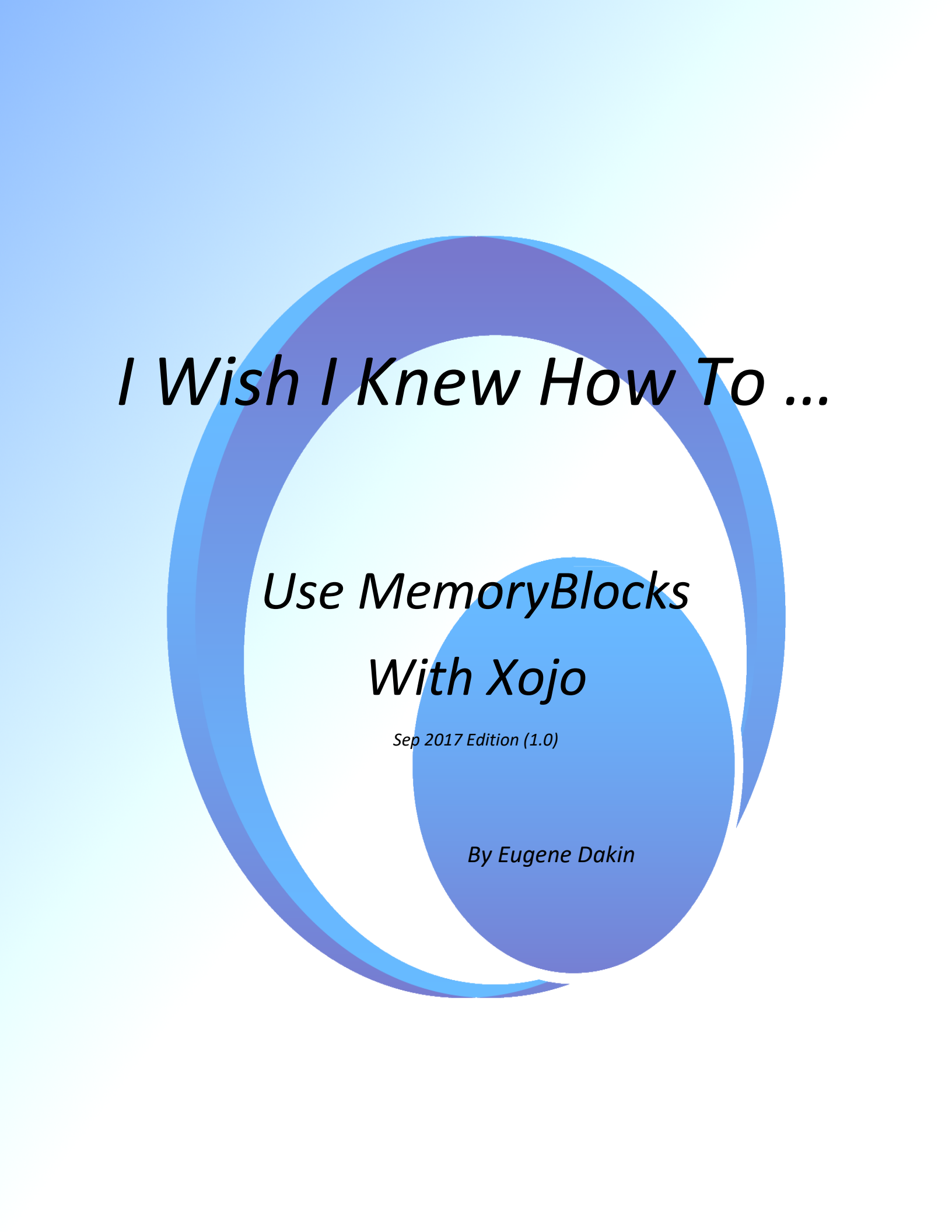




I Wish I Knew How To ...

