

# SuperCDEFs™

World-Class Controls for the Discerning Developer

copyright © 1996-99 Water's Edge Software

(Version 1.3)

---

## U s e r M a n u a l

- 
- Absolutely professional CDEFs
- 
- For all Macintoshes and System versions
- 
- Full color and multi-monitor support
- 
- In 680x0, PowerMac and Fat/Safe format
- 
- The finishing touch for a complete 3D look
- 

Featuring :

14 buttons, 9 tabs and 11 sliders (flat/3D text, flat/3D body, raised/inset title, soft/bold shadows, variable shapes) plus a thermometer and hierarchy triangles.

Customizable check boxes plus "undefined" state

Slider options: tick marks (1 side/2 sides/none), scale (forward/reversed/none), snap or step to mouse, smart scaling, and more



When Apple first introduced Macintosh in 1984, they provided programmers with the basic controls needed in applications: buttons and scroll bars. They were available in one style, and only in black and white.

Over the decade that followed, application development evolved to a highly refined art form with programmers making use of color and crafting style refinements such as the "3D look" that is seen in today's applications. Apple's standard controls, although updated, did not keep pace with the state-of-the-art in application development.

continued...

Table of Contents	
Foreword.....	1
Who Needs SuperCDEFs?.....	2
CDEFs Explained.....	2
What is a CDEF.....	2
How to use a CDEF.....	3
Examples.....	3
Using CDEFs in your application.....	4
Automated CDEFs and beyond.....	4
About SuperCDEFs.....	5
Before you continue.....	5
Button CDEFs.....	6
Tab CDEFs.....	14
Slider CDEFs.....	21
Thermometer CDEF.....	28
Hierarchy Triangle CDEF.....	30
Special Offers.....	33
Acknowledgments.....	36
License & Warranty.....	37

## Who Needs SuperCDEFs?

...continuation

In mid 1997, with the introduction of the Appearance Manager in MacOS 8, Apple finally supplied developers with a greater variety of controls such as sliders and tabs, and a fresh “3D look” for controls. This also meant that developers now had to consider that their applications might run on a Macintosh that may or may not be equipped with the Appearance Manager’s new controls.

SuperCDEFs bridges the gap with a collection of high quality contemporary controls. They were designed and crafted by professional Macintosh developers to be as reliable as Apple’s controls and as easy to implement. Our controls, however, have a few notable differences: they offer styling that ranges from a subtle improvement over Apple’s original design, to a bold “3D look.”

SuperCDEFs also goes far beyond the standard button and scroll bar controls originally issued by Apple. We include additional types of controls to let you realize your creative potential, and to let you include the features that are found in today’s commercial applications.

What kinds of developers need SuperCDEFs? Any or all of the following:

- Commercial developers who don’t have the time, resources or inclination to create professional controls themselves
- Developers who need the finishing touch for a complete “3D look” interface
- Someone who needs sliders and/or tabs, controls that are not provided by Apple on Macs that are not equipped with the Appearance Manager (pre MacOS 8).
- Developers who have tried shareware or freeware controls and have been disappointed, or are ready for top-notch controls used by professionals
- Someone who wants to retain a traditional Macintosh interface, but with a few minor changes to clean up Apple’s shortfalls
- Someone who wants the absolute best in CDEFs
- Someone who may have a tight budget

## CDEFs Explained

This section details the following:

- What is a CDEF
- How to use a CDEF
- About our CDEFs
- Automating CDEFs

What is a CDEF

A CDEF is a Control DEF inition. It is a resource (whose type is 'CDEF') that gives your application the ability to use a particular style or type of control. Push buttons, radio buttons, check boxes, scroll bars and esoteric things like sliders are all controls that are typically made with a CDEF. Apple’s standard controls, those being push buttons, radio buttons, check boxes, and scroll bars, are all CDEFs.

By using a custom CDEF (other than Apple’s), you can give your application controls that look different from Apple’s standard ones. This includes (but is not limited to) 3D buttons that are needed to complete a “3D look” interface. You can also give your application additional functionality that does not normally exist, as is the case with tabs and sliders.

### How to use a CDEF

A CDEF is a resource, so you will need to know how to use a resource editor such as Apple's ResEdit. The CDEF can be located anywhere your application can access a resource. Apple's CDEFs are located in the System file thereby making them accessible to all applications. Your application will likely store its CDEFs in its own resource fork to make it accessible only to your own application.

### Including a CDEF in your application:

All you need to do to give your application access to a CDEF is to copy the CDEF to your application's resource fork. Application development environments such as Metrowerks' CodeWarrior, and Symantec's C/C++ and THINK Pascal all let you include resource files when you are working with a project, so it is best to copy the required CDEFs to the resource file that is used by your project and to let your compiler copy the contents of that resource file to your application's resource fork. The CDEF will then be accessible to your application during development and debugging, and it will eventually be compiled into your finished application's resource fork.

