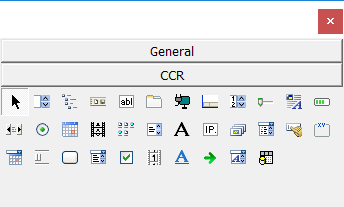
**Common Controls Replacement**

**User Guide and Utility**

For the past 6 years, a VBForums user named Krool has been developing a set of replacements for the Windows Common Controls. Replacing these controls has been discussed by many and successfully done by none, until now. Krool has been working on this primarily by himself with lots of debugging and feedback within the forum. In the middle of 2017 Krool took on the task of addressing a replacement for the MSFlexGrid control. He decided to do this development in a separate set of threads on the forum although it is very similar to the other replacement controls.

His controls will show up in your Toolbox similar to what is shown below. They function just like other controls that are built-in to VB6. Just drop them onto forms or onto other controls. Those that you can add are shown below.



What can these new controls do for you and programs you write or maintain?

* They use Unicode. If you look on the Internet you will find a control here and a control there which use Unicode but Krool’s two packages allow you to do 34 Windows common controls in one cohesive package.
* They have enhancements beyond what is found in the common controls VB6 and VBA programmers have been using for years.
* They can use themes, a complicated set of things made simpler here so that each of your programs can have a modern look and not look like it came out of Windows 95.
* The controls can be embedded into your code so that your final EXE file has no dependencies and is just the one executable file that does not require registration or installation (e.g., you could run it from a thumb drive). Alternatively, it can use the traditional control versions (using .OCX files) but with “side-by-side” so that the .OCX files don’t have to be registered on the end-user’s PC but rather reside in your program’s folder. These two techniques enable you to have truly portable solutions for your users.
* The packages get more stable all the time but users and the author are available to help with troubleshooting, inserting new features, etc. all the time.
* I have included in this package a user guide, not to each of the controls themselves but the packages in their entirety and also a utility that helps you keep up-to-date on the latest controls but also to enable you to use the OCX versions for your program development but use the StdEXE version to make the compiled code program which is totally self-contained and has no external dependencies.

[User Guide](#UserGuide)

[Introduction](#Introduction)

[Overview](#Overview)

[Controls Included](#ControlsIncluded)

[Terminology](#Terminology)

[User Guide for the Programmer](#UserGuideForTheProgrammer)

[VBCCR StdEXE Version Guide](#VBCCRStdEXEVersion)

[VBCCR OCX Version Guide](#VBCCROCXVersionGuide)

[VBFlexGrid User Guide](#VBFlexGridUserGuide)

[VBA Usage](#VBAUsage)

[OCX2StdExe Utility](#OCX2StdExeUtility)

[Update OCX Version](#UpgradeOCX)

[Compile Without the OCX Files](#CompilewoOCXs)

[Running OCX2StdEXE with Commandline Options](#CommandlineOptions)

[How I Manage Krool’s System](#IManage)

[Version History](#versionHistory)

# User Guide

Krool has a package of routines that do an incredible number of things but there are many things the programmer should understand to fully utilize these packages. Items addressed include:

* Installation
* The two different approaches, compiled .OCX controls and controls embedded you’re your program that get compiled each time with your code. How to choose which version to use and why? Can or should you use both of them?
* Set-up
* How to set up for the frequent updates to minimize the hassles of frequent updates as bugs are fixed and features are added.
* Required type libraries.
* What else do you need to use these control packages?
* What are visual themes and should I use them? How?
* What does “side-by-side mean? Should I use this and how?
* What are manifests, should I use them and how should I use them?

## Introduction

VB6 controls come in two versions. The simplest is when the source code for the control is in your program and when your code is compiled the code for the control is compiled with it. The compiled code for the control becomes part of your program. Your program is standalone and does not rely on any external files in order to run nor do any files need to be copied to or registered on the users’ PC’s. The disadvantage of this method is that the control’s code must be compiled every time you compile your code as you are developing the program. Also, most companies who author controls do not want to give away their code so they won’t distribute the source code for each control. Thus, this method of making controls is not very common.

The other way to do controls is to compile one or more controls into a file that has an OCX extension, (which stands for OLE Control eXtension or just ActiveX controls). The programmer who uses the OCX file never sees the source code and the code doesn’t need to be compiled over and over. The disadvantage of this type of control package is that the controls are not a part of the program and must be sent to each end-user and, for reasons well beyond this document, the OCX file must be registered on each user’s computer (we’ll get to side-by-side later).

Krool produced the first version of the Common Controls Replacement in late 2012. These were the un-compiled versions of source code that you would include in your code. He calls this version the StdEXE which presumably means that the standard versions of these controls get compiled into your EXE file. A bit over 5 years later (January 2017) he started providing the same controls but in the pre-compiled OCX version. So now we programmers have the same controls available in both forms.

There is no user guide for how to use each of the controls but since each of the controls is a replacement for other common controls it can be argued that such handholding is not needed for the largely very veteran VB6 programmers. However, there are several other aspects of these control sets that without some support makes them more difficult to fully sue than they have to be. Hopefully this document addresses some of those points.

Well after Krool developed his controls package of common controls he decided to add another one that had been left out of his package, a replacement for the MSFlexGrid control. He decided that although there is a lot of commonality with the first package, it is a separate package and use of it with the original package can be done but it can be confusing.

When using the StdEXE version of either set of controls, you need a type library, “OLEGuids and interface definitions”, while you are in the IDE. Also, when using the StdEXE version of the FlexGrid control, another type library is required. OCX versions do not require type libraries because in essence they are included in the OCX file.

One of the major advantages of Krool’s controls is that they are designed with the capability of using “visual styles” so that your programs don’t look like they came out of Windows 95. However, in order to use these styles you need to now how to turn this feature on and for many of us this is not intuitively obvious. Once it is turned on, though, it is really impressive.

If you use the OCX version of the controls you have to distribute the OCX file to your users with your program. Up until when VB6 came out Windows was designed such that these pre-compiled controls, dynamic link libraries, device drivers, etc. would all be registered on each user’s system and only one version (ostensibly the latest one) would be one each user’s system so that once installed and registered, many programs could make use of the same registered code. This was designed to save space on the hard drive, reduce memory usage, etc. but it caused many more problems than it solved. By the time VB6 came out, a new system had been designed and was being implemented to enable programs to use the centrally-registered files or for a program to have its own support files of its own, not copied to the central repository (the Windows System folder) and registered but instead included “side by side” with the program. If you use the StdEXE version of the controls you don’t need this (at least for these controls) because all of the code is compiled into yours but if you use the OCX version this side-by-side solution may be desirable because it enables you to distribute programs that don’t need to be installed and that don’t have components that must be installed and registered.

If you use the StdEXE version of the controls there are no “versions” and when Krool issues an update you just copy the new files over the old ones and keep going. However, the OCX version does have versions (like all pre-compiled code, see long paragraph above) and it is not a trivial matter to modify each of your programs to use the newer version. My utility takes care of this.

Since we have both the OCX and the StdEXE versions of the controls, it is possible to improve your programming experience by enabling you to use the OCX version during development (easier to use and much shorter compile time) but then do the final compile with the StdEXE version to get all of the control code included in the program such that there is no OCX file to include with the program and you don’t have to deal with the side-by-side complications. My utility takes care of this.

## Overview

There are two versions of each control set; one is a single file with the extension of OCX that contains the pre-compiled version of the controls. This would need to be registered on your (programmer) computer.

You reference this in your program and then you have access to all of the controls. This is simple from a programmer’s perspective but when you distribute your finished EXE or DLL file you have to include the OCX file. Further, this file either needs to be part of a “side-by-side” solution or it needs to be copied to and registered on the user’s PC.

The other approach Krool calls the StdEXE version. In this one has all of the code in a large group of files (153 currently) that are all un-compiled and have to be added as modules to your program. Each time you compile your program all of these controls get compiled too. There is a major hassle of even knowing which files to include in your program for which controls, some of the names of subs, functions and variables can conflict with names you use and compiling each time of all of this control code takes a long time. I have a fairly fast PC and just compiling the ComCtlsDemo program Krool provides takes over 25 seconds. I compile a lot in VB6 as I code, partly just to check syntax and logic errors, and a compile time of 25+ seconds is very irritating especially when I know I can get the same code to compile in the OCX version in 2.8 seconds. I love the fact that there are no external dependencies in the final executable but I hate the compile time. If only there was a way to use the OCX version with its fast compile times during development but a final compile using the internal controls to eliminate the need to have a separate OCX file to distribute with the program. My utility does that.

Also, there are many updates to these controls. The un-compiled files (the StdEXE version) really don’t have version numbers and as long as you copy the latest files over the earlier ones in the same location you won’t have any issues. However, this is not true of the OCX version. I put mine in C:\Windows\SysWOW64 and as long as we have the same version we can just copy the new OCX file over the old one. But there have been new features added over time so we have had versions 1.1, 1.2, 1.3, 1.4, 1.5 and now 1.6. If you developed a program using version 1.5 controls you will have a problem using version 1.6 controls because changes have to be made to each project file, each control file in that project that uses Krool’s controls and the manifest if you use one. My utility takes care of that, allowing you to switch from any OCX version to any other OCX version of the CommonControls that are installed on your PC (including the standalone FlexGrid controls).

## 

## Controls Included

Below is a list of controls found in Krool’s packages. All are in the Common Controls replacement package except for VBFlexGrid which is in the VBFlexGrid Control package.

|  |  |  |  |
| --- | --- | --- | --- |
| Animation | FrameW | MCIWnd | SysInfo |
| CheckBoxW | HotKey | MonthView | TabStrip |
| ComboBoxW | ImageCombo | OptionButtonW | TextBoxW |
| CommandButtonW | ImageList | Pager | ToolBar |
| CommandLink | IPAddress | ProgressBar | TreeView |
| CommonDialog | LabelW | RichTextBox | UpDown |
| CoolBar | LinkLabel | Slider |  |
| DTPicker | ListBoxW | SpinBox | VBFlexGrid |
| FontCombo | ListView | StatusBar |  |

If you have any programming experience, you have undoubtedly seen and used many of these controls. Once you get Krool’s system installed, usage is quite easy since the controls behave very similarly to existing controls. We will mainly cover how to get each of the systems installed and set-up to use.

## Terminology

Krool’s controls are wonderful but many of the terms used within and around his packages can be very confusing (to me at least). Here are my interpretations of some of the terms.

**ActiveX** – Wikipedia [article](https://en.wikipedia.org/wiki/ActiveX). This is a software framework instituted by Microsoft in 1996 using the earlier OLE and COM technologies. For this document we will consider ActiveX as the basis of our controls.

**Control** – A component represented by an icon in your Toolbox that can be placed on a form. Most are visible but some, such as a Timer, are not. The simplest ones in VB6 are included in the as-installed VB6 program package. Others, such as Krool’s controls and many others from Microsoft and other vendors, must be added. For amore information on making your own controls please see [this](https://pages.cpsc.ucalgary.ca/~saul/vb_examples/index.html) web page and especially tutorial #10.

