BitShift Obtain result of left- or right-shifted 32-bit value

#include <ToolUtils.h> **Toolbox Utilities**

```
long    BitShift(op , count );
long    op ;        32-bit values to be shifted
short   count;      positive shifts left; negative shifts right
        returns     result of (op << count) or (op >> (-count))
```

BitShift returns the value of bit-shifted a 32-bit operand. Bit shifting can be used for fast multiplication and division by a power of 2.

op is a 32-bit long operand.

count specifies the direction and extent of the shift operation. The bits are shifted ABS(*count*) positions. *count* is always treated MOD 32 (should range from -31 to +31).

- <0 (negative values) shift to the right
- >0 (positive values) shift to the left
- 0 Nothing happens

Returns: a long integer; the result of (*op* **<<** *count*) or (*op* **>>** (–*count*)).

Notes: Bits of *op* are shifted either right or left, depending on the sign of *count* . Bits shifted off the end are lost and the vacated positions are cleared to 0. Note that a left shift is the same as an unsigned multiplication by 2, 4, 8, etc. A right shift is the same as unsigned division by 2, 4, 8, etc.

This capability is native to the CPU and can be performed much faster using the C **>>** and **<<** (bitwise shift) operators or the assembler LSL or LSR opcodes.

```
long    x, operand;
short   count;

x = BitShift( operand, count );     /* is equivalent to . . . */
x = operand << count;               /* if count is positive or . . . */
x = operand >> (-count);            /* if count is negative */
```