM prices(10)
prices(1) = 1.42     ;First Index
prices(10) = 9.99    ;Last Index

DIM names$(20,32)    ;20 strings of length 32 chars
names$(1) = "Pig Dog Bay"

; Integer arrays are already predefined, %a[] to %z[] each with 64 elements
; Use () for indices 0-63
%m(63) = 999

;Use [] to extend across multiple sequential integer arrays
```

```
%d[64] = 42          ;Same as %e(0) = 42
%a[1663] = 88        ;Same as %z(63) = 88, 1663 = 26 * 64 -1
```

Decisions

```
IF answer=42 THEN PRINT "What is the question?"
IF %x=10 THEN %x=0: ELSE %x+=1

10 IF a$="spectrum" : PRINT "Good"
20 ELSE IF a$="commodore" : PRINT "Bad"
25     PRINT "and Ugly!"
30 ELSE PRINT "Try again"
40 ENDIF

;Jump to statement
ON x : GO TO 1000 : GO TO 2000 : PRINT "x is 2" : ELSE PRINT "else is optional"

;Select expression
PRINT n?("n is 0", "n is 1", "n is >1")

;Need to use brackets with AND
IF %g=19 AND (i>200) THEN SPRITE %g,%i,32,%2,%1
```

Loops

```
FOR %I=0 TO 10 STEP 2
    PRINT %I          //0,2,4,6,8,10
NEXT %I

REPEAT
   INPUT n
   IF n=42 THEN EXIT 300       //EXIT will jump to line 300
REPEAT UNITL n<100

REPEAT
  INPUT n
  WHILE n>=0.                  //stay in loop if true
  WHILE n<100
REPEAT UNTIL 0                 //Loop forever as 0 = false
```

Structure

```
;Use labels instead of line numbers
10 GO TO @setUpGame
1000 @setUpGame : PRINT "Setting Up"

10 GOSUB 100
90 STOP
100 REM Subroutine
110 RETURN
```

Procedure Args are pass by value, unless using REF. Args are also local.

```
PROC double(42) TO %a
DEFPROC double(%x)
  LOCAL %a
  %a = %x * 2
ENDPROC = %a
```

Multiple return values

```
PROC getTime() TO h,m,s
```

```
DEFPROC getTime()
  LOCAL a$ = TIME$
  LOCAL h=VAL(a$(12 TO 13))
  LOCAL m=VAL(a$(15 TO 16))
  LOCAL s=VAL(a$(18 TO ))
ENDPROC = h,m,s
```

```
DEF FN sq(x)=x*x
```

READ, DATA, RESTORE

```
40 RESTORE 1000
50 READ %i,name$
1000 DATA 42,"ZX Next"

;Labels also work
RESTORE @WaveData
```

PRINT / INPUT

AT is Y(0-21),X(0-31), POINT is X,Y. For layer 2 you can print at the bottom of the screen normally reserved for INPUT, POINT AT Y(0-23),X(0-31).

```
PRINT "Col 1","Col 2"
PRINT "Toge";"ther"
PRINT "New"'"Line"

PRINT AT 22,31;"*"           ;Prints * bottom right
LAYER 1,1: PRINT POINT 248,176;"*"

;Fixed width spacing
PRINT "Name";TAB 8;"Score";TAB 8;"Rank"


INPUT "What is your name? ";name$
INPUT LINE a$               ;Removes quotes

5 REM Handling key presses
10 IF INKEY$ <> "" GO TO 10     ;Wait for user to release key
20 IF INKEY$ = "" GO TO 20      ;Wait for user to press key
30 PRINT INKEY$
40 GOTO 10

REM n=0 Keyboard Joystick, n=1 Joystick 1, n=2 Joystick 2
%p = INPUT n                    ;Bits: 3210udlr (fire buttons bits 4-7)
```

Random

```
%r = %RND 100         ;fast random integer 0-99 - no casting
%r = INT(RND * 100)   ;Casting Float to Int, slow
myRand = RND * 100    ;0 to 99.999999
myRand = RND(100)     ;0 to 99 real values rounded to nearest whole number

RANDOMIZE             ;Pick a random seed
RANDOMIZE seed        ;Seed 1-65535, 0 - random

; NextBASIC's %CODE bit 0 can change RND's behaviour:
```

```
%CODE = 0             ;%RND n and RND(n) return values 0..n-1
%CODE = 1             ;%RND n and RND(n) return values 0..n
```

Timing

```
PAUSE n               ;n 1/50th second (when in for 50Hz display mode)

;TIME reads System var FRAMES which increase every 1/50s
start = TIME
PRINT "Time taken: ";(TIME-start)

TIME$                ;Returns RTC string 2024-01-3 10:55:47

;Update RTC with network time
.nxtp time.zx.in.net 12300 -z=UTC
```

Expressions

% at the beginning denotes integer expression

