_MEMPUT

The _MEMPUT statement writes data to a portion of a designated memory block at an OFFSET position.

Syntax

> _MEMPUT *memoryBlock*, *bytePosition*, *sourceVariable* [AS *type*]

Parameters

- *memoryBlock* is a _MEM variable type memory block name created by _MEMNEW or the _MEM function.
- *bytePosition* is the *memoryBlock*.OFFSET start position plus any bytes needed to read specific values.
- The *sourceVariable* type designates the size and *bytePosition* it should be written to. It can be a variable, array or user defined type.
- *bytePosition* can be converted AS a specific variable *type* before being written to the *memoryBlock* as bytes.

Description

- The _MEMPUT statement is similar to the PUT file statement, but *bytePosition* is required.
- The *memoryBlock*.OFFSET returns the starting byte position of the block. Add bytes to move into the block.
- The variable type held in the memory block can determine the next *byte position* to write a value.
- LEN can be used to determine the byte size of numerical or user defined variable types regardless of the value held.
- STRING values should be of a defined length. Variable length strings can actually move around in memory and not be found.

Description

Example: _MEMPUT can be used just like POKE without DEF SEG.

```
DIM o AS _MEM
o = _MEM(d&)
_MEMPUT o, o.OFFSET + 1, 3 AS _UNSIGNED _BYTE  'POKE
v = _MEMGET(o, o.OFFSET + 1, _UNSIGNED _BYTE) 'PEEK
PRINT v 'prints 3
PRINT d& 'print 768 because the 2nd byte of d& has been set to 3 or 3 * 256
```

See also

- _MEMGET, _MEMGET (function)
- _MEM, _MEM (function)
- _MEMIMAGE, _MEMNEW
- _MEMFREE, _MEMCOPY

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