Array and Data Storage:

Arrays in Memory

- _DEFINE (statement) defines a range of untyped variable names according to their first character as a datatype.
- _PRESERVE (REDIM option) preserves the existing element values when an array is resized.

- CLEAR (statement) resets all variable values and array element values to 0 or null strings and closes all open files.
- DIM (statement) dimensions(sizes) a static array and defines the type.
- $DYNAMIC (metacommand) defines that all arrays are dynamic or changeable in size.
- ERASE (array statement) clears a static array of all values and totally removes a dynamic array.
- LBOUND (array function) returns the lowest valid index (lower boundary) of an array.
- OPTION BASE (statement) sets the default starting index of an array to 0 or 1.
- REDIM (statement) re-dimensions the number of elements in a dynamic(resizeable) array and defines the type.
- SHARED (statement) designates variable values that can be shared with sub-procedures without using parameters.
- STATIC (statement) defines a variable or list of variables that will retain their values after the sub-procedure is exited.
- $STATIC (metacommand) defines that all arrays are static or unchangeable in size.
- SWAP (statement) trades the values of two numerical or string values or array elements.
- UBOUND (array function) returns the highest valid index (upper boundary) of an array.

See also: Arrays

Fixed Program DATA

- DATA (statement) creates a field of built-in program data values separated by commas.
- READ (statement) reads the DATA from the data field sequentially.
- RESTORE (statement) sets the data pointer to the start of all DATA or a specified DATA field.

- **WARNING! Do not place DATA fields after SUB or FUNCTION procedures! QB64 will FAIL to compile properly!**

Qbasic allowed programmers to add DATA fields anywhere because the IDE separated the main code

from other procedures.

(Return to Table of Contents)

Colors and Transparency:

- _ALPHA (function) returns the alpha channel transparency level of a color value used on a screen page or image.
- _ALPHA32 (function) function returns the alpha channel level of a 32 bit color value only.
- _BACKGROUNDCOLOR (function) returns the current background color.
- _BLEND (statement) turns on alpha blending for the current image or a specific one.
- _BLEND (function) returns if blending is enabled or disabled for the current window or a specified image handle.
- _BLUE (function) returns the palette intensity OR the blue component intensity of a 32-bit image color.
- _BLUE32 (function) returns the blue component intensity of a 32-bit image color.
- _CLEARCOLOR (statement) sets a specific color to be treated as transparent in an image
- _CLEARCOLOR (function) returns the current transparent color of an image.
- _COPYPALETTE (statement) copies the color palette intensities from one image to another image or screen page.
- _DEFAULTCOLOR (function) returns the current default text color for an image handle or page.
- _DONTBLEND (statement) turns off alpha blending for an image handle.
- _GREEN (function) returns the palette index OR the green component intensity of a 32-bit image.
- _GREEN32 (function) returns the green component intensity of a 32-bit image color.
- _NEWIMAGE (function) prepares a custom sized program SCREEN or page surface that can use 256 or 32 bit colors.
- _PALETTECOLOR (statement) sets the color value of a palette entry of an image using 256 color modes or less (4 or 8 BPP).
- _PALETTECOLOR (function) returns the 32 bit attribute color setting of an image or screen page handle's palette.
- _PIXELSIZE (function) returns the color depth (Bits Per Pixel) of an image.
- _RED (function) returns the palette index OR the red component intensity of a 32-bit screen.
- _RED32 (function) returns the red component intensity of a 32-bit image color.
- _RGB (function) returns the closest palette attribute index OR the LONG 32 bit color value in 32 bit screens.
- _RGB32 (function) returns the LONG 32 bit color value only.
- _RGBA (function) returns the closest palette attribute index OR the LONG 32 bit color value in _ALPHA screens.

- _RGBA32 (function) returns the LONG 32 bit _ALPHA color value only.
- _SETALPHA (statement) sets the alpha channel transparency level of some or all of the pixel colors of an image.

- CLS (statement) clears the screen and can set the background color in QB64.
- COLOR (statement) sets the current text color attribute or _RGB value to be used or background colors in some screen modes.
- INP (function) returns the RGB color intensity values from color port register &H3C9 for a specific attributes.
- OUT (statement) sets the color port access mode and sets the RGB color intensity values using &H3C9.
- PALETTE (statement) sets the Red, Green and Blue color attribute intensities using a RGB multiplier calculation.
- PALETTE USING (statement) sets the color intensity settings using a designated array.
- POINT (function) returns a pixel coordinate color attribute or the LONG _RGB color value of a 32 bit color.
- PRESET (statement) sets a pixel coordinate to the background color or a color specified.
- PSET (statement) sets a pixel coordinate a specified color or uses the current color when not designated.
- SCREEN sets the screen mode of a program which may determine the number of colors available in legacy modes.

(Return to Table of Contents)

Console Window

- $CONSOLE (QB64 Metacommand) creates a console window throughout the program.
- _CONSOLE (statement) can be used to turn the console window OFF or ON or designate it as _DEST _CONSOLE for output.
- _CONSOLETITLE (statement) creates a title for the console window using a literal or variable string.

- $SCREENHIDE (QB64 Metacommand) hides the program window throughout the program until $SCREENSHOW is used.
- _SCREENHIDE hides the main program window in a section of code until _SCREENSHOW is used.
- $SCREENSHOW (QB64 Metacommand) displays the main program window throughout the program after $SCREENHIDE has been used.
- _SCREENSHOW displays the main program window in a section of code after

_SCREENHIDE has been used.

- SHELL (QB64 function) executes a DOS command or calls another program. Returns codes sent by END or SYSTEM.
- _SHELLHIDE (function) hides a DOS command or call to another program. Returns codes sent by END or SYSTEM.

See also: Console Window and C++ Console window Library.

(Return to Table of Contents)

Conditional Operations:

- AND (boolean) returns True if all of the arguments are True.
- NOT (boolean) returns the opposite condition of an argument.
- OR (boolean) returns True if one of the arguments is True.
- XOR (boolean) returns True if only one of two arguments are True.

Relational Operations	
Operation	**Description**
a = b	Tests if a is equal to b.
a <> b	Tests if a is not equal to b; equivalent to (NOT (a = b)).
a < b	Tests if a is less than b.
a > b	Tests if a is greater than b.
a <= b	Tests if a is less than or equal to b; equivalent to (NOT (a > b)).
a >= b	Tests if a is greater than or equal to b; equivalent to (NOT (a < b)).

See also: Logical Operations: and Relational Operations

(Return to Table of Contents)

Definitions and Variable Types:

- _BIT {` numerical type) values of 0 (bit off) or -1 (bit on). Unsigned of 0 or 1.
- _BYTE {%% numerical type) values from -128 to 127 (one byte or 8 _BITs). Unsigned from 0 to 255.
- _DEFINE (statement) defines a range of untyped variable names according to their first character as a datatype.

- _FLOAT {## numerical type) values offer the maximum floating-point decimal precision available using QB64.
- _INTEGER64 (&& numerical type) values -9223372036854775808 to 9223372036854775807. Unsigned to 18446744073709551615.
- _MEM (variable type) contains read only dot elements for the OFFSET, SIZE, TYPE and ELEMENTSIZE of a block of memory.
- _OFFSET(%& variable type) can store any memory offset integer value when using DECLARE LIBRARY or _MEMory only.
- _UNSIGNED {~ numerical type) defines an integer numerical value as being positive only in QB64.

- COMMON (statement) shares common variable values with other Linked or CHAINed programs.
- COMMON SHARED (statement) shares common variable values with all sub-procedures and other Linked or CHAINed programs.
- CONST (statement) defines one or more named numeric or string shared values which can not change in a program.
- DEFDBL (statement) defines undefined variable starting letters AS DOUBLE variables instead of the SINGLE type default.
- DEFINT (statement) defines undefined variable starting letters AS INTEGER variables instead of the SINGLE type default.
- DEFLNG (statement) defines undefined variable starting letters AS LONG variables instead of the SINGLE type default.}}
- DEFSNG (statement) defines undefined variable starting letters AS SINGLE variables instead of the SINGLE type default.
- DEFSTR (statement) defines undefined variable starting letters AS STRING variables instead of the SINGLE type default.
- DIM defines a variable or size a static array and can define the type of value it returns.
- DOUBLE {# numerical type) an 8 byte floating decimal variable type with numerical values up to 15 decimal places.
- INTEGER {% numerical type) a two byte variable type with values from -32768 to 32767. Unsigned to 65535.
- LONG {& numerical type) Integer values can be from -2147483648 to 2147483647. _UNSIGNED values to 4294967295.
- OPTION BASE (statement) sets the default starting index of an array to 0 or 1.
- REDIM defines and sizes a dynamic(changeable) array and can define the type of value returned.
- SHARED (statement) designates variable values that can be shared with sub-procedures without

using SUB parameters.