```
;Modulo
%17 MOD 6           ;Returns 5

;Bitwise Operators
%!$f                ;NOT returns $fff0
%x & $0f            ;AND use $ for hexadecimal
%x | BIN 01001100   ;OR use @ or BIN for binary notation
%x << 2             ;Shift right 2 places (mul by 4)
%x >> 4             ;Shift left 4 places (div by 16)
%x ^| $y            ;XOR (^ is like an up arrow character on the speccy)

;Logical
x AND y             ;Gives 0 if y is zero, x if y non-zero
x OR y              ;Gives x if y is zero, 1 if y non-zero
NOT x               ;0->1, non-zero->0

%1 + RND 6          ;Random integer 1-6

%i = -42            ;Integers are unsigned, %i is 65494 (2's complement)
PRINT %SGN {i}      ;Cast unsigned integer to signed integer
```

Strings

```
"abcdefg"(2 To 4)      ;Gives "bcd"
"abcdefg"( TO 5)       ;Gives "abcde"
"abcdefg"(3 TO)        ;Gives "cdefg"

a$(5 TO 8) = "****"

"abc"+"def"
"abc"*2
"abc"*-1               ;Reverse string
"abcdef"*0.5           ;Gives "abc"
```

Modifiers

String$[modifiers]

```
;+ Lower case, - upper case, < strip leading spaces, > strip trailing spaces
"    Hello There!    "[<+->]
Result: "hELLO tHERE!"
```

Tokenisation

```
{"sin (pi/4)"}        ;Gives "SIN(PI/4)" using BASIC tokens
VAL{"sin (pi/4)"}     ;Gives 0.7071
i=VAL(a$)             ;Convert string to number
```

Graphics

Layers

Origin - Top Left, except Layer 0 - Origin Bottom Left. Note that layer 3 is not yet supported in BASIC.

```
Layer      Pixels       Colours      Scheme           Name
0          256x192      15           32x24 cells      Standard Spectrum(ULA)
1,0        128x96       256          per pixel        LoRes (EnhancedULA)
1,1        256x192      256          32x24 cells      StandardRes (EnhancedULA)
1,2        512x192      256          monochrome       Timex HiRes (EnhancedULA)
1,3        256x192      256          32x192 cells     Timex HiColour (EnhancedULA)
2,0 - Disable layer 2
2,1        256x192      256          per pixel        RRRGGGBB Colour -
2,2        320x256      256          per pixel        - PALETTE has no effect -
2,3        640x256      16           per pixel        - in layer 2
3,0        320x256      256          40x32 cells      Text Mode
3,1        640x256      256          80x32 cells      Text Mode
3,2        320x256      256          40x32 cells      Graphics Mode - 16 colours per cell
3,3        640x256      256          80x32 cells      Graphics Mode - 16 colours per cell
```

Commands

```
LAYER 0                   ;Default layer
LAYER 2,0                 ;Disable layer 2
LAYER CLEAR
LAYER OVER order
LAYER ERASE x1,y1,x2,y2,c             ;c is optional, default is transparency colour
LAYER DIM x1,y1,x2,y2                 ;Set clipping window
LAYER AT x,y                          ;Move layer, layer will wrap, good for scrolling background
```

Colours

0-Black, 1-Blue, 2-Red, 3-Magenta, 4-Green, 5-Cyan, 6-Yellow, 7-White

```
BORDER, PAPER, INK, BRIGHT, FLASH
INVERSE 1            ;Swaps INK and PAPER
OVER 1               ;XOR mode for text
ATTR line,col        ;Get attribute value for the cell

;Palettes are a table of colour values for EnhancedULA layers
;B bit: RRRGGGBB
;9 bit: R RRGGGBBB
;9 bit palettes require 2 bytes per colour
PALETTE DIM bits           ;8 or 9
PALETTE FORMAT inkCount    ;0,1,3,7,15,31,63,127 or 255
PALETTE FORMAT 0           ;Disable enhanced ULA, normal colour mode
PALETTE CLEAR              ;Resets all palettes
```

```
PALETTE OVER value         ;Transparency: value - RRRGGGBB (8 bit), index (9 bit)

;Select palette 0 or 1
LAYER PALETTE num
;Load a bank into the palette
LAYER PALETTE num BANK bank,offset     ;num = 0,1 select palette table, BANK
;Update single palette value
LAYER PALETTE num, index, val          ;Write a val at index into layer palette num
```

Sprites

Loading

```
BANK NEW bankId
LOAD "spaceship.spr" BANK bankId
SPRITE CLEAR           ;Clear out any old data
SPRITE BANK bankId     ;Pass in the sprite patterns
```

Display