At this point, your application has access to the CDEF but it has not done anything with it.

### The CDEF's Resource ID and proclD:

As required by Macintosh standards, a control's proclD is calculated using the following formula:

'CDEF' resource ID x 16 + variant code

Your CDEF's resource ID can be in the range of 0 to 2047, but it is safest if you use 128 or higher to prevent non-intentional conflicts with Apple's standard CDEFs. Apple's standard push button, check box and radio button's resource ID is 0. Their scroll bar's resource ID is 1, and in System 7 or later, the pop-up menu resource is 63. If you include a CDEF in your application with the same number as Apple's CDEF, your application will use the custom CDEF in place of Apple's (a simple way to easily replace standard Apple buttons with 3D buttons throughout your application).

The variant code is made up of 4 bits (the low 4 bits of the proclD) and is used by the CDEF in any way the CDEF wants. For example, Apple's button CDEF uses these bits to determine the button's style:

pushButProc = 0  
checkBoxProc = 1  
radioButProc = 2

Whenever you get custom CDEFs, read the accompanying documentation to determine how these 4 bits are used. Some CDEFs' proclD is identical to Apple's making it easy to replace standard Apple controls without changing your application (this is what we do). Others use different variant code altogether.

- Example #1: Using 3D buttons instead of Apple's buttons in your application

Here are the steps you would follow to replace Apple's standard buttons with 3D buttons throughout your application. This includes push buttons, check boxes and radio buttons.

1. Get a CDEF that makes 3D buttons. SuperCDEFs are a good choice.
2. Copy the CDEF to the resource file that is used by your project.
3. Make sure that the CDEF's resource ID is 0.
4. Use Apple's standard proclDs when creating buttons in your application. Those standard proclDs are: pushButProc, checkBoxProc and radioButProc. In other words, you don't have to change anything if your application was already using Apple's standard buttons.

- Example #2: Using 3D buttons and Apple's buttons

Here are the steps you would follow to let your application use a custom CDEF without losing access to Apple's standard buttons.

1. Get a CDEF that makes 3D buttons. SuperCDEFs are a good choice.
2. Copy the CDEF to the resource file that is used by your project.
3. Give the CDEF a resource ID that does not conflict with Apple's. Use numbers 128 or higher to be safe. In this example, set your CDEF's resource ID to 128 in your project resource file.
4. The proclD you will use when creating your buttons still follows the formula: 'CDEF' resource ID x 16 + variant code. If the CDEF uses the same variant codes as Apple's CDEFs you would declare and use the following proclDs in place of Apple's proclD when creating custom buttons:  
pushButProc3D = 2048; {128 x 16 + 0}  
checkBoxProc3D = 2049; {128 x 16 + 1}  
radioButProc3D = 2050; {128 x 16 + 2}

#### Using CDEFs in your application

You typically create controls in your application using the Macintosh toolbox's NewControl routine. Like all other controls, your application must detect a mouse-down event in the control and call TrackControl while the user's mouse button is down. Your application must also disable controls on inactive windows, enable them when a window activates, and update them when a window needs to be refreshed. If your control displays text using the window's font, your application must remember to correctly set the window's font, font size and style before any interaction with the control.

#### Automated CDEFs and beyond

Tools Plus Libraries and Framework (by Water's Edge Software) simplify the creation and management of controls by letting you create a working control with a single line of code. Tools Plus's NewButton or NewScrollBar routines not only create a control, they also make virtually any control work and become fully automatic without writing additional code. In fact, when you program with Tools Plus, you can create just about any element of your user interface with a single line of code, and it works! This includes windows, floating palettes, tool bar, editing fields, pop-up menus, buttons, entire dialogs, and much more.

If you need to go beyond the scope of a CDEF to create a truly powerful picture button, Tools Plus also includes extensive facilities for making PICTs and icons into a wide variety of buttons including animated buttons. Tools Plus's picture buttons are the best anywhere, and make a good complement for SuperCDEFs. Tools Plus also has 3D panels and group boxes. Together, Tools Plus and SuperCDEFs give you everything you need for a complete "3D look" user interface.

