



I Wish I Knew How To ...

*Program Plugins
With Xojo on Windows*

May 2015 Edition (2.1)

For Visual Studio 2013 Community Edition

By Eugene Dakin

Table of Contents

Chapter 1 - Introduction to Xojo and the Environment	8
Preview of Chapter 1	8
What is Xojo?.....	8
Describing and Defining the User Interface	9
Setting the run-time properties	12
Chapter 2 – Install Xojo	15
Install Visual Studio C++ 2013	16
Chapter 3 - Plugin Install Fundamentals.....	19
Add Existing Xojo Windows 7, 8, and 10 Plugin (.rbx or xojo_plugin)	19
Data Types	22
Plugin SDK Reference	23
REALBuildString	23
REALclassDefinition	23
REALconstant.....	25
REALcontrol	26
REALcontrolBehaviour.....	27
REALenum.....	29
REALmethodDefinition	30
REALmodule Definition.....	31
REALproperty.....	32
REALRegisterClass.....	33
REALRegisterMethod.....	33
REALRegisterModule	33
REALstructure	34
Chapter 4 – Create a Manual Global Plugin (.xojo_plugin)	35
Condensed Notes	35
Step by Step.....	37
IGNORE C/C++ Deprecation	44
Chapter 5 – Module Plugins.....	47
Module Method	47
Condensed Notes	48
Explained	50
Module Constants	53

Condensed Notes	53
Explained	55
Module Properties	57
Condensed Notes	57
Explained	59
Module String Properties	62
Condensed Notes	63
Explained	65
Module Structure	68
Condensed Notes	69
Explained	71
Module Enum	75
Condensed Notes	75
Explained	77
Chapter 6 – Adding a Class.....	80
Class Method.....	80
Condensed Notes:	81
Explained	83
Class Number Properties.....	87
Condensed Notes	87
Explained	90
Class String Property	95
Condensed Notes	95
Explained	98
Class Constants.....	101
Condensed Notes	101
Explained	103
Shared Property	106
Condensed Notes	106
Explained	109
Shared Method.....	112
Condensed Notes	112
Explained	114
Define Default Values.....	117
Condensed Notes	117
Explained	120
Chapter 7 - Xojo Plugin Packaging	125
Package Example 6-7 Plugin.....	125
Plugin Picture	127
Chapter 8 – Controls.....	129

Line Control	129
Condensed Notes	130
Explained	135
Drawing Shapes	142
Condensed Notes	143
Explained	147
Control Number Property	154
Condensed Notes	155
Explained	159
Control String Property	168
Condensed Notes	168
Explained	172
Control Method	181
Condensed Notes	181
Explained	185
Control Constants	193
Condensed Notes	193
Explained	197
Control Click	204
Condensed Notes	205
Explained	208
Control MouseUp	216
Condensed Notes	216
Explained	220
Control GotFocus Keyboard	228
Condensed Notes	229
Explained	233
Control KeyDown	241
Condensed Notes	241
Explained	245
ASCII Control Character Tables	254
Control Open and Close	264
Condensed Notes	265
Explained	269
Control MouseWheel	277
Condensed Notes	277
Explained	281
Control Mouse Enter and Exit	289
Condensed Notes	290
Explained	294
Index	302

Module Properties

Properties are variables which can be used in the dll or within Xojo. An example would be the radius of a circle which can be changed – not all circle radiuses are the same. A property is able to have its value changed, many times, according to what the user and/or programmer would like. Data types which can be created in a property are listed in the Data Types section of Chapter 3.

The quick steps to create a module property are shown below.

