I Wish I Knew How To …

Use MemoryBlocks With Xojo

Sep 2017 Edition (1.0)

By Eugene Dakin

Table of Contents

Chapter 2 – Classic Bitwise

Comparing bits is very helpful when performing advanced programming tasks, such as writing a driver for an electrical component, or controlling a program over the internet, and the list just keeps getting bigger. Bits can also be associated with Boolean values, because there are only one of two possible choices. Bits can be on (1) or off (0), while Boolean can be true (1) or false (0).

Bits are very useful and one way to retrieve some of the data from bits is to perform logical comparisons, and there are typically 4 types: AND, OR, XOR, and NOT. These values are commonly referred to in mathematical tables called logic tables, or truth tables. Each of these functions will be shown below.

AND

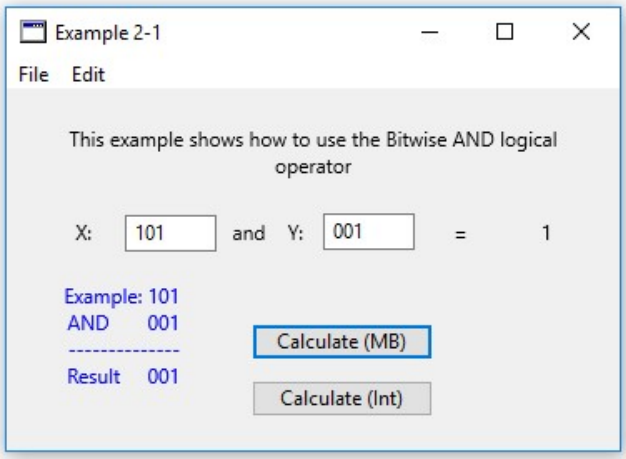
The AND logical comparison means that both X ***and*** Y inputs must be true in order for the output to be true. If one of the inputs are false (0), then the output will also be false (0). The AND comparison is a binary operator which has two operands (X and Y Input). Below is the truth table for this logic.

Table 6. AND Truth Table

AND Inputs		Output
X	Y	Z
0	0	0
0	1	0
1	0	0
1	1	1

Each of the columns are added together with logic math for the AND comparator. There are a few methods to perform AND logic, and one is with MemoryBlocks while the other is with Integer variables. Both are included in Example 2-1.

Figure 21. Example 2-1: Logical AND Screen Grab



The example AND text in blue shows that 101 AND 001 results in 001. From the truth table, 1 AND 1 is 1, 0 AND 0 is 0, 1 AND 0 is 0. Xojo automatically removes the preceding 0's, and the value 001 is converted to 1.

Below is the MemoryBlock method to perform Bitwise AND.

Code 14. Example 2-1: MemoryBlock Bitwise

```
Sub Action() Handles Action
  //Create 3 MemoryBlock Variables
  )//8 bytes = 64 bits
  Dim X as New MemoryBlock(8)
  Dim Y as New MemoryBlock(8)

  //Populate X and Y with binary text data
  X.UInt64Value(0) = Val("&b" + TFX.Text)
  Y.UInt64Value(0) = Val("&b" + TFY.Text)

  //Calculate Bit AND result
  Dim Z as New MemoryBlock(8)
  Z.UInt64Value(0) = Bitwise.BitAnd(X.UInt64Value(0),Y.UInt64Value(0))

  //Show Bit AND result
  LblAnswer.Text = Z.UInt64Value(0).ToBinary
End Sub
```

Two memory blocks are created with a size of 8 bytes, and this is equal to 64 bits (8*8=64). Each memory block variable is populated with values from the two text field controls in Window1. A third memory block is created to hold the result of the AND value. The Bitwise bitand command is used to logically compare these two values and return the result from the bit logical comparison.

Code 15. Example 2-1: Integer Bitwise

```
Sub Action() Handles Action
  //Create 3 integer variables
  Dim X, Y, Z as Integer
  //Populate X and Y with binary text data
  X = Val("&b" + TFX.Text)
  Y = Val("&b" + TFY.Text)

  //Calculate Bit AND result
  Z = Bitwise.BitAnd(X,Y)

  //Show Bit AND result
```

```
  LblAnswer.Text = Z.ToBinary
End Sub
```

Using bitwise with an integer is very like the memory block example. Integer variables are created, binary text from the text fields are converted to an integer value and this value is then stored in the X and Y variables. The bitwise AND function works on the two variables and returns the result which is then shown in the label lblAnswer.

This example shows how to use classic memory blocks and integer variables to perform the logical AND function on values and show the result to the user.

OR

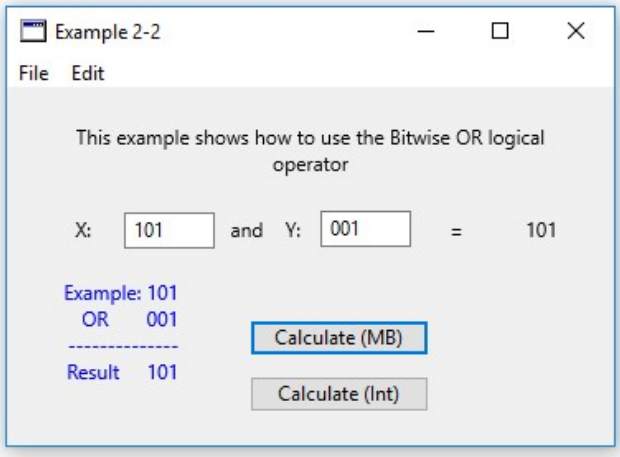
The OR logical comparison means that either X ***or*** Y inputs can be true for the output to be true. If one of the inputs are true (1), then the output will also be true (1). The OR comparison is a binary operator which has two operands (X and Y Input). Below is the truth table for this logic.

Table 7. AND Truth Table

OR Inputs		Output
X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	1

Each of the columns are added together with logic math for the OR comparator. There are a few methods to perform OR logic, and one is with MemoryBlocks while the other is with Integer variables. Both are included in Example 2-2.

Figure 22. Example 2-2: Logical OR Screen Grab



Index

The 'I Wish I Knew' series contains technical data and advice that makes sense and contains practical and numerous examples with explanations to allow you to ease into the steep programming curve. You can extend Xojo applications today!

This book "I Wish I Knew How to … Use Memory Blocks with Xojo" shows you how to use the powerful memory management of your computer to increase the program execution speed and to be able to more comfortably use this data type.

Examples include, structures to memoryblocks and back, mixed data, saving memoryblock data with binary stream files, parameters with ByRef and ByVal, Ones' Complement, Twos' Complement, Structures, New Framework Append, mixed data, and more. All of these examples use the classic and new framework memoryblock versions on Windows, Mac, and Ubuntu OS's.

There are 10 chapters and contains over 180 pages with over 55 example programs.
Many screenshots have been added to show the results of the code with an index to help find topics quickly.

This is one of many books at xDevLibrary. This book can be purchased at http://XojoLibrary.com where many great Xojo resources are available.

Happy programming!

Eugene

Eugene Dakin MBA, Ph.D., P.Chem., is an author of Xojo reference materials and has many years of experience in the programming industry. Another great reference book is *I Wish I Knew How To … Program OpenGL Core*.

ISBN: 978-1-927924-18-1