## About SuperCDEFs

Water's Edge Software CDEFs are written to make them easy to use and compatible with any application. The following features ensure this:

- For any model Macintosh (including Power Macintosh)
- Requires System 5 or later
- Retains the look of the System version on which it is running
- Button CDEFs use the same variant codes as Apple's CDEFs for easy integration into existing applications
- Supports all monitor setting (1 bit, 2 bit, 4 bit, and 8 bit or higher using colors, gray scale, or black and white settings)
- Supports multiple monitor setups displaying perfectly on each monitor regardless of its settings (i.e., a SuperCDEF can be displayed half on a color monitor and half on a 1-bit black and white monitor, and it will still look and work perfectly).
- Fully utilizes Color QuickDraw and System 7 features (but does not require them)
- Our CDEFs look and feel great! They make an excellent addition to Macintosh applications that need the "final touch" to complete a professional looking interface.

Before you continue...

Before you continue reading this manual and get into the details of SuperCDEFs' controls, this document assumes that you already know what a CDEF is and how to:

- Install a CDEF in your application
- Correctly number the CDEF's resource ID
- Create a control in your application using a CDEF
- Make controls work

If you are using Tools Plus libraries, all you need to do is read up on the NewButton and NewScrollBar routines. If you are not using Tools Plus, you should be familiar with the Control Manager, and have a working knowledge of the Window Manager and Event Manager.

## CDEFs and The Appearance Manager (MacOS 8 and later)

The Appearance Manager (available starting in MacOS 8) supports CDEFs just as they are used now. Current CDEFs, those that are not written to take advantage of the Appearance Manager's new services, will not inherit new features that exist only in the Appearance Manager, specifically, the Appearance Manager's ability to change "themes." All CDEFs are affected by this, not just SuperCDEFs.

Apple recommends that application running on MacOS 8 and later use standard system CDEFs whenever possible. However, until the day arrives when all applications run exclusively on MacOS 8 or later, SuperCDEFs will serve a vital role in providing developers with an appearance and types of controls that are not available on Macs without an Appearance Manager, such as 3D buttons, tabs, sliders, thermometers, hierarchy triangles and others. It is estimated that this situation will continue for some time as Mac users slowly make the transition to MacOS 8.

Advanced application development systems such as Tools Plus libraries even let you use or ignore custom CDEFs depending on the capabilities of the Mac that is running your application. This is an excellent way to use Apple's 3D Appearance Manager controls if they are available, or SuperCDEFs if the Appearance Manager is not available.

## Button CDEFs

This section details the button CDEFs that are available in SuperCDEFs. The following items are discussed:

- How our buttons are similar to Apple's
- How our buttons are different from Apple's (extra features)
- Variant codes and other settings
- Pictures of all button CDEFs available in SuperCDEFs

How our buttons are similar to Apple's

- Identical round-corners in push buttons
- Identical positioning of text, check box square, and radio button circle
- Same use of control color table
- Identical appearance and use of colors for selecting, highlighting and disabling
- Same variant codes
- Multiple line titles are created using a carriage return character
- Can be used with or without Color QuickDraw
- Colors map perfectly across all monitor settings and multiple monitors
- Apple user interface guidelines were observed in designing and creating the control

How our buttons are different from Apple's (extra features)

- The inside of the radio button's circle and the check box's square are always white to produce a crisper, cleaner look. Apple uses the control's body color which does not present well when the button is on a color background.
- Like Apple check boxes, our CDEFs support the selected ("X") and unselected (empty square) states. We also support an "undefined" state where a dash ("-") appears inside the check box.
- Our CDEFs let you replace the check box's "x" with an icon of your own design. We include an anti-aliased check mark for good measure.
- Most of our CDEFs feature a 3D look for the check box's square, the radio button's circle and the push button's body. A soft 3D and bold 3D look are available.
- Most of our CDEFs feature 3D text that can be either raised or inset. A soft 3D and a bold 3D look are available.
- We also include mini-buttons that reduce the size of the check box's square and radio button's circle. This works better in some cases where small fonts are used or space is restricted.
- Our buttons observe the control's color table just like Apple's CDEFs. Additionally, if the button does not have its own color table or the color table is set to the default black on white, our buttons take on a light gray coloring for their 3D effects. This lets you have 3D buttons just by adding our CDEFs to your application without having to code anything differently.
- Our CDEF resources are bigger because they run on any Mac and any System version.
- Our CDEFs are available as 680x0 only, PowerMac only, or Fat/Safe Binary format (a single CDEF resource that runs on both 680x0 Macs and on PowerMacs in native mode)
- Apple's buttons are functional while ours provide a variety of looks ranging from "just like Apple's only better" to a complete 3D look.