Condensed Notes

- 1) Start Visual Studio 2013 Community Edition
- 2) Start *File->New->Project*
- 3) Select *Templates->Visual C++,* and select *Win32 Project*
- 4) Choose the location for the files to be placed, I made Example5-3 folder at the location:
D:\user\Xojo\Eugene\Plugins\Chapter5\Example5-3
- 5) Name the project *Prop*
- 6) Press the *OK* button
- 7) Press *Next* button
- 8) Select *DLL, Export Symbols, and Precompiled Header (unselect everything else)*
- 9) Press *Finish*
- 10) Copy the following files from the directory: C:\Program Files (x86)\Xojo\Xojo 2015r1\Extras\PluginsSDK\Includes and paste them in the directory:
D:\user\Xojo\Eugene\Plugins\Chapter5\Example5-3\Prop\Prop (rb_plugin.h, rb_plugin_cpp.h, REALplugin.h, and WinHeader++.h)
- 11) Copy the following file from the directory: C:\Program Files (x86)\Xojo\Xojo 2015r1\Extras\PluginsSDK\GlueCode and paste them in the directory:
D:\user\Xojo\Eugene\Plugins\Chapter5\Example5-3\Prop\Prop (PluginMain.cpp)
- 12) Remove the following files from Visual Studio: stdafx.h, stdafx.cpp, ReadMe.txt, targetver.h, and dllmain.cpp.
- 13) Drag and drop files from the folder: D:\user\Xojo\Eugene\Plugins\Chapter5\Example5-3\Prop\Prop into the Visual Studio Solution Explorer: rb_plugin.h, rb_plugin_cpp.h, REALplugin.h, WinHeader++.h, and PluginMain.cpp.
- 14) Delete the file Prop.h
- 15) Delete all the code in Prop.cpp and add the following code:

```
#include "WinHeader++.h"  
#include "rb_plugin.h"  
  
// Dimension the methods
```

```

// use DOUBLE (not double) for Xojo
static DOUBLE RadiusGetter(void);
static void RadiusSetter(DOUBLE RadiusValue);

//Make properties
REALproperty PropModule[] = {
    { "", "RadiusValue", "double", REALconsoleSafe, (REALproc)RadiusGetter,
    (REALproc)RadiusSetter },
};

//Create property module
REALmoduleDefinition PropModuleDefinition = {
    kCurrentREALControlVersion, //Version
    "Prop", //Module name
    nil, //Method names
    0, //Name count
    nil, //Constant array
    0, //constant count
    PropModule, //Module properties
    sizeof(PropModule) / sizeof(REALproperty), //Property count
    nil, //Structure array
    0, //structure count
    nil, //Enum array
    0, //enum count
};

//Perform property functions
static DOUBLE DBRadius;
static DOUBLE RadiusGetter(void)
{
    //Return the DOUBLE value
    return DBRadius;
}
static void RadiusSetter(DOUBLE AValue)
{
    //Store the double value
    DBRadius = AValue;
}
void PluginEntry(void){
    REALRegisterModule(&PropModuleDefinition);
}

```

- 16) Change Prop Property->C/C++->Precompiled Headers to *Not Using Precompiled Headers* in Debug and Release Configuration.
- 17) In the main IDE, select Release Win32, and *Rebuild* solution. There should be two *#pragma deprecated* warnings and there should be 1 succeeded, 0 failed, and 0 skipped
- 18) Copy the dynamic link library Prop.dll from the D:\user\Xojo\Eugene\Plugins\Chapter5\Example5-3\Prop\Release folder and place it in the Xojo plugin folder at: C:\Program Files (x86)\Xojo\Xojo 2015r1\Plugins
- 19) Start Xojo and add a pushbutton action event and add the following code and run the program

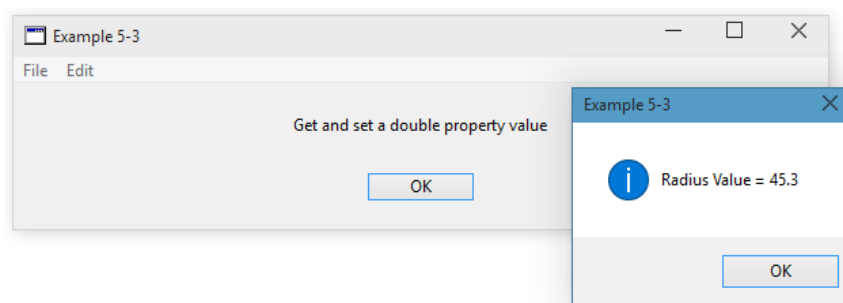
```
//Assign a value  
Prop.RadiusValue = 45.3  
//Get the property value  
MsgBox "Radius Value = " + CStr(prop.RadiusValue)
```

The number 45.3 should appear in the messagebox which was a properties stored and retrieved from the plugin.

Explained

Example 5-3 shows how to set a property value with a Xojo Plugin and also to get the value. On Xojo the property value will use code that is the same as a module created in Xojo.

Figure 39. Example 5-3: Xojo Screen Grab



One property is created in Visual Studio 2013 and has two methods, a setter and a getter method. The setter method takes a numerical value and places it in the property value. The getter method retrieves the property value and returns this value back to the variable in Xojo.

Code for this example is shown below.

