BitTst Determine state of a bit in a bit string

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

Boolean	**BitTst(***bytePtr*, *bitNum* **);**	
Ptr	*bytePtr* ;	address of the byte at the start of "bit string"
long	*bitNum* ;	0-based ID of bit to check
	returns	Is the bit a 1?

This returns TRUE if a specified bit is a 1; FALSE if it is a 0.

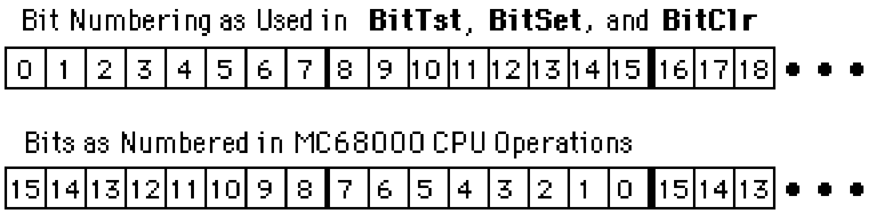
bytePtr is the address of the first byte of a sequence of bytes.

bitNum identifies the bit to test, as a positive offset from the first bit in the byte addressed by *bytePtr*.. Bits are identified by a logical mapping (matching that used for screen pixels), rather than the normal high-to-low numbering used in CPU operations. See Notes, below.

Returns: a Boolean value indicating the state of the bit. It will be one of:

FALSE	Bit *bitNum* is 0 (by convention, white or OFF)
TRUE	Bit *bitNum* is 1 (black or ON)

Notes: This function does some address arithmetic to overcome difficulties surrounding MC68000 even-address restrictions and the normal right-to-left bit ordering. The result is that you can treat any area of memory (as much as 16 Megabytes) as a string of sequentially-numbered bits.


Bit Numbering as Used in **BitTst**, **BitSet**, and **BitClr**
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 • • •
Bits as Numbered in MC68000 CPU Operations
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 15 14 13 • • •

Any bit in the bit string can be accessed individually via **BitTst**, **BitSet**, and **BitClr**. Other Toolbox **Bit**Xxx functions apply to bitwise operations between long integers and not relevant for C programmers.