**Control Values:**

Like ordinary buttons, the control's value (ctrlValue field in the ControlRecord) indicates if the button is selected or not. A value of one (1) indicates the button is selected, and zero (0) indicates the button is unselected. Additionally, any value other than 0 and 1 in a check box is considered to be "undefined," as shown below:



Our CDEFs can also display a custom icon in place of the traditional "x" inside a check box. Set the control's minimum limit (ctrlMin field in the ControlRecord) to the negative value of your 'cicn' resource ID. For example, if you have a 'cicn' resource ID of 240, set the control's minimum limit to -240. You can use any resource ID that is 128 or higher for this purpose. The 'cicn' must be exactly 10 x 10 pixels in size and it must exclude the check box's frame. For mini-buttons, a specially created set of CDEFs that have a smaller check box, the icon must be exactly 6 x 6 pixels. The cicn's black and white component is displayed on Macs that don't have Color QuickDraw. The example below shows a check box using a check mark icon (included with SuperCDEFs) in a selected state:



The roundness of the push button's corners is automatically calculated using the same formula that Apple uses, so any code you have to draw a default button frame around the button will still work perfectly. If you want a slightly different look for push buttons, you can set the control's maximum limit (ctrlMax field in the ControlRecord) to any value greater than 1 but less than 32767. That value will be used for the oval height and width in the toolbox's FrameRoundRect routine when calculating the push button's outline. The smaller the number, the closer the corners get to becoming square. The examples below show the different looks you can attain by varying a push button's maximum limit. The left one is the default. The right one uses a value of 2.

**Variant Codes**

Our CDEFs use variant codes that are identical to Apple's. This lets you easily replace Apple's standard buttons with our attractive 3D buttons in your application. The variant codes are as follows:

- bit 3 (+8) Use window's font
- bit 2 (+4) Inset text, otherwise text is raised. This bit is ignored if the CDEF does not display 3D text.
- bit 1 (+2) Button is a radio button, or...
- bit 0 (+1) Button is a check box

By adhering to these standards as established by Apple, you can continue to use the standard constants for button proclDs.

- pushButProc = 0
- checkBoxProc = 1
- radioButProc = 2
- useWFont = 8



## Colors:




Our button CDEFs default to black text and a black frame, a white body, and in the case of check boxes and radio buttons, a background that is the same color and the window's content color. This is just like an ordinary button. You can assign colors to individual parts of the button by creating a color table and attaching it to the control. The following code shows you how to create a color table and attach it to the button:

```
var
  hColorTable: CCTabHandle;      {Handle to button's color table}
begin
  hColorTable := CCTabHandle(NewHandleClear(SizeOf(CtlCTab)));
  hColorTable^.ctSize := 3;
  hColorTable^.ctTable[0].Value := cFrameColor;
  hColorTable^.ctTable[0].RGB := myFrameRGB;
  hColorTable^.ctTable[1].Value := cBodyColor;
  hColorTable^.ctTable[1].RGB := myBodyRGB;
  hColorTable^.ctTable[2].Value := cTextColor;
  hColorTable^.ctTable[2].RGB := myTextRGB;
  hColorTable^.ctTable[3].Value := -1;
  hColorTable^.ctTable[3].RGB := myBackgroundRGB;
  SetCtlColor(hControl, hColorTable);
```

Notice the last element of the color table does not have a standard Apple value. The value of "-1" is ignored by standard CDEFs but is used by our button CDEF to specify the button's background color. This optional element makes it easier to have a number of controls on different colored backgrounds within the same window. If you are using Tools Plus libraries, this entry in the color table corresponds to a button's "background color."

If you are using SuperCDEFs in a dialog ('DLOG' resource), the easiest way to create a color table is to create your button as a control ('CNTL' resource), and to include a 'cctb' (control color table) resource with the same number as your 'CNTL'. The 'cctb' resource can contain your entire color table. You can also use this strategy to create controls without using a dialog by creating a 'CNTL' resource for your control and including a related 'cctb' resource. The toolbox's GetNewControl is used to load the 'CNTL' resource and to create a control. If you are using Tools Plus libraries, the LoadButton routine performs similarly.

Pictures of all button CDEFs available in SuperCDEFs

<p>Description : Apple's standard button CDEF File Name : none, part of MacOS</p> <ul style="list-style-type: none"> <li>• This CDEF is shown here for comparison purposes only, and is not part of SuperCDEFs</li> <li>• Notice how the inside of radio buttons and check boxes does not look well defined when the button is on a color background.</li> </ul>	
<p>Description : Replacement for standard Apple CDEF File Name : Flat Ctrl/Flat Title</p> <ul style="list-style-type: none"> <li>• Just like Apple's buttons except the inside of the check box's square and the radio button's circle are always white</li> <li>• A subtle improvement on a tried and true idea.</li> </ul>	
<p>Description : 3D control with standard text File Name : 3D Ctrl/Flat Title</p> <ul style="list-style-type: none"> <li>• Unobstructive 3D look uses soft highlights and shadows</li> <li>• Check box is inset into the window</li> <li>• Unselected radio button is raised like a dome</li> <li>• Selected radio button is inset into the window</li> <li>• The push button looks like it's being pushed into the window when it is being clicked (see the picture at the end of this section)</li> <li>• Title's text is unaltered (no 3D effect)</li> <li>• Perfect when you need a very subtle 3D look while retaining as much of the traditional Macintosh character as possible.</li> </ul>	