```
SPRITE PRINT 1         ;1-Enable, 0-Disable sprites
SPRITE BORDER 1        ;Display over border, 0 behind

SPRITE s,x,y,p,f,rf,mx,my   ;Display sprite, s - sprite id, p - pattern, f - flags, rf - relative flags,
mx/my scaling 0(1x),1(2x),2(4x),3(8x)
;Flags
;Bit 0 - 0 invisible, 1 visible
;Bit 1 - 1 rotate clockwise by 90
;Bit 2 - Vertical mirror
;Bit 3 - Horizontal mirror

@1101 - Rotate 180, mirror X+Y
@1111 - Rotate 270 clockwise

;Relative Flags
;Bit 0 - 0 Composite, 1 Relative (Anchor sprite only)
;Bit 1 - Pattern is relative to the acnhor
;Bit 2 - Palette offset is relative to the acnhor
```

Animation

Smoothly animate and move sprites by batching updates and sync'ing with display redraw

```
;Animate by changing pattern number
;However updating sprites immediately gives flicker and tear
SPRITE 0,152,119,0,1
SPRITE 0,152,119,1,1

;Turn off immediately displaying sprite updates from SPRITE cmd
;Now in batch mode
SPRITE STOP

;Update Sprite
SPRITE 0,%x,%y,0,1     ;Update x and y
;Display sprite updates but wait for 50Hz interrupt
;This makes sprite updates smoother
SPRITE MOVE INT y      ;y optional scanline

;To disable batch mode
SPRITE RUN
```

Automatic Movement

Instead of updating the sprite position, specify co-ords to move between. Here the sprite will move x 0 to 200, then 200 to 0 forever

```
;Parameters are optional, can skip ,,
SPRITE CONTINUE
  s,                          ;must specify sprite ID
  x1 TO x2 STEP xs RUN,       ;x1 min value, x2 max value (x1<x2)
  y1 TO y2 STEP ys STOP,
  p1 to p2,                   ;Pattern animation
  f,                          ;Flags see below
  r,                          ;Rate: 0 every SPRITE MOVE, 1 every other, 2 skip two ...255
  d                           ;Delay: 0 - none, 1 skip 1 SPRITE MOVE, 2 skip 2 ...255

Flags Bits
1:0   00 reflect, 01 stop this direction start other direction,10(2) - Stop this direction, 11(3) stop and
invisibile
2     Flip Y mirror bit when Y limits reached
3     Flip X mirror bit when X limits reached
4     Pattern, 0 - wrap to lower limit, 1 - bounce between limits
5     1 - disable when reaches limits
6     1 - update pattern even when stationary
7     1 - Set mirror bits according to direction of travel

;Clearer to split the parameters over several calls instead of one huge command
SPRITE CONTINUE 0,0 TO 200 STEP 1 RUN.       ;Set up X
SPRITE CONTINUE 0,,0 TO 192 STEP 1 STOP      ;Set up Y
SPRITE CONTINUE 0,,,,@0001                   ;Set up Flags, follow a rectangular route
SPRITE MOVE INT                              ;Repeatedly call this

SPRITE PAUSE s1 [TO s2]
SPRITE CONTINUE s1 [TO S2]
```

Status

```
%f = SPRITE CONTINUE s                      ;bit 0: auto enabled, bit 1: moving y axis, bit 2: moving x
axis
% SPRITE s                                    ;1 sprite is visible, 0 invisible
%v = SPRITE AT(s,n)                         ; n = 0: x co-ord, 1: y co-ord, 2: pattern, 3: x step, 4: y
step, 5: delay

;If the sprite is smaller than its bounding box 16x16, use overlap, 0-7
;Returns 0 - no collision, or sprite ID if there is a collision
SPRITE OVER (s1, s2 [TO s3] [,xOverlap, yOverlap])
```

Files

Drives c: boot, m: ramdisk, t: tape

SAVE & LOAD

```
LOAD"t:"                  ;Load from tape

SAVE "m:"                 ;Doesn't save anything but selects drive m, RAM disk
SAVE "test.scr" SCREEN$   ;Save layer 0 pixels+attr
LOAD "test.scr" SCREEN$

SAVE "test.scr" LAYER     ;Saves current layer data (but not palette table)
SAVE "test.scr" LAYER 0   ;Same as SCREEN$
LOAD "test.scr" LAYER     ;Loads data into current layer

DIM a(100)
SAVE "array.arr" DATA a()  ;Save array
SAVE "int.arr" INT         ;Save all integer array values

LOAD "array.arr" DATA x()  ;Loads array, no need to DIM beforehand
LOAD "int.arr" INT         ;Load all integer array values
```

```
LOAD "test.scr" BANK 5     ;Layer 0 is located bank 5
LOAD "test.scr" BANK 5, 0, 6144
LOAD "test.scr" CODE 16384, 6144