*Use MemoryBlocks
With Xojo*

Sep 2017 Edition (1.0)

By Eugene Dakin

Table of Contents

Chapter 1 – Introduction to Programming with MemoryBlocks	7
Why use Memory Blocks?.....	7
Introduction to Memory Blocks - Quick start	8
Get MemoryBlock Address	12
MemoryAddress Pointing	14
Bits and Bytes.....	15
How is Data Read	17
How is Data Stored (Endian Type)	17
Variables.....	19
Data Types and Sizes.....	20
Reading 8 of 64 bits.....	22
Reading 16 of 64 bits.....	22
Reading 32 of 64-bits	22
Reading 64 of 64-bits	22
Mutable and Immutable	23
Decimal, Hexadecimal, Octal, and Binary Output.....	24
Decimal, Hexadecimal, Octal, and Binary Input.....	26
Ways to Create a MemoryBlock	28
MemoryBlock to Method.....	31
Using Nil in MemoryBlocks	31
MemoryBlocks and the Heap/Stack.....	31
Encoding.....	33
Data Types.....	33
Chapter 2 – Classic Bitwise.....	35
AND	35
OR.....	37
XOR.....	39
NOT	41
Bitwise ShiftLeft	43
Bitwise ShiftRight	45
Ones' Complement	48

Twos' Complement	51
Chapter 3 – Classic Reading and Writing	54
Wide String Types	54
C Style String	57
Adding WStrings.....	59
Integers	61
Integer Addition	62
Doubles	63
Manual Fixed MB List.....	65
Structure Fixed List	67
Chapter 4 – Classic MemoryBlock Methods	70
ByRef and ByVal	70
ByVal Input Parameter.....	71
ByRef Input Parameter.....	74
Pass Structure to Method	77
Chapter 5 – Classic Binary Stream	80
Write Text to Binary File	80
Mixed Data Types.....	83
Structure Binary File.....	87
MB to Binary Stream.....	91
Chapter 6 – New Framework Reading and Writing	95
Create MemoryBlock String.....	97
String Encoding Type.....	99
Integers	101
Integer Addition	102
Doubles	104
Manual Fixed MB List.....	106
Structure Fixed List	108
Chapter 7 – New Framework MemoryBlock Methods	111
ByRef and ByVal	111
ByVal Input Parameter.....	112
ByRef Input Parameter.....	115
Pass Structure to Method	118

Convert Data Classic MB to New MB and Back	120
Chapter 8 – New Framework Append, replace, etc.....	122
Append Text.....	122
Insert Text	124
Remove Text	126
Left and Right Replacement.....	127
Mid Replacement.....	129
Append Mixed Integer	131
Insert Mixed Integer	133
Remove Integer.....	136
Chapter 9 – New Framework Binary Stream	139
Write Text to Binary File	139
Mixed Data Types.....	142
Structure Binary File.....	147
MB to Binary Stream.....	152
Chapter 10 – New Framework Bitwise	157
AND	157
OR.....	159
XOR.....	161
NOT	163
Bitwise ShiftLeft	165
Bitwise ShiftRight	167
Ones' Complement	170
Twos' Complement	173
Appendix A – Number Data Type Ranges	176
Appendix B: Frequently Asked Questions.....	177
Index.....	178

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

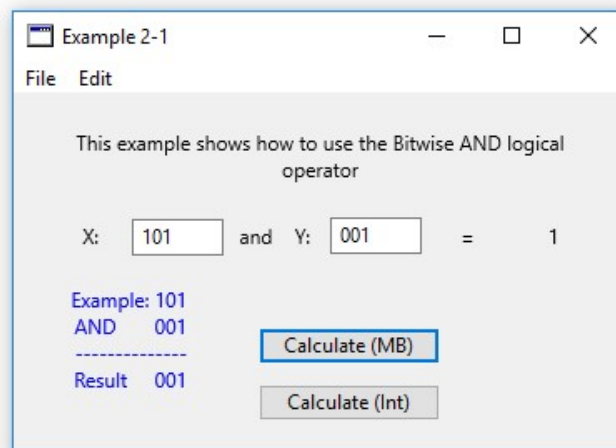
The AND logical comparison means that both X **and** Y inputs must be true in order for the output to be true. If one of the inputs are false (0), then the output will also be false (0). The AND comparison is a binary operator which has two operands (X and Y Input). Below is the truth table for this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