Description : 3D control with 3D text  
 File Name : 3D Ctrl/3D Title

- Unobstructive 3D look uses soft highlights and shadows for both the control and the text
- Title is displayed using a subtle 3D effect and can be inset or raised
- Great for a 3D look that is suitable for contemporary applications

Raised:	<input type="radio"/> Title	<input type="checkbox"/> Title	<input type="radio"/> Title	<input type="checkbox"/> Title	
	<input checked="" type="radio"/> Title	<input checked="" type="checkbox"/> Title	<input checked="" type="radio"/> Title	<input checked="" type="checkbox"/> Title	
Inset:	<input type="radio"/> Title	<input type="checkbox"/> Title	<input type="radio"/> Title	<input type="checkbox"/> Title	
	<input checked="" type="radio"/> Title	<input checked="" type="checkbox"/> Title	<input checked="" type="radio"/> Title	<input checked="" type="checkbox"/> Title	

Description : 3D control with heavily shadowed 3D text  
 File Name : 3D Ctrl/3D+ Title

- 3D look uses soft highlights and shadows for the control
- Text is displayed using a bold 3D effect with heavy shadows and bright highlights
- Presents well on a dark background or with a dark color scheme

Raised:	<input type="radio"/> Title	<input type="checkbox"/> Title	<input type="radio"/> Title	<input type="checkbox"/> Title	
	<input checked="" type="radio"/> Title	<input checked="" type="checkbox"/> Title	<input checked="" type="radio"/> Title	<input checked="" type="checkbox"/> Title	
Inset:	<input type="radio"/> Title	<input type="checkbox"/> Title	<input type="radio"/> Title	<input type="checkbox"/> Title	
	<input checked="" type="radio"/> Title	<input checked="" type="checkbox"/> Title	<input checked="" type="radio"/> Title	<input checked="" type="checkbox"/> Title	

Description : Heavily shadowed 3D control with standard text  
 File Name : 3D+ Ctrl/Flat Title

- Relatively unobstructive 3D look uses strong highlights and shadows for the control
- Title's text is unaltered (no 3D effect)
- Well suited when you need a 3D look while retaining much of Macintosh's traditional character

	<input type="radio"/> Title	<input type="checkbox"/> Title	<input type="radio"/> Title	<input type="checkbox"/> Title	
	<input checked="" type="radio"/> Title	<input checked="" type="checkbox"/> Title	<input checked="" type="radio"/> Title	<input checked="" type="checkbox"/> Title	

Description : Heavily shadowed 3D control and text

File Name : 3D+ Ctrl/3D+ Title

- Bold 3D look for both the control and the title
- Title can be inset or raised
- Ideal when you need a bold 3D look. Better when used with a dark background or dark color scheme.



Description : Miniature replacement for standard Apple CDEF

File Name : Mini Flat Ctrl/Flat Title

- Similar to Apple's buttons except the inside of the check box's square and the radio button's circle are always white, and the box or circle is miniaturized for use with smaller fonts or where space is at a premium.



Description : Miniature 3D control with standard text

File Name : Mini 3D Ctrl/Flat Title

- Unobstructive 3D look uses soft highlights and shadows
- Check box is inset into the window
- Unselected radio button is raised like a dome
- Selected radio button is inset into the window
- The push button looks like it's being pushed into the window when it is being clicked (see the picture at the end of this section)
- Title's text is unaltered (no 3D effect)