MERGE "utils.bas"          ;Will overwrite if same line number
VERIFY "program.bas"       ;Verify tape program
```

Commands

Recommend use command line with more columns

```
CAT                        ;Display contents of directory (also DIR and LS)
CAT EXP                    ;Filename, flags, date (last updated), size
CAT "c:/home/*.bas"
CAT TAB                    ;List storage devices
CAT ASN                    ;Show drive assignments
.ls                        ;.ls --help for more options
.lstap san*.tap            ;Info about a TAP file

mkdir "tmp"                ;Also .mkdir
rmdir "tmp"
cd "tmp"
.cd ..                     ;.cd doesn't require quotes
pwd                        ;Print working directory

COPY "c:/*.bas" TO "m:"
MOVE "screen.sl2" TO "title.sl2"
ERASE "tmp.bas"
```

Attributes

```
MOVE "nexions.bas" TO "+p"     ;Write protect
MOVE "nexions.bas" TO "-p"     ;Remove write protect
MOVE "system.bin" TO "+s"      ;Hide file in CAT listing, will show in CAT EXP
.chmod --help                  ;Similar functionality, see help
```

Channels and Streams

0-15 Channels, 0-3 are used by the system, 0,1 Input, 2 screen, 3 printer

```
;Write a file
OPEN #4,"i>/home/tmp.txt"    ;u> Appends to the file
PRINT #4;"Hello World"
CLOSE #4

;Read a file
OPEN #4,"o>/home/tmp.txt"
DIM #4 TO %s                 ;Size of file
FOR %i=0 TO %s-1
  NEXT #4 TO %x              ;Read byte, also %x=NEXT #4
  PRINT CHR$(x);
NEXT %i
CLOSE #4

;Quick version of the above
COPY "/home/tmp.txt" TO #2   ;Outputs the file to the screen

;Change Position
GOTO #n, pos
```

Best practice is to close a stream before opening and use ON ERROR to close as well

Other Streams

Here we temporarily redirect screen output to a string var

```
DIM t$(100)
OPEN #2,"v>t$"
.time
CLOSE #2                    ;Hand channel back to the system

;Send text file to string var
COPY "/home/tmp.txt" TO #4
```

Memory Streams

```
CLEAR 29999
OPEN #8,"m>30000,1000"
COPY "/home/tmp.txt" TO #8
```

Ports

Display mouse X,Y,Wheel+Buttons

```
PRINT AT 0,0;IN 64479;",";IN 65503;",";IN 64223;"     "
```

IN OUT

```
IN 254                      ;Read port 254
OUT 254,$d4                 ;BEEPER on/off bit
```

Next Registers

```
REG 20                      ;Returns global transparency colour (default 227)
REG 20,42                   ;Set GTC

;9275 - NextReg Select
;9531 - NextReg Value
OUT 9275,20: OUT 9531,42    ;Same as REG 20,42
```

Memory

```
POKE USR "a",0,1,2,3,4,5,6,7   ;UDG A

POKE $4000,"Hello World",13    ;Poke takes multiple params: address, val1...valn
a$ = PEEK$($4000,~13)          ;Read string until terminator 13 (CR)

DPOKE $4000, $FF02             ;Pokes 02 at $4000, then $FF at $4001

;Memory above RAMTOP is protected from being cleared,
;eg safely store data/machine code above RAMTOP
```

```
CLEAR 30000                    ;Clear + Sets RAMTOP to 30000
```

Banks

Next BASIC using 16K banks, whilst NextZXOS uses 8K.

```
BANK NEW b                     ;Reserve an available bank, b
BANK b CLEAR                   ;Release bank b

BANK 100 COPY TO 90            ;Copy contents of 1 bank to another
BANK src_bank COPY [,src_off,len] TO dest_bank[,dest_off]

BANK 100 ERASE                 ;Fill bank 100 with 0's
BANK n ERASE [offset,len][,][value]
```

Banking Code

From the prompt/command line (NOT CODE)

```
BANK n LINE x,y                ;Copy lines x to y to bank n
BANK n LIST                    ;List code in bank
BANK n LIST PROC name()        ;List code in proc
SAVE "name.bnk" BANK n         ;Save the code in the bank
```

From code

```
BANK NEW %b
LOAD "utils.bnk" BANK %b         ;Load code into bank
BANK %b PROC timer()
BANK %b GO TO 500
BANK %b RESTORE
BANK %b CLEAR                    ;Release bank when finished
```

Links

- Home / Forum (https://www.specnext.com/)
- GeTiT (https://zxnext.uk/)
- Wiki (https://wiki.specnext.dev/)
- Facebook Group (https://www.facebook.com/groups/specnext)