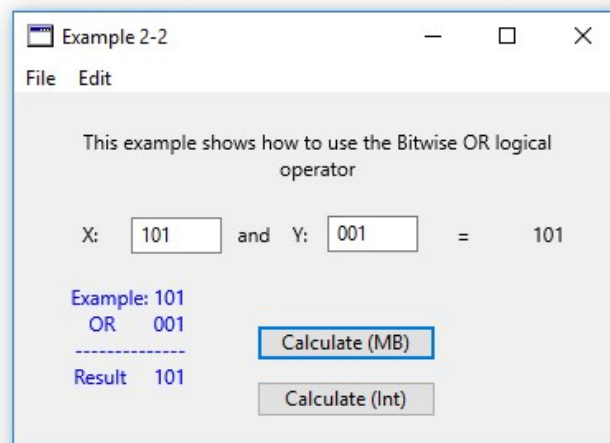
The OR logical comparison means that either X **or** Y inputs can be true for the output to be true. If one of the inputs are true (1), then the output will also be true (1). The OR comparison is a binary operator which has two operands (X and Y Input). Below is the truth table for this logic.

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

- &b, 26
- &h, 26
- &o, 26
- AND, 35, 157
- Append, 123
- ASCII, 96
- Binary Operator, 35, 39, 157, 161
- BinaryStream, 81
- Bits, 15
- Bitwise, 35, 157
 - AND, 35, 157
 - NOT, 41, 163
 - OR, 37, 159
 - ShiftLeft, 43, 165
 - XOR, 39, 161
- Boolean, 20
- ByRef, 70, 111
- Byte, 20
- Bytes, 15
- ByVal, 70, 111
- Classic Integers
 - Byte, 61
 - Int16Value, 61
 - Int32Value, 61
 - Int64Value, 61
 - Int8Value, 61
 - Long, 61
 - Short, 61
 - UInt16Value, 61
 - UInt32Value, 61
 - UInt64Value, 61
 - UInt8Value, 61
 - UShort, 61
- Code, 32
- Compare Logic, 35, 157
- Constants, 32
- ConvertEncoding, 85
- Create, 82
- Data Types
 - Byte, 176
 - Double, 176
 - Int16Value, 176
 - Int32Value, 176
 - Int64Value, 176
 - Int8Value, 176
 - Integer, 176
 - Long, 176
 - Short, 176
 - Single, 176
 - UInt16Value, 176
 - UInt32Value, 176
 - UInt64Value, 176
 - UInt8Value, 176
 - UInteger, 176
 - UShort, 176
- Desktop, 81
- Encoding
 - ASCII, 96
 - UTF16, 96
 - UTF16BigEndian, 96
 - UTF16LittleEndian, 96
 - UTF32, 96
 - UTF32BigEndian, 96
 - UTF32LittleEndian, 96
 - UTF8, 96
- Encodings, 86
- Endian Type, 17
- Examples
 - 01-01
 - MemoryBlock Intro, 11
 - 01-02
 - MemoryBlock Address, 12
 - 01-03
 - Show Memory Address, 14
 - 01-04
 - Variable vs MemoryBlock, 20
 - 01-05
 - Data Types and Sizes, 22
 - 01-06
 - Mutable/Immutable, 24
 - 01-07
 - Data Type Output, 25
 - 01-08
 - Data Type Input, 27
 - 02-01
 - MemoryBlock AND, 36
 - 02-02
 - MemoryBlock OR, 38