- SINGLE (! numerical type) a 4 byte floating decimal variable type with numerical values up to 7 decimal places.
- STATIC (statement) defines a variable or list of variables that will retain their values after the sub-procedure is exited.
- STRING ($ variable type) one byte text variable with ASCII code values from 0 to 255.
- TYPE (statement) defines variable types that can hold more than one variable type value of a fixed byte length.

See also: QB64 Variable Types and C++ Variable Types

(Return to Table of Contents)

DOS and OS Environment:

- _DEVICE$ (function) returns a STRING expression listing device names and input types of system input devices.
- _DEVICES (function) returns the number of input devices found on a computer system.
- _DIREXISTS (function) returns -1 if the directory folder name string parameter exists. Zero if it does not.
- _CLIPBOARD$ (statement) sends STRING data to the Clipboard.
- _CLIPBOARD$ (function) returns the current contents of the Clipboard as a string.
- _CWD$ (function) returns the current working directory path as a STRING.
- _DONTWAIT (SHELL action) allows the program to continue without waiting for the other program to close.
- _FILEEXISTS (function) returns -1 if the file name string parameter exists. Zero if it does not.
- _HIDE (SHELL action) hides the DOS screen output during a shell.
- _LASTBUTTON (function) returns the number of buttons available on a specified number device listed by DEVICE$.
- _OS$ (function) returns the QB64 compiler version in which the program was compiled as [WINDOWS], [LINUX] or [MACOSX] and [32BIT] or [64BIT].
- _SCREENCLICK simulates clicking the mouse at a position on the screen to get focus.
- _SCREENIMAGE captures the current desktop screen image.
- _SCREENPRINT simulates keyboard entries on the desktop.
- _SHELLHIDE (function) executes a DOS command or calls another program. Returns codes sent by END or SYSTEM.
- _STARTDIR$ (function) returns the user's program calling path as a STRING.

- CHDIR (statement) changes the program path to a new path.
- COMMAND$ (function) returns command line parameters sent when a program is started.
- ENVIRON (statement) sets the environmental settings of the computer. NOT IMPLEMENTED!
- ENVIRON$ (function) returns the environmental settings of the computer.
- FILES (statement) displays a specified list of files.
- MKDIR (statement) creates a new directory folder in the designated path.
- NAME (statement) renames a file.
- RMDIR (statement) removes an empty directory folder from the specified path.
- SHELL (statement) performs a command line operation in DOS.
- SHELL (QB64 function) executes a DOS command or calls another program. Returns codes sent by END or SYSTEM.

See also: Console Window to display DOS commands.

(Return to Table of Contents)

Error Codes

The following table describes the error codes that are reported by the **QB64** compiler, as well as possible causes and solutions:

QB/64 Error Codes			
Code	**Description**	**Common cause/Resolution**	**QB64 Differences**
1	NEXT without FOR	Missing loop end or look for a missing END IF or END SELECT statement.	none
2	Syntax error	Mistyped keyword statements or puctuation errors can create syntax errors.	none
3	RETURN without GOSUB	Place sub-procedure line label after program END or EXIT in a SUB-procedure.	none
4	Out of DATA	A READ has read past the end of DATA. Use RESTORE to reset to data start. .	none
5	Illegal function call	A parameter passed does not match the function type or exceeds certain function limitations. See Illegal Function.	none
6	Overflow	A numerical value has exceeded the limitations of a variable type.	none
7	Out of memory	A module has exceeded the 64K memory limitation of QB. Try breaking the code up to smaller modules.	no limit
8	Label not defined	GOTO or GOSUB tries to branch to a label that doesn't exist.	none

9	Subscript out of range	An array's upper or lower dimensioned boundary has been exceeded.	none
10	Duplicate definition	You can't define a variable twice with DIM, the first time a variable is used it is also defined.	none
11	Division by zero	You cannot divide any value by zero! Even using MOD.	none
12	Illegal in direct mode	A statement (like DIM) in the Immediate window wasn't allowed.	N/A
13	Type mismatch	A SUB or FUNCTION parameter does not match the procedure Declaration.	none
14	Out of string space	A module has exceeded the 32767 text character limit. Use SUB print procedures.	no limit.
16	String formula too complex.	A string formula was too long or INPUT statement requested more than 15 strings	N/A
17	Cannot continue.	The program while debugging has changed in a way that it cannot continue.	no debug
18	Function not defined.	The function used by the program must be defined. Did you include the .bi file while using a library?	none
19	No RESUME.	Use RESUME at the end of the ON ERROR GOTO routine, not RETURN.	none
20	RESUME without error.	RESUME can only be used in an error handler called by ON ERROR GOTO.	none
24	Device timeout.	Use DS0 (DSzero)in the OPEN COM statement to avoid modem timeouts.	none
25	Device fault.	Device not connected or does not exist.	none
26	FOR without NEXT.	Missing loop end or look for a missing END IF or END SELECT statement.	none
27	Out of paper	A printer paper error when using LPRINT.	none
29	WHILE without WEND.	WEND must end a WHILE loop. Otherwise look for missing END IF or END SELECT	none
30	WEND without WHILE	Missing loop end or look for a missing END IF or END SELECT statement.	none
33	Duplicate label	Line numbers or labels cannot be used twice in a procedure.	none
35	Subprogram not defined.	Often occurs when the Quickbasic Library is not used with CALL ABSOLUTE or INTERRUPT or a SUB or FUNCTION procedure has not been created for a CALL.	none
37	Argument-count mismatch	The number of sub-procedure parameters do not match the call.	none
38	Array not defined	Arrays using more than 10 elements must be DIMensioned.	none
40	Variable required.	LEN cannot read literal numerical values. A GET or PUT statement must specify a variable when reading	none

		or writing a file opened in BINARY mode.	
50	FIELD overflow.	A FIELD statement tried to allocate more bytes than were specified for the record length of a random access file.	none
51	Internal error.	An internal malfunction occured in QuickBASIC or QB64.	none
52	Bad file name or number.	The filename must follow the rules for filenames in the OS and use filenumbers from 1 and 255. Use FREEFILE to avoid duplicate OPEN file numbers.	none
53	File not found.	File not in current directory or path. Use _FILEEXISTS to verify file names.	none
54	Bad file mode.	File access mode does not match a current OPEN file procedure.	none
55	File already open.	CLOSE a file to open it in a different mode.	none
56	FIELD statement active.	**WRITEME**	N/A
57	Device I/O error.	**WRITEME**	N/A
58	File already exists.	The filename specified in the NAME statement was identical to a file that already exists.	none
59	Bad record length.	Record length exceeds 32767 bytes or is 0	none
61	Disk full.	The amount of data to write to the disk was more than the free space available, remove some files you don't need and try again.	none
62	Input past end of file.	Check for the end of file with EOF when reading from a file.	none
63	Bad record number.	GET read exceeds number of records in a RANDOM file.	none
64	Bad file name	File name contains illegal characters or exceeds 12 characters.	none
67	Too many files	Over 15 files are open in Qbasic.	none
68	Device unavailable.	Device does not exist, busy or not connected.	none
69	Communication-buffer overflow.	**WRITEME**	N/A
70	Permission denied	A file or port is in use on a network, blocked, read only or locked.	none
71	Disk not ready.	Disk is busy or has no media.	none
72	Disk-media error.	Improper media format or bad data.	none
73	Feature unavailable.	Based on the DOS version available.	none
74	Rename across disks.	**WRITEME**	N/A
75	Path/File access error.	File or path cannot be accessed.	none
76	Path not found.	Path is not access-able or does not exist. Use _DIREXISTS to check paths.	none
97	(no error message)	Can be used to trigger an error trap event with ERROR 97, nothing else will cause this error, so	none

create your own errors for ON ERROR.

N/A means Not Available or Not Applicable to QB64.

QB64 Error Codes			
Code	**Description**	**Common cause/resolution**	**QB Differences**
258	Invalid handle	Zero or bad handle values cannot be used by the QB64 procedure creating the error.	N/A

(Return to Table of Contents)

Error Trapping:

- $CHECKING (Metacommand) turns off or on error event checking and strips error code from compiled programs.
- _ERRORLINE (function) returns the actual text code line where a program error occurred.

- ERDEV (function) returns an error code from the last device to create an error. NOT IMPLEMENTED!
- ERDEV$ (function) returns the string name of the last device to declare an error. NOT IMPLEMENTED!
- ERR (function) returns the error code number of the last error to occur.
- ERROR (statement) simulates a program error based on the error code number used.
- ERL (function) returns the closest line number before an error occurred if the program uses them.

- ON ERROR (statement) GOTO sends the program to a line number or label when an error occurs. Use to avoid program errors.

- RESUME (statement) error statement sends the program to the NEXT code line or a designated line number or label .

See the Error Code Table reference.

(Return to Table of Contents)

Event Trapping:

- _AUTODISPLAY (statement) enables the automatic display of the screen image changes previously disabled by _DISPLAY.
- _DELAY (statement) suspends program execution for a SINGLE value of seconds down to milliseconds.
- _DISPLAY (statement) turns off automatic display while only displaying the screen changes when called.
- _EXIT (function) prevents a user exit and indicates if a user has clicked the close X window button or CTRL + BREAK.
- _FREETIMER (function) returns a free TIMER number for multiple ON TIMER(n) events.
- _MOUSEINPUT (function) reports any changes to the mouse status and MUST be used to read those changes.
- _SHELLHIDE (function) returns the code sent by a program exit using END or SYSTEM followed by an INTEGER value.

- OFF turns event checking off and does not remember subsequent events.
- ON turns event checking on.
- ON COM(n) (event statement) executes when there is a value in the serial port specified. NOT IMPLEMENTED!
- ON ERROR GOTO (event statement) executes when a program error occurs
- ON KEY(n) (event statement) executes when a keypress or keypress combination occurs.
- ON PEN (event statement) executes when a PEN event occurs. NOT IMPLEMENTED!
- ON PLAY(n) (event statement) executes when the background music queue is low. NOT IMPLEMENTED!
- ON STRIG(n) (event statement) executes when a joystick button event occurs. NOT IMPLEMENTED!
- ON TIMER(n) (event statement) executes when a timed event occurs. QB64 can use multiple numbered timers.
- ON UEVENT (event statement) executes when a user event occurs. NOT IMPLEMENTED!
- ON...GOSUB (event statement) branches to a line number or label according to a numerical ordered list of labels.
- ON...GOTO (event statement) branches to a line number or label according to a numerical ordered list of labels.
- STOP suspends event checking and remembers subsequent events that are executed when events are turned back on.
- TIMER (function) returns the number of seconds past the previous midnite down to a QB64 accuracy of one millisecond.
- TIMER (statement) enables, turns off or stops timer event trapping. In QB64 TIMER(n) FREE

can free multiple timers.

- WAIT (statement) normally used to delay program display execution during or after vertical retrace periods.

(Return to Table of Contents)

File Input and Output:

- _DIREXISTS (function) returns -1 if the directory folder name string parameter exists. Zero if it does not.
- _FILEEXISTS (function) returns -1 if the file name string parameter exists. Zero if it does not.

- ACCESS (clause) used in a networking OPEN statement to allow READ or WRITE access to files.
- APPEND (file mode) opens or creates a file that can be appended with data at the end.
- BINARY (file mode) opens or creates a file that can be byte accessed using both GET and PUT.
- BLOAD (statement) opens a binary file and loads the contents to a specific array.
- BSAVE (statement) creates a binary file that holds the contents of a specified array.
- CHDIR (statement) changes the program path to a new path.
- CLOSE (statement) closes a specified file or all open files.
- EOF (file function) returns -1 when the end of a file has been read.
- FIELD (statement) creates a STRING type definition for a random-access file buffer.
- FILEATTR (function) can return a file's current file mode or DOS handle number.
- FILES (statement) displays a specified list of files.
- FREEFILE (file function) returns a file access number that is currently not in use.
- GET (file I/O statement) reads file data by byte or record positions.
- INPUT (file mode) only OPENs existing sequential files for program INPUT.
- INPUT (file statement) reads sequential file data that was created using PRINT # or WRITE #.
- INPUT$ (function) reads a specific number of bytes from random or binary files.
- IOCTL (statement) sends a message to an open IOCTL compatible device. NOT IMPLEMENTED!
- IOCTL$ (function) receives messages from an open device. NOT IMPLEMENTED!
- KILL (statement) deletes a specified file without asking for verification. Remove empty folders with RMDIR.
- LINE INPUT (file statement) reads an entire text row of printed sequential file data.
- LOC (function) finds the current file location or size of a COM port receive buffer.
- LOCK (statement) prevents access to a file.
- LOF (file function) returns the size of a file in bytes.
- MKDIR (statement) creates a new folder in the designated path.

- NAME (statement) renames a file AS a new file name.
- OPEN (file I/O statement) opens a specified file FOR an access mode with a set reference number.
- OUTPUT (file mode) opens or creates a new file that holds no data.
- PRINT (file statement) writes text and numerical data into a file.
- PRINT USING (file statement) writes template formatted text into a file.
- PUT (file I/O statement) writes data into a RANDOM or BINARY file by byte or record position.
- RANDOM (file mode) opens or creates a file that can be accessed using both GET and PUT.
- RESET (statement) closes all files and writes the directory information to a diskette.
- RMDIR (statement) removes an empty folder from the specified path.
- SEEK (function) returns the current read or write byte position in a file.
- SEEK (statement) sets the current read or write byte position in a file.
- UNLOCK (statement) unlocks access to a file.
- WIDTH (statement) sets the text width of a file.
- WRITE (file statement) writes numerical and string data to a sequential file using comma separators.

(Return to Table of Contents)

Fonts and Unicode:

- _FONT (function) creates a new alphablended font handle from a designated image handle
- _FONT (statement) sets the current _LOADFONT function font handle to be used by PRINT or _PRINTSTRING.
- _FONTHEIGHT (function) returns the font height of a font handle created by _LOADFONT.
- _FONTWIDTH (function) returns the font width of a MONOSPACE font handle created by _LOADFONT.
- _FREEFONT (statement) frees a font handle value from memory

- _LOADFONT (function) loads a TrueType font (.TTF) file of a specific size and style and returns a font handle value.

- _MAPUNICODE (statement) maps a Unicode value to an ASCII character code value.

- _PRINTMODE (function) returns the present _PRINTMODE status as a numerical value.
- _PRINTMODE (statement) sets the text or _FONT printing mode on a background image when

using PRINT or _PRINTSTRING.

- _KEEPBACKGROUND (1): Text background transparent. Only the text is displayed over anything behind it.
- _ONLYBACKGROUND (2): Text background is only displayed. Text is transparent to anything behind it.
- _FILLBACKGROUND (3): Text and background block anything behind them like a normal PRINT. Default setting.

- _PRINTSTRING (statement) prints text or custom font strings using graphic column and row coordinate positions.
- _PRINTWIDTH (function) returns the width in pixels of the _FONT or text string that a program will print.

- PRINT (statement) prints numeric or string expressions to the program screen.
- PRINT USING (statement) prints template formatted numeric or string values to the program screen.
- WRITE (screen I/O statement) writes a comma-separated list of values to the screen.

See also: Unicode, Unicode Code Pages or ASCII Code Table

(Return to Table of Contents)

Game Controller Input (Joystick):

- _AXIS (function) returns a SINGLE value between -1 and 1 indicating the maximum distance from device axis center 0.
- _BUTTON (function) returns -1 when a device button is pressed and 0 when button is released.
- _BUTTONCHANGE (function) returns -1 when a device button has been pressed and 1 when released. Zero indicates no change.

- _DEVICE$ (function) returns a STRING expression listing a designated numbered input device name and types of input.
- _DEVICEINPUT (function) returns the _DEVICES number of an _AXIS, _BUTTON or _WHEEL event.
- _DEVICES (function) returns the number of input devices found on a computer system including the keyboard and mouse.