Figure 40. Example 5-3: Variable Property VS 2013 Code

```

#include "WinHeader++.h"
#include "rb_plugin.h"

// Dimension the methods
// use DOUBLE (not double) for Xojo
static DOUBLE RadiusGetter(void);
static void RadiusSetter(DOUBLE RadiusValue);

//Make properties
REALproperty PropModule[] = {
    { "", "RadiusValue", "double", REALconsoleSafe, (REALproc)RadiusGetter, (REALproc)RadiusSetter },
};

//Create property module
REALmoduleDefinition PropModuleDefinition = {
    kCurrentREALControlVersion, //Version
    "Prop", //Module name
    nil, //Method names
    0, //Name count
    nil, //Constant array
    0, //constant count
    PropModule, //Module properties
    sizeof(PropModule) / sizeof(REALproperty), //Property count
    nil, //Structure array
    0, //structure count
    nil, //Enum array
    0, //enum count
};

//Perform property functions
static DOUBLE DBRadius;
static DOUBLE RadiusGetter(void)
{
    //Return the DOUBLE value
    return DBRadius;
}

static void RadiusSetter(DOUBLE AValue)
{
    // Store the DOUBLE value
    DBRadius = AValue;
}

void PluginEntry(void){
    REALRegisterModule(&PropModuleDefinition);
}

```

Above code for Example 5-3 begins the same way as other module plugins with the WinHeader++ and rb_plugin headers being *included*. Both methods are declared at the beginning of code and then the functioning code can be placed after REALmoduleDefinition. If functional code is placed after REALmoduleDefinition and no declaration of the method exists at the before REALmoduleDefinition (example: `static DOUBLE RadiusGetter(void);`) then an error will occur when compiling the program.

REALproperty defines the properties used for a single property. When a number is to be placed in RadiusValue, the RadiusSetter method is called. Getting data from the property will require the RadiusGetter method to be called. The array PropModule is can store many properties.

REALproperty has no value ("") for the group, the sub name to be used in Xojo for accessing this property is RadiusValue, and this is a double value. There are a seven possible flags for REALproperty (see Chapter 3 PluginSDK Reference) and this is console safe, which can be used in console and desktop projects. Both RadiusGetter and RadiusSetter procedures (REALproc) are declared.

Code 8. Property Module Definition

```
//Create property module
REALmoduleDefinition PropModuleDefinition = {
    kCurrentREALControlVersion, //Version
    "Prop", //Module name
    nil, //Method names
    0, //Name count
    nil, //Constant array
    0, //constant count
    PropModule, //Module properties
    sizeof(PropModule) / sizeof(REALproperty), //Property count
    nil, //Structure array
    0, //structure count
    nil, //Enum array
    0, //enum count
};
```

The property definition begins with the version of the module which is almost always used the constant `kCurrentREALControlVersion`. The name of the module is the word that is typed before the added methods and in this case is *Prop*, which is the shortened version for property. All of the properties have been defined in the array *PropModule* and are added and the number of constants (using `sizeof`) are also added. Since there are no methods, constants, structures, or enums, then these are all left as Nil and 0 values. Later examples in this chapter will show how some of these are used.

The functional part of the properties are then coded.

Code 9. Functional Properties

```
//Perform property functions
static DOUBLE DBRadius;
static DOUBLE RadiusGetter(void)
{
    //Return the DOUBLE value
    return DBRadius;
}
static void RadiusSetter(DOUBLE AValue)
{
    // Store the DOUBLE value
    DBRadius = AValue;
}
```

The variable `DBRadius` (shortened for `DoubleRadius`) is only used in VS 2013 which holds the `RadiusValue` number. When Xojo wants to retrieve the value from the property `RadiusValue`, the method `RadiusGetter` gets the value by Xojo and returns it in this method. Similarly, when a

new value is placed in the property (Prop.RadiusValue = 45.3) then the value is stored with the RadiusSetter method.

All of the created properties which have been built need to be registered in the module.

```
void PluginEntry(void){  
    REALRegisterModule(&PropModuleDefinition);  
}
```

The pluginEntry is where the properties are registered with the REALRegisterModule command. All of the properties are added – usually in arrays.

Once the dll has been built with the Build-> Rebuild Solution menu item, the dll is copied to the Xojo plugin folder. Xojo is restarted to load the plugin data and the following code is added to a pushbutton in a new Xojo program.

Code 10. Example 5-3: Xojo Program with dll Properties

```
//Assign a value  
Prop.RadiusValue = 45.3  
//Get the property value  
MsgBox "Radius Value = " + CStr(Prop.RadiusValue)
```

Code where the value of 45.3 is placed in the property calls the RadiusSetter method in the dll, and retrieving the data (Cstr(Prop.RadiusValue)) gets the value by calling RadiusGetter in the dll.

Example 5-3 shows how to create a double numerical property that can get and set the value.