Description : Miniature 3D control with 3D text File Name : Mini 3D Ctrl/3D Title					
<ul style="list-style-type: none"> <li>• Unobstructive 3D look uses soft highlights and shadows for both the control and the text</li> <li>• Title is displayed using a subtle 3D effect and can be inset or raised</li> </ul>					
Raised:		<input type="radio"/> Title	<input type="checkbox"/> Title	<input type="radio"/> Title	<input type="checkbox"/> Title
		<input checked="" type="radio"/> Title	<input checked="" type="checkbox"/> Title	<input checked="" type="radio"/> Title	<input checked="" type="checkbox"/> Title
Inset:		<input type="radio"/> Title	<input type="checkbox"/> Title	<input type="radio"/> Title	<input type="checkbox"/> Title
		<input checked="" type="radio"/> Title	<input checked="" type="checkbox"/> Title	<input checked="" type="radio"/> Title	<input checked="" type="checkbox"/> Title
Description : Miniature 3D control with heavily shadowed 3D text File Name : Mini 3D Ctrl/3D+ Title					
<ul style="list-style-type: none"> <li>• 3D look uses soft highlights and shadows for the control</li> <li>• Text is displayed using a bold 3D effect with heavy shadows and bright highlights</li> <li>• Presents well on a dark background or with a dark color scheme</li> </ul>					
Raised:		<input type="radio"/> Title	<input type="checkbox"/> Title	<input type="radio"/> Title	<input type="checkbox"/> Title
		<input checked="" type="radio"/> Title	<input checked="" type="checkbox"/> Title	<input checked="" type="radio"/> Title	<input checked="" type="checkbox"/> Title
Inset:		<input type="radio"/> Title	<input type="checkbox"/> Title	<input type="radio"/> Title	<input type="checkbox"/> Title
		<input checked="" type="radio"/> Title	<input checked="" type="checkbox"/> Title	<input checked="" type="radio"/> Title	<input checked="" type="checkbox"/> Title
Description : Miniature heavily shadowed 3D control with standard text File Name : Mini 3D+ Ctrl/Flat Title					
<ul style="list-style-type: none"> <li>• Relatively unobstructive 3D look uses strong highlights and shadows for the control</li> <li>• Title's text is unaltered (no 3D effect)</li> </ul>					
		<input checked="" type="radio"/> Title	<input type="checkbox"/> Title	<input type="radio"/> Title	<input type="checkbox"/> Title
		<input checked="" type="radio"/> Title	<input checked="" type="checkbox"/> Title	<input checked="" type="radio"/> Title	<input checked="" type="checkbox"/> Title

Description : Miniature heavily shadowed 3D control and text  
 File Name : Mini 3D+ Ctrl/3D+ Title

- Bold 3D look for both the control and the title
- Title can be inset or raised
- Ideal when you need a bold 3D look. Better when used with a dark background or dark color scheme.

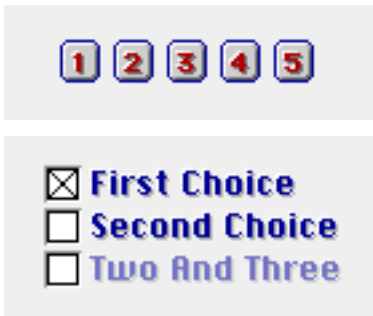
Raised:	<input type="radio"/> Title	<input type="checkbox"/> Title	<input type="radio"/> Title	<input type="checkbox"/> Title	<input type="radio"/> Title	<input type="checkbox"/> Title	<input type="radio"/> Title	<input type="checkbox"/> Title	<input type="radio"/> Title	<input type="checkbox"/> Title
	<input checked="" type="radio"/> Title	<input checked="" type="checkbox"/> Title	<input checked="" type="radio"/> Title	<input checked="" type="checkbox"/> Title	<input checked="" type="radio"/> Title	<input checked="" type="checkbox"/> Title	<input checked="" type="radio"/> Title	<input checked="" type="checkbox"/> Title	<input checked="" type="radio"/> Title	<input checked="" type="checkbox"/> Title
Inset:	<input type="radio"/> Title	<input type="checkbox"/> Title	<input type="radio"/> Title	<input type="checkbox"/> Title	<input type="radio"/> Title	<input type="checkbox"/> Title	<input type="radio"/> Title	<input type="checkbox"/> Title	<input type="radio"/> Title	<input type="checkbox"/> Title
	<input checked="" type="radio"/> Title	<input checked="" type="checkbox"/> Title	<input checked="" type="radio"/> Title	<input checked="" type="checkbox"/> Title	<input checked="" type="radio"/> Title	<input checked="" type="checkbox"/> Title	<input checked="" type="radio"/> Title	<input checked="" type="checkbox"/> Title	<input checked="" type="radio"/> Title	<input checked="" type="checkbox"/> Title

3D Push Buttons

All our 3D push buttons appear as though they are being pushed into the window when they are clicked by the user. The example at right shows a 3D push button with 3D text in its standard unselected state (top), and as it is being selected by the user (bottom). Notice that the 3D text retains its raised character even when the button is highlighted.