- 02-03
 - MemoryBlock XOR, 40
- 02-04
 - MemoryBlock NOT, 42
- 02-05
 - MemoryBlock ShiftLeft, 44
- 02-06
 - MemoryBlock ShiftLeft, 46
- 02-07
 - Ones Complement, 49
- 02-08
 - Twos Complement, 52
- 03-01
 - WString, 55
- 03-02
 - CString, 57
- 03-03
 - WString Addition, 60
- 03-04
 - Integer Addition, 62
- 03-05
 - Double Addition, 64
- 03-06
 - Fixed-Length MB List, 66
- 03-07
 - Fixed-Length Struct List, 68
- 04-01
 - ByVal Int Variable and MB, 72
- 04-02
 - ByRef Int Variable and MB, 75
- 04-03 Structures Method, 78
- 05-01 Text BinaryStream, 81
- 05-02 Mixed BinaryStream, 84
- 05-03 Structure BinaryStream, 88
- 05-04 MemoryBlock and BinaryStream, 92
- 06-01 New Framework MemoryBlock, 97
- 06-02 String Encoding, 99
- 06-03 Get/Set Integers, 101
- 06-04
 - Integer Addition, 102
- 06-05
 - Double Addition, 104
- 06-06
 - Fixed-Length MB List, 107
- 06-07
 - Fixed-Length Struct List, 109
- 07-01
 - ByVal Int Variable and MB, 113
- 07-02
 - ByRef Int Variable and MB, 116
- 07-03 Structures Method, 119
- 07-04 Classic MB to New and Back, 121
- 08-01 Mutable Append, 123
- 08-02 Mutable Insert, 124
- 08-03 Mutable Remove, 126
- 08-04 Mutable Left Right, 128
- 08-05 Mutable Mid, 130
- 08-06 Append UInt32, 132
- 08-07 Insert Mixed Integer, 134
- 08-08 Remove Mixed Integer, 137
- 09-01 Text BinaryStream, 140
- 09-02 Mixed BinaryStream, 143
- 09-03 Structure BinaryStream, 148
- 09-04 MemoryBlock and BinaryStream, 153
- 10-01
 - MemoryBlock AND, 158
- 10-02
 - MemoryBlock OR, 160
- 10-03
 - MemoryBlock XOR, 162
- 10-04
 - MemoryBlock NOT, 164
- 10-05
 - MemoryBlock ShiftLeft, 166
- 10-06
 - MemoryBlock ShiftLeft, 168
- 10-07
 - Ones Complement, 171
- 10-08
 - Twos Complement, 174
- Extensions, 81
- FileType, 81
- Filter, 81
- Global, 32
- Insert, 125
- Insert Mutable, 124
- Int16, 20
- Int32, 20
- Int32Value(0), 8
- Int64, 20
- Int8, 20
- Left, 128
- Left Most Bit, 48, 51, 170, 173
- Literal Numerical Types, 26
- Logic Tables, 35, 157
- Logical Comparisons, 35, 157

MemoryBlocks Quickstart, 8
Mid, 98, 130
Most Significant Bit, 48, 51, 170, 173
Mutable Remove, 126
New, 32
New MemoryBlock, 8
Nibble, 22
Nil, 32
NOT, 41, 163
Ones' Complement, 22, 48, 170
OR, 37, 159
Pop, 32
Popped, 32
Ptr, 20
Push, 32
Remove, 127
Right, 128
SaveAsDialog, 81
ShiftLeft, 43, 165
ShowModal, 81
Signed, 16, 48, 51, 170, 173
SpecialFolder, 81
Stack, 32

Structure BinaryFile, 87, 147
SuggestedFileName, 81
Truth Table, 35, 157
Twos' Complement, 51, 173
UInt16, 20
UInt32, 20
UInt64, 20, 85
UInt8, 20, 85
Unary Operator, 41, 163
Unsigned, 16, 48, 51, 170, 173
UTF16, 96
UTF16BigEndian, 96
UTF16LittleEndian, 96
UTF32, 96
UTF32BigEndian, 96
UTF32LittleEndian, 96
UTF8, 96
Val, 64
Variables, 19
Why MemoryBlocks?, 7
Write, 82
XOR, 39, 161

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