Module String Properties

String properties in a module are slightly different than numerical values because the string reference should be locked and/or unlocked with use, otherwise the string can be released which could cause errors with a dangling pointer. This method also includes the previous

Index

- #pragma deprecated, 38
- _CRT_SECURE_NO_WARNINGS, 44
- Add Class, 79
- Add Existing Plugin, 19
- ASCII Control Characters, 253
- Change Code Property, 13
- Class
 - Constants, 100
 - Default Values, 116
 - Method, 79
 - Number Property, 86
 - Shared Method, 111
 - Shared Property, 105
 - String Property, 94
- Class adding, 79
- Close, 263
- Control
 - REALcontrolAcceptFocus, 26
 - REALcontrolFocusRing, 26
 - REALcontrolIsHIViewCompatible, 27
 - REALcontrolOwnsCursor, 26
 - REALcontrolRequiresComposite, 27
 - REALdontEraseBackground, 27
 - REALdontTryHIViewize, 27
 - REALinvisibleControl, 26
 - REALopaqueControl, 26
- Control Click, 203
- Control Constants, 192
- Control GotFocus Keyboard, 227
- Control KeyDown, 240
- Control Method, 180
- Control Mouse Enter, 288
- Control Mouse Exit, 288
- Control Mouse Wheel, 276
- Control MouseUp, 215
- Control Number Property, 153
- Control Open and Close, 263
- Control Shape Drawing, 141
- Control String Property, 167
- Controls, 128
- Create a Plugin from Template, 46
- Data Types, 22
 - boolean, 22
 - Byte, 22
 - CFStringRef, 22
 - char, 22
 - color, 22
 - const char, 22
 - const unsigned char, 22
 - Const wchar_t, 22
 - CString, 22
 - currency, 22
 - Currency, 22
 - double, 22

- float, 22
- Int16, 22
- Int32, 22
- Int64, 22
- Int8, 22
- Integer, 22
- long, 22
- Object, 22
- PString, 22
- Ptr, 22
- RInt64, 22
- REALarray, 22
- REALbasic Array Type, 22
- REALCurrency, 22
- REALObject, 22
- REALString, 22
- short, 22
- Single, 22
- String, 22
- UInt16, 22
- UInt32, 22
- UInt64, 22
- UInt8, 22
- unsigned char, 22
- unsigned long, 22
- unsigned RInt64, 22
- unsigned short, 22
- Variant, 22
- void, 22
- WString, 22
- deprecated, 38
- Deprecation Ignore, 44
- DrawRect, 147
- Examples
 - 01-01 Xojo Program, 14
 - 04-01 Manual Plugin, 43
 - 05-01 Module Method, 49
 - 05-02 Module Constants, 54
 - 05-03 Module Property, 59
 - 05-04 Module String Property, 65
 - 05-05 Module Structures, 71
 - 05-06 Module Enum, 77
 - 06-01 Class Method, 82
 - 06-02 Class Number Property, 89
 - 06-03 Class String Property, 97
 - 06-04 Class Constants, 102
 - 06-05 Shared Property, 108
 - 06-06 Shared Method, 113
 - 06-07 Initial Property Values, 119
 - 08-01 Line Control Code, 132, 135
 - 08-02 Shapes Control Code, 146
 - 08-03 Control Number Property, 158
 - 08-04 Control String Property, 171
 - 08-05 Control Method, 184
 - 08-06 Control Constant, 196
 - 08-07 Control Click Event, 207
 - 08-08 Control Mouse Up Event, 219
 - 08-09 GotFocus Control Event, 232
 - 08-10 KeyDown Control Event, 244
 - 08-11 Open and Close Control Event, 268
 - 08-12 Mouse Wheel Control Event, 280
 - 08-13 Mouse Pointer Enter and Exit Control Events, 293
- Global, 31
- Graphics
 - ClearRect, 141
 - Clip, 141
 - DrawCautionIcon, 141
 - DrawLine, 141
 - DrawNotelcon, 141
 - DrawObject, 141
 - DrawOval, 141
 - DrawPicture, 141
 - DrawPolygon, 141
 - DrawRect, 141
 - DrawRoundRect, 141
 - DrawStopIcon, 141
 - DrawString, 141
 - FillOval, 142
 - FillPolygon, 142

FillRect, 142
FillRoundRect, 142
Pixel, 142
StringDirection, 142
StringHeight, 142
StringWidth, 142
IDE Interface, 9

Ignore Deprecation, 44

Installing Xojo, 15

Introduction to Xojo, 8

KeyDown, 240

Line Control, 128

Manual Plugin, 35

Manual Plugin Condensed, 35

Manual Plugin Step by Step, 37

Module

- Constant, 52
- Enum, 74
- Method, 46
- Property, 56
- String Property, 61
- Structure, 67

