NextBASIC additional changes (Updated 10 Oct 2017)

This document describes planned new commands and features for NextBASIC. Many of these are largely complete already, but some are still to be implemented; the features as described will be available for the official January launch (as NextOS v2.00). It is in addition to the previous document which mostly concerned partition and file commands.

Updates: 10 Oct 2017

There will no longer be a restriction on the address for user-defined character sets.

Clarified that changing character set size also causes the window print position to be moved to the start of the next line.

Clarified that window save/load is costly in terms of memory.

Clarified that all commands using the standard "s" channel (not just **PRINT**) will operate in the currently-selected layer/mode.

Window control code 26 is now "auto-pause" instead of "fill with byte".

Added new integer expressions section.

Added new token codes for >> and <<.

The following new command (not in the main text) will be added:

BANK *n* **CLEAR**
Marks bank *n* as free for use by other parts of the system (eg dot commands). Banks are marked as used by BASIC by commands that modify them (eg **BANK..POKE/COPY/ERASE/USR** and **LOAD..BANK**). All marked banks are released by a **NEW** command.

Updates: 5 Oct 2017

Updated text to clarify some details of how different attributes are handled in different modes.

The following new commands (not in the main text) will be added:

LAYER DIM *x1*,*y1*,*x2*,*y2*
Sets the clip window for the current layer from (*x1*,*y1*) to (*x2*,*y2*). Areas of the layer outside this window are not visible.

SPRITE DIM *x1*,*y1*,*x2*,*y2*
Sets the clip window for sprites from (*x1*,*y1*) to (*x2*,*y2*). Any part of a sprite outside this window is not visible.

POINT *x*,*y*,*var*
Checks the pixel on the current layer at (*x*,*y*) and stores the value in variable *var*.
The value will be 0 or 1 for standard Spectrum modes and Timex hi-res and hi-colour modes (pixel off or on). The value will be 0-255 for lo-res and layer 2 (actual pixel colour).

Initial version as at 3 Oct 2017

Deprecated commands

As well as those previously noted, the following command is no longer available:
COPY *f$* **TO SPECTRUM FORMAT**

This is no longer necessary, since it is now possible to load headerless files as **CODE** files.

New keyword tokens

The following keyword tokens are defined in addition to SPECTRUM, PLAY and all the 48K BASIC tokens:

<<	$98
>>	$99
BANK	$9a
TILE	$9b
LAYER	$9c
PALETTE	$9d
SPRITE	$9e
PWD	$9f
CD	$a0
MKDIR	$a1
RMDIR	$a2

New errors

Invalid mode
Direct command only

New commands (**PALETTE, BANK, SPRITE, LAYER, TILE**) issue the following errors:

4 Out of memory
5 Out of screen
A Invalid argument
B Integer out of range
K Invalid colour
M RAMTOP no good
Invalid mode
Direct command only

Extended commands

The standard **LOAD**, **SAVE** and **VERIFY** commands (which work with tape as well as SD/RAMdisk files) now also have the following new variants (all with **LOAD**, **SAVE** or **VERIFY**; only **LOAD** shown):

LOAD *f$* **BANK** *n*
load/save/verify 16K of data in memory bank *n*

LOAD *f$* **BANK** *n,offset,len*
load/save/verify *len* bytes of data starting at offset *offset* in bank *n*

LOAD *f$* **LAYER**
load/save/verify screen for the active layer (like **SCREEN$** but may use 6.75K, 12K or 48K of data depending upon active layer)

Memory bank access

The *Next* comes with between 1MB and 2MB of RAM, divided into 16K banks. These are numbered as follows under *NextOS*:

```
0..7        Same as the standard RAM banks on all 128K Spectrums.
8..47       Additional RAM banks available on 1MB Nexts.
48..111     Further additional RAM banks available on 2MB Nexts.
```

(the remaining 256K is used for ROMs and the DivMMC interface, and is unavailable to users).

Under *NextOS* the memory capacity is shown in the on-screen menus. It can also be queried programmatically by examining the new system variable, *MAXBNK*, which contains the number of the highest usable bank in the system (normally 47 or 111).