**OCX** – Often, all of us use controls that are pre-compiled into files having an extension of .OCX (which stands for OLE Control eXtension or just ActiveX controls). These files have to be registered on the developer’s PC and they have to be distributed and registered on each user of the developer’s program(s). One .OCX file can contain multiple controls. The programmer (you) needs to have an OCX installed and registered on his system but the user needs to have the OCX file on his system too, generally registered but sometimes “side-by-side” with the compiled program that is being run which uses the OCX control(s).

**StdEXE** – Krool uses this term to designate the other method of including ActiveX controls in a program. There are a number of source code files including standard and class modules, property page files etc. that get compiled into an OCX control. Alternatively, all of these can be distributed by the author and put into each of our programs as source code. Normally the authors of a controls package do not want to distribute the source code for their controls for a variety of reasons but Krool has elected to do that with his packages. The advantage of doing this is that your program has all of the control code compiled within it and thus no files need to be distributed or registered for your program to work. Everything is self-contained in your compiled code. A drawback of this from the programmer’s perspective is that all of the controls code gets compiled over and over as you develop your program and compile it over and over.

**VBCCRxx** – VBCCR stands for Visual Basic Common Controls Replacement and “xx” refers to the OCX version which at present can be 1.1 (xx=11), 1.2, 1.3, 1.4, 1.5 or 1.6 (xx=16). As code that is distributed in compiled form gets modified and expanded over time, the author (Krool) has to issue different versions and each must be on the user’s PC (and the programmer’s PC). For example, suppose you write a program that uses VBCCR16.ocx and you send it to a colleague who has VBCCR13.ocx installed but not VBCCR16.ocx. It won’t run because it is looking fro the other ocx file when it starts up. That’s a disadvantage of OCX files. If you use the StdEXE version, the code is compiled directly into your code and you don’t have to distribute, install or register any external file to use the controls. The main advantages of the OCX version over the StdExE versions are a) compile time for your programs is almost 10x faster because the form code doesn’t need to be re-compiled each time and b) all of the various .BAS, .CTL, .PAG files that make up each control don’t clutter up the programmers list of files to manage.

**ComCtlsDemo** – This is a sample program that shows each of Krool’s controls. This file is updated regularly and is always found at the bottom of the first post in [this](http://www.vbforums.com/showthread.php?698563-CommonControls-(Replacement-of-the-MS-common-controls)) thread on VBForums. This package uses the StdEXE concept of compiling the code for all of the controls into the executable.

This is much more than just a sample file. The source code in this file is all there is for his package and you will use it (or most of it) when you make your own programs that use his controls.

It doesn’t have (or need) version numbers like the OCX version since there are no files to register on the developer or user’s PC’s (it is source code that gets embedded ina program each time the program is compiled). I download most every update and to avoid confusion I rename each of the .zip files to a file on my hard drive with the date of the file in the name of the .zip file. Because of VBForum size limitations, he posts the file with a Word extension of .docx but it is really a .zip file so when you download it you do a Save As and then cut the .docx off of the filename. At the time of this writing, the latest one Krool posted is 11 November 2018 (look at the small italics line at the very bottom of the post below the download link) and the file to download is named ComCtlsDemo.zip.docx so when I downloaded that file I renamed it to ComCtlsDemo 2018-11-11.zip on my hard drive.

**VBCCR OCX Version** – After about 4½ years of having just the StdEXE version, Krool released a pre-compiled version called VBCCRxx.OCX where xx is the version number. The earliest one is version 1.1 so the first file was VBCCR11.ocx. As of this writing, the latest version is 1.6.13 so the file is VBCCR16.ocx. The latest version is always at the bottom of the 1st post in [this](http://www.vbforums.com/showthread.php?841929-VB6-ActiveX-CommonControls-%28Replacement-of-the-MS-common-controls%29&p=5129155#post5129155) thread on VBForums. The .zip file online contains the .ocx file as well as some resource files (discussed below) related to “side-by-side” execution of the programmer’s EXE file and also making your programs themed so that what you display on-screen doesn’t look like it came out of Windows 95. I save the .zip file with it renamed to include the version number. For example, the latest one which is called “VBCCR16.OCX.rar.docx” was copied to my hard drive with the name “VBCCR.OCX v1.6.13.rar” (see above note for why the file is available with the .docx extension). In addition to containing the .ocx file, the zip file has all of the code in case you wanted to put make your own .ocx file (not recommended) or just to learn from what Krool has done.

Copy the .ocx file to you Windows system folder. If you run 32-bit Windows this will typically be C:\Wnidows\System32 but if you run 64-bit Windows then your 32-bit folder for these types of files is C:\Windows\SysWOW64. (In the unlikely event that you put Windows somewhere other then C:\Window then use that path instead). Since you are copying to a system folder you will need elevated permission. It is okay to copy over a previous version if the xx is the same (in my example the xx is 16 so I can copy over the 1.6.12 or 1.6.11 versions) and if the xx is a new one (like for version 1.6.0) then there is no overwrite but you should use regsvr32 to register this control with your system.

This .OCX file is loaded into your project using Ctrl-T in the VB6 IDE (or Project|Components) and selecting the appropriate control. For version 1.6 you would select “VB Common Controls Replacement 1.6 Library” by clicking the check mark next to it.

With the reference to this OCX, all of the controls with show up in your toolbox for use like any other controls you can put on your forms. Because the .OCX version of the controls is pre-compiled, the controls aren’t compiled every time you compile your program like has to happen with the StdEXE version where the controls are embedded into your code. On the other hand, a drawback is that the .OCX file ahs to accompany your EXE or DLL file when you distribute it because it was not compiled into it.

**Type Library** – To get Krool’s VBCCR and VBFlexGrid controls in the StdEXE versions (not the OCX versions) to work you need to use a type library he has provided called “OLEGuids.tlb” which is in the “OLDGuids” folder he distributes with each of the control packages. I copied oleguids.tlb into my system directory and then I can use Project|References to specify it by clicking on the check box next to “OLE Guid and interface definitions”. The file he is currently distributing has a date stamp of 9 June 2017 so it hasn’t changed in some time. This file is only needed while you are developing your code if you use the StdEXE version. It does not need to be distributed with the final compiled program. (Note-If you use Krool’s controls with VBA you do ***not*** need this file because you have to use the OCX version which already has this type library compiled into it.)

**Visual Styles (Themes)** – This started in XP but didn’t really didn’t see much use until Vista and alter. With visual styles you can get more modern-looking forms but since these came into vogue after VB6 came out, there is no simple method of using these in your VB6 programs. Krool’s code is set up to use these visual styles but without a couple of interesting steps you won’t get them. In any event his code provides Unicode and some enhanced features compared to the original controls but to get visual styles you have to specify this in a manifest and then embed the manifest into a resource file referenced by your program. It sounds harder than it is and I will cover how to do that later. 9Note- VBA doesn’t do styles

**Side-by-Side Assemblies** – If your program requires a DLL or an OCX file to run, it can use one if it already exists on the user’s system, has been registered and is the right version. Starting around the time when VB6 came out, there was a movement to get away from this system to avoid what many call “DLL Hell” (Google that for some fun). To make a long story short, Windows allows programs to run without having to register support files as long as the support files are located along side the EXE or DLL file (or in a sub-folder of it). A lot of corporate users have their PC’s set to not allow any new programs to be installed by blocking registration of thee support files and this is one way around that. In order to specify that the support files are side-by-side you have to specify this in a manifest (see below). It used to be that it was okay to have the manifest in the same folder as the executable file or embed into the file but later versions of Windows strongly prefer the manifest file to be included in a resource file. All of this sound crazy but Krool provides some support for all of the compelxities and my utility (hopefully) takes the pain out of the rest.

**Manifest File** – A manifest file is XML-based and can specify many things for Windows to do to control the program. An example is that you can specify in a manifest that the program needs to run with elevated privileges. For our situation, though, the manifest file is of use to use for 2 reasons: 1) we can tell Windows that we want to use version 6.0 f Microsoft’s CommonContrls dll file which is the one that give us the visual styles with Krool’s code and 2) we can tell give it information such that the .OCX file (if you use that instead of the StdEXE one) is side-by-side with the executable you will be making. This by itself is bad enough but Windows now wants this manifest file included in the program’s resource file.

**Resource File** – You can put many things into a VB6 resource file (extension .RES) including icons, graphics images, international strings, etc. You can also put a manifest file and you will do exactly that to get the visual styles and/or the side-by-side stuff going. Krool provides two resource files with the OCX version, one for just side-by-side and one for side-by-side plus visual styles. In the StdEXE package there is a resource file in the Resources folder that provides for visual styles (don’t need side-by-side if you don’t use the .OCX version). A user named LaVolpe has a utility on VBForums ([here](http://www.vbforums.com/showthread.php?845909-VB6-Manifest-Creator-II)) that enables you to pull out the manifest information from a resource file, edit it and then put it back in. Hopefully you won’t have to use that utility (I use a part of LaVolpe’s code in my utility as discussed later). The point of defining it here is to ensure that the programmer knows that side-by-side and visual styles need to specify a resource file and that the resource file needs to specify these internally.

**Windows System Folder** – The Windows system folder holds many system-related files including DLL’s, type libraries, registered controls etc. For 32-bit VB6 and 32-bit VBA, this folder can be one of two things. If you are using a 32-bit operating system this folder will be “C:\Windows\System32”. Unfortunately if you are using a 64-bit Windows operating system, all of its 64-bit DLL’s, type libraries and controls go into the system32 folder (?) so Microsoft puts all of the 32-bit files like this into a folder named “SysWOW64” (**W**indows 32-bit **O**n **W**indows **64**-bit). All of the 32-bit files go into “C:\Windows\SysWOW64”.

If you are using all of this in 64-bit VBA you will only deal with the 64-bit Windows folder which is always “C:Windows\System32”.

**VBFlexGrid** – Krool’s original controls included 33 controls but not a replacement for the MSFlexGrid control (Microsoft provides a file MSFLXGRD.OCX which Krool’s package upgrades and replaces). The approach he has taken is very similar to what he has done with the other controls. There are separate versions that are standalone like the VBCCR ComCtlsDemo package, this one called VBFlexGridDemo found [here](http://www.vbforums.com/showthread.php?848839-VBFlexGrid-Control-(Replacement-of-the-MSFlexGrid-control)), and the corresponding OCX version of it, VBFLXGRD12.OCX, found [here](http://www.vbforums.com/showthread.php?855931-VB6-ActiveX-VBFlexGrid-%28Replacement-of-the-MSFlexGrid-control%29&p=5236525#post5236525). Both of these versions are independent of the VBCCRxx controls. My utility deals with these and the VBCCR controls as if they are all part of the same package.

## User Guide for the Programmer

Below is a discussion of usage for each version. After that is a discussion of my recommended use which is a simple hybrid of each option that (hopefully) takes advantage of the best features of each option and then some.

VBCCR – Two versions exist for this set of 33 controls (all of the ones in the previous table except the last one). Until now, programmers have had to decide whether to use either the StdEXE or OCX versions; there has been no way to use both of them. After the user guides for the two versions I will show you another, hopefully better, way of using these controls so you can take advantage of the easier and faster OCX version for development yet produce a final executable with the code for the controls in your control with the StdEXE version.

### VBCCR – StdEXE Version

This is the version in which all of the control code is compiled into your program. You will include the appropriate source code in your program and when you compile the controls become part of your program.

Getting the latest version – You might be tempted to think that there is a download available that has all of the controls, a user guide, etc. but that’s not the case. Krool has a demonstration project on the VBForums web site and from that you get access to all of his controls (interesting approach but it works…). Krools’ demo project with all of the controls has been [here](http://www.vbforums.com/showthread.php?698563-CommonControls-(Replacement-of-the-MS-common-controls)) since 10 Nov 2012. There are over 54 pages of comments and discussion in the thread. Most deal with various bugs and user questions and feature addition requests as Krool has worked the package for the past 6 years.

The important thing is that at the bottom of the first post is a downloadable file named “**ComCtlsDemo.zip.docx**” which is always the latest version to download. It has the .DOCX extension because VBForums has a lower size limit on .ZIP files than .DOCX files and this file exceeds the file size limit for .ZIP files. It really is a .ZIP file so as you download it (or after) rename it by knocking off the .DOCX part of the name, leaving the file ComCtlsDemo.zip.

The name of this file on the web site is always the same. I recommend that you look at the last line in this first post and note the date and then put the date in the filename. For example, I am looking at the first post and at the bottom it says that it was last edited by Krool on November 11, 2018 so when I right-click on the link I tell it to save it into a set of folders where I keep all of my downloaded archive files and I save it with the name “**ComCtlsDemo 2018-11-11.zip**” so I can differentiate it from prior copies I have downloaded.

Now here is an important observation. ComCtlsDemo contain all of the files for the controls and these files are not supposed to be altered so you can put these files into a central location that all of your programs will access (a library). This location doesn’t need to change and you can always delete the existing files and put the newest version’s files in the same folders. Also, all of your programs access all of these files where they are so you don’t need to copy all of these files into your individual project folders. That greatly simplifies usage and updating for newer versions.

Where to put the downloaded files when unzipping - I have a library folder into which I put all sorts of files for use in my programs. I do not put files into here that will change. This contains files I can use in all or my programs without modifications. In my Library folder I have a sub-folder for Controls & Forms. Within that I have a folder for Krool’s controls named VBCCR and in that I have made a sub-folder called “Current”. Whenever I download the latest update from Krool, I first delete all the files and folders in Current and then I unzip the new files into Current. This is important because programs I am working on that already that use these controls will continue to find these controls in the same places and won’t even know they are the new versions. Fortunately Krool keeps the names of his files and folders the same as he debugs and adds features to the controls.

So now that we have the files on our PC in a location where we can use them for all of our programs, we can just start using them. Right? Well, sort of. There are some additional steps to take in order to use his controls. Follow the steps below for any new or modified program you create.

Type library - You need to be able to access a type library named “OLEGuids.tlb” that is included with Krool’s sample program. This file is in a sub-folder of Current named “OLEGuids”. You will need this file during your editing and compiling but your compiled program doesn’t need it and you don’t distribute it with your executable. I put mine into my Windows System folder so I always know where it is and I register it in Windows 10 with regsvr32. Fortunately this type library file doesn’t change very often so you don’t have to do this step often. OLEGuids.tlb is dated 9 June 2017 so it hasn’t changed in 18 months.

In VB6 you will use the commands Project | References to select this type library. If you have registered this file using regsvr32 then you can find it in the list of Available References with the name “OLE Guid and interface definitions” but if you haven’t registered it yet you can click on “Browse…” and go find it.

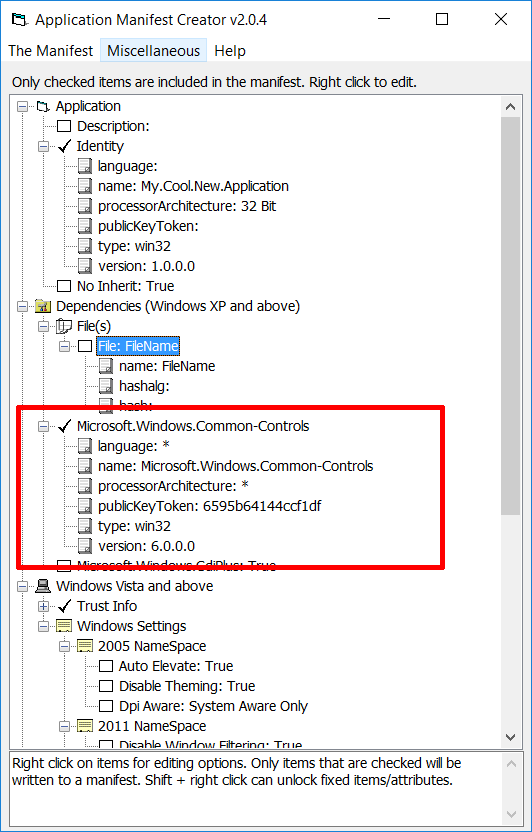
Visual Styles - In order to use visual styles (themes) in your programs so they don’t look like old Windows programs you have to specify the use of Windows’ Common Controls library version 6.0 because it has support for visual styles. The way to do that is to include the specification for this in a manifest file. It used to be that you made a file including the name of your executable that had a “.Manifest” extension and when you distributed your program you included this file in the same folder as your executable file. Later versions of Windows discourage this but recommend that you include the manifest in the EXE file as part of an embedded resource file. VB6 can use resource files to hold a number of different things such as internationalization strings, icons, etc. in addition to manifests.

So we have to get the visual style specification into a manifest and then get the manifest file inside of a VB6 resource file. In Krool’s package look in Current\Resources to find a file named “Resources.res” which is a resource file Krool made that contains the directions to enable visual styles (or theming). If you are not using a resource file for anything else you can simply copy this Resources.res file to wherever you project file (.VBP) resides. I will show you in a minute how to embed that in your project. But for now let’s consider what to do if you already have a resource file and we want to add our manifest information to that resource file (whether it already has manifest information in it or not).

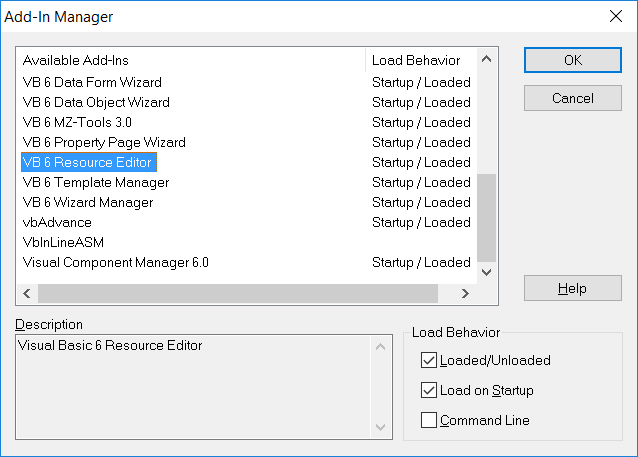
Please note that the VB6 IDE was not designed to use visual styles. [Here](http://www.vbforums.com/showthread.php?693111-VB6-IDE-solving-UAC-and-Visual-Style-issues&highlight=) is a VBForums post by Krool that shows how to get a resource file with embedded manifest that will run VB6 with elevated UAC as well as incorporating visual styles. You don’t need this to use Krool’s controls but if you want to see the “prettier” forms then this may be worth it to you. Note that it involves using another utility called ResourceHacker to get the resource file into the VB6.EXE file.

As far as using visual styles in your programs, I find it strange that the instructions to use visual styles are part of a manifest file which we have to embed into a resource file which is then embedded within our executable files. Krool supplies a resource file with visual styles in the Resources folder under Current that you can copy to your project and embed.

Advanced resource/manifest/visual styles note – You may want other things in your resource file in addition to the visual styles setting. It is not a trivial thing to work with these files. The manifest part is XML and it is contained in a non-XML resource file. I have found another utility on VBForums by a user named LaVolpe ([here](http://www.vbforums.com/showthread.php?845909-VB6-Manifest-Creator-II)) that lets us make or edit a manifest from scratch or from a manifest file or extracted from a resource file so that we can edit it and then we can specify that it is put into a resource file. Below is a screenshot of LaVolpe’s utility running with Resources.res loaded. I highlighted the section that specifies Windows Common Controls version 6.0.0.0 to be used. This is what specifies visual styles. In general for a new program you should be able to just copy the Resources.res to your project folder and use it without having to edit it. BTW, you don’t *need* this file but you won’t get any of the modern looking controls in your program without it.



So now you have a resource file that specifies using visual styles. How do we get it into our project? In VB6, go to AddIns | AddIn Manager… and you will see something like the following.



Select “VB6 Resource Editor” and ensure that “Loaded/Unloaded” and “Load On Startup” are both checked. Then, back in your main project go to Project | Add New Resource File… and then select your .res file from the dialog box that pops up. Now your resource file should show up in the navigator pane under Related Documents.

Side-by-Side – One reason for using the StdEXE version is that unless you use some other specialized controls or other files, your executable has no dependencies so there is no reason to be concerned about side-by-side. I will discuss it more in the user guide for the OCX version of Krool’s controls.

Files to Include in Your Project - Each of the controls has a number of files specific to the control which need to be inserted into your project as shown below.

| **Control** | **Files in Current Folder** |
| --- | --- |
| Animation | Builds\Animation\Animation.ctl |
|  | Builds\Animation\PPAnimationGeneral.pag |
| CheckBoxW | Builds\CheckBoxW\CheckBoxW.ctl |
| ComboBoxW | Builds\ComboBoxW\ComboBoxW.ctl |
| CommandButtonW | Builds\CommandButtonW\CommandButtonW.ctl |
| CommandLink | Builds\CommandLink\CommandLink.ctl |
|  | Builds\CommandLink\PPCommandLinkGeneral.pag |
| CoolBar | Builds\CoolBar\CbrBand.cls |
|  | Builds\CoolBar\CbrBandProperties.cls |
|  | Builds\CoolBar\CbrBands.cls |
|  | Builds\CoolBar\CoolBar.ctl |
|  | Builds\CoolBar\PPCoolBarBands.pag |
|  | Builds\CoolBar\PPCoolBarGeneral.pag |
| DTPicker | Builds\DTPicker\DTPicker.ctl |
|  | Builds\DTPicker\PPDTPickerGeneral.pag |
| FontCombo | Builds\FontCombo\FontCombo.ctl |
| FrameW | Builds\FrameW\FrameW.ctl |
| HotKey | Builds\HotKey\HotKey.ctl |
| ImageCombo | Builds\ImageCombo\ImageCombo.ctl |
|  | Builds\ImageCombo\ImcComboItem.cls |
|  | Builds\ImageCombo\ImcComboItems.cls |
|  | Builds\ImageCombo\PPImageComboGeneral.pag |
| ImageList | Builds\ImageList\ImageList.ctl |
|  | Builds\ImageList\ImlListImage.cls |
|  | Builds\ImageList\ImlListImages.cls |
|  | Builds\ImageList\PPImageListGeneral.pag |
|  | Builds\ImageList\PPImageListImages.pag |
| IPAddress | Builds\IPAddress\IPAddress.ctl |
| LabelW | Builds\LabelW\LabelW.ctl |
| LinkLabel | Builds\LinkLabel\LinkLabel.ctl |
|  | Builds\LinkLabel\LlbLink.cls |
|  | Builds\LinkLabel\LlbLinks.cls |
|  | Builds\LinkLabel\PPLinkLabelGeneral.pag |
| ListBoxW | Builds\ListBoxW\ListBoxW.ctl |
| ListView | Builds\ListView\ListView.ctl |
|  | Builds\ListView\LvwColumnHeader.cls |
|  | Builds\ListView\LvwColumnHeaders.cls |
|  | Builds\ListView\LvwGroup.cls |
|  | Builds\ListView\LvwGroups.cls |
|  | Builds\ListView\LvwListItem.cls |
|  | Builds\ListView\LvwListItems.cls |
|  | Builds\ListView\LvwListSubItem.cls |
|  | Builds\ListView\LvwListSubItems.cls |
|  | Builds\ListView\LvwVirtualListItem.cls |
|  | Builds\ListView\LvwVirtualListItems.cls |
|  | Builds\ListView\PPListViewGeneral.pag |
|  | Builds\ListView\PPListViewImageLists.pag |
|  | Builds\ListView\PPListViewSorting.pag |
| MCIWnd | Builds\MCIWnd\MCIWnd.ctl |
| MonthView | Builds\MonthView\MonthView.ctl |
|  | Builds\MonthView\PPMonthViewGeneral.pag |
| OptionButtonW | Builds\OptionButtonW\OptionButtonW.ctl |
| Pager | Builds\Pager\Pager.ctl |
|  | Builds\Pager\PPPagerGeneral.pag |
| ProgressBar | Builds\ProgressBar\PPProgressBarGeneral.pag |
|  | Builds\ProgressBar\ProgressBar.ctl |
| RichTextBox | Builds\RichTextBox\PPRichTextBoxGeneral.pag |
|  | Builds\RichTextBox\RichTextBox.ctl |
|  | Builds\RichTextBox\RichTextBoxBase.bas |
| Slider | Builds\Slider\PPSliderAppearance.pag |
|  | Builds\Slider\PPSliderGeneral.pag |
|  | Builds\Slider\Slider.ctl |
| SpinBox | Builds\SpinBox\PPSpinBoxGeneral.pag |
|  | Builds\SpinBox\SpinBox.ctl |
| StatusBar | Builds\StatusBar\PPStatusBarGeneral.pag |
|  | Builds\StatusBar\PPStatusBarPanels.pag |
|  | Builds\StatusBar\SbrPanel.cls |
|  | Builds\StatusBar\SbrPanelProperties.cls |
|  | Builds\StatusBar\SbrPanels.cls |
|  | Builds\StatusBar\StatusBar.ctl |
| SysInfo | Builds\SysInfo\SysInfo.ctl |
| TabStrip | Builds\TabStrip\PPTabStripGeneral.pag |
|  | Builds\TabStrip\PPTabStripTabs.pag |
|  | Builds\TabStrip\TabStrip.ctl |
|  | Builds\TabStrip\TbsTab.cls |
|  | Builds\TabStrip\TbsTabs.cls |
| TextBoxW | Builds\TextBoxW\PPTextBoxWText.pag |
|  | Builds\TextBoxW\TextBoxW.ctl |
| ToolBar | Builds\ToolBar\PPToolBarButtons.pag |
|  | Builds\ToolBar\PPToolBarGeneral.pag |
|  | Builds\ToolBar\TbrButton.cls |
|  | Builds\ToolBar\TbrButtonMenu.cls |
|  | Builds\ToolBar\TbrButtonMenus.cls |
|  | Builds\ToolBar\TbrButtonProperties.cls |
|  | Builds\ToolBar\TbrButtons.cls |
|  | Builds\ToolBar\ToolBar.ctl |
| TreeView | Builds\TreeView\PPTreeViewGeneral.pag |
|  | Builds\TreeView\TreeView.ctl |
|  | Builds\TreeView\TvwNode.cls |
|  | Builds\TreeView\TvwNodes.cls |
| UpDown | Builds\UpDown\PPUpDownGeneral.pag |
|  | Builds\UpDown\UpDown.ctl |

If you want any individual control to be available in your project, from within the IDE, press Ctrl-D and then navigate to the appropriate folder and highlight all of the files in the folder to import and press Enter.

A challenge is that Krool’s package includes all of the controls and it is unlikely you will need all of them. The demo program utilizes all of them and the EXE file resulting from this is 4.2 MB so if you include all of the controls in your program you will add about 4 MB to the file size. In these days of multi-gigabyte RAM and hard drives this may not be the consideration that it once was.

There are also some files that have to be present whether you use one control or all of them. These are:

|  |
| --- |
| Builds\ComCtlsBase.bas |
| Builds\VTableHandle.bas |
| Builds\VTableSubclass.cls |
| Builds\ISubclass.cls |
| Common\Common.bas |
| Common\VisualStyles.bas |

Finally, if you use the control MCIWnd.ctl or property pages for CoolBar, Imagelist, RichTextBox or StatusBar you must also include the file “Builds\CommonDialog.cls” in your project.

You may be tempted to not include code for controls you are not using. You do save some size in your final executable file but just know that the control(s) left out do not appear in your toolbox for even possible use without including the above code for the individual controls. If you are sure you won’t use a given control it is okay to leave it out. As far as I know, each of the controls is independent of the others.

There is one more file in Krool’s ComCtlsDemo package, Common\Startup.bas, which is really for the demo program. You don’t need this file but there are some concepts in that file that need to be part of your program.

Sub Main – Krool’s controls rely on some Microsoft code that needs to run before any forms are loaded or shown. To use the new controls you must specify the Startup Object as Sub Main instead of any Form and you must have the correct start-up code in Sub main before you reference any control. This is set in Project | Properties on the General tab. If you don’t have a start-up routine in a Sub named Main you need to put one in your program.

You need to have a call to one of Krool’s routines to provide protection for callbacks etc. built into his programs so that you don’t crash in the IDE. Also, there is some start-up code required for your program to use the visual styles enabled by using Microsoft’s Common Controls 6.0. Thus, the first two lines in your Sub Main should be:

Call ComCtlsInitIDEStopProtection ' in Builds\ComCtlsBase.bas

Call InitVisualStyles ' in Common\VisualStyles.bas

Now you can put the rest of your code to show Forms, do calculations etc. Unless you use other .OCX control files, your final executable will contain all of the code including the controls so the executable is standalone and does not need anything other than the standard VB6 support files included with Windows by Microsoft.

You may have some issues with names of subroutines and variables being the same as ones that you use. If you use the .OCX version of the controls, most of that is hidden and is not a concern. However, when you include all of the various files for the controls to be compiled within your program, you now have 153 new files and you may have some naming conflicts. If you decide to keep your names and change Krool’s, just know that every time you download and use and update your will have to edit his files to rename these (not just the name of the subroutine but every other routine that calls it). I reluctantly decided to change some conflicting names in my code so I wouldn’t have to keep updating his files with new downloads. I don’t like this but it is a small price to pay for these controls.

Note – I don’t use the above approach because I don’t like the long compile times although I do like a fully self-contained executable. That’s part of the reason I wrote the utility I will discuss later. It allows you to develop using the .OCX version (compiles so much faster) and then do the final compile through my utility with the StdEXE version so you have the self-contained executable.

### VBCCR OCX Version Guide

Krool has the equivalent of the StdEXE package for VBCCR in a pre-compiled form as a more typical single controls file with an OCX extension found [here](http://www.vbforums.com/showthread.php?841929-VB6-ActiveX-CommonControls-%28Replacement-of-the-MS-common-controls%29&p=5129155#post5129155). Advantages versus the StdEXE-based controls are: 1) simpler to use as it needs one .OCX file instead of 153 individual files to include in each of your programs, 2) compile times for your program are faster since the OCX file is already compiled and 3) programmers are used to OCX systems available from Microsoft and other vendors. Disadvantages are: 1) the OCX file has to accompany the executable to the users, 2) the OCX file must either be registered on the user’s PC or employ a more-complicated side-by-side solution.

Getting the latest version - From [this site](http://www.vbforums.com/showthread.php?841929-VB6-ActiveX-CommonControls-%28Replacement-of-the-MS-common-controls%29&p=5129155#post5129155) download the file at the bottom of the first post. We are now in version 1.6 of the OCX version of the controls so the file is listed for download as “VBCCR16.OCX.rar.docx”. It has the .docx extension due to size limitations of other types of files on VBForums. It is really a RAR file. I include the current version in the filename when I download the file so I can keep track of the various downloads. The first post says the current version if 1.6.13 so when I downloaded this file is used SaveAs and I named it “VBCCR16.OCX v1.6.13.rar” (I drop the .docx extension and add the new version number). Inside this file is the .OCX file and a zip file containing the source code if you wanted to generate your own version of the .OCX (I don’t recommend this because it will now be a separate package from Krool’s; it basically has the same files that the StdEXE version has).

Use of the files in your program - Now in your new program you will use all of these files. Press Ctrl-T or do Project|Compenents and select “VB Common Controls Replacement 1.6 Library” (that’s file vbccr16.ocx). To get these into a VBA project do Tools | Additional Controls… in the VBA IDE.

Finally, you need to set the resource file that contains the manifest to enable visual styles and optionally side-by-side (VB6 only; not applicable for VBA). There are also 2 VB6 resource files in the first post on VBForums: “VBCCR16SideBySide.res” and “VBCCR16SideBySideAndVisualStyles.res” which are for exactly what they say in the title. These are not zipped but instead are the actual resource files so you should just save the links as files and use them. I will cover how to use one of these in just a bit.

Setting up your project - Just like with the StdEXE version you cannot start your program with a Form; you must have a Sub Main and start it first so you can run some code that is required to be run before your first form is called or referenced. The easiest thing to do is to include the standard module VisualStyles.bas from the StdEXE version and then in your program, make sure in the General tab of Project | Properties is set to use Sub Main and in your Sub Main you should have the following line of code before loading, referencing or showing any forms:

InitVisualStyles

Now you are ready to go. All of the controls will show up in your Toolbox for use in your forms.

Note – In the StdEXE version there is some code Krool has included to provide crash protection when you use the controls in the IDE. You need the same protections with the OCX version but thy have been compiled into the OCX so you don’t need to call the IDE protection code.

Compiling your code – Nothing special here other than using that procedure discussed above in Sub Main. Krool says his code is IDE-safe and that has been my experience as well.

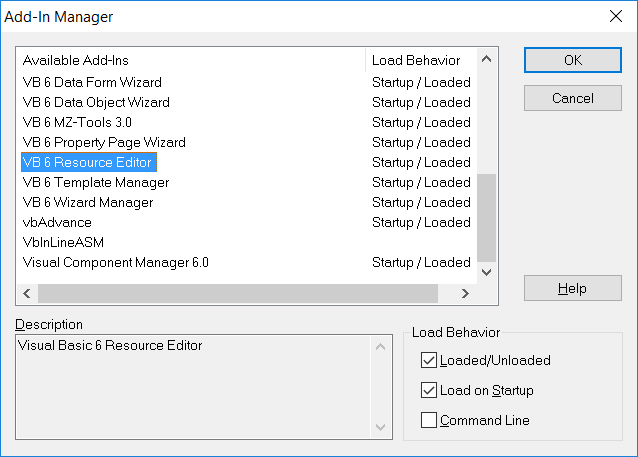
You have an option of using a manifest file to specify two things that may be of interest to you. First, I think you will want to take advantage of themes/visual styles since this capability is built into Krool’s code and is relatively easily accessed. Also, the manifest can specify that you want a “side-by-side” assembly. Normally you have to include separate controls and ActiveX DLL’s with your program and they must be registered on the user’s system in the Windows System folder. Some organizations don’t allow this so they can’t even use a program with external dependencies such as the VBCCRxx.OCX controls file discussed here. Around the time VB6 came out there was a move to get away from this and to allow something else. In theory, having the latest version of controls (.OCX’s) and DLL’s in your system folder is efficient because many programs can access that one file instead of each program having its own OCX’s or DLL’s. In practice this caused a lot of problems (for fun, Google “DLL Hell” sometime). So with the right manifest you can specify that the OCX and/or DLL files your program uses can be with your programs and they don’t have to be registered with the user’s system. It is not quite as slick as including the code inside the program (like with the StdEXE version of Krool’s code) but it is the next best thing.

Manifests can include other things but for purposes of this user guide we will focus only on the two that can affect Krool’s package. “Back in the day” you used to be able to put the manifest file next to the EXE when you distributed your package to your users but recent versions of Windows really want you to include the manifest in the program. There really isn’t a way to do that so the manifest gets included inside a resource file which can be and is used by many programming languages including VB6.

Manifests are in XML and are strange because the file size must be exactly a multiple of 4 bytes. Fortunately Krool has provided two resource files that I discussed 2 pages back that can just be included into your program. One is for side-by-side and the other is for enabling visual styles plus side-by-side. Suppose you have downloaded from Krool’s web site the file “VBCCR16SideBySideAndVisualStyles.res” and want to use it. Obviously this is the one that provides both visual styles and side-by-side. You do not need to edit this file but you do need to reference it.

Note – The resource file can be left in a common location as a library file used by multiple of your programs. Note though if you add to it or modify the file as you centrally use it, these changes will show up in all programs using that resource file. If you are concerned about this, simply copy the resource file to your project folder (same on that holds your .vbp file) and reference it separately for your project.

So how do you get this resource file into your project? In VB6, go to AddIns | AddIn Manager… and you will see something like the following:



Select “VB6 Resource Editor” and ensure that “Loaded/Unloaded” and “Load On Startup” are both checked. Then, back in your main project go to Project | Add New Resource File… and then select your .res file from the dialog box that pops up. Now your resource file should show up in the navigator pane under Related Documents which is listed below all of the forms, standard and class modules.

Now you are ready to go and to use your compiled program.

Using the ExE or DLL – You do not need a manifest to be included with the files you distribute since by including it in the resource file it is now part of your executable.

You do need to include VBCCRxx.OCX with your program. If you use the side-by-side approach you can include it in the same folder or a sub-folder of the folder you put the executable file in. Please note that if you do use side-by-side, Windows looks for this OCX file in the executable’s folder even on your PC or anyone else’s even though you may have copied it to the System folder and registered it (at least that’s my experience). So I recommend that you do not move the OCX file out of the system folder but rather make a copy of it in the same folder where you have your EXE file on your PC. Please check my utility before you decide to do side-by-side. I have a solution whereby you use the OCX version of the controls during development then use the utility to do a commandline compile with the StdEXE version of the controls so you don’t need any side-by-side solutions sine the code is all embedded in the executable (at least for Krool’s controls; if you use someone else’s OCX files you may still want to do side-by-sie for their controls).

If you don’t specify side-by-side you won’t need to do anything on your PC since you already have it registered but you need to have your installation program install it to the user’s system directory and register it.

### VBFlexGrid User Guide

The MSFlexgrid replacement package from Krool is almost identical to what we have discussed above for the other 33 Common Control replacements Krool has done. This one came much later than the others so Krool decided to keep it separate. Perhaps at some point these two packages get merged but for now just think of VBFLXGRD as just like VBCCR except it has one control instead of 33.

You can download the StdEXE version from VBForums [here](http://www.vbforums.com/showthread.php?848839-VBFlexGrid-Control-(Replacement-of-the-MSFlexGrid-control)) and the OCX version [here](http://www.vbforums.com/showthread.php?855931-VB6-ActiveX-VBFlexGrid-%28Replacement-of-the-MSFlexGrid-control%29&p=5236525#post5236525). All of the issues relating to the VBCCR counterpart are the same with the FlexGrid control.

My utility will help you manage both of these seamlessly.

### VBA Usage

The StdEXE version of the VBCCR and VBFLXGRD controls will not work in VBA because VBA does not allow you to have controls embedded into the code. Any controls in addition to the ones built-in to VBA must be ActiveX controls (i.e., the OCX version).

The OCX versions do work in VBA. The type library is not required since it is effectively compiled into the OCX file. Each user have the OCX file registered and referenced in his VBA project on whatever PC’s are running the code.

Also, there are no styles in VBA so this feature is not available. If your VBA forms are ugly this code won’t help that.

These controls do give the VBA user access to Unicode versions of the controls although many VBA controls in recent years display Unicode characters anyway (don’t allow Unicode text in properties at design-time though). It does not enable Unicode in the IDE code editor; the editor remains ANSI with or without these controls. Finally, many of Krool’s controls are enhanced versus what Microsoft provides so this could be an advantage.

## OCX2StdExe Utility

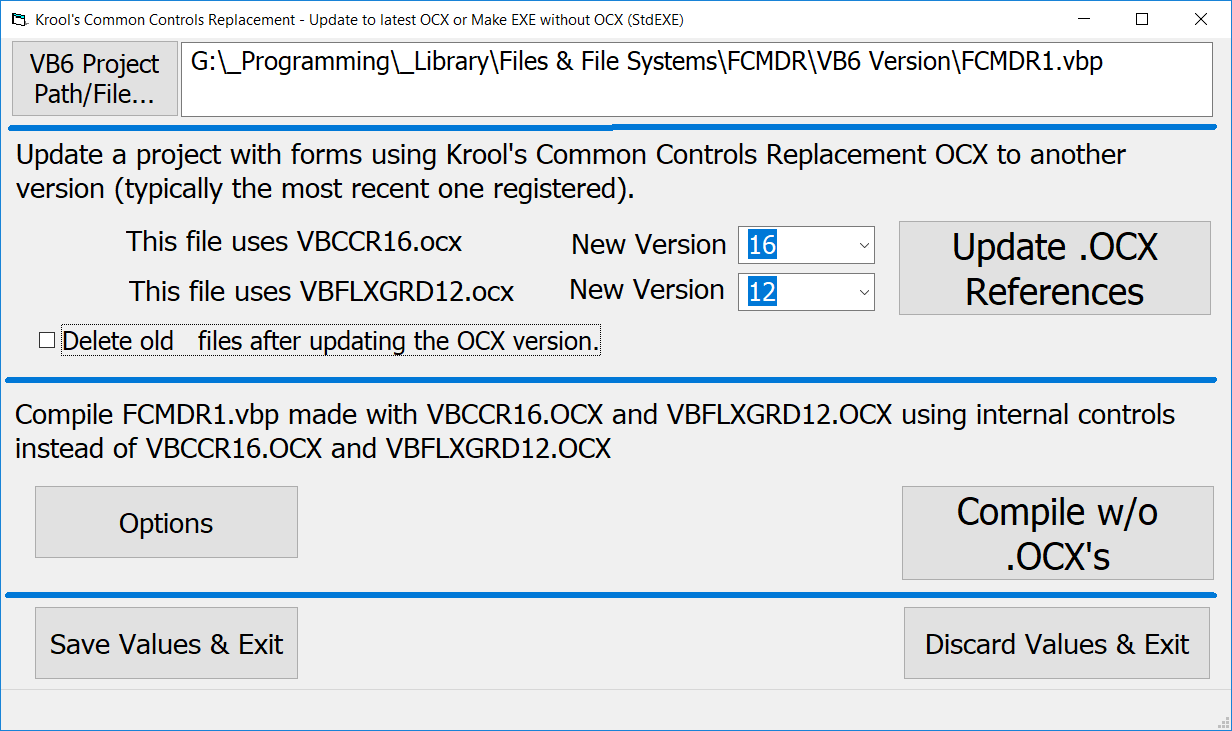
I have used these for a while and have thought about how to manage these to not drive myself crazy and to get an efficient coding environment. Things I don’t like are:

* The StdEXE versions enable a standalone executable file with no dependencies which is great but the compile times are a killer.
* The OCX versions compile much more quickly but now you have an OCX to distribute and possibly register for each executable you make.
* Updates to the OCX packages are great because they have a lot of bug fixes and feature additions but it is a real pain when a new version comes out. For each of your programs you have to find which forms use the controls from the text in the project file, manually change the references to the new name of VBCCRxx.OCX and/or VBFLXGRDxx.OCX in each form manually with a text editor, if you use a manifold with side-by-side you have to replace all of the names and GUID strings in the manifest part of the resource file and lastly you have to manually edit the project file (.vbp) to reference the new ocx file(s). Once is an experience and many times is nuts.
* Since Krool’s OCX version doesn’t come with a sample file it is not entirely straightforward how to get the OCX version to use visual styles even though it might be referenced in the manifest/resource file. There is some initialization code that is required that is not included in the OCX package. You can make one from the StdEXE version but it takes parts of several standard modules.
* I have a lot of other initialization code and I don’t want my code to get mixed up with Krool’s code.

So I wrote a VB6 utility that manages these issues and more:

* You set up your development program to use the OCX version(s) because it is simpler (1 file instead of 153) and your project compiles much faster than the StdEXE version during development.
* You can use VBCCRxx.OCX or VBFLXGRDxx.OCX or both. I am trying to treat these two control packages as if they are one and the same (I have no inside knowledge but my expectation is that Krool will combine these in the not-too-distant future anyway).
* Whenever a new version of VBCCRxx.OCX or VBFLxGRDxx comes out, you download it and copy it to your system folder and register it. If you have1.6.13 which is in VBCCR16.OCX and your overwrite the version 1.6.12 copy that was already registered and in the system folder you don’t have to do anything extra but it you have a program that is using version 1.5 and you now download version 1.6.xx you can use my utility to specify that your existing program that had been using VBCCR15.OCX should be upgraded to VBCCR16.OCX. the utility takes care of making the appropriate changes to your project file (.vbp), all form files (.frm and .frx) and even to the data in the resource file (.res) that references the old file for visual styles and/or side-by-side. All of your existing files are saved so you can easily restore them if for some reason you wanted to regress to an earlier version (not recommended). In fact, the utility can take projects using any version and switch to an earlier version. So you could go from version 1.6 down to 1.1 (why you would want to do this I don’t know but you could).
* You can develop and maintain your project as you normally do with other projects when you use OCX controls. At some point, though, you likely will want to switch the compile to use the StdEXE version so you can get the single file, no dependency program and you don’t mind the compile time just this once. You can easily do this and not change any of your existing files. You just get a new self-contained executable file.
* Your compiled code is smaller with the StdEXE commandline compilation because it includes only those controls that are actually used in the project. Your original files are not modified. I make temporary files that contain references to the StdEXE controls (the 153 files) instead of the one OCX. You have the option of retaining the temporary files (have different names than the files for the OCX version) so you can recompile the new StdEXE version again if you need or want to. Also, the property pages are optionally not included in the StdExE version reducing the executable file size because the commandline compile does not make use of property pages anyway (IDE does).

The utility is (hopefully) easy to use. This document covers the version of the utility written for VB6; there is an almost identical version written for Excel if you prefer to do that. The Excel version is included in the package along with the VB6 version. Below is a screen shot of the VB6 version. At the top of the form is the VB6 project file to be updated or compiled. You can enter the path in the text box or click on the button to the left to search for it. The project file needs to be a project that uses the OCX versions of Krool’s controls (either or both). The OCX versions have versions and it is not trivial to change from one of the versions to a newer one because references to the OCX file to be used is embedded in code for the forms that use the controls, the resource file if it has an embedded manifest and the project file itself. If you specify a project file for a project that either doesn’t use Krool’s controls or uses the StdEXE version then you won’t be able to do either the OCX upgrade or the commandline compile.



**Update OCX Version** – Our form advises you what current version of either VBCCRxx.OCX and/or VBFLXGRD.OCX you are using in the project. The sample above shows that the specified project is using VBCCR15.OCX and VBFLXGRD11.OCX. It also shows the versions of each that you have installed on the PC running this utility, defaulting to the most recent. In the sample above, it is showing VBCCR16.OCX and VBFLXGRD12.OCX which are the latest available at the time of this document. Although you can’t see it from the picture, the drop-downs include some earlier versions as well.

You have an option to keep or delete the old version files after the upgrade. In general yo won’t need to keep the old files but if you choose to keep them, you can find them in the same folder(s) as the new ones but with the extensions added not the names of the old OCX version numbers. For example, if you used Krool’s controls on “myForm.frm” when it is converted to version 16 from 15, the old file can be retained and if it is it would be named “myForm.frm.ocxCCR15ocxFlex11” so you can clearly identify it as the old file left behind after an upgrade. I recommend that once you develop trust in this utility that it won’t delete files it shouldn’t, there really is no reason to save the old versions so you can click on the choice to delete the old files.

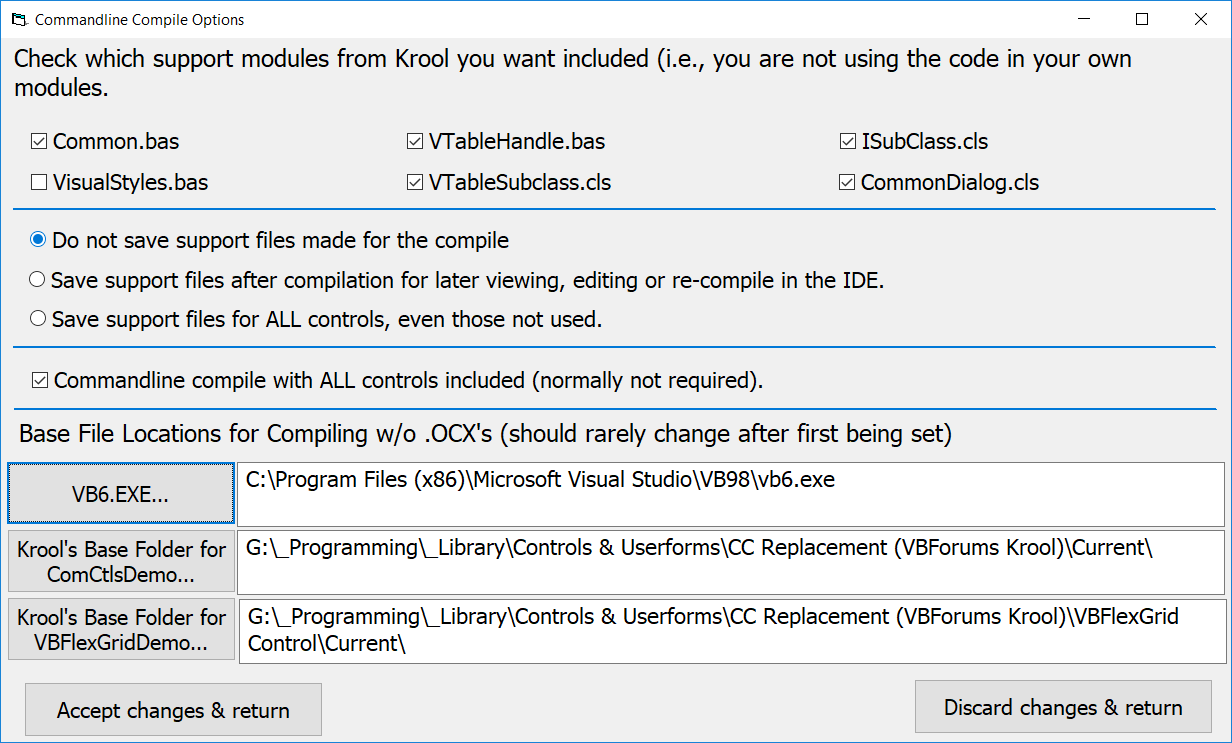
You can actually go to earlier versions of the controls that the current one if the older OCX file(s) is registered on your PC. In general, newer versions have more features but more importantly they also have bug fixes so I generally discourage going back to an earlier version. Another item to consider is that VBCCR16 added a new control (ComboFont” that doesn’t exist in earlier versions so if you specify going from version 1.6 or later to version 1.5 or earlier, references to this control have to be cut out which is likely not what you want (assuming you have used it).

Once you have chosen the version(s) to go to and whether or not to save the old files, you just click on “Update .OCX References” to get your project changed. All of the controls will have the same settings as before you upgraded to the new version.

**Note** – You can update the OCX version of the utility using the utility itself but make sure that if you do this, you compile the utility to the EXE form and then run that. Otherwise, the modules, forms, classes etc. will be open in the VB6 IDE and the upgrade will not go as you anticipate.

**Compile Without the OCX Files** – This choice from the main menu allows you to compile your program with the StdEXE files embedded into your program such that after the compilation you will not need the OCX file any more. It does not change any of your files that you have been using for development using the OCX versions of the controls.

This option should be used *after* you get your program to compile and run with the OCX version. It is much easier and more productive to develop the program using the OCX version and then use the StdEXE version to make an executable file that can be distributed. Note that all of this presumes that you have downloaded and registered the OCX version(s) of VBCCR and VBFlexGrid and also have downloaded and unzipped the equivalent versions of the StdEXE version(s) as well.



At the very bottom of the form are entries for 3 important items for this utility to function properly. If you are going to compile the project by swapping the OCX version for a temporary StdEXE version then you need to know where VB6.EXE is because this utility will execute it later when you click on “Compile w/o .OCX’s” above.

There are two locations towards the bottom of the form to specify the locations of Krool’s StdEXE versions (ComCtlsDemo and VBFlexGridDemo) for you to specify. You do not need these to specify an update to the OCX versions but you do need to specify these if you wish to do a commandline compilation with the StdExE controls instead of the OCX version of the controls. These files do not ever get modified so I put them in my library and I always keep the latest versions in a folder called “Current”. You don’t need to call the folder by that name but you do need somewhere to put the new files downloaded for VBCCRxx and/or VBFLXGRDxx.

The other choice to make for the commandline compile is whether you wish to retain the temporary files used for the commandline compile for possible future use. If you select to save the support files you can later open the VB6 project (that now uses StdEXE instead of the OCX controls) and you can safely edit, compile and run the project. I normally do not save these support files because I prefer editing with the OCX file and when I want a new self-contained utility I just re-run this utility.

Note – If you elect to save the support files so you can open the StdEXE version in the VB6 IDE, there are some modifications I have to make that are transparent to you but are important to note so you aren’t surprised. For example, I do not include any control property page files (.pag extension) since these are only used within the IDE and we skip the IDE with the commandline compile. However, if you want to save the support files so you can later open this in the IDE then we need to make sure that we adjust your project file to reference the appropriate .pag files so you can open it in the IDE. Likewise, in the commandline compile we aren’t concerned about crashing the IDE so we skip the code used to provide IDE crash protection for Krool’s controls. But if you want to open these later this protection is restored. I will discuss how to do this later in my section on how to set up your project to develop with OCX and finalize with StdEXE.

In addition, when you use the OCX version of the controls there is coding already in the OCX file that provides a safe environment in the IDE which is necessary due to the hooks and callbacks Krool uses. In a commandline compile you do not need this code but if you elect to save the support files I add a conditional compilation constant into the project set to true:

ImplementIDEStopProtection = -1

Then somewhere in your initialization code (I have a discussion on this later) you need something like the following:

#If ImplementIDEStopProtection = True Then

' The following sub is in ComCtlsBase.bas

ComCtlsInitIDEStopProtection

' constant set to False for commandline compile of Krool's controls

#End If

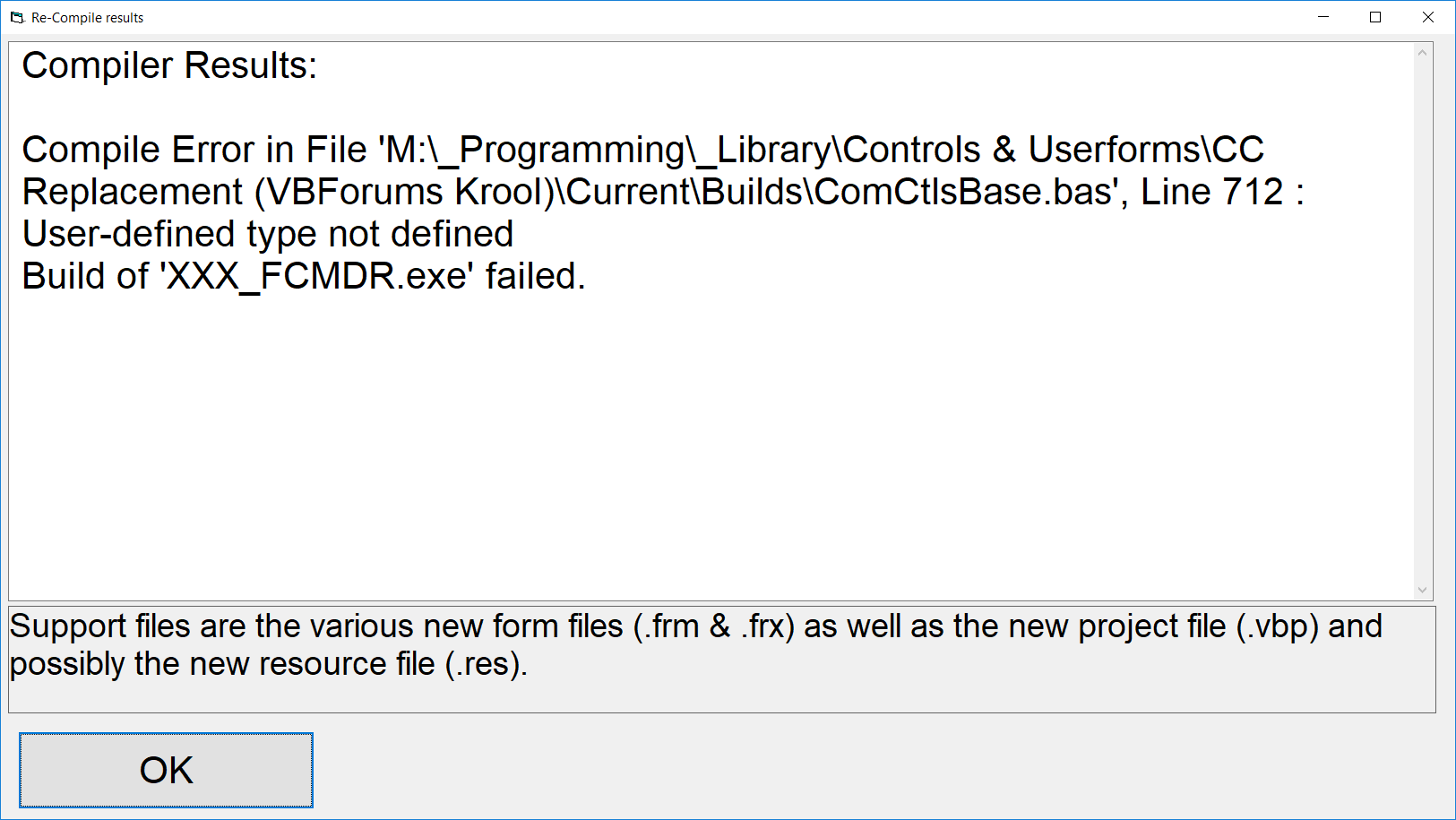
The above is never required for the OCX versions of the controls. It is only required for the StdEXE version if you bring up the project in the IDE and the way this occurs using my utility is if the support files for the commandline compile are saved and reopened in the IDE and the only time this conditional constant is set to be True is when I include the line in your saved .vbp file to set it True.

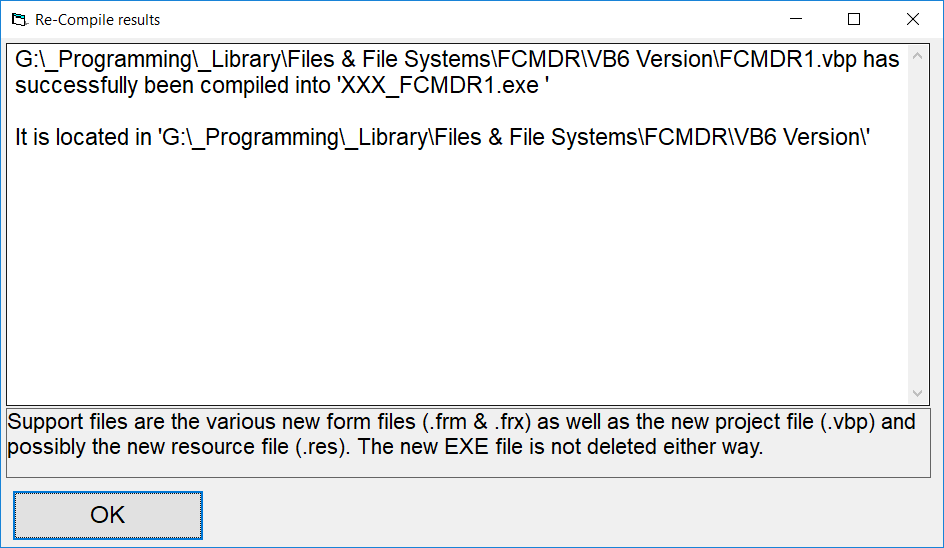
Compiling – When you click on the “Compile w/o .OCX’s” button, the utility will look through your project file, all of your controls, modules and the manifest/resource file for references to the OCX controls and it changes them to the StdEXE controls and the references to the StdEXE controls are put in the .vbp file which is then compiled using the VB6.EXE program from a commandline (we shell out to an elevated command prompt).

None of your project files are modified. Every one that is modified is saved in a new file of the same name with “XXX\_” at the start to indicate a new file. For example, if you are using “myProgram.vbp” it would be saved as “XXX\_myProgram.vbp” and the executable file would be named “XXX\_myProgram.exe”. If you elect not to save the support files, all of the XXX\_ files are deleted after the compile except the executable file. (Note – If you use the Excel version of this utility the prefix is YYY\_ instead of XXX\_.)

**NOTE** – Please know that the first few times you use this utility to do a commandline compilation you are likely to get some naming conflicts. The names of all of the control, page property, class files etc. (basically everything inside of the Builds folder) are “hidden” in the OCX but all of the file names and all of the public variables, types, procedures enums, etc. are all visible when you have your program compile with all of these controls included with your program. At this point you get to decide whether to rename your code or Krool's. I would like to have kept mine and renamed Krool's (I had 2 conflicts) but I did not because I didn’t want to track down each use of his and because I didn’t want to go through the renaming hassle every time I downloaded and update. But either way will work.

Below are two screen shots, one of a successful compile and one that failed.





### Running OCX2StdEXE with Commandline Options

There is now a commandline version of the utility that can do either OCX version upgrade or OCX compile using the StdEXE version just like described above but without the input form. Note that in both cases if the full path to your project has any spaces in it that the full path must be in quotes.

OCX2StdExe ProjectPathAndName [/s[1][2]] [/A[-][+]]

This means do a compile of the project that uses the OCX version with the StdEXE version to embed the code in the executable. If you specify /S then no files are saved after the compile other than the executable (assuming a successful compile). If you specify /S1 then the support files for the used controls are saved and if you specify /S2 then all support files for all controls are saved regardless of whether the controls are used in the project. If yo udo not specify a /S switch then whatever value was saved from the last time yo used the program with the dialog box is re-used.

You can also specify whether or not all of the StdEXE controls are used in the commandline compile (this is different from whether any of the files are saved after the compile). Specifying /A or /A+ says use all controls and /A- means do not use all controls. If you do not specify the /A switch then whatever the currently saved value for /A is from you last run is used.

Switch case does not matter. /S is the same as /s.

The compiled version of the program (if successful compile) will be found in the same folder as the original project file and the executable will have “XXX\_” appended to the front of the file name.

OCX2StdExe ProjectPathAndName /u [/CCRxx] [/FLEXxx] [/d]

This causes the specified project to have its OCX controls changed to the values specified in /CCRxx or /FLExxx switches. The “xx” specifies the version to use (must be registered on this PC first). If no “xx” is specified then the latest version registered on the PC running the utility is used. As with the regular version of OCX2STDExe, the old files are saved with a new extension. Specifying /d or /D causes all of those old files containing the old OCX references to be deleted.

## How I Manage Krool’s System

As you probably have figured out, I use Krool’s OCX versions of the controls during development and then use my utility to switch to a compile using Krool’s StdEXE version so I have a self-contained executable to distribute. There are a few techniques I have learned that I will share with you that will hopefully make this whole process very simple.

* I always use the latest version of the OCX files found [here](http://www.vbforums.com/showthread.php?841929-VB6-ActiveX-CommonControls-%28Replacement-of-the-MS-common-controls%29&p=5129155#post5129155) for VBCCRxx and [here](http://www.vbforums.com/showthread.php?855931-VB6-ActiveX-VBFlexGrid-%28Replacement-of-the-MSFlexGrid-control%29&p=5236525#post5236525) for VBFLXGRDxx. The latest versions are at the end of post #1. When you save the VBCCRxx file note that you do a Save As and drop the .docx extension since it is a trick Krool does to avoid a file size limit on VBForums for zip files. I recommend putting the file on your hard drive with the current version as part of the name. The current VBCCRxx file is version 1.6.13 so I save the file “VBCCR16.OCX.rar.docx” as “VBCCR16.OCX v1.6.13.rar”. the latest VBFLXGRDxx version is 1.2.6 at least for now the file size is small enough that is still ahs the .zip extension. I would save this file which is “VBFLXGRD12.OCX.zip” as “VBFLXGRD12.OCX v1.2.6.zip”. While you are on the web page getting the OCX files you should also get the .RES files (resource files) because those will be helpful a bit later.
* Now you have to get the OCX files that are inside the .RAR and .ZIP files into your System folder to use them. If you are using a 32-bit version of Windows you will want to get the OCX files into C:\Windows\System32 and if you are using a 64-bit version of Windows, put them in C:\Windows\SysWOW64. Note that the OCX file doesn’t have the minor version in the file name. For example, the two most recent VBCCRxx.OCX versions are 1.6.12 and 1.6.13 but each is a file named VBCCR16.OCX. If an older version is already in your system folder you can just overwrite it. Note that with any operating system after XP you will have to use an elevated CMD prompt or a file manager such as Directory Opus (my favorite) that takes care of elevation for you. I don’t think you have to put the OCX file in the system folder but I always do that if for no other reason than I know where it is.
* If you haven’t used Krool’s OCX controls at all or you have an updated version (say 1.6 instead of 1.5) then you will need to register the OCX file using regsver32 (don’t need if you just overwrote an older file of the same name).
* If you have updated OCX versions (e.g. from 1.5 to 1.6) then you will want to run my utility for each of your projects that sue the earlier version and update to the most recent OCX version.
* You need to look into the OLEGuids folder of either VBCCRxx or VBFLXGRDxx and copy the type library OLEGuids.tlb to you system folder and register it. Fortunately this file doesn’t change very often (current version is date 9 June 2017) but it is worth checking every now and then to make sure you aren’t using an out-of-date type library.
* In your program you must reference the type library in the IDE via Project | References and put a check mark by “OLE Guid and interface definitions”.
* Enable the controls in the IDE via Project | Components (or Ctrl-T) and then select the appropriate controls files. For VBCCR version 1.6 you would put a check mark by “VB Common Controls Replacement 1.6 Library”. For VBFLXGRD version 1.2 you would put a check mark by “VB FlexGrid Control 1.2”. All of the controls should now show up in your toolbox.
* The default start action in a VB6 program is to load and display a form. You can’t do this with these controls because there is some initialization code that needs to run before any form is referenced, loaded or displayed. First you need to have Sub Main in a standard module in your project. Then you need to change the setting in Project | Properties on the General tab to make the Startup Object be a call to Sub Main instead of any forms. Then within Sub Main you need some initialization code so that you don’t crash when you call the first form. There are 2 ways to do this.
  + The first is to use Krool’s code directly, albeit in a strange way. The OCX control package doesn’t contain any guides or code on how to use it (it does have some code in later versions that would enable you to make your own OCX file from source code but that isn’t the same as a go-by for use). If you download the StdEXE version you will find a folder called “Common” and inside that are files “Common.bas” and “VisualStyles.bas”. There is a sub in VisualStyles.bas called “InitVisualStyles” which is what you need to run before calling a form but if you just include VisualStyles.bas in your program you will find that it needs some routines from Common.bas so you have to load it too. So if you include these two files in your project, your first line in your Sub Main would be a call to InitVisualStyles and then you can call your forms.
  + The route I use (as you can see in the source code for my utility), I have a general standard module in my library called mVB6Core.bas in which I put enough of the code from Common.bas and VisualStyles.bas so that I can run the sub InitVisualStyles and I can do a lot of other stuff I always do (checking whether we are in the IDE or running compiled code, the current Windows version, etc.). Below is the code I have in my general initialization routine called UCCoreInit (line 1695):

If OSVer >= 600 Then

Dim ICC As InitCC

If App.LogMode <> 0 Then Call InitReleaseVisualStyles(AddressOf ReleaseVisualStyles)

ICC.dwSize = LenB(ICC)

ICC.dwICC = &H4000&

InitCommonControlsEx ICC

Else

InitCommonControls

End If

* You would think you are good to go. Not quite yet. We have to setup visual styles for each control on each form when they are loaded. Somewhere in the initialization code for each form you need to call SetupVisualStyles and pass the form to the sub. This code is in VisualStyles.bas and it is also part of my mVB6Core.bas library. This sub SetupVisualStyles makes sure that all of the controls on the form are able to use visual styles. So you either put this call in the Form\_Load sub for each form you use. My method is close to this. I always have my forms use my class library clResizer.cls and even if I turn resizing off for a form I still have in the initialization code for my form a call to this class module which in turn calls the sub SetupVisualStyles. This way I didn’t have to make that call be a part of the code for each new form since my class library handles it. But either way works just as well.
* On to manifest and resource files. Krool’s controls have nothing to do with resource files per se other than being a container for a manifest file. You are not required to use a manifest or a resource file but if you want to take advantage of visual styles, side-by-side or high DPI monitors then you must use a manifest and embed it into a resource file and then load that resource file into your executable. If you are planning to continue to use the OCX version and not do the final compile with the StdEXE version of the controls then you will likely want to use the side-by-side option. I do ***not*** do this. There is no advantage to specifying side-by-side on your development PC and in fact it causes a hassle because when you make the EXE file on your PC with the OCX version, if you have specified side-by-side on your own PC you have to have the OCX file in the same folder as your EXE to run the EXE even if it also resides and is registered in your Windows system folder.

Instead, I use a resource file named OCX2StdEXE.res for all new projects (included with my utility) that specifies the use of visual styles and high DPI monitors but does not specify the use of side-by-side. If there are other files you need to have side-by-side it is okay to put them in the resource file, just don’t bother putting Krool’s OCX controls in there.

Note – Even if you specify side-by-side and then decide to use my utility nothing bad will happen. My utility will strip out the side-by-side specification from inside the resource file since it is not needed or wanted when all of the code for the controls is included in the executable. As usual, your original files (the ones you have been using to develop with the OCX version) are unchanged by this.

* Now it is time to plan a bit for later. We are using the OCX controls for development. We do not need to worry about using these in the IDE and crashing it because Krool has taken that into account in his OCX. When you do a commandline compile I handle that as well. An issue arises when we take code that does not need it (the OCX controls themselves plus your code) and you specify that you want a commandline compile but you want to save the support files so you can open it later in the IDE. Krool controls the IDE Stop protection code via a conditional compilation constant called “ImplementIDEStopProtection”. In his demo for the StdEXE version there is a file named ComCtlsBase and on line 3 is this conditional compilation constant set to True which later in that module cause the IDE stop protection code to be compiled. We don’t want that to happen in the commandline compile so for the commandline compile I use a copy of ComCtlsBas.bas with the conditional compilation constant commented out. In the post-commandline compile not only do you want this constant set True for all modules you want to call the sub “ComCtlsInitIDEStopProtection” which turns on the IDE Stop protection when you run your/his code in the IDE. I account for this in the library standard module I include with the utility called mVB6Core.bas. If you look on line 1643 you will see the following code:

#If ImplementIDEStopProtection = True Then

' If you use Krool's controls, the OCX version does not need IDE protection

' but the StdEXE version does. If you are using the OCX version and then compile using

' this utility (commandline with switch to StdEXE) and elect to keep the files you will

' need to execute the following sub. At the end of the compile part of my utility I add

' ImplementIDEStopProtection = True to the list of compilation constants in the new

' .vbp project file (the one that now references the individual controls not the OCX.

' The following sub is in ComCtlsBase.bas

ComCtlsInitIDEStopProtection ' constant set to False for commandline compile of Krool's controls

#End If

Since normally the conditional compilation constant is 0 (it is not a visible part of the OCX version of the controls) then the call above to ComCtlsInitIDEStopProtection does not occur (and it shouldn’t) but if you specify to save support files in the utility, the saved project file sill have an additional conditional compilation constant ImplementIDEStopProtection set to -1 (True). The only time the procedure ComCtlsInitIDEStopProtection is even present is in this post-commandline compile situation with the saved support files so the above code does nothing outside of that situation. If you use your own initialization code you need something like this in your code if you want to be able to open the post-commandline compile support file version. Note that this is not really necessary. If you are able to compile your program with the OCX version of the controls then it is extremely likely that the commandline compile of your code with the StdEXE version of the controls will be successful and you will never need to even look at the support files (and will likely not even keep them).

* So now you can develop your program. You will be using the OCX version of the controls. You can use the VBCCRxx.OCX and/or the VBFLXGRDxx.OCX packages.
* If a new version of the OCX controls comes out ([here](http://www.vbforums.com/showthread.php?841929-VB6-ActiveX-CommonControls-%28Replacement-of-the-MS-common-controls%29&p=5129155#post5129155) for VBCCRxx and [here](http://www.vbforums.com/showthread.php?855931-VB6-ActiveX-VBFlexGrid-%28Replacement-of-the-MSFlexGrid-control%29&p=5236525#post5236525) for VBFLXGRDxx) that has the same main version as the one you are using (say you have 1.6.13 installed and 1.6.14 comes out) just copy the OCX file over the existing one. If a new main version comes out (current is 1.6 so 1.7 would be the next one) simply copy the OCX to your system folder and use regsvr32.exe to register it and then use my utility to adjust each of your programs that use the older version to the new version.
* Also keep track of the StdEXE releases as part of Krool’s ComCtlsDemo ([here](http://www.vbforums.com/showthread.php?698563-CommonControls-(Replacement-of-the-MS-common-controls))) and the VBFlexGridDemo ([here](http://www.vbforums.com/showthread.php?848839-VBFlexGrid-Control-(Replacement-of-the-MSFlexGrid-control))). There are no versions per se but the most recent version on the web site corresponds very closely (if not exactly) with the latest OCX version. There used to be some differences (OCX version tended to lag a bit behind in terms of bug fixes and additional features) but for the past year Krool has done a very good job of keeping the two the same.
* Whenever you want to send out a version of your executable, run my utility and select the option to compile without the .OCX’s.
* When doing a commandline compile you will need files for each control. The utility determines which ones are needed and includes them. However, there are some general files in the Common and Builds folder of ComCtlsDemo that may or may not be included. As the developer I give you the choice of including any of them or not. You will find these in the Function DoCompile in the module zVBandVBA starting on line 171. My values are shown below.

Const IncludeStartupbas As Boolean = False

Const IncludeCommonbas As Boolean = True

Const IncludeVisualStylesbas As Boolean = False

Const IncludeISubclasscls As Boolean = True

Const IncludeVTableHandlebas As Boolean = True

Const IncludeVTableSubclasscls As Boolean = True

Const IncludeCommonDialogcls As Boolean = True

I do not include Startup.bas because it is specific to Krool’s demo program although I have used bits of his routine in my own.

I do not include VisualStyles.bas because I have incorporated that code into my initialization routine for both the OCX and StdEXE versions. VisuaStyles requires some functions and subs from Common.bas but I only took a part of them along with VisualStyles and made them Private so that for the commandline compile I can bring in the whole of Common.bas. Finally, specify that CommonDialog.bas can be included. Note that it is not automatically included. It is only included if/when the control MCIWnd.ctl is used or one or more of the property pages for CoolBar, Image<List, RichtextBox or StatusBar are used. Property pages are nt used in the commandline compile option but if you elect to save the support files for later re-compiling then the property page files are included.

**Version History**

| **Version** | **Date** | **Comments** |
| --- | --- | --- |
| 0.9.0 | 2 Jul 2017 | * Initial VB6 & Excel releases for beta test. |
| 0.9.1 | 4 Jul 2017 | * Works with no controls but 1 or more references to CommonDialog * Comparisons within the .vbp file are done uppercase * Does not include reference to stdol2.tlb if the .vbp file already has that reference. |
| 0.9.2 | 9 Jul 2017 | * Correctly handles situation where no controls are used but CommonDialog is called in a Form (used to just check for this in class and standard modules). |
| 0.9.3 | 31 Jul 2017 | * Fixed path error for forms in new .vbp file |
| 0.9.4 | 23 Aug 2017 | * Reworked some relative path stuff |
| 0.9.6 | 11 Jul 2018 | * Adjusted to function name change in mUCCore of myQuickOpen to FileCreateOrOpen. |
| 0.9.7 | 1 Oct 2018 | * Add support for VBCCR16 * Broke out VB6 core code into own mVB6Core.bas, parallel with mUCCore for VBA |
| 0.9.8 | 24 Nov 2018 | * Many, many changes |
| 0.9.9 | 1 Dec 2018 | * Add support for VBFLXGRD.OCX and VBFlexGridDemo.vbp (StdEXE) * Combined all input onto 1 form in VB6 version * Made 3 option buttons for amount of post-compile support file saves (none, controls used, all controls) * Modified logic to include or exclude various modules from Common and Builds in compile * Modified logic around conditional compilation constant ImplementIDEStopProtection especially for post-compile re-use * New logic for include or exclude of CommonDialog.cls in compile &/or post-compile support file save. * Use enums to help manage code around whether VBCCRxx &/or VBFLXGRDxx are used in a given project. * Added commandline option for Upgrade or Compile. |
| 0.9.10 | 11 Dec 2018 | * Fixed bug in DoCompile if VBCCRxx or VBFLXGRDxx was not installed/registered * Fixed bug in DoCompile on extracting manifest from resource file when resource file had no embedded manifest |
| 0.9.11 | 12 Dec 2018 | * In DoCompile look in class and standard modules for references to VBCCRxx and/or VBFLXGRDxx (had done forms and the resource file but not .bas and .cls files) * On fmInput, cbutProjFile defaulted to this program's path even when a current file was already in the dialog box. Now, goes to that file's folder. |
| 0.9.12 | 13 Dec 2018 | * New check box on fmInput for DoCompile to force the use of all of StdEXE controls in the compile which is not mornally needed but if someone puts a reference to a property of a control in the OCX-based project we can see it as part of VBCCRxx but without going through each of the properties for each control we wouldn't know which control it is connected to. I don't think this is a normal occurrence so I put it there just in case someone did this in their OCX-based code. |
| 0.9.13 | 19 Dec 2018 | * Code clean-up * Moved all compile options to 2nd form |