Exploring Your Options



You can get a variety of different looks out of a single CDEF just by selecting a different color scheme. In the top example at left, we use a strongly shadowed 3D control with strongly shadowed 3D text. The font is Geneva 9pt bold. This works well on these small buttons because we've used lighter colored text instead of the traditional black. The following colors are used:

Blue frame	RGB values: 0, 0, 38000
Dark gray body	RGB values: 52428, 52428, 52428
Red letters	RGB values: 45000, 0, 0
Background	RGB values: 61166, 61166, 61166

Strong shadows work well when placed on dark colors. The mini-button's body is a medium gray, so we risk losing much of the shadow's effects if we had used a softly shadowed control or softly shadowed text. As you can see, the buttons almost appear to jump off the window, yet they are very readable and well defined.

The check boxes' color scheme incorporates blue text and a light gray background (their RGB values are detailed earlier). With the text being placed on a light background, we want to preserve clarity and therefore elect to use softly shadowed text.

Notice how the 3D text's highlights and shadows are balanced when the button is disabled to retain the distinctive 3D look while remaining crisp and legible.

## Tab CDEFs

This section details the tab CDEFs that are available in SuperCDEFs. The following items are discussed:

- How our tab CDEFs are similar to Apple's CDEFs
- How our tab CDEFs are different from Apple's CDEFs (extra features)
- Variant codes and other settings
- Pictures of all tab CDEFs available in SuperCDEFs

How our tab CDEFs are similar to Apple's CDEFs

- Multiple line titles are created using a carriage return character
- Our tabs observe the control's color table just like Apple's CDEFs
- Can be used with or without Color QuickDraw
- Colors map perfectly across all monitor settings and multiple monitors
- Apple user interface guidelines were observed in designing and creating the control

How our tab CDEF differs from Apple's CDEFs (extra features)

- As of this writing, Apple does not have an equivalent control for tabs.
- Our tabs feature a 3D look for the control's body and/or the title. A soft 3D and bold 3D look are available.
- 3D text can be raised or inset.
- You can optionally populate an additional element of the color table for complete control over the tab's colors (detailed later).
- These tabs can be attached to 3D panels with highlights and shadows, or flat panels.
- Our tab's variant codes differ from standard Apple CDEFs
- Our CDEF resources are bigger because they run on any Macintosh and any System version.
- Our CDEFs are available as 680x0 only, PowerMac only, or Fat/Safe Binary format (a single CDEF resource that runs on both 680x0 Macs and on PowerMacs in native mode)

### Implementing Tabs:

You should implement tabs in your application as you would a set of radio buttons: one control per tab, and only one tab in a set can be selected at a time. Visually, tabs go against the norm in terms of what a selected control should look like. Controls typically darken when they are selected. A tab is normally darkened when it is unselected to indicate that it is “pushed back” and is not part of the active layout. As seen below, clicking a tab brings it forward to join the main layout whereas the previously selected tab is darkened and moved back.



Because you have access to the control’s color table, you can design your own color scheme for selected and unselected tabs. An alternative scheme is using a medium gray for the panel’s body, and unselected tabs can be lightened, or “faded” into the background.

If you are writing your application using Tools Plus libraries, use the `NewButton` routine to create a tab and to make it work automatically.

### Colors:

Our tab CDEFs default to black text and a black frame on a white background, just like an ordinary button. Unless otherwise specified, the deselected color defaults to a medium gray color. You can assign colors to individual parts of the tab by creating a color table and attaching it to the control. The following code shows you how to create a color table and attach it to the tab:

```
var
  hColorTable: CCTabHandle;      {Handle to tab's color table}
begin
  hColorTable := CCTabHandle(NewHandleClear(SizeOf(CtlCTab)));
  hColorTable^.ctSize := 3;
  hColorTable^.ctTable[0].Value := cFrameColor;
  hColorTable^.ctTable[0].RGB := myFrameRGB;
  hColorTable^.ctTable[1].Value := cBodyColor;
  hColorTable^.ctTable[1].RGB := myBodyRGB;
  hColorTable^.ctTable[2].Value := cTextColor;
  hColorTable^.ctTable[2].RGB := myTextRGB;
  hColorTable^.ctTable[3].Value := -1;
  hColorTable^.ctTable[3].RGB := myDeselectedRGB;
  SetCtlColor(hControl, hColorTable);
```

Notice the last element of the color table does not have a standard Apple value. The value of “-1” is ignored by standard CDEFs but is used by our tab CDEF to specify the tab’s body color when it is not selected. This is an optional element in the color table and does not have to be included. If you are using Tools Plus libraries, this entry in the color table corresponds to a button’s “background color.”

If you are using SuperCDEFs in a dialog (‘DLOG’ resource), the easiest way to create a color table is to create your tab as a control (‘CNTL’ resource), and to include a ‘cctb’ (control color table) resource with the same number as your ‘CNTL’. The ‘cctb’ resource can contain your entire color table. You can also use this strategy to create controls without using a dialog by creating a ‘CNTL’ resource for your control and including a related ‘cctb’ resource. The toolbox’s `GetNewControl` is used to load the ‘CNTL’ resource and to create a control. If you are using Tools Plus libraries, the `LoadButton` routine performs similarly.



