

Contents

Welcome

[The RIP Family](#)
[Introducing RipTABS](#)
[Registration & Licensing](#)
[Feedback & Support](#)
[New In This Version](#)
[Installation & Packing List](#)
[Interface Primer - Folder Tab Dialogs](#)
[What RipTABS Does For You](#)

Implementation

[Requirements](#)
[About The Sample Source Code](#)
[Building Your Dialogs \(Resource Editor\)](#)
[Adding The Support Code](#)
[Final Touch-Ups](#)

Tips, Tricks & Notes

[Things To Know & Remember](#)
[Troubleshooting](#)

Reference

[RipTABRegister](#)
[RipTABUnregister](#)
[CTabDialog::CTabDialog](#)
[CTabDialog::InitData](#)
[CTabDialog::OnOK](#)
[CTabDialog::OnCancel](#)
[CTabDialog::OnInitDialog](#)
[CTabDialog::SwitchToNotification](#)
[CTabDialog::SwitchFromNotification](#)
[CTabDialog::OnPaint](#)
[CTabDialog::OnLButtonDown](#)

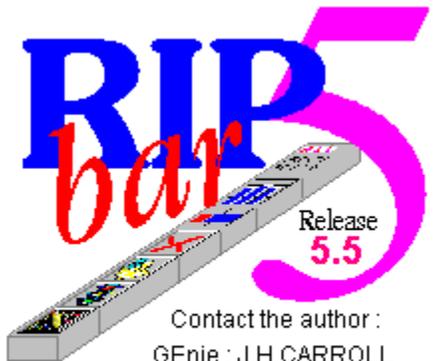
The RIP Family

RipTABS is a library derived from code originally written for a number of other applications that sport the RIP title. Take a moment to read about them.



The RIP family of applications / utilities is a collection of shareware programs written for Windows and Windows NT. Each is designed to fulfill a purpose in an elegant, easy-to-use fashion. All RIP applications have a registration fee which you are expected to pay if you continue to use them. The amounts asked for are reasonable and I ask that you think twice about continuing to use something you haven't paid for. None of the RIP applications have disabled functions, or annoying "Register Me" reminders-- so I leave it up to you.

RipBAR - The Application ToolBar For Windows & Windows NT



Contact the author :
GEnie : J.H.CARROLL
CompuServe 74017,3242
Internet 74017.3242@compuserve.com

©1993, 1994 Jonathan Carroll

RipBAR is an application toolbar that you may set up with applications and documents of your choice. Configuration options for each item include the ability to set up startup directory, and command line parameters. Additionally, files may be dropped onto the application icons sitting on the bar-- causing the application to launch and open that file automatically.

The Windows 3.1 version displays the time, memory information, and the remaining resources available to the system. The *Resource Tracker* (Windows 3.1 only) is included to allow you to monitor an

application's use of memory and resources.

The Windows NT version displays the time, along with available physical and paged RAM. It also includes the PErformance Tool -- displaying summary information about CPU performance.

In addition, RipBAR supports attaching sound to certain types of events, HotKey activation, program groups, a Post-It style Notes feature and much more.

Was recently written up in *PC Week Magazine*.

RipSPACE - The Disk Space Reporting Tool For Windows & Windows NT



RipSPACE is an application written for Windows & Windows NT that analyzes a drive and the space that each sub directory on it, consumes. In other words, you'll now be able to find out how much space your Windows installation is taking up, or how much drive space you'll gain if you delete the WordPounder word processor you installed but never use.

RipSPACE also allows you to define file types that are to be included in a file report. You can then tell how much total space is being taken up by certain file types-- i.e. Bitmaps, Dynamic Link Libraries and executable files.

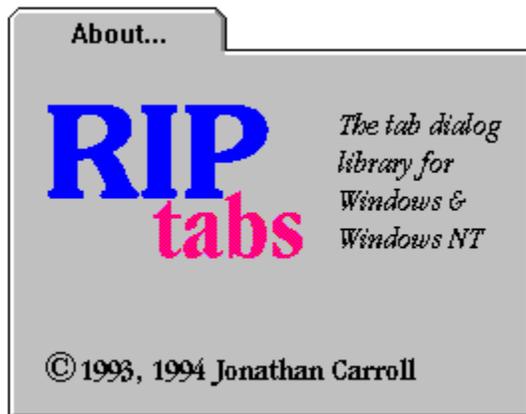
RipFIND (April 1994)

RipFIND is an application written for Windows & Windows NT that simplifies the process of locating files. RipFIND can be instructed to perform searches based on a variety of criteria or qualifiers.

RipFIND qualifiers include DOS-style file specification, string-based specification, and file content specification.

Jonathan Carroll

Introducing RipTABS



(See the end of this section for important information regarding this unregistered version)

RipTABS is a C++ library designed for applications that use Microsoft Foundation Class Library support. It provides developers with an easy-to-use path to the dialog folder tab interface that is becoming popular in a number of new Windows applications including Word 6 and Excel 5 from Microsoft. For more information on what tab dialogs look like and why they are important, see [Interface Primer - Folder Tab Dialogs](#).

The RipTABS library is provided for both Windows 3.1 (Windows For Workgroups 3.1, 3.11) and Windows NT. These are available as separate downloads from CompuServe. (Both appear in the MSLANG forum libraries.)

Note that the Windows 3.1 version is currently provided for the Large memory model only.

Specifically, RipTABS provides you with an MFC-based CDialog class called CTabDialog from which you can derive dialog classes anywhere where you derive from CModalDialog or CDialog.

RipTABS is particularly easy to implement because it shouldn't be necessary to change much of your existing code to make use of the folder tab interface-- in other words, much of the new functionality you'll get will be for free!

Please read carefully through all of the topics in this help file as there are a number of issues concerning memory management and the like that you should be aware of. Resisting the temptation to save a few minutes by not reading through the documentation now, will save you hours of trouble later.

Using the Unregistered Version

If you have not registered RipTABS, then it is presumed that you are evaluating this library-- you're expected to pay the registration fee if you actually use it in your programs.

The unregistered version will always place the library title and a copyright notice in the title of any RipTABS dialog. You should not attempt to modify the dialog box caption text through a call to SetWindowText or some other call because the library code checks the title from time to time, and if it finds the text has been modified, the dialog box will stop painting (updating) itself. The dialog will also stop working properly if you set the dialog style such that there is no title bar.

Other than the pre-set dialog box caption text, this library is fully functional with respect to the registered / release version.

Registration & Licensing

Registration

RipTABS is available free from various online service including CompuServe. What you get, is an evaluation copy of the library, documentation, and some sample source code. You should have enough to determine whether you like the RipTABS implementation of tab dialogs and whether it is suitable for your purposes.

You should know that the unregistered version will always place the library title and a copyright notice in the title of any RipTABS dialog. You should not attempt to modify the dialog box caption text through a call to SetWindowText or some other call because the library code checks the title from time to time, and if it finds the text has been modified, the dialog box will stop painting (updating) itself. The dialog will also stop working properly if you set the dialog style such that there is no title bar.

Should you want to continue using it, there is a \$40 registration fee. For this fee, you'll have access to support (via electronic mail) for the library, notification of new releases. Printed documentation is also available for a nominal charge. Of course, you'll also get a release version of the library that doesn't set the caption text of dialogs with RipTABS copyright information and versions for memory models other than the Large Memory Model.

The \$40 will get you one complete version of the compiled library-- either for Windows or Windows NT. If you purchase both libraries at once, the package price is \$60.

Source code is available for \$100 and you can request compiled builds of the library for no additional charge.

License

When you purchase a RipTABS library, you may use it to build applications and you can distribute and/or sell your applications that include RipTABS dialogs *without any additional royalties*.

*However, you may **not** redistribute the library itself-- nor may you include the RipTABS library in any other library that you build for the purpose of delivering some kind of "enhancements" tool to third parties. In other words, RipTABS may *only* be included in an *application* (whether that inclusion be in the form of an executable file or dynamic link library.)*

You must purchase one copy of RipTABS for every developer who will be using the library.

Feedback & Support

Feedback

This is the second release of the RipTABS dialog library.

Should you find a problem with the library, or have any suggestions for improvement (with either the library or this documentation) please do not hesitate to contact me at one of the addresses listed below.

Support

Support for RipTABS (i.e. implementation, design etc.) is provided to those who have registered RipTABS. You can contact me at one of the addresses listed below.

Reaching Me

CompuServe (*preferred method*)

74017,3242

Internet

74017.3242@compuserve.com

GEnie

J.H.CARROLL

Conventional Mail

Jonathan Carroll
28 Parkland
Pointe Claire, Quebec
Canada, H9R 2E8

New In This Version

This is Version 2.0 of RipTABS.

The following is new or improved in the latest release:

- RipTABS contains two new functions -- RipTABSRegister() and RipTABSUnregister(). See the tutorial and Reference sections for more information-- including [RipTABRegister](#) and [RipTABUnregister](#).
- RipTABS now supports the use of hotkeys to switch between tabs. For more information on how this works, see [Interface Primer - Folder Tab Dialogs](#).
- Full support is provided for modeless dialogs-- a number of omissions in the last release made modeless dialogs a problem. A new constructor for the CTabDialog class is provided for this purpose.
- Tab names are now controls that can receive focus-- this focus is displayed visually by a bounding rectangle.
- When switching tabs, focus is now initially set to the active tab-- emulating the behavior in products like Microsoft Word and Excel.
- The [Things To Know & Remember](#) section has been added. It contains a number of caveats about using RipTABS, a few troubleshooting tips and highlights of important things to remember.
- Problems that sometimes occurred when more than one control in the dialog had the same accelerator key, are now fixed.
- Some people reported that during the link process, Visual C++ reported that some symbols (MFC ones) were defined twice-- this problem has been fixed.

Installation & Packing List

RipTABS is available in versions for both Windows and Windows NT. The header file is common between the two versions-- just the library file is different. The demo Windows NT lib file is RIPTAB32.LIB and the Windows one is RIPTAB16.LIB. For descriptions of other libraries included with the registered version of RipTABS, see below.

The ZIP file you got, should contain two files-- the release notes in Windows Write format, and RIPT_LIB.ZIP.

To finish installing RipTABS, simply create a new directory anywhere on your system and extract all the files from RIPT_LIB.ZIP into it-- use the -d option (or whatever option your UnZip utility supports) to extract the files in the proper directory structures-- Building the sample application depends on the directory structure that has been set up for you in the ZIP file.

After extracting the files, there should be the RipTABS Windows Help file in a directory called HELP, the RipTABS include file in a directory called INCLUDE, and the sample code (including the executable file) in a directory called SAMPLE.

There will also be a CTL3D DLL in a directory called SYSTEM. If you don't already have the same file in your Windows SYSTEM directory or if the one you have is older than the one supplied here with RipTABS, move it to your SYSTEM (or SYSTEM32 in NT) directory.

In the LIB directory, the files you have will depend on the version of RipTABS you have.

For Windows 16 bit:

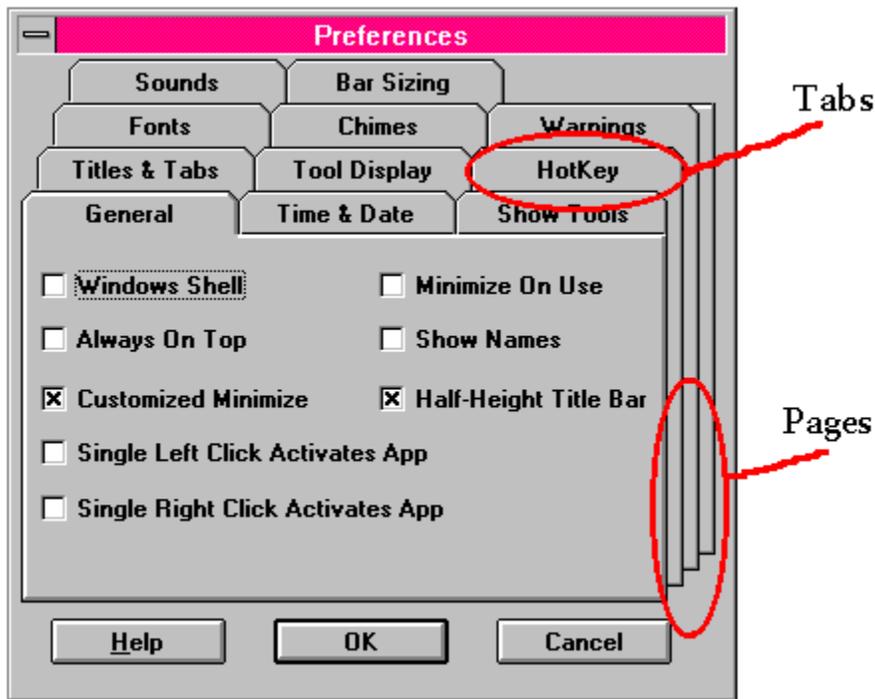
- RIPTAB16.LIB (*demo version*) Large Memory Model
- RIPT_MCR.LIB (*registered version*) Medium Memory Model
- RIPT_LCR.LIB (*registered version*) Large Memory Model

For Windows NT:

- RIPTAB32.LIB (*demo version*)
- RIPT_NCR.LIB (*registered version*)

Be sure to include the .LIB file for linking in any project you're going to use the RipTABS library with.

Interface Primer - Folder Tab Dialogs



Quick Introduction To Tabs

It's no secret -- the widespread acceptance of Windows and other graphically-oriented operating environments has allowed applications to enable the user to do far more things in a wider variety of fashions that text-based systems could ever hope to offer.

One dilemma for those who write software is trying to present information and options to the user without becoming overwhelming.

In the next while, you'll notice the "Folder Tab" interface becoming increasingly popular. Folder Tabs allow for the logical grouping of options and settings within one dialog-- doing away with millions of nested dialogs or menu items. It also takes closer to the "desktop" metaphor.

The Folder Tab implementation in RipBAR is designed to closely resemble (both visually and in terms of behavior) those of Microsoft. (Take a look at Word V6 or Excel V5)

Tab Interface Behavior

The Folder Tab dialog presents groups of options in *pages*. Each page has a number of tabs. For those with a mouse or similar input device, navigating the dialog is easy-- just click on a tab to bring that folder to the front.

Keyboard navigation is possible. Pressing CTRL+TAB will cycle forward through the tabs and pressing CTRL+SHIFT+TAB will cycle backwards through the tabs. In addition, when the tab title has input focus (the tab will be highlighted by a bounding rectangle), typing the first letter of another tab will cause that tab group to become active.

Each time a new tab group is activated, the tab for that group gains focus. Pressing the TAB key will begin the navigation through the rest of the dialog's controls.

What RipTABS Does For You

In this section, we'll discuss the functionality that RipTABS provides to your applications and cover in a general sense what RipTABS is doing "behind the scenes". If you haven't already read the material and looked at the figure in [Interface Primer - Folder Tab Dialogs](#), it would be helpful to do that now before proceeding any further here.

Now that you know what tab dialogs look like, we'll discuss a little bit about how they behave (in case you haven't used an application that makes use of them).

Tab dialogs are actually made up of several dialogs. First, you have what we are going to call *Master* dialogs-- this is the dialog that frames all of the tab pages and usually contains a number of controls that are common to all tab pages-- i.e. OK, Cancel and possible Help. (Note, this is a deviation from the Microsoft style dialogs, as the Master dialog normally contains no controls.) Next you have what we'll call the *Child* dialogs-- there is one Child dialog for each tab page in the Master dialog.

You might for instance, have one Master dialog and five Child dialogs-- which means you'll have a tab dialog with five tab pages.

RipTABS does a number of jobs for you-- making the implementation of tab dialogs *much* easier than having to write the code from scratch. (There's almost 5000 lines of code that has already been implemented for you in the library.) Here's what RipTABS does :

- Scans the tab titles you pass it, and decides how big each tab needs to be.
- Counts the tabs and resizes your dialog dynamically to make room for the tabs. Tabs are arranged in multiple rows (stacked and offset) if necessary.
- Draws tabs and the 3D borders.
- Handles user interface responsibilities such as responding to left-button mouse clicks, and hotkey and CTRL-TAB key presses to switch between dialog pages.
- Dynamically manages memory to accommodate the Master and multiple Child dialogs.

All that you have to do, is derive your own dialog class from RipTABS' CTabDialog class and pass RipTABS the list of tab titles, and Child dialog resource names.

Requirements

You need Microsoft Visual C V1.5 with the Microsoft Foundation Classes V2.5 that comes with it.

To use RipTABS, you must link the Microsoft Foundation Classes into your application as MFC support is required. As of this release, there was a known problem linking to the DLL version of the MFC libraries-- hopefully a future version of the library will address this problem.

If you are using the demo version of RipTABS and Windows 3.1, you must make sure your application is built with for the Large Memory Model-- a Medium Memory Model version of the library will be made available to registered users.

About The Sample Source Code

The source code for the RipTABS sample application (RSAMP16.EXE for Windows and RTAB32.EXE for Windows NT) is provided. It was built with Visual C++ and its AppWizard. Hardly anything was changed with respect to the generated code-- most of the RipTABS dialog code was added in a source code file call RTABDIAL.CPP and RTABDIAL.H.

The sample application demonstrates both Modal and Modeless dialogs with single and multiple pages of tab groups. Play with the sample a little to find out how the tab dialog behaves.

The Visual C++ project (.MAK) file was included (RSAMP16.MAK for Windows and RSAMP32.MAK for Windows NT) so that you can quickly compile the app yourself. If you are going to try to recompile the provide source code and project file, you'll have to insure to update the project file with the proper paths to the various libraries etc.

The following sections discuss what code you add to your application to implement tab dialogs-- the sample source shown in this guide is taken from the sample source code provided.

The sample source code also makes use of CTL3D -- the DLL from Microsoft that gives a 3D look to dialogs. It is recommended that you also use CTL3D in applications that use the RipTABS library because the two offer each other a complimentary "look". If you don't have the CTL3D .LIB and .H file, you'll need to remove references to CTL3D from the sample source code before you can compile it. To do this :

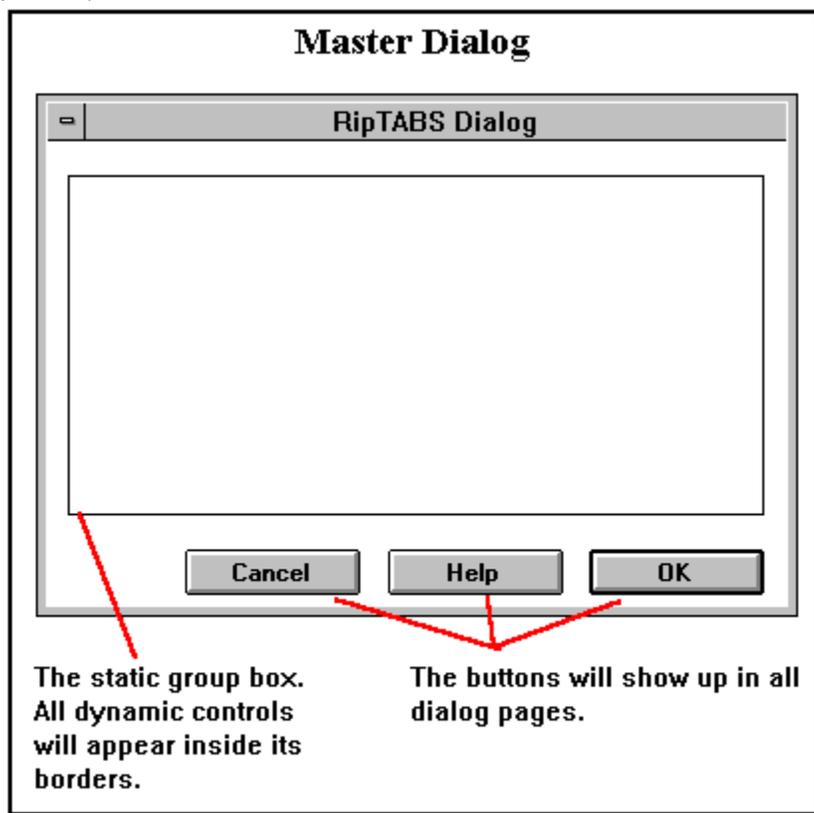
- Remove the reference to CTL3D.H from STDAFX.H
- Remove the initialization of CTL3D from InitInstance in RTAB_SAM.CPP. (There are two lines to remove)
- Remove the reference to CTL3D in ExitInstance in RTAB_SAM.CPP.

Building Your Dialogs (Resource Editor)

For your first implementation of a RipTABS dialog, go simple.

The Folder Tab interface is designed to allow for a dialog to present a lot of information and options to the end user in a relatively small amount of space. Pick an existing dialog that contains a lot of controls that can be logically grouped into two or more "families".

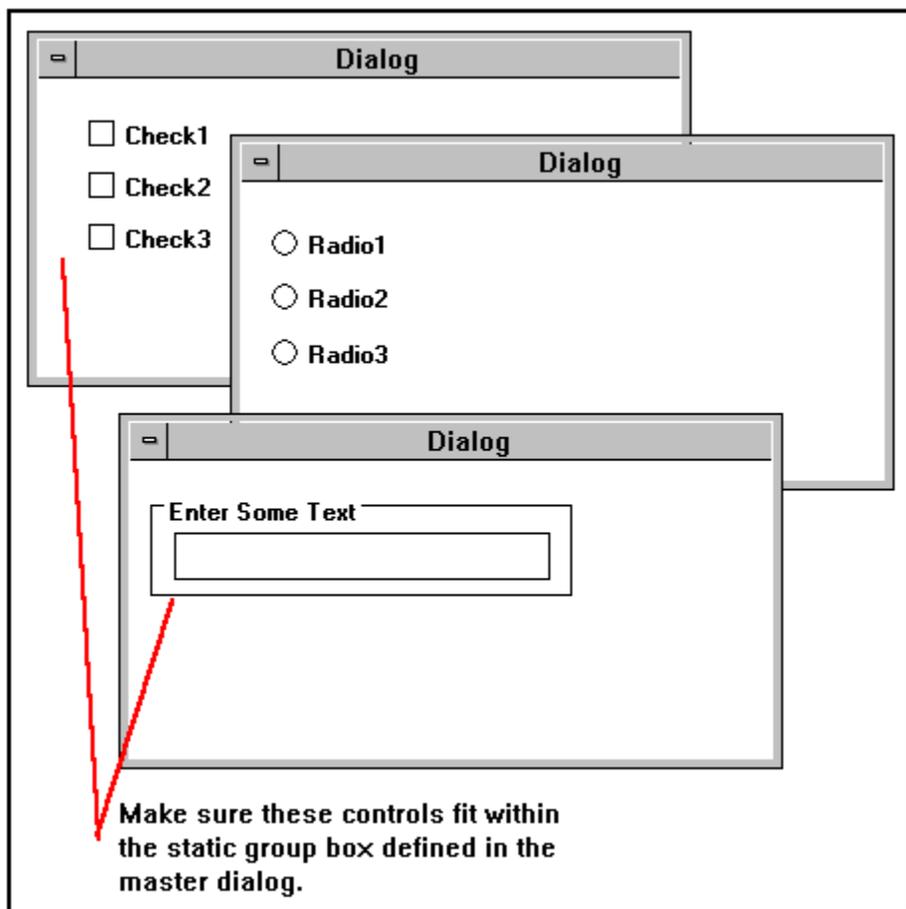
Create a new dialog with your resource editor and make it as large as necessary to fit the largest group of controls and three buttons-- Help, OK, and Cancel. This is known as your *Master* dialog. (see the next picture.)



In this dialog place three buttons (Help, OK, and Cancel) at the bottom right of the dialog. Make sure that your OK button has a resource ID of IDOK and the Cancel button has a resource ID of IDCANCEL. (This ensures appropriate default behavior for OnOK and OnCancel.)

Next, draw a group box that extends from the top left of the dialog to the bottom right, but *does not contain the three buttons*. Give this group box a unique resource ID-- doesn't matter what, but it must be unique. All controls will be drawn inside of this box. Save this dialog.

Now go ahead and create additional dialogs. In each one, place the controls that you've designated as being a group. (See the next picture)



It doesn't matter exactly how big each dialog is, but the controls must be placed relative to the upper left corner so that they'd fit if they were pasted into the master dialog's group control. You don't need OK / Cancel buttons in these dialogs. You also don't need to worry about the Caption text for each one-- only the controls from each dialog are used.

Again, the ID for these dialogs should be *string based* and not numeric. (i.e. IDs like "ADIALOG" are ok, IDs like IDD_DIALOG are not.)

Adding The Support Code

[Initializing and Uninitializing RipTABS](#)

[Deriving Your Dialog - Header File](#)

[Dialog Initialization - Overriding OnInitDialog](#)

[Implementing Help - Switch Notification Usage](#)

[Implementing OnOK](#)

[Implementing OnCancel](#)

[Creating & Displaying The Dialog](#)

Initializing and Uninitializing RipTABS

Before you access RipTABS functions from your code, you must call RipTABRegister() from your application's InitInstance member function.

Also, before you terminate, you must call RipTABUnregister()-- it is recommended you do this from your application's ExitInstance member function.

Examples of these two calls follow:

Code Sample - Initializing RipTABS library

```
BOOL CRtab_samApp::InitInstance()  
{  
    RipTABRegister();  
    ...  
}
```

Code Sample - Uninitializing RipTABS library

```
int CRtab_samApp::ExitInstance()  
{  
    RipTABUnregister();  
    ...  
}
```

Deriving Your Dialog - Header File

Declaring your dialog class is easy -- simply derive your class from CTabDialog instead of CDialog. You'll also want a member variable -- a two dimensional CPtrArray* array. The example below (part of the supplied sample source code) also includes overrides for OnInitDialog, message map functions to respond to the OK, Cancel and Help buttons, and a CString variable to hold the text that corresponds to the active page of our dialog.

Code Sample - Defining our tab dialog class.

```
class CRTabDialog : public CTabDialog
{
// Construction
public:
    CRTabDialog(CWnd* pParent = NULL);    // standard constructor
// Dialog Data
   //{{AFX_DATA(CRTabDialog)
    enum { IDD = IDD_RTABDIALOG };
    // NOTE: the ClassWizard will add data members here
    //}}AFX_DATA
// Implementation
protected:
    virtual void SwitchToNotification(int TabIndex, CString* TabName);
    // Generated message map functions
   //{{AFX_MSG(CRTabDialog)
    virtual BOOL OnInitDialog();
    virtual void OnOK();
    afx_msg void OnClickedHelp();
    virtual void OnCancel();
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
protected:
    CPtrArray*        DlgOptionsList[2];
    CRtab_samView*    MyView;
    CString           ActivePage;
};
```

Dialog Initialization - Overriding OnInitDialog

The following code is an example of what goes into our OnInitDialog function. We initialize our CPtrArray* array. We are creating two CPtrArrays. Each of these CPtrArrays holds pointers to character strings. One CPtrArray (in this case DlgOptionsList[0]) holds the titles of our dialog tabs, and the other (DlgOptionsList[1]) holds the resource names of our dialogs (excluding the master dialog).

Please note that the two CPtrArrays hold pointers to regular C character arrays (strings) and *not* CString objects. The CStringPtr function you see in the following example takes a literal string as an argument and returns a char*, which we cast to void* for storage in the CPtrArray.

After setting up the arrays, we call CTabDialog::InitData (See [CTabDialog::InitData](#)) which takes three arguments : a pointer to the array of tab titles, a pointer to the array of resource names, and the resource value of the group box in the master dialog.

Finally, before leaving our OnInitDialog, we call the CTabDialog version of OnInitDialog. (See [CTabDialog::OnInitDialog](#))

Once you have performed these steps, you may initialize dialog box controls (from the Master dialog or *any of the Child dialogs*) such as placing default text into edit controls, placing check marks in check box controls etc. **Don't attempt to do this before calling CTabDialog::InitData and CTabDialog::OnInitDialog.**

You should return 0 or FALSE from your OnInitDialog routine-- this tells Windows that you have specifically set focus to a control in your dialog (something RipTABS does).

*Note : You **should not** delete the two CPtrArrays-- RipTABS will delete them after the dialog box closes. Likewise, RipTABS will delete the strings created in memory for the tab titles.*

Code Sample - Initializing dialog, building our CPtrArrays

```
BOOL CTabDialog::OnInitDialog()
{
    DlgOptionsList[0]= new CPtrArray();
    DlgOptionsList[0]->SetSize(0, 20);
    DlgOptionsList[1]= new CPtrArray();
    DlgOptionsList[1]->SetSize(0, 20);

    // mode info was here
    DlgOptionsList[0]->Add((void*)CStringPtr("Radio Buttons"));
    DlgOptionsList[0]->Add((void*)CStringPtr("Check Boxes"));
    DlgOptionsList[0]->Add((void*)CStringPtr("Edit Control"));

    DlgOptionsList[1]->Add((void*)CStringPtr("RADIO_DIALOG"));
    DlgOptionsList[1]->Add((void*)CStringPtr("CHECK_DIALOG"));
    DlgOptionsList[1]->Add((void*)CStringPtr("EDIT_DIALOG"));

    CTabDialog::InitData(DlgOptionsList[0], DlgOptionsList[1], IDC_RSTATIC);
    CTabDialog::OnInitDialog();

    // place your own controls initialization code here ...

    return TRUE; // return TRUE unless you set the focus to a control
}
```

Implementing Help - Switch Notification Usage

RipTABS provides two notification functions for determining when the user has clicked on a tab in our dialog and thus, switched pages. If you need to know what page a user has switched *from* or what page he has switched *to*, you'll need to know about **SwitchFromNotification** and **SwitchToNotification**.

SwitchFromNotification (See [CTabDialog::SwitchFromNotification](#)) gets called to let us know what page the user switched from. Two parameters are passed : TabIndex and TabName. TabIndex is the zero-based index of the page that the user switched from. (This corresponds to the index in the array we passed in, in our call to InitData earlier.) TabName is a pointer to a CString-- the tab title text.

SwitchToNotification (See [CTabDialog::SwitchToNotification](#)) gets called to let us know what page the user switched to. Two parameters are passed : TabIndex and TabName. TabIndex is the zero-based index of the page that the user switched to. (This corresponds to the index in the array we passed in, in our call to InitData earlier.) TabName is a pointer to a CString-- the tab title text.

In the example below, we make use of SwitchToNotification and assign the tab title to our own CString variable. We use this to simulate tracking of help topics-- in other words, we'll be able to provide context sensitive help in our dialog-- based on the active page.

Code Sample - Use SwitchToNotification to implement a crude form of context sensitive help.

```
void CTabDialog::SwitchToNotification(int TabIndex, CString* TabName)
{
    ActivePage= *TabName;
}

void CTabDialog::OnClickedHelp()
{
    CString MessageText="Display Help For : ";
    MessageText+=ActivePage;
    MessageBox(MessageText, "RipTABS Sample");
}
```

In the example above (OnClickedHelp), we display a dialog with the name of the currently-active tab page. It wouldn't be a big leap from here to implement context-sensitive help. (you'd probably track the tab index rather than the title though.)

Implementing OnOK

If you are going to implement a message map function for when the user clicks on the OK button, you should remember to call `CDialog::OnOK` (See [CDialog::OnOK](#)) before you finish.

Note that even though your controls are spread across many child dialogs, they are all accessible from your `CDialog`.

Code Sample - Responding to user's click on OK button

```
void CDialog::OnOK()
{
    char str[101];
    CString CheckBoxStatus, RadioButtonStatus, EditControlStatus;
    if(IsDlgButtonChecked(IDC_CHECK1))
        MyView->CheckBoxStatus = "1";
    if(IsDlgButtonChecked(IDC_CHECK2))
        MyView->CheckBoxStatus = "2";
    if(IsDlgButtonChecked(IDC_CHECK3))
        MyView->CheckBoxStatus = "3";
    ...
    CDialog::OnOK();
}
```

Implementing OnCancel

If you are going to implement a message map function for when the user clicks on the Cancel button, you should remember to call `CDialog::OnCancel` (See [CDialog::OnCancel](#)) before you finish.

Note that even though your controls are spread across many child dialogs, they are all accessible from your `CDialog`.

Code Sample - Responding to user's click on the Cancel Button

```
void CDialog::OnCancel()
{
    // TODO: Add extra cleanup here

    CDialog::OnCancel();
}
```

Creating & Displaying The Dialog

The following code creates and displays the dialog.

```
void CRtab_samView::OnDoRipTABDialog()  
{  
    CRTabDialog diag(this);  
    diag.DoModal();  
}
```

Final Touch-Ups

As you modify, re-arrange your tab dialog, you may have to make slight adjustments to your master dialog size-- i.e. add a little more space on the left, or bottom etc. There is some degree of imprecision in the translation of dialog units and precise pixel mapping. On the whole though, you shouldn't have to do much to get things looking just right.

Things To Know & Remember

This section contains some important things to remember when using RipTABS, and a few caveats that you should bear in mind as you design and work with your dialogs.

- RipTABS switches between groups by disabling and hiding controls in any group that isn't currently active. This means that if you need for certain controls to be disabled or hidden, it isn't good enough to simply mark the control as disabled or hidden in your resource editor. Respond to the CTabDialog::SwitchToNotification message and find out what group has been made active. Then send hide and disable messages to any controls that need them.
- Remember to return 0 or FALSE from your CTabDialog::OnInitDialog member function for any dialog derived from CTabDialog-- this tells Windows that you've specifically set focus to a control and that you want to override the default behaviour of setting focus to the first control in a dialog. RipTABS sets focus to the tab of whatever group has been made active. If you don't return 0 from OnInitDialog, you're effectively overriding RipTABS' behaviour.

Troubleshooting

This section contains solutions for some commonly reported problems.

- If your CTabDialog implementation doesn't seem to be drawing tabs or otherwise behaving as a tab dialog, check to make sure that in the message map for your dialog, you are using CTabDialog and not CWnd or CDialog as the template for messages... i.e.
BEGIN_MESSAGE_MAP(CRTabMDialog, CTabDialog) works,
BEGIN_MESSAGE_MAP(CRTabMDialog, CDialog) doesn't.
- If one of the pages in your dialog appears blank and then as you switch between groups, the rest of the pages start appearing blank as well, it is probably because you have passed RipTABS a bad dialog resource name in your CTabDialog::InitData call. (Maybe a typo)

RipTABRegister

```
RipTABRegister();
```

This function takes no parameters. It initializes RipTABS' internal structures. Call this from your application's InitInstance member function. It **must** be called before using any other RipTABS function. Remember to call RipTABUnregister before your app terminates.

RipTABUnregister

```
RipTABUnregister();
```

This function takes no parameters. It frees up RipTABS' internal structures. Call this from your application's ExitInstance member function.

CFileDialog::CFileDialog

```
CFileDialog(int theDialog, CWnd* pParent = NULL);  
CFileDialog(int theDialog, CWnd* pParent = NULL);
```

Call this as part of the constructor of your derived dialog.

The first version of the constructor is for modal dialogs-- it takes two parameters-- the resource ID of the master dialog, and the parent window.

The second version takes no parameters and is for modeless dialogs. After calling the CFileDialog constructor, you'll have to call the Create function to construct your dialog. The sample source code demonstrates the use of a modeless dialog.

CFileDialog::InitData

```
void InitData(CPtrArray* theOptionsList1, CPtrArray* theOptionsList2, int  
TheStaticBox);
```

Call this after you've set up your arrays of tab titles and child dialog resource names. (The first two parameters) *TheStaticBox* is the resource ID of the static group box in the Master dialog that will surround all the controls from your child dialogs. *Do not pass in the default IDC_STATIC* ID that AppStudio defaults to--- this must be a *unique* identifier.

CFileDialog::OnOK

```
virtual void OnOK();
```

If you are going to override OnOK for your own purposes (to information from the dialog's controls), be sure to finish off by calling the CFileDialog version of OnOK because this function performs important cleanup steps.

CFileDialog::OnCancel

```
virtual void OnCancel();
```

If you are going to override `OnCancel` for your own purposes (to information from the dialog's controls), be sure to finish off by calling the `CFileDialog` version of `OnCancel` because this function performs important cleanup steps.

CFileDialog::OnInitDialog

```
virtual BOOL OnInitDialog();
```

Be sure to call the CFileDialog version of OnInitDialog as this performs the necessary construction of tabs and pages etc.

CFileDialog::SwitchToNotification

```
virtual void SwitchToNotification(int TabIndex, CString* TabName);
```

A notification function that gets called when a user clicks on a dialog tab and thus switches pages. TabIndex is the zero-based index of the page the user is switching to. (Corresponds to the array you passed in during InitData.) TabName is a pointer to the title (text) of the page the user switched to.

CFileDialog::SwitchFromNotification

```
virtual void SwitchFromNotification(int TabIndex, CString* TabName);
```

A notification function that gets called when a user clicks on a dialog tab and thus switches pages. TabIndex is the zero-based index of the page the user is switching from. (Corresponds to the array you passed in during InitData.) TabName is a pointer to the title (text) of the page the user switched from.

CFileDialog::OnPaint

```
afx_msg void OnPaint();
```

This member function paints the interior of the dialog-- tabs, titles etc. If you are going to override the OnPaint for some reason, be sure to call the CFileDialog version of it.

CFileDialog::OnLButtonDown

```
afx_msg void OnLButtonDown(UINT flags, CPoint point);
```

This member function handles left-button clicks to see if the user has clicked on a tab. If you need to check for this message to, be sure to call the CFileDialog version as well.