Mouse Enter, 288

Mouse Exit, 288

Mouse Wheel, 276

MsofficeAutomation.rbx Plugin, 21

Office Plugin, 21

Open, 263

Package Plugin Example, 124

packaging, 124

Picture, 126

Plugin Fundamentals, 19

Plugin Office, 21

Plugin Packaging, 124

Plugin Picture, 126

Plugin Template, 46

PluginMain.cpp, 39

Plugins Folder, 21

Precompiled Headers, 42

Private, 31

Protected, 31

Public, 31

rb_plugin.h, 39

rb_plugin_cpp.h, 39

REALBuildString, 23, 121

REALclassDefinition, 23, 82

REALClassDefinition, 84

REALconstant, 25, 100, 102

REALconstantFlags

- REALScopePrivate, 25
- REALScopeProtected, 25
- REALScopePublic, 25

REALconstantFlags

- REALScopeGlobal, 25

- REALcontrol, 26
- REALcontrolBehaviour, 27
- REALenum, 29, 77
- REALenumFlags
 - REALScopeGlobal, 30
 - REALScopePrivate, 30
 - REALScopeProtected, 30
 - REALScopePublic, 30
- REALGetControlBounds, 148, 152, 164, 177, 190, 202, 213, 225, 238, 250, 273, 286, 298
- REALgraphics, 147
- REALGraphicsDrawString, 147, 148
- REALLockString, 65, 66, 98, 110, 121, 179
- REALmethodDefinition, 30, 82
 - REALconsoleOnly, 31
 - REALconsoleSafe, 30
 - REALScopeGlobal, 31
 - REALScopePrivate, 31
 - REALScopeProtected, 31
 - REALScopePublic, 31
- REALmoduleDefinition, 31
- REALplugin.h, 39
- REALproperty, 32, 89
- REALProperty, 63
- REALpropertyFlags
 - REALconsoleOnly, 32
 - REALconsoleSafe, 32
 - REALpropRuntimeOnly, 32
 - REALScopeGlobal, 32
 - REALScopePrivate, 32
 - REALScopeProtected, 32
 - REALScopePublic, 32
- REALRegisterClass, 33
- REALRegisterControl, 148
- REALRegisterEnum, 77
- REALRegisterMethod, 33
- REALRegisterModule, 33
- REALRegisterStructure, 72
- REALSetPropValue, 147, 148
- REALString, 64
- REALstructure, 34
- REALStructure, 69, 72
- REALstructureFlags
 - REALScopeGlobal, 34
 - REALScopePrivate, 34
 - REALScopeProtected, 34
 - REALScopePublic, 34
- REALUnlockString, 66, 98, 110, 122, 179
- Reference, 23
- Runtime Properties, 12
- Scope
 - Global, 31
 - Private, 31
 - Protected, 31
 - Public, 31
- SDK Reference, 23
- SDL, 38
- Security Development Lifecycle, 38

Setting Runtime Properties, 12

stateChangedFunction

kActivationChanged, 29

kBoundsChanged, 29

kEnabledChanged, 29

kVisibilityChanged, 29

Structure

Declaration, 30

Flags, 30

Method Name, 30

REALBuildString, 23

REALclassDefinition, 23

REALconstant, 25

REALcontrol, 26

REALcontrolBehaviour, 27

REALenum, 29

REALmethodDefinition, 30

REALmoduleDefinition, 31

REALproperty, 32

REALregisterClass, 33

REALregisterMethod, 33

REALregisterModule, 33

REALstructure, 34

Setter Function, 30

User Interface, 9

What is Xojo?, 8

Wheel, 276

Windows, 10

WinHeader++.h, 39

Xojo Installer Files, 15

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 ... Program Plugins with Visual Studio 2013 with Xojo for Windows" shows you how to create a plugin with FREE community Visual Studio 2013. All code has been tested on Xojo 2015 r2.

The book is written as a guide and reference to Xojo programmers who program Desktop Applications in Windows. Dynamic Link Libraries are created with Visual Studio to allow extra functionality in Xojo.

There are 8 chapters and contains over 300 pages with over 27 example programs.

Examples include creating a manual plugin, plugins from prebuilt templates, class constants, classes, modules, and controls. 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 Great White Software. This book can be purchased at <http://great-white-software.com/rlibrary/> where many great Xojo and Real Studio resources are available.

Happy programming!

Eugene

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

ISBN: 978-1-927924-09-9