NextOS uses the first 9 RAM banks as follows:

```
0           Standard 48K Spectrum memory (at 49152-65535)
1           RAMdisk
2           Standard 48K Spectrum memory (at 32768-49151)
3           RAMdisk
4           RAMdisk
5           Standard 48K Spectrum memory (at 16384-32767)
6           RAMdisk
7           Used for workspace and data structures by NextOS
8           Used for additional screen data (in lo-res, Timex hi-res and Timex
            hi-colour modes) and other data by NextOS
```

Banks 9+ are always available to the programmer, and can be accessed using the new **BANK** command (and extended **LOAD/SAVE/VERIFY..BANK..** commands seen previously).

Generally banks 0..8 cannot be used in **BANK** commands.
Bank 0 can be used, but only if **CLEAR** has first been used to set *RAMTOP* to below 49152.
Bank 2 can be used, but only if **CLEAR** has first been used to set *RAMTOP* to below 32768.
Banks 1,3,4,6 can be used if the **BANK 1346 USR** command has been executed.
Banks 5,7,8 can never be used.

The following new commands are available to manipulate banks:

BANK *n* **POKE** *offset*,*value*
POKE a byte *value* at offset *offset* (0-16383) in bank *n*

BANK *n* **PEEK** *offset*,*var*
PEEK a byte at offset *offset* (0-16383) in bank *n*, and store the value in numeric variable *var*

BANK *n* **COPY TO** *n2*
Copy all 16K from bank *n* to bank *n2*

BANK *n* **COPY** *offset*,*len* **TO** *n2*,*offset2*
Copy *len* bytes starting at offset *offset* in bank *n* to offset *offset2* in bank *n2*

BANK *n* **ERASE**
BANK *n* **ERASE** *value*
Fill all 16K of bank *n* with *value* (zero is used if *value* not specified)

BANK *n* **ERASE** *offset*,*len*
BANK *n* **ERASE** *offset*,*len*,*value*
Fill *len* bytes at offset *offset* in bank *n* with *value* (zero is used if *value* not specified)

BANK 1346 USR
Allow banks 1,3,4,6 to be used in the **BANK** command. This will delete all files on the RAMdisk and unmap it from any drive it is currently mapped to (usually M:).

BANK 1346 FORMAT
Release banks 1,3,4,6 for use by the RAMdisk again. (The RAMdisk will need to be mapped back to a drive using the **MOVE..IN** command).

Palette manipulation

The *Next* provides 6 palettes: 2 palettes each (numbered 0 and 1) for sprites, ULA modes, and layer2. All can be manipulated in BASIC. Note that the Editor will use ULA palette 1 so it is safe to muck around with palette 0 without risk of being unable to see what's going on (the current palette will be restored by the Editor when BASIC is running).

The following new palette manipulation commands are available:

PALETTE DIM *n*
Palettes being specified in the **LAYER PALETTE BANK** and **SPRITE PALETTE BANK** commands use *n* bits per colour (n=8 or 9), ie 256 bytes or 512 bytes (default value is *n*=9).

PALETTE FORMAT *n*
Enable the ULANext extended palette with *n* INKs (1,3,7,15,31,63,127 or 255)
When the ULANext extended palette is enabled, BRIGHT and FLASH are not allowed (in standard and Timex hi-colour modes), and INK and PAPER accept the appropriate new range of values.
If *n*=0, disables the ULANext extended palette and uses standard attributes with 8 inks, 8 papers, bright and flash.

PALETTE OVER *n*
Sets the global transparency colour to *n* (default value is 227)

PALETTE CLEAR
Resets all palettes and related settings to defaults. This is also done by **NEW**.

Sprites

The *Next* provides 64 sprites (size 16x16 pixels). These can be manipulated with the following new commands:

SPRITE BANK *b*
Defines all 64 sprite patterns using the 16K of data (256 bytes per sprite) in bank *b*.

SPRITE BANK *b*,*offset*,*p*,*n*
Defines *n* sprite patterns starting with pattern *p*. Pattern data begins at offset *offset* in bank *b*.

SPRITE PALETTE *n*
Switch to using sprite palette *n* (0 or 1)

SPRITE PALETTE *n* **BANK** *b*,*offset*
Set sprite palette *n* from bank *b*, at offset *offset*. Either 256 bytes or 512 bytes of data is used, depending upon the **PALETTE DIM** setting.

SPRITE PALETTE *n*,*i*,*v*
Set sprite palette *n*, index *i* to value *v*
NOTE: *v* is always specified as a 9-bit value RRRGGGBBB (0-511) regardless of the **PALETTE DIM** setting, and can be conveniently specified using the standard Spectrum **BIN** function.

SPRITE PRINT *n*
Enable (*n*=1) or disable (*n*=0) sprites

SPRITE BORDER *n*
Enable (*n*=1) or disable (*n*=0) sprites over the border

SPRITE *s*,*x*,*y*,*i*,*f*
Set sprite *s* to image *i*, position (*x*,*y*) with flags *f*, which is a bitmask:

```
bit 0=visible flag
bit 1=rotate flag
bit 2=Y-mirror flag
bit 3=X-mirror flag
bits 4..7=palette offset (or zero)
```

Again the **BIN** function can be used to specify this more conveniently.

SPRITE CLEAR
Resets the sprite attributes and global settings to defaults. This is also done by **NEW**.

Layers and modes

The *Next* provides various new graphics modes, to which *NextBASIC* gives access using the **LAYER** command.

There are conceptually 3 layers of graphics which can be seen on the screen at the same time (the 3 layers can be placed in any front-to-back order). The top layer is usually the sprites, which are manipulated with the **SPRITE** command. The other 2 layers are manipulated by the **LAYER** command. The "bottom" of these two layers can only be seen where the "top" layer has the transparency colour (227, or bright magenta).

Layer 1 is the ULA screen, and by default is the bottom layer.
This can be in any of 4 different modes:

- mode 0: lo-res mode (128x96 pixels, each can be any of 256 colours)
- mode 1: standard Spectrum screen mode (256x192 pixels, with 32x24 attributes)
- mode 2: Timex hi-res mode (512x192 pixels, monochrome but with 8 different selectable global ink/paper combinations)
- mode 3: Timex hi-colour mode (256x192 pixels, with 32x192 attributes)

Layer 2 is 256x192 pixels, each can be any of 256 colours. By default it is the top layer but disabled, so does not usually obscure the layer 1 screen.

The **LAYER** command allows either layer to be selected (and for layer 1, any of the 4 available modes to be selected). After the **LAYER** command takes effect, all of the following standard Spectrum commands take place on the selected layer/mode (until another **LAYER** command is issued):

- **INK**, **PAPER**, **BRIGHT**, **FLASH**, **OVER**, **INVERSE**
- **CLS**
- **PLOT**, **DRAW**, **CIRCLE**
- **PRINT**, **LIST**, **CAT** etc (through the standard "s" channel, usually on stream 2)

NOTE: The **ATTR**, **POINT** and **SCREEN$** functions do not take account of the layer/mode settings, and only refer to the standard Spectrum screen.

BRIGHT and **FLASH** are only valid in standard and hi-colour modes (and only when the ULANext extended palette is not enabled).
INK and **PAPER** values can range from 0..255 in lo-res and layer 2.

The **LAYER** command also allows you to select layer 0. This is the default layer/mode when *NextOS* starts and is identical to the standard Spectrum screen mode used on 48K/128K Spectrums. This is the mode you should select in order to load and run standard Spectrum software.

You can switch back and forth between layer 1 and layer 2 without affecting what is on the screen (as long as you always select the same layer 1 mode each time). This allows BASIC programs to enable and manipulate both layer 1 and layer 2 screens, and use transparent areas so that both can be seen together.

The following **LAYER** commands are available:

LAYER 0
Select layer 0, standard Spectrum mode

LAYER 1,0
Select lo-res mode

LAYER 1,1
Select standard resolution mode

LAYER 1,2

Select Timex hi-res mode

LAYER 1,3
Select Timex hi-colour mode

LAYER 2
Select layer2

LAYER 2,0
Select layer2, and disable displaying it

LAYER 2,1
Select layer2, and enable displaying it

LAYER PALETTE *n*
Switch to using palette *n* (0 or 1) for the current layer

LAYER PALETTE *n* **BANK** *b*,*offset*
Set palette *n* for the current layer from bank *b*, at offset *offset*. Either 256 bytes or 512 bytes of data is used, depending upon the **PALETTE DIM** setting.

LAYER PALETTE *n*,*i*,*v*
Set palette *n* for the current layer, index *i* to value *v*
NOTE: *v* is always specified as a 9-bit value RRRGGGBBB (0-511) regardless of the **PALETTE DIM** setting, and can be conveniently specified using the standard Spectrum **BIN** function.

LAYER AT *x*,*y*
(Layer 2 or lo-res only).
Set the display offset for the top-left of the screen for the current layer to *x*,*y*. This is used for scrolling effects.

LAYER OVER *n*
Set sprite/layer SLU ordering:

n=**BIN** 000	sprites over layer2 over ULA (layer1) - the default
n=**BIN** 001	layer2 over sprites over ULA (layer1)
n=**BIN** 010	sprites over ULA (layer1) over layer2
n=**BIN** 011	layer2 over ULA (layer1) over sprites
n=**BIN** 100	ULA (layer1) over sprites over layer2
n=**BIN** 101	ULA (layer1) over layer2 over sprites

LAYER BANK *n*,*m*
(Layer 2 only). Set current banks *n*..*n*+2 as frontbuffer (to be displayed) and banks *m*..*m*+2 as backbuffer (for rendering). These values can be the same and both default to 9.

LAYER ERASE *x*,*y*,*w*,*h*
LAYER ERASE *x*,*y*,*w*,*h*,f
(Layer 2 or lo-res only).
Fill region width *w* pixels, height *h* pixels, top-left corner *x*,*y* with value *f*. If *f* is not specifed, the current global transparency value (usually 227) is used.

LAYER CLEAR
Reset all layer information to defaults. This is also done by **NEW**.
Resets banks, mode, layer2 enable, layer offsets, layer ordering.

Differences between layer 0 and layer 1 mode 1.
Layer 0 behaves in exactly the same way as the screen always has on 48K and 128K Spectrums. Layer 1 mode 1 has the same resolution and attributes, but behaves in a slightly different manner under *NextBASIC*. It shares this same behaviour with all other layer 1 modes.

In layer 0, the standard Spectrum memory map is in force (ROM, RAM 5, RAM 2, RAM 0). However, in all layer 1 modes, the top 8K of RAM 5 is replaced with 8K from the *NextOS* RAM 8 bank. This is done so that BASIC still has access to the same amount of memory as usual (~41K); without this change, it would lose about 6K to the new screen modes.

The other main differences are:

Layer 0 pixel coordinates (used by **PLOT**, **DRAW**, **CIRCLE**) run from (0,0) at the bottom left on the main screen area to (255,175) at the top right. The bottom two screen lines are not normally accessible to these commands. However, in layer 1 modes, pixel coordinates run from (0,0) at the top left of the screen to (255,191) at the bottom right (511,191 in hi-res mode, 127,95 in lo-res mode).

Layer 0 **PRINT** coordinates (on channel "s") are in character squares, defined as (0,0) at the top left and (21,31) at the bottom right (again, the lower screen is not usually accessible).

Layer 1 modes all use a full-screen system-defined text window for any **PRINT**s directed to channel "s". Therefore they generally use the same control codes as other text windows (except justify and save/load are not available), and **AT** coordinates are defined using a pixel line number and a character position (the number of positions depending upon the character set size selected).

Layer 1 modes do not support "9" to mean contrast (for **PAPER/INK**) or "8" to mean transparent (for **PAPER/INK/BRIGHT/FLASH**).

Timex Hi-Res colour scheme

The colour scheme for hi-res mode is selected using **INK** (either as a direct command or by **PRINT**ing to the hi-res screen or a window). This will immediately change the whole colour scheme. The colour schemes available (can be altered using ULANext palettes) are:

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

Tiling commands

For layer 2 and lo-res modes, there are new commands available to draw complete screens (or sections of a screen) from a set of tiles and a tilemap.

Tiles are either 8x8 pixels in size or 16x16 pixels in size. This allows a 16K bank to hold 256 8x8 tiles or 64 16x16 tiles. Tiles are numbered 0..255. Therefore, a complete set of 8x8 tiles occupies a single 16K bank, and a complete set of 16x16 tiles occupies 4 16K banks. If you use 16x16 tiles, you can restrict the tile numbers used and therefore reduce the memory requirements (eg if you need 64 or fewer different tiles, only 1 16K bank is required).

A tilemap is a linear map of 8-bit tile numbers. The user can specify any width up to 2048 tiles; each row of tiles follows directly after the previous one. The tilemap must be fully contained in a single 16K bank. This gives a maximum tilemap size of 256x64, 128x128, 2048x8 etc.

Information on layer 2 and lo-res tilemaps is stored separately, so you can use both. The **TILE** commands affect the currently selected layer/mode. They are:

TILE BANK *n*
Define bank *n* as containing the tiles (up to 4 banks *n*..*n*+3 if 16x16 tiles)

TILE DIM *n*,*offset*,*w*,*tilesize*
Define bank *n* as containing the tilemap, starting at offset *offset* in the bank. The tilemap is width *w* (1-2048) and uses 8x8 (*tilesize*=8) or 16x16 (*tilesize*=16) tiles.

TILE
TILE AT *x*,*y*
Draw entire screen from tilemap, from tile offset *x*,*y* in the tilemap (0,0 if not specified).

TILE *w*,*h*
TILE *w*,*h* **AT** *x*,*y*
TILE *w*,*h* **TO** *x2*,*y2*
TILE *w*,*h* **AT** *x*,*y* **TO** *x2*,*y2*
Draw section of screen from tilemap.
Number of tiles to draw is width *w*, height *h*.
Draw from tile offset *x*,*y* in the tilemap (or 0,0 if not specified).
Draw to tile offset *x2*,*y2* on the screen (or 0,0 if not specified).

Text window changes

There are some changes to the text window channels from those used in the +3e.

As noted earlier, there are 4 system-maintained full-screen windows which are used for all **PRINT**ing through the standard "s" channel when one of the layer 1 modes is selected, and most of the changes were made to accommodate this.

Windows can only be used in the same layer/mode that was active when they were defined. Control codes not listed here behave in exactly the same was as on +3e v1.43.

Control code	Differences
0	On user-defined windows, turns justification off (as +3e) On system windows, increases the current character set width (can range from 3 to 8 pixels), and moves the cursor to the start of the next line.
1	On user-defined windows, turns justification on (as +3e) On system windows, decreases the current character set width (can range from 3 to 8 pixels), and moves the cursor to the start of the next line.
2	On user-defined windows, saves window contents (as +3e) On system windows, causes the size 8 character set to be replaced with the character set defined by the CHARS system variable.
3	On user-defined windows, restores window contents (as +3e) On system windows, causes the size 3..7 character sets to be regenerated
15	Wash window. This does nothing on layer 2 or lo-res windows.
17,n	PAPER n. Not allowed in Timex hi-res mode.
18,n	FLASH n. Only allowed in standard or Timex hi-colour modes, and only if ULANext is not enabled.
19,n	BRIGHT n. Only allowed in standard or Timex hi-colour modes, and only if ULANext is not enabled.
24,n	ATTR n. Not allowed in lo-res and layer2 modes. In Timex hi-res mode it does the same as INK n (16,n).
25,n	Kern adjust. This moves the position within the window left by *n* pixels, and can be used for primitive kerning. Previously this control code turned on or off extended UDGs for codes 165-255 instead of keyword tokens. Under *NextOS* extended UDGs are always used (LIST will expand keywords so keyword token codes will not normally be seen by windows anyway)
26,n	Auto-pause every n pixel lines. After each n pixel lines have been scrolled out of the window, output will automatically pause until the SPACE key is pressed (the bottom right char in the window will be inverted to indicate SPACE is being waited for). If set to zero (the default), auto-pause is disabled.
30,n	On user-defined windows, selects justification mode 0, 1 or 2 (as +3e). On system windows, changes the current character set width to

n (can be 3,4,5,6,7 or 8 pixels), and moves the cursor to the start of the next line.

31,n — On user-defined windows, selects whether embedded codes are permitted in justify mode (as +3e).
On system windows, causes the size *n* character set to be replaced with the character set defined by the CHARS system variable.

User character sets

If the default character set(s) are replaced using control codes 2, 3 or 31 in a system window, any subsequent text printed in any window (which doesn't have its own user-defined character set) will use the new character set(s).

The system-defined character sets are partially shared: sizes 3 and 4 use the same set (only the leftmost 3 pixels are used for size 3), and similarly so do sizes 5 and 6. This should be borne in mind when replacing system character sets using control code 31.

Window input

Text windows now support the **INPUT** command, as in *ResiDOS*. If you use **INPUT #**, then a cursor is added to the window at the current position. The user can then input any text desired, using the left and right arrows to move along the text input so far, or the up and down arrows to move to the start or end of the text. The DELETE key deletes the character to the left of the cursor, and the ENTER key completes the input. Depending upon memory available, the entire size of the window can be used in the input, although care is taken to ensure that no input character is ever scrolled off the top of the window. An absolute maximum of 255 characters is allowed in the input line.

Window definitions

Windows are still defined using character squares as before. In lo-res mode, this means the maximum window size is 16x12 (not 32x24). In hi-res mode, characters are considered to be 16 pixels wide, so the maximum window size is still 32x24 for this mode.

Memory constraints

It should be noted that saving/loading window contents (only available on user-defined windows) is a costly operation. The amount of memory required for each character square is:

- 9 bytes (standard resolution mode)
- 16 bytes (Timex hi-res or hi-colour modes)
- 64 bytes (lo-res or layer2 modes)

For example, a 10x10 window in layer2 would require 6400 bytes of available memory for saving the contents.

BASIC Program Extensions

It is now possible to write BASIC programs larger than the usual ~41K with a little extra effort. Sections of BASIC programs can be copied into any memory bank available to the user (and saved/loaded with the **SAVE/LOAD..BANK** commands), and the program can then switch between lines in the "main" program area and a bank.

The following new commands are available to manage banked sections of BASIC programs:

BANK *n* **LINE** *x*,*y*
Copies lines *x* to *y* inclusive from the main program to bank *n*. The total number of bytes used in the bank will be shown.
Once this has been done it is not possible to change or delete any lines in the banked section (except by completely overwriting the bank's contents using another **BANK...LINE** command).

BANK *n* **LIST**
BANK *n* **LIST** *l*
List lines (optionally from *l*) in bank *n*

BANK *n* **MERGE**
Copy all lines back from bank *n* into the main program

BANK *n* **GOTO** *l*
GOTO line *l* in bank *n*. To GOTO the main program from a banked section, use *n*=-1.

BANK *n* **GOSUB** *l*
GOSUB line *l* in bank *n*. To GOSUB the main program from a banked section, use *n*=-1.

BANK *n* **RESTORE** *l*
Set the DATA pointer to line *l* in bank *n*

Notes

Any **GOTO** or **GOSUB** within a banked section will go to a line in the same bank.

Any **RETURN** will always return to the calling bank.

DEF FN statements must be in the main program; they will not be searched for in banked sections.

Lines in banks can have the same numbers as main program lines.

Renumbers won't affect or take into account lines in banked sections.

Commands that affect program lines can only be used as direct commands, and not be part of a program. These are:
ERASE *first*,*last*
LINE *first*,*last* **TO** *start*,*step*
BANK *n* **LINE** *x*,*y*
BANK *n* **MERGE**

Integer variables and expressions

For additional speed and memory efficiency, NextBASIC provides a new integer expression evaluator. All integer values are treated as unsigned 16-bit values, and all operations are performed modulo 65535, with no checks for overflow/underflow (except division by zero, which results in error 6, Number too big).

A fixed set of integer variables are provided: the user cannot define additional variables. The two main advantages of a fixed set of variables are:

- speed of access (all integer variables are at a known location)
- memory usage - the integer variables are stored in some of the RAM 8 bank reserved by NextOS, and hence do not use any space in the normal BASIC/variables area

All integer variables are erased to zero at **RUN**, **CLEAR** and **NEW**. Note, however, that integer variables are not saved/loaded along with BASIC programs, as is the case with normal floating-point and string variables.

There are 26 integer variables provided, named **A** to **Z** (can also be referred to in lower-case, **a** to **z**). There are also 26 integer variable arrays provided, named **A()** to **Z()** (or **a()** to **z()**), each containing 64 elements, numbered 0 to 63.

Note that array elements are numbered from 0, not 1 as in normal floating-point/string arrays. Also note that integer array element **A(0)** is *not* the same as integer variable **A**.

The following unary operators may precede any integer value or expression:

+	unary plus
-	unary minus
!	bitwise not

Literal numbers can be specified in decimal (the default), hexadecimal (preceded by the **$** symbol) or binary (preceded by the **@** symbol), eg:

```
32767
$ed01
@11100010
$FF
```

The following binary operators are available:

+	add
-	subtract
*****	multiply
/	divide
%	modulus (remainder)
<<	shift left
>>	shift right
&	bitwise AND
|	bitwise OR
^	bitwise XOR
<	less than
>	greater than
=	equal to
<=	less than or equal to
>=	greater than or equal to
<>	not equal to

The six relational operators always produce a result of $0000 for false and $ffff for true.

Operations are performed in strictly left-to-right order, unless overridden by the use of parentheses.

An integer expression can be used in any BASIC line where a numeric expression

is normally expected. To indicate an integer expression instead of a floating point expression, a **%** symbol must always precede an integer expression (further % symbols within the integer expression are treated as the modulus operator).

Similarly, integer variables can be used in assignments (such as **LET**) by preceding their name with a %.

It is *not* possible to access standard numeric variables or functions within an integer expression, or to access integer variables or operations within a standard numeric expression.

It is possible to assign an integer expression to a standard normal numeric variable, or vice-versa, and the value will be converted appropriately. For example, all the following assignments are valid:

LET %A=2*PI*radius
assigns truncated floating point calculation to integer variable A

LET %B=%B+(A(7)<<3)
shifts integer array element A(7) left 3 bits and adds to integer variable B

LET addr=%x(1)<<8+x(0)
calculates standard numeric variable *addr* from low and high bytes in integer array X elements 0 and 1

Note that **DEF FN** does not support user-defined integer functions.

System variable changes

The following system variables have been changed (same format as +3 manual):

X1	5B68H (23400)	FLAGN	Flags for the NextOS system (was XLOC)
1	5B69H (23401)	MAXBNK	Maximum available RAM bank (was YLOC)
1	5B73H (23411)	TILEBNKL	Tiles bank for lo-res (was RC LINE)
1	5B74H (23412)	TILEML	Tilemap bank for lo-res (was RC LINE+1)
1	5B75H (23413)	TILEBNK2	Tiles bank for layer2 (was RC START)
1	5B76H (23414)	TILEM2	Tilemap bank for layer2 (was RC START+1)
X1	5B77H (23415)	NXTBNK	Bank containing NXTLIN (was RC STEP)
X1	5B78H (23416)	DATABNK	Bank containing DATADD (was RC STEP+1)
N1	5B7BH (23419)	L2SOFT	Softcopy of layer2 port (was DUMPLF)
X1	5C7FH (23679)	GMODE	Graphical layer/mode flags (was P POSN)

The following system variables have been inserted where STRIP1 and STRIP2 were, within the temporary TSTACK area (STRIP1 and STRIP2 have been moved up to accommodate them). This means that there are now only a guaranteed 91 bytes of TSTACK when calling +3DOS (115 including STRIP1 and STRIP2):

2	5B7CH (23420)	TILEWL	Width of lo-res tilemap
2	5B7EH (23422)	TILEW2	Width of layer2 tilemap
2	5B80H (23424)	TILEOFFL	Offset in bank for lo-res tilemap
2	5B82H (23426)	TILEOFF2	Offset in bank for layer2 tilemap
2	5B84H (23428)	COORDSL	x,y coords of last point plotted (lo-res)
2	5B86H (23430)	COORDS2	x,y coords of last point plotted (layer 2)
2	5B88H (23432)	COORDSULA	x,y coords of last point (standard)
2	5B8AH (23434)	COORDSHR	x,y coords of last point (hi-res)
2	5B8CH (23436)	COORDSHC	x,y coords of last point (hi-colour)
1	5B8EH (23438)	INKL	INK colour for lo-res mode
1	5B8FH (23439)	INK2	INK colour for layer2 mode
1	5B90H (23440)	ATTRULA	Attributes for standard mode
1	5B91H (23441)	INKHR	INK colour for hi-res mode
1	5B92H (23442)	ATTRHC	Attributes for hi-colour mode
1	5B93H (23443)	INKMASK	Softcopy of ULANext inks mask (or 0)
N8	5B94H (23444)	STRIP1	Stripe one bitmap (moved)
N8	5B9CH (23452)	STRIP2	Stripe two bitmap (moved)