- _LASTAXIS (function) returns the number of axis available on a specified number device listed

by _DEVICE$.
- _LASTBUTTON (function) returns the number of buttons available on a specified number device listed by DEVICE$.
- _LASTWHEEL (function) returns the number of scroll wheels available on a specified number device listed by _DEVICE$.

- _MOUSEMOVEMENTX (function) returns the relative horizontal position of the mouse cursor compared to the previous position.
- _MOUSEMOVEMENTY (function) returns the relative vertical position of the mouse cursor compared to the previous position.
- _WHEEL (function) returns -1 when a device wheel is scrolled up and 1 when scrolled down. Zero indicates no activity.

- ON STRIG(n) (event statement) directs program flow upon a button press event of a game controller device.
- STICK (function) returns the directional axis coordinate values from 1 to 254 of game port (&H201) or USB controller devices.
- STRIG (function) returns the True or False button press status of game port (&H201) or USB controller devices.
- STRIG(n) (statement) enables, suspends or disables event trapping of STRIG button return values.

See also: Windows Game Pad or SDL Joystick Libraries.

(Return to Table of Contents)

Graphic GL Special Commands

Full list of QB64 _GL prefixed statements and functions

- _COPYIMAGE (function) can copy a software surface to a hardware accelerated surface handle using mode 33.
- _DISPLAY (statement) renders surfaces visible in the _DISPLAYORDER previously set in the QB64GL program.
- _DISPLAYORDER (GL statement) sets the rendering order of _SOFTWARE, _HARDWARE and _GLRENDER with _DISPLAY.
- _LOADIMAGE (function) can load images as hardware accelerated using mode 33.

- _MOUSESHOW (statement) a special string parameter after command in GL allows some special cursor shapes.
- _PUTIMAGE (statement) can place GL surfaces and allows the _SMOOTH action to blend stretched surfaces.

(Return to Table of Contents)

Graphics and Imaging:

- _AUTODISPLAY (statement) enables the automatic display of the screen image changes previously disabled by _DISPLAY.
- _CLIP (PUT action) allows placement of an image partially off of the screen.
- _COPYIMAGE (function) function duplicates an image handle from a designated handle.
- _COPYPALETTE (statement) copies the color palette intensities from one image to another image or screen page.
- _DEST (statement) sets the current write image or page. All graphics will go to this image.
- _DEST (function) returns the current write destination image or page.
- _DISPLAY (statement) turns off automatic display while only displaying the screen changes when called.
- _DISPLAY (function) returns the handle of the current image that is displayed on the screen.
- _FULLSCREEN (function) returns the present full screen mode setting of the screen window.
- _FULLSCREEN (statement) sets the full screen mode of the screen window. Alt + Enter can do it manually.
- _FREEIMAGE (statement) releases an image handle value from memory when no longer needed.
- _HEIGHT (function) returns the height of an image handle or current write page.
- _ICON (function) places an image in the title bar using a _LOADIMAGE handle. (Icons can be used in GL only!)
- _LOADIMAGE (function) loads a graphic file image into memory and returns an image handle.
- _MAPTRIANGLE (statement) maps a triangular portion of an image to a destination image or screen page.
- _NEWIMAGE (function) prepares a window image or page surface and returns the handle value.
- _PIXELSIZE (function) returns the color depth (Bits Per Pixel) of an image.
- _PRINTSTRING (statement) prints text or custom font strings using graphic column and row coordinate positions.
- _PUTIMAGE (statement) maps a rectangular area of a source image to an area of a destination image in one operation
- _SCREENIMAGE (function) creates an image of the current desktop and returns an image

handle.

- _SOURCE (statement) establishes the image SOURCE using a designated image handle
- _SOURCE (function) returns the present image _SOURCE handle value.
- _WIDTH (function) returns the width of an image handle or current write page.

- CIRCLE (statement) is used in graphics SCREEN modes to create circles, arcs or ellipses.
- CLS (statement) clears a screen page or the program SCREEN. QB64 can clear with a color.
- COLOR (statement) sets the current text color attribute or _RGB value to be used or background colors in some screen modes.
- DRAW (statement) uses a special type of string expression to draw lines on the screen.
- GET (graphics statement) used to store a box area image of the screen into an INTEGER array.
- LINE (statement) used in graphic SCREEN modes to create lines or boxes.
- PAINT (statement) used to color enclosed graphic objects with a designated fill color and border color.
- PALETTE (statement) can swap color settings, set colors to default or set the Red, Green, Blue color palette.
- PALETTE USING (statement) sets all RGB screen color intensities using values from an array.
- PCOPY (statement) copies one source screen page to a destination page in memory.
- PMAP (function) returns the physical or WINDOW view coordinates.
- POINT (function) returns the pixel COLOR attribute or _RGB value at a specified graphics coordinate.
- PRESET (statement) sets a pixel coordinate to the background color or a designated color.
- PSET (statement) sets a pixel coordinate to the default color or designated color attribute.
- PUT (graphics statement) statement is used to place GET saved or BSAVEd images stored in an array.
- SCREEN sets the screen mode of a program. No statement defaults the program to SCREEN 0 text only mode.
- STEP (relational statement) is used to step through FOR loop values or use relative graphical coordinates.
- VIEW (graphics statement) creates a graphics view port area by defining the coordinate limits to be viewed.
- WINDOW (statement) defines the coordinate dimensions of the current graphics viewport.

See also: Bitmaps, Icons and Cursors, SAVEIMAGE, GIF Images

(Return to Table of Contents)

Keyboard Input:

- _EXIT (function) prevents a program user exit and indicates if a user has clicked the close X window button or CTRL + BREAK.

- _KEYDOWN (function) returns whether modifying keys like CTRL, ALT, SHIFT, and any other keys are pressed.
- _KEYHIT (function) returns ASCII one and two byte, SDL Virtual Key and Unicode keyboard key press codes.

- _SCREENPRINT (statement) simulates typing text into another OS program using the keyboard.

- INKEY$ (function) returns the ASCII string character of a keypress.
- INPUT (statement) requests a STRING or numerical keyboard entry from a program user.
- INPUT$ (function) used to get a set number of keypress characters or bytes from a file.
- INP (function) returns a scan code value from keyboard register &H60(96) to determine key presses.
- KEY n (event statement) is used to assign a "softkey" string to a key and/or display them.
- KEY(n) (event statement) assigns, enables, disables or suspends event trapping of a keypress.
- KEY LIST (statement) lists the 12 Function key soft key string assignments going down left side of screen.
- LINE INPUT (statement) requests a STRING keyboard entry from a program user.
- ON KEY(n) (event statement) defines a line number or label to go to when a specified key is pressed.
- ON PEN (event statement) enables event handling for a lightpen. NOT IMPLEMENTED!
- PEN (event function) returns requested information about the lightpen device used. NOT IMPLEMENTED!
- PEN (statement) enables/disables or suspends event trapping of the lightpen device. NOT IMPLEMENTED!
- SLEEP (statement) pauses the program for a specified number of seconds or a until a key press.

See also: Keyboard scancodes, ASCII Codes references or Hot Keys for Windows.

(Return to Table of Contents)

Libraries:

- _OFFSET (function) returns the memory offset of a variable when used with DECLARE DYNAMIC LIBRARY only.
- _OFFSET(variable type) can be used store the value of an offset in memory when using DECLARE DYNAMIC LIBRARY only.

- ALIAS (statement) tells the program that you will use a different name than the name used in the Library file.
- BYVAL (statement) used to pass a parameter's value with sub-procedures from a Library.
- DECLARE LIBRARY (QB64 Only) allows the use of OS specific, SDL or C++ external library SUB and FUNCTION procedures
- DECLARE DYNAMIC LIBRARY (QB64 Only) declares DYNAMIC, CUSTOMTYPE or STATIC library(DLL) SUBs or FUNCTIONs.
- END DECLARE (QB64 Only) required at the END of the block of Library declarations in QB64.

QB64 also supports $INCLUDE text code file Libraries. QB64 does not support QLB Libraries or OBJ files.

See also: C++ Variable Types

(Return to Table of Contents)

Logical Bitwise Operations:

- AND (operator) the bit is set when both bits are set.
- EQV (operator) the bit is set when both are set or both are not set.
- IMP (operator) the bit is set when both are set or both are unset or the second condition bit is set.
- OR (operator) the bit is set when either bit is set.
- NOT (operator) the bit is set when a bit is not set and not set when a bit is set.
- XOR (operator) the bit is set when just one of the bits are set.

The results of the bitwise logical operations, where *A* and *B* are operands, and *T* and *F* indicate that a bit is set or not set:

Operands		Operations																	
A	B	NOT	B	A	AND	B	A	OR	B	A	XOR	B	A	EQV	B	A	IMP	B	
T	T	F		T	T		T	F		T	T								
T	F	T		F	T		T	T		F	F								
F	T	F		F	T		T	T		F	T								
F	F	T		F	F		F	F		T	T								

Relational Operations return negative one (-1, all bits set) and zero (0, no bits set) for *true* and *false*, respectively.

This allows relational tests to be inverted and combined using the bitwise logical operations.

(Return to Table of Contents)

Mathematical Functions and Operations:

- _ROUND (function) rounds to the closest EVEN INTEGER, LONG or _INTEGER64 numerical value.

- + addition operation
- - subtraction or negation operation
- * multiplication operation
- / decimal point division operation
- \ integer division operation
- ^ exponential operation
- MOD integer remainder operation

- ABS (function) returns the the positive value of a variable or literal numerical value.
- ATN (function) or arctangent returns the angle in radians of a numerical tangent value.
- CDBL (function) closest double rounding function
- CINT (function) closest integer rounding function
- CLNG (function) closest long integer rounding function
- COS (function) cosine of a radian angle
- CSNG (function) closest single rounding function
- EXP (function) returns the value of e to the power of the parameter used.
- FIX (function) rounds positive or negative values to integer values closer to 0
- INT (function) rounds to lower integer value

- LOG (function) natural logarithm of a specified numerical value.
- SIN (function) sine of a radian angle.
- SQR (function) square root of a positive number.
- TAN (function) returns the ratio of SINe to COSine or tangent value of an angle measured in radians.

See also: Mathematical Operations and Logical Operations:

(Return to Table of Contents)

Memory Handling and Clipboard:

- _CLIPBOARD$ (function) returns the current STRING contents of the system Clipboard.
- _CLIPBOARD$ (statement) sets and overwrites the STRING contents of the current system Clipboard.
- _MEM (function) returns _MEM block referring to the largest continuous memory region beginning at a designated variable's offset.
- _MEM (variable type) contains read only dot elements for the OFFSET, SIZE, TYPE and ELEMENTSIZE of a block of memory.
- _MEMCOPY (statement) copies a value from a designated OFFSET and SIZE TO a block of memory at a designated OFFSET.
- _MEMELEMENT (function) returns a _MEM block referring to a variable's memory (but not past it).
- _MEMEXISTS (function) verifies that a memory block exists for a memory variable name or returns zero.
- _MEMFILL (statement) fills a designated memory block OFFSET with a certain SIZE and TYPE of value.
- _MEMFREE (statement) frees a designated memory block in a program. Only free memory once!
- _MEMGET (statement) reads a designated value from a designated memory OFFSET
- _MEMGET (function) returns a value from a designated memory block and OFFSET using a designated variable TYPE.
- _MEMIMAGE (function) returns a _MEM block referring to a designated image handle's memory
- _MEMNEW (function) allocates new memory with a designated SIZE and returns a _MEM block referring to it.
- _MEMPUT (statement) places a designated value into a designated memory _OFFSET
- _OFFSET (function) returns the memory offset of a variable when used with DECLARE _LIBRARY or _MEM only.

- _OFFSET(%& numerical type) can be used store the value of an offset in memory when using DECLARE LIBRARY or _MEM only.

Functions and statements using QB64's emulated 16 bit memory

- DEF SEG (statement) defines the segment address in memory.
- FRE (function) returns the amount of Memory available in bytes to running programs. NOT IMPLEMENTED!
- PEEK (function) returns the value that is contained at a certain memory address offset.
- POKE (statement) sets the value of a specified memory address offset.
- SADD (function) returns the address of a STRING variable as an offset from the current data segment.
- SETMEM (function) is used to increase, decrease or return the current "far heap" byte size. NOT IMPLEMENTED!
- VARPTR (function) returns an INTEGER value that is the offset pointer of the memory address within it's segment.
- VARPTR$ (function) returns a STRING representation of a variable's memory address value
- VARSEG (function) returns an INTEGER value that is the segment part of a variable or array memory address.

See also: Screen Memory or Using _OFFSET

(Return to Table of Contents)

Metacommands

Metacommands are commands that affect a program globally after they are in used. Error checking can be turned OFF or ON.

QB64 Metacommands do NOT allow commenting or REM!

- $CHECKING:OFF/ON (QB64 only) turns event and error checking ON and OFF. ON (default) can only be used after it is OFF.
- $CONSOLE creates a console window throughout the program.
- $SCREENHIDE hides the program window throughout the program until $SCREENSHOW is used.
- $SCREENSHOW displays the main program window throughout the program only after $SCREENHIDE or _SCREENHIDE has been used.

Qbasic Metacommands do not require commenting or REM in QB64!

- '$DYNAMIC defines that all arrays are dynamic or changeable in size using DIM or REDIM.

- '$INCLUDE: 'filename$' includes a text library file with procedures to be used in a program. Comment both sides of file name also.
- '$STATIC defines that all arrays are static or unchangeable in size using DIM.

(Return to Table of Contents)

Mouse Input:

- _AXIS (function) returns a SINGLE value between -1 and 1 indicating the maximum distances from device center 0.
- _BUTTON (function) returns -1 when a device button is pressed and 0 when button is released. 2 is the mouse center or scroll button
- _BUTTONCHANGE (function) returns -1 when a device button has been pressed and 1 when released. Zero indicates no change.
- _DEVICE$ (function) returns a STRING expression listing device names and input types of system input devices.
- _DEVICEINPUT (function) returns the _DEVICES number of an _AXIS, _BUTTON or _WHEEL event. Mouse is normally _DEVICEINPUT(2).
- _DEVICES (function) returns the number of input devices found on a computer system. The mouse is normally device 2.
- _EXIT (function) prevents a program user exit and indicates if a user has clicked the close X window button or CTRL + BREAK.
- _LASTAXIS (function) returns the number of axis available on a specified number device listed by _DEVICE$.
- _LASTBUTTON (function) returns the number of buttons available on a specified number device listed by DEVICE$.
- _LASTWHEEL (function) returns the number of scroll wheels available on a specified number device listed by _DEVICE$.
- _MOUSEBUTTON (function) returns whether a specified mouse button number has been clicked. 3 is the mouse center or scroll button
- _MOUSEHIDE (statement) hides the OS mouse pointer from view.
- _MOUSEINPUT (function) must be used to monitor and read all changes in the mouse status.
- _MOUSEMOVE (statement) moves the mouse cursor pointer to a designated coordinate.
- _MOUSEMOVEMENTX (function) returns the relative horizontal position of the mouse cursor.
- _MOUSEMOVEMENTY (function) returns the relative vertical position of the mouse cursor.
- _MOUSESHOW (statement) displays(default) the mouse cursor after it has been hidden.
- _MOUSEWHEEL (function) returns a positive or negative count the mouse scroll wheel clicks since the last read.

- _MOUSEX (function) indicates the current horizontal position of the mouse pointer.
- _MOUSEY (function) indicates the current vertical position of the mouse pointer.
- _SCREENCLICK simulates clicking the mouse at a position on the screen to get focus.
- _WHEEL (function) returns -1 when a device wheel is scrolled up and 1 when scrolled down. Zero indicates no activity.

- CALL ABSOLUTE (statement) used to access Interrupt vector &H33 to work with the mouse. Functions 0 to 3 implemented.
- INTERRUPT (statement) used to access Interrupt vector &H33 to work with the mouse. Functions 0 to 3 implemented.

See also: SDL Mouse Library

(Return to Table of Contents)

Numerical Manipulation and Conversion:

- &B Binary base number prefix used in QB64 to represent _BITs on as 1 or off as 0.
- _CV (function) used to convert _MK$ ASCII string values to specified numerical value types.
- _MK$ (function) converts a specified numerical type into an ASCII string value that must be converted back using _CV.
- _PRESERVE (REDIM action) preserves the current contents of an array, when re-dimensioning it.
- _UNSIGNED {numerical type) defines a numerical value as being positive only using QB64.

- ABS (function) returns the the positive value of a variable or literal numerical value.
- ASC (function) returns the ASCII code number of a certain STRING text character or a keyboard press.
- CDBL (function) converts a numerical value to the closest DOUBLE-precision value.
- CDECL (statement) used to indicate that the external procedure uses the C-language argument order.
- CHR$ (function) returns the character associated with a certain ASCII character code as a STRING.
- CINT (function) returns the closest INTEGER value of a number.
- CLEAR (statement) clears all variable values to 0 or null strings and closes all open files.
- CLNG (function) rounds decimal point numbers up or down to the nearest LONG integer value.
- CSNG (function) converts a numerical value to the closest SINGLE-precision number.
- CVD (function) converts STRING values to DOUBLE numerical values.
- CVDMBF (function) converts a 8-byte Microsoft Binary format string value to a DOUBLE

precision number.

- CVI (function) converts 2 byte STRING values to INTEGER numerical values.
- CVL (function) converts 4 byte STRING values to LONG numerical values.
- CVS (function) converts 4 byte STRING values to SINGLE numerical values.
- CVSMBF (function) converts a 4-byte Microsoft Binary format string value to a SINGLE-precision number.
- DIM (statement) used to declare a variable type or dimension a STATIC array.
- ERASE (array statement) clears a STATIC array of all values and totally removes a $DYNAMIC array.
- HEX$ (function) converts decimal INTEGER values to hexadecimal STRING number values. Prefix values with &H
- INT (function) rounds a numeric value down to the next whole number or INTEGER value.
- LEN (function) returns the byte size of strings or numerical variables.
- OCT$ converts decimal numerical values to Octal STRING number values. Prefix values with &O
- RANDOMIZE (statement) seeds the RND random number generation sequence.
- REDIM (statement) re-dimensions the number of elements in a dynamic(resizeable) array.
- RND (function) returns a randomly generated number from 0 to .9999999
- SGN (function) returns the sign as -1 for negative, zero for 0 and 1 for positive numerical values.
- STR$ (function) converts a numerical value to a STRING value.
- SWAP (statement) trades the values of two numerical types or strings.
- VAL (function) converts number strings into numerical values until it runs into a non-numeric character.

(Return to Table of Contents)

Port Input and Output (COM and LPT):

- COM(n) (statement) used in an OPEN COM statement to open "COM1" or "COM2".

- GET (file I/O statement) reads port data data by byte or record positions.

- INP (function) returns a value from port hardware address. NOT IMPLEMENTED for port access!
- LOC (function) finds the current file location or size of a COM port receive buffer.

- ON COM(n) (event statement) branches to a line number or label when there is a value in the serial port specified.
- OPEN COM (statement) opens a computer serial COMmunications port.
- OUT (statement) sends values to register or port hardware addresses. Use with care! NOT IMPLEMENTED for port access!

- PRINT (file statement) writes text and numerical data to a port transmit buffer.
- PUT (file I/O statement) writes data into a RANDOM or BINARY port by byte or record position.

See Port Access Libraries for other ways to access COM and LPT ports..

(Return to Table of Contents)

Print formatting

- LPRINT USING (statement) prints template formatted STRING text to an LPT or USB printer page.
- PRINT USING (statement) prints template formatted STRING text to the screen.
- PRINT USING (file statement) prints template formatted STRING text to a text file.

Template is a literal or variable string using the following formatting characters:

&	Prints an entire string value. STRING length should be limited as template width will vary.
\ \	Denotes the start and end point of a fixed string area with spaces between(LEN = spaces + 2).
!	Prints only the leading character of a string value. Exclamation points require underscore prefix.
#	Denotes a numerical digit. An appropriate number of digits should be used for values received.
^^^^	After # digits prints numerical value in exponential E+xx format. Use ^^^^^ for E+xxx values.*
.	Period sets a number's decimal point position. Digits following determine rounded value accuracy.
,.	Comma to left of decimal point, prints a comma every 3 used # digit places left of the decimal point.
+	Plus sign denotes the position of the number's sign. + or - will be displayed.

-	Minus sign (dash) placed after the number, displays only a negative value's sign.
$$	Prints a dollar sign immediately before the highest non-zero # digit position of the numerical value.
****	Prints an asterisk in any leading empty spaces of a numerical value. Adds 2 extra digit positions.
****$**	Combines ** and $$. Negative values will display minus sign to left of $.
_	Underscore preceding a format symbol prints those symbols as literal string characters.

Note: Any string character not listed above will be printed as a literal text character.

* Any # decimal point position may be specified. The exponent is adjusted with significant digits left-justified.

(Return to Table of Contents)

Printer Output (LPT and USB):

- _PRINTIMAGE (statement) prints an image stretched to the size of the paper setting of an LPT or USB printer.

- LPOS (function) returns the current parallel(LPT) printer head position.
- LPRINT (statement) prints text to an LPT or USB printer page.
- LPRINT USING (statement) prints template formatted STRING text to an LPT or USB printer page.

QB64 will use the default system printer selected. _PRINTIMAGE images will be stretched to the paper size setting.

(Return to Table of Contents)

Program Flow and Loops:

- _DEST (statement) sets the current write image or page. All graphics will go to this image.
- _DEST (function) returns the current write destination image or page.
- _EXIT (function) prevents a user exit and indicates if a user has clicked the close X window button or CTRL + BREAK.
- _SOURCE (statement) establishes the image SOURCE using a designated image handle
- _SOURCE (function) returns the present image _SOURCE handle value.
- _SHELLHIDE (function) returns the code sent by a program exit using END or SYSTEM

followed by an INTEGER value.

- CALL (statement) sends code execution to a subroutine procedure in a program.
- CALLS (statement) transfers control to a procedure written in a different programming language. NOT IMPLEMENTED!
- CASE (SELECT CASE statement) used within a SELECT CASE block to specify a conditional value of the compared variable.
- CASE ELSE (SELECT CASE statement) used in a SELECT CASE block to specify an alternative to other CASE values.
- CASE IS (SELECT CASE statement) used within a SELECT CASE block to specify a conditional value of the compared variable.
- DO...LOOP (loop statement) used in programs to repeat code or return to the start of a procedure.
- ELSE (statement) used in IF...THEN statements to offer an alternative to other conditional statements.
- ELSEIF (statement) used in IF...THEN block statements to offer an alternative conditional statement.
- END (statement) ENDs a conditional block statement, a sub-procedure or ends the program with "Press any key..."
- END IF (IF statement end) ENDs an IF statement block.
- ERROR (error statement) used to simulate an error in a program.
- EXIT (statement) exits a loop, function or sub-procedure early.
- FOR...NEXT (statement) a counter loop procedure that repeats code a specified number of times.
- GOSUB (statement) send the program to a designated line label procedure in the main module or a SUB procedure.
- GOTO (statement) sends the program to a designated line number or label.
- IF...THEN (statement) a conditional flow statement or block of statements.
- LOOP end of a DO...LOOP procedure that repeats code until or while a condition is true.
- RESUME (error statement) an error statement that can return the program to the NEXT code line or a specific line number.
- RETURN (statement) a sub-procedure statement that returns the program to the code immediately after the procedure call.
- RUN (statement) clears and restarts the program currently in memory or executes another specified program.
- SELECT CASE (statement) determines the program flow by comparing the value of a variable to specific values.
- SHELL (DOS statement) directly accesses the Operating System's command line procedures.

- SLEEP (statement) stops program progression for a specific number of seconds or until a keypress is made.
- STEP (relational statement) is used to step through FOR loop values or use relative graphical coordinates.
- STOP (statement) is used when troubleshooting a program to stop the program at a specified code line.
- SYSTEM (statement) immediately exits a program and closes the program window.
- UNTIL (conditional statement) continues a DO LOOP procedure until a condition is true.
- WHILE (statement) continues a DO LOOP procedure while a condition is true.
- WHILE...WEND (statement) a loop procedure that repeats code while a condition is true.

(Return to Table of Contents)

Sounds and Music using Sound Card:

- _SNDBAL (statement) attempts to set the balance or 3D position of a sound.
- _SNDCLOSE (statement) frees and unloads an open sound using a _SNDOPEN or _SNDCOPY handle.
- _SNDCOPY (function) copies a sound to a new handle so that two or more of the same sound can be played at once.
- _SNDGETPOS (function) returns the current playing position in seconds of a designated sound handle.
- _SNDLEN (function) returns the length of a sound in seconds of a designated sound handle.
- _SNDLIMIT (statement) stops playing a sound after it has been playing for a set number of seconds.
- _SNDLOOP (statement) loops the playing of a specified sound handle.
- _SNDOPEN (function) loads a sound file with certain capabilities and returns a handle value.
- _SNDPAUSE (statement) pauses a specified sound handle if it is playing.
- _SNDPAUSED (function) returns the pause status of a specified sound handle.
- _SNDPLAY (statement) plays a designated sound file handle.
- _SNDPLAYCOPY (statement) copies a sound, plays it and automatically closes the copy using a handle parameter
- _SNDPLAYFILE (statement) a simple command to play a sound file with limited options for SYNC and volume.
- _SNDPLAYING (function) returns whether a sound handle is being played.
- _SNDRATE (function) returns the sample rate frequency per second of the current computer's sound card.
- _SNDRAW (statement) plays sound wave sample frequencies created by a program.
- _SNDRAWDONE (statement) pads a _SNDRAW stream so the final (partially filled) buffer section is played.

- _SNDRAWLEN (function) returns the length, in seconds, of a _SNDRAW sound currently queued.
- _SNDRAWOPEN (function) returns a handle to a new, separate _SNDRAW audio stream.
- _SNDSETPOS (statement) changes the current/starting playing position of a sound in seconds.
- _SNDSTOP (statement) stops a playing or paused sound handle.
- _SNDVOL (statement) sets the volume of a sound handle being played.

- BEEP (statement) makes a beep sound when called or CHR$(7) is printed.
- ON PLAY(n) (event statement) specifies the line-number or label to branch to when the background music queue is low. NA!
- PLAY(n) (event function) returns the number of notes currently in the background music queue. NOT IMPLEMENTED!
- PLAY (music statement) uses a custom string statement to play musical notes.
- SOUND (statement) creates sounds of a specified frequency for a set duration.

(Return to Table of Contents)

String Text Manipulation and Conversion:

- _CLIPBOARD$ (function) returns the current STRING contents of the system Clipboard.
- _CLIPBOARD$ (statement) sets the STRING contents of the current system Clipboard.
- _CV (function) used to convert _MK$ ASCII string values to numerical values.
- _MK$ (function) converts any numerical type into an ASCII string value that must be converted back using _CV.

- ASC (statement) allows a QB64 program to change a character at any position of a predefined STRING.
- HEX$ (function) returns the base 16 hexadecimal representation of an INTEGER value as a STRING.
- INSTR (function) searches for the first occurance of a search STRING within a string and returns the position.
- LCASE$ (function) changes the uppercase letters of a STRING to lowercase.
- LEFT$ (function) returns a part of a STRING from the start a designated number of character places.
- LEN (function) returns the number of bytes or characters in a STRING value.
- LSET (statement) left-justifies a fixed length string expression based on the size of the STRING.
- LTRIM$ (function) returns a string with all leading spaces removed.

- MID$ (statement) returns a portion of a string from the start position a designated number of characters.
- MKD$ (function) converts a DOUBLE numerical value into an 8 byte ASCII STRING value.
- MKDMBF$ (function) converts a double-precision number to a string containing a value in Microsoft Binary format.
- MKI$ (function) converts a numerical INTEGER value to a 2 byte ASCII string value.
- MKL$ (function) converts a numerical LONG value to a 4 byte ASCII string value.
- MKS$ (function) converts a numerical SINGLE value to a 4 byte ASCII string value.
- MKSMBF$ (function) converts a single-precision number to a string containing a value in Microsoft Binary format.}}
- OCT$ (function) returns the base 8 Octal representation of an INTEGER value.
- RIGHT$ (function) returns a set number of characters in a STRING variable starting from the end.
- RSET (statement) right-justifies a string according to length of the string expression.
- RTRIM$ (function) returns a string with all of the spaces removed at the right end of a string.
- SPACE$ (function) returns a STRING consisting of a number of space characters.
- STR$ (function) converts a numerical value to a STRING.
- STRING ($ variable type) one byte text variable with ASCII code values from 0 to 255.
- STRING$ (function) returns a STRING consisting of a single character repeated a set number of times.
- SWAP (statement) used to exchange two string variable or array element values.
- UCASE$ (function) returns a string with all letters as uppercase.
- VAL (function) converts a string number value to a numerical value.

(Return to Table of Contents)

Sub procedures and Functions

Qbasic and QB64

- CALL (statement) sends code execution to a SUB procedure in a program. Parameter brackets are required when used.
- CALL ABSOLUTE (statement) used to access Interrupts on the computer or execute assembly type procedures.
- CALLS (statement) transfers control to a procedure written in a different programming language. NOT IMPLEMENTED!
- CHAIN (statement) changes seamlessly from one program module to another.
- DECLARE (BASIC statement) used to tell that a SUB or FUNCTION is created to be used in the program. NOT USED by QB64!

- DEF FN (statement) statement defines a function in the main module. NOT IMPLEMENTED!
- END (statement) ends a SUB or FUNCTION procedure.
- EXIT (statement) exits a SUB or FUNCTION procedure early.
- FUNCTION (statement) a procedure that holds ONE return value in the function's name which is a variable type.
- GOSUB (statement) sends the program to a sub program that uses a line number or label.
- $INCLUDE (metacommand) used to insert a source code text file into your program at the point of the insertion.
- INTERRUPT (statement) a built in assembly routine for accessing computer information registers.
- RETURN (statement) used in GOSUB procedures to return to the original call code line.
- RUN (statement) flow statement that clears and restarts the program currently in memory or executes another specified program.
- SHARED (statement) defines a variable or list of variables as shared with the main program module.}}
- SHELL (statement) allows a program to use OS command lines.
- STATIC (statement) defines a variable or list of variables that will retain their values after the sub-procedure is exited.
- SUB (statement) procedures are programs within programs that can return multiple calculations.

(Return to Table of Contents)

TCP/IP Networking and Email:

All Statements and Functions Compile in QB64 Only!

- _CONNECTED (function) returns the connection status of a TCP/IP connection handle.
- _CONNECTIONADDRESS$ (function) function returns a connected user's STRING IP address value.

- _OPENCLIENT (function) connects to a Host on the Internet as a Client and returns the Client status handle.
- _OPENCONNECTION (function) open's a connection from a client that the host has detected and returns a status handle.
- _OPENHOST (function) opens a Host which listens for new connections and returns a Host status handle.

- CLOSE (statement) closes an opened internet connection using the handle assigned in an OPEN

statement.

- GET (TCP/IP statement) reads unformatted(raw) data from an opened connection using the connection handle.

- INPUT (TCP/IP statement) reads a formatted message from an opened connection using the connection handle.

- PUT (TCP/IP statement) sends unformatted(raw) data to an open connection using a user's handle.
- PRINT (TCP/IP statement) sends formatted message to an open connection handle.

See also: TCP/IP Message Format, IP Configuration, Downloading Files and WGET

(Return to Table of Contents)

Text on Screen:

- _CONTROLCHR OFF allows ASCII characters 0 to 31 to be used as text characters. ON(default) resets to default usage.
- _FONT (function) creates a new alphablended font handle from a designated image handle
- _FONT (statement) sets the current _LOADFONT function font handle to be used by PRINT or _PRINTSTRING.
- _MAPUNICODE (statement) maps a Unicode value to an ASCII character code value.
- _PRINTSTRING (statement) prints text or custom font strings using graphic column and row coordinate positions.
- _SCREENPRINT (statement) simulates typing text into a Windows program using the keyboard.

- CHR$ (function) returns the text character associated with a certain ASCII character code as a one byte STRING.
- CLS (statement) clears a screen page or the program SCREEN. QB64 can clear with a color.
- COLOR (statement) used to change the color of the text and background in some legacy screen modes.
- CSRLIN (function) returns the current print cursor row position on the screen.