**Note :** Your tabs' appearance can vary greatly depending on the colors you select. In this manual, we have chosen a subtle color scheme. There are samples at the end of this chapter that show off our tabs using much more pronounced colors that may be more to your liking.

#### Control Values:

Like ordinary buttons, the control's value (contrlValue field in the ControlRecord) indicates if the tab is selected or not. A value of one (1) indicates the tab is selected, and zero (0) indicates the tab is unselected.

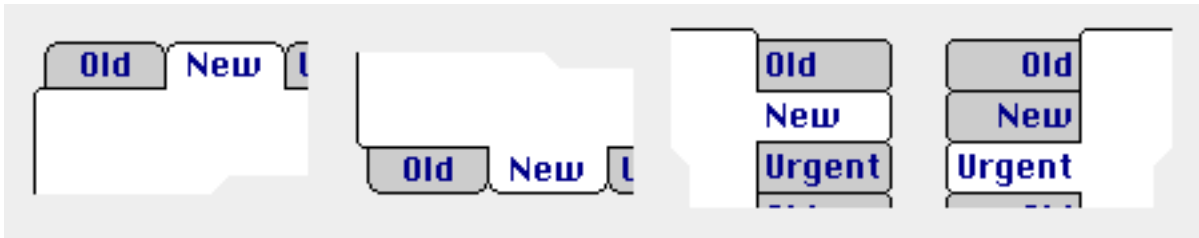
Always set the control's minimum limit (contrlMin field in the ControlRecord) to zero (0).

Set the control's maximum limit (contrlMax field in the ControlRecord) to 1 to produce round corners that are a standard size. You can manually change the size of the tab's round corners by setting the control record's maximum limit to a value that is greater than 1, although this is usually unnecessary because our tabs are designed to look attractive with default settings.

#### Variant Codes:

The variant codes used by our tabs are as follows:

- bit 3 (+8) Use window's font (same as Apple's useWFont)
- bit 2 (+4) Inset text, otherwise text is raised. This bit is ignored if the CDEF does not display 3D text.
- bits 0-1 Tab's orientation indicating which side is "attached" to the rest of the layout:
  - (+0) Tab is attached by its bottom
  - (+1) Tab is attached by its left side
  - (+2) Tab is attached by its top
  - (+3) Tab is attached by its right side



Attached at bottom

Attached at top

Attached at left

Attached at right

It's a good idea to establish a standard set of constants for tab orientation and to use them throughout all your applications. The following constants are recommended:

```

tabProc          = 32000
rightTabProc     = 32001
bottomTabProc    = 32002
leftTabProc      = 32003
inset3DTabTitle = 4
  
```

The above constants assume that your tab CDEF's resource ID is 2000 (2000 x 16 = 32000 plus your variant code for orientation).

TrackControl

Many developers do not implement the toolbox’s TrackControl routine correctly, but these Tab CDEFs account for those inconsistencies and are very forgiving. If any control has an action proc defined, as is indicated when a control’s contrlAction field has a non-nil (not zero) value, then your application should call TrackControl with its actionProc parameter set to -1. If this is the case, the tab is instantly selected when a mouse-down occurs in the control.

If you pass a value of nil in TrackControl’s actionProc parameter, as many programmers do, these tab CDEFs will highlight much like a radio button does when TrackControl is called. The tab will work perfectly but it will not respond instantly when a mouse-down occurs in the control. This is something you may decide to do on purpose.

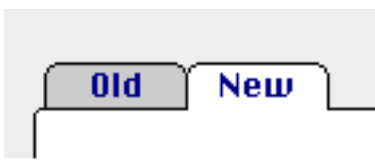
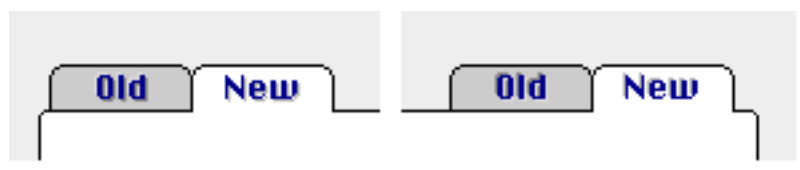
If you are writing your application using Tools Plus, you don’t need to do anything to make the CDEF work. Tools Plus responds by telling your application that a tab was selected as soon as the user’s mouse goes down in the control.

Pictures of the Tab CDEFs available in SