_MEM

The _MEM variable type can be used when working with memory blocks. It has no variable type suffix.

Syntax

```
DIM m AS _MEM
```

Contents

Description

Variable TYPE:

- Memory DOT values are actually part of the built in memory variable type in QB64. The following TYPE is built in:

```
TYPE memory_type
  OFFSET AS _OFFSET       'start location of block(changes with byte position)
  SIZE AS _OFFSET         'size of block remaining at offset(changes with position)
  TYPE AS _OFFSET         'type description of variable used(never changes)
  ELEMENTSIZE AS _OFFSET  'byte size of values inside the block(never changes)
  IMAGE AS LONG           'the image handle used when _MEMIMAGE(handle) is used
  SOUND AS LONG           'the sound handle used when _MEMSOUND(handle) is used
END TYPE

The above TYPE is for clarification purposes only. It doesn't need to be pasted in
a program to use _MEM.

IMPORTANT NOTE: As of Build 20170802/57 onward (early v1.2 development), mem.TYPE has
been changed to be an _OFFSET, just as mem.SIZE and mem.ELEMENTSIZE.
```

Usage

- The _MEM type contains the following **read-only** elements where *name* is the _MEM variable name:

name.**OFFSET** is the current start position in the memory block AS _OFFSET. Add bytes to change position.
name.**SIZE** is the remaining size of the block at current position in bytes AS _OFFSET
name.**TYPE** is the type (represented as bits combined to form a value) AS _OFFSET:

.TYPE values (version 1.000 and up incl. all QB64-PE releases)

- [bit 0] 1* byte types (_BYTE)
- [bit 1] 2* byte types (INTEGER)
- [bit 2] 4* byte types (LONG or SINGLE)

- [bit 3] 8* byte types (DOUBLE or _INTEGER64)
- [bit 4] 16* byte types (reserved for future use)
- [bit 5] 32* byte types (_FLOAT)
- [bit 6] 64* byte types (reserved for future use)
- [bit 7] 128 = integer types (_BYTE, INTEGER, LONG, _INTEGER64) (added to *)
- [bit 8] 256 = floating point types (SINGLE, DOUBLE, _FLOAT) (added to *)
- [bit 9] 512 = STRING types (fixed length or variable length)
- [bit 10] 1024 = _UNSIGNED types (added to *+128)
- [bit 11] 2048 = pixel data usually from _MEMIMAGE (added to 1+128+1024 for 256 color screens, or 2+128+1024 for text screens, or 4+128+1024 for 32-bit color screens)
- [bit 12] 4096 = _MEM TYPE structure (NOT added to 32768)
- [bit 13] 8192 = _OFFSET type (added to 4+128+[1024] or 8+128+[1024] or future_size+128+[1024])
- [bit 14] 16384 = data created/defined by _MEMNEW(size) or _MEMNEW(offset,size)
- [bit 15] 32768 = a custom, user defined type (ie. created with TYPE name ... END TYPE)
- [bit 16] 65536 = an array of data (added to other type values defining the array's data type)

Note: If a future integer, float or other type doesn't have a size that is 1,2,4,8,16,32,64,128 or 256 it won't have a size-bit set.

Versions prior to 1.000 (never use with QB64-PE releases)

- 1 = Integer types such as _BYTE, INTEGER, LONG, _INTEGER64 or _OFFSET
- 2 = _UNSIGNED variable types. Value must be added to the variable type value.(2 cannot be used by itself)
- 3 = ALL _UNSIGNED INTEGER type values.(add 1 + 2)
- 4 = Floating point types such as SINGLE, DOUBLE or _FLOAT
- 8 = STRING
- 0 = unknown(eg. created with _MEMNEW) or user-defined-types

- **Note: _OFFSET values cannot be cast to other variable types reliably. _MEM is a reserved custom variable type.**
- **_MEM cannot reference variable length STRING variable values. String values must be designated as a fixed-length string.**

Examples

Example 1: Demonstration of .IMAGE to determine an image's dimensions, .TYPE to verify the type and _MEMEXISTS to check image has not been freed

```
SCREEN _NEWIMAGE(500, 500, 32)
i = _LOADIMAGE("qb64_trans.png", 32)
_PUTIMAGE (0, 0), i
DIM m AS _MEM
m = _MEMIMAGE(i)
'try uncommenting the following line and see what happens
'_MEMFREE m
t = m.TYPE
IF t AND 2048 THEN
  PRINT "this is/was an image"
  IF _MEMEXISTS(m) THEN 'check if memory m is still available
    PRINT t AND 7; "bytes per pixel"
```

```
    PRINT "image handle "; m.IMAGE
    PRINT "image width"; _WIDTH(m.IMAGE)
    PRINT "image height"; _HEIGHT(m.IMAGE)
  ELSE PRINT "Memory already freed!"
  END IF
END IF
```

Example 2: Converts the current destination SCREEN 13 image memory altered by PSET to a STRING value. SCREEN 13 only.

```
SCREEN 13
PSET (0, 0), ASC("H") 'top left corner of screen
PSET (1, 0), ASC("E")
PSET (2, 0), ASC("L")
PSET (3, 0), ASC("L")
PSET (4, 0), ASC("O")

DIM m AS _MEM
m = _MEMIMAGE(0)  'copy the screen memory to m
x1$ = _MEMGET(m, m.OFFSET, STRING * 5) 'get at block start position
LOCATE 2, 1:PRINT LEN(x1$) 'prints 5 bytes as size is STRING * 5
PRINT x1$ 'prints HELLO as ASCII character values
PRINT m.OFFSET; m.SIZE; m.ELEMENTSIZE
_MEMFREE m
```

```
 5
HELLO
 5448320  6400  1
```

Explanation: When a numerical _BYTE value is converted to a STRING, each byte is converted to an ASCII character. The QB64 IDE will capitalize _MEM dot values.

```
m.SIZE = 320 * 200 = 6400 bytes
m.ELEMENTSIZE = 1 byte
```

Example 3: Using _MEM to convert _OFFSET to _INTEGER64.

```
DIM x AS INTEGER
DIM m AS _MEM
m = _MEM(x)
PRINT m.OFFSET
PRINT ConvertOffset(m.OFFSET)


FUNCTION ConvertOffset&& (value AS _OFFSET)
$CHECKING:OFF
DIM m AS _MEM 'Define a memblock
m = _MEM(value) 'Point it to use value
$IF 64BIT THEN
```

```
    'On 64 bit OSes, an OFFSET is 8 bytes in size.  We can put it directly into an Integer64
    _MEMGET m, m.OFFSET, ConvertOffset&& 'Get the contents of the memblock and put the values there directly into ConvertOffset&&
$ELSE
    'However, on 32 bit OSes, an OFFSET is only 4 bytes.  We need to put it into a LONG variable first
    _MEMGET m, m.OFFSET, temp& 'Like this
    ConvertOffset&& = temp& 'And then assign that long value to ConvertOffset&&
$END IF
_MEMFREE m 'Free the memblock
$CHECKING:ON
END FUNCTION
```

Explanation: The above will print two numbers which should match. These numbers will vary, as they're representations of where X is stored in memory, and that position is going to vary every time the program is run. What it should illustrate, however, is a way to convert _OFFSET to _INTEGER64 values, which can sometimes be useful when trying to run calculations involving mem.SIZE, mem.TYPE, or mem.ELEMENTSIZE.

See also

- _MEM (function), _MEMELEMENT
- _MEMNEW, _MEMCOPY, _MEMFREE
- _MEMGET, _MEMPUT, _MEMFILL
- _MEMIMAGE, _MEMSOUND

Navigation:
Main Page with Articles and Tutorials
Keyword Reference - Alphabetical
Keyword Reference - By usage