- INPUT (statement) requests a STRING or numerical keyboard entry from a program user.
- KEY LIST (statement) vertically lists all the ON KEY(n) softkey strings associated with each function key F1 to F12.
- LINE INPUT (statement) requests a STRING keyboard entry from a program user.
- LOCATE (statement) locates the screen text row and column positions for a PRINT or INPUT procedure.
- POS (function) returns the current print cursor column position.
- PRINT (statement) prints numeric or string expressions to the program screen.
- PRINT USING (statement) prints template formatted numeric or string values to the program screen.
- SCREEN (statement) sets the screen mode of a program. No statement defaults the program to SCREEN 0 text mode.
- SCREEN (function) returns the ASCII code of a text character or the color attribute at a set text location on the screen.
- SPACE$ (function) returns a string consisting of a number of space characters.
- SPC (function) used in PRINT and LPRINT statements to print or output a number of space characters.
- STR$ (function) returns the STRING representation of a numerical value.
- STRING$(function) returns a STRING consisting of a single character repeated a set number of times.
- TAB (function) used in PRINT and LPRINT statements to move to a specified text column position.
- VIEW PRINT (statement) defines the boundaries of a text viewport PRINT area.
- WIDTH (statement) changes the text dimensions of certain SCREEN modes or printer page widths
- WRITE (screen I/O statement) writes a comma-separated list of values to the screen.

See also: Fonts and Unicode or ASCII Code Table

(Return to Table of Contents)

Time, Date and Timing:

- _AUTODISPLAY (statement) enables the automatic display of the screen image changes previously disabled by _DISPLAY.
- _DELAY (statement) suspends program execution for a SINGLE value of seconds down to milliseconds.
- _DISPLAY (statement) turns off automatic display while only displaying the screen changes when called.

- _FREETIMER (function) returns a free TIMER number for multiple ON TIMER(n) events.
- _KEYDOWN (function) returns whether modifying keys like CTRL, ALT, SHIFT, and any other keys are pressed.
- _KEYHIT (function) returns ASCII one and two byte, SDL Virtual Key and Unicode keyboard key press codes.
- _LIMIT (statement) sets the loop repeat rate of a program to so many per second, relinquishing spare cpu cycles.

- DATE$ (function) returns the present computer date in a mm-dd-yyyy string format
- DATE$ (statement) sets the computer date in a mm-dd-yyyy string format if allowed by the OS.
- INKEY$ (function) can be used in a loop to wait for a keypress or a [Ctrl] + letter key combination.
- INPUT (statement) can be used to wait for an [Enter] key press or a text or numerical menu entry.
- INPUT$ (function) can be used to wait for a key press or a fixed length text entry.
- ON KEY(n) (event statement) executes when a keypress or keypress combination occurs.
- ON TIMER(n) (event statement) executes when a timed event occurs. QB64 can use multiple numbered timer events.
- SLEEP (statement) pauses the program for a specified number of seconds or a until a key press is made.
- TIME$ (function) returns the present computer time in a hh:mm:ss 24 hour string format
- TIME$ (statement) sets the computer time in a hh:mm:ss 24 hour string format if allowed by the OS.
- TIMER (function) returns the number of seconds past the previous midnite down to a QB64 accuracy of one millisecond.
- TIMER (statement) enables, turns off or stops timer event trapping. In QB64 TIMER(n) FREE can free multiple timers.
- WAIT (statement) normally used to delay program display execution during or after vertical retrace periods.

(Return to Table of Contents)

Window and Desktop:

All Statements and Functions except SCREEN Compile in QB64 Only!

- _FULLSCREEN (function) returns the present full screen mode setting number of the screen window.

- _FULLSCREEN (statement) sets the full screen mode of the screen window. Alt + Enter can do it manually.
- _HEIGHT (function) returns the height of a _SCREENIMAGE handle to get the desktop resolution.
- _ICON (statement) creates a program icon from an image file handle created by _LOADIMAGE. Cannot use .ICO files!
- _NEWIMAGE (statement) function prepares a window image surface and returns the handle value.
- $RESIZE (Metacommand) used with ON allows a user to resize the program window where OFF does not.
- _RESIZE (function) returns -1 when a program user wants to resize the GL program screen.
- _RESIZEHEIGHT (function) returns the requested new user screen height when $RESIZE:ON allows it.
- _RESIZEWIDTH (function) returns the requested new user screen width when $RESIZE:ON allows it.
- _SCREENCLICK simulates clicking the mouse at a position on the screen to get focus.
- _SCREENEXISTS (function) returns a -1 value once a screen has been created.
- $SCREENHIDE (QB64 Metacommand) hides the program window throughout the program until $SCREENSHOW is used.
- _SCREENHIDE (statement) hides the main program window in a section of code until _SCREENSHOW is used.
- _SCREENIMAGE (function) creates an image of the current desktop and returns an image handle.
- _SCREENMOVE (statement) positions the program window on the desktop using designated coordinates or _MIDDLE.
- _SCREENPRINT (statement) simulates typing text into a Windows program using the keyboard.
- $SCREENSHOW (QB64 Metacommand) displays the main program window throughout the program after $SCREENHIDE.
- _SCREENSHOW (statement) displays the main program window in a section of code after _SCREENHIDE has been used.
- _SCREENX (function) returns the current program window's upper left corner column position on the desktop.
- _SCREENY (function) returns the current program window's upper left corner row position on the desktop.
- _TITLE (statement) sets the program name string in the title bar of the program window.
- _WIDTH (function) returns the width of a _SCREENIMAGE handle to get the desktop resolution.

- SCREEN sets the screen mode of a program. No statement defaults the program to SCREEN 0 text mode.

See also: Console Window, SDL Frameless Window Library, Windows Hot Keys or Focus on Program.

(Return to Table of Contents)

QB64 Programming Symbols:

QB64 and QB Symbols:

[Note: All symbols below can also be used inside of literal quoted strings except for quotation marks.]

Print, Input or File Formatting

- ; Semicolon after a PRINT stops the invisible cursor at end of the printed value. Can prevent screen rolling!
- , Comma after a PRINT tabs the invisible cursor past the end of the printed value.
- " Quotation mark delimits the ends of a literal STRING value in a PRINT, INPUT or LINE INPUT statement.
- ? Question mark is a shortcut substitute for the PRINT keyword. Will change to PRINT when cursor leaves the code line.

Program Code Markers

- ' Apostrophe denotes a program comment, to ignore a code line or a Qbasic Metacommand. Same as using REM.
- , Comma is a statement variable or DATA, SUB or FUNCTION parameter separator.
- : Colons can be used to separate two procedure statements on one code line.
- $ Dollar sign prefix denotes a Qbasic Metacommand. Only **QB64**'s event $CHECKING should NOT be commented.
- ( ) Parenthesis enclose a math or conditional procedure order, SUB or FUNCTION parameters or to pass by value.
- + Plus concatenation operator MUST be used to combine literal string values in a variable definition.
- " Quotation mark delimits the ends of a literal STRING value. Use CHR$(34) to insert quotes in a text STRING.
- REM or apostrophe are used to make comments or ignore code or precedes a Metacommand.
- _ Underscore at the end of a code line is used to continue a line of code to the next program line

in **QB64 only**.

(Return to Table of Contents)

Variable Name Type Suffixes

- $ STRING text character type: 1 byte
- ! SINGLE floating decimal point numerical type (4 bytes)
- # DOUBLE floating decimal point numerical type (8 bytes)
- ## _FLOAT **QB64** decimal point numerical type (32 bytes)
- ~ _UNSIGNED **QB64** whole positive numerical type when it precedes the 6 numerical suffixes below:
- % INTEGER whole numerical type (2 bytes)
- & LONG whole numerical type (4 bytes}
- && _INTEGER64 **QB64** whole numerical type (8 bytes)
- ` _BIT **QB64** whole numerical type (1 bit)(Key below tilde(~) or CHR$(96))
- %% _BYTE **QB64** whole numerical type (1 byte)
- %& _OFFSET **QB64** whole numerical pointer address type (any byte size required)

(Return to Table of Contents)

Numerical Base Prefixes

- &B Binary base 2 Digits 0 or 1 [**QB64**]
- &O Octal base 8 Digits 0 to 7
- &H Hexadecimal base 16: Digits 0 to F

Mathematical Operations

- + Addition operator or sign
- - Subtraction operator or sign
- * Multiplication operator
- / Division (floating decimal point) operator
- \ Integer division operator
- ^ Exponential operator
- MOD Integer Remainder division operator

Relational Operations

- = (Equal to condition)
- > (Greater than condition)

- < (Less than condition)
- <> (Not equal to condition)
- >= (Greater than or equal to condition)
- <= (Less than or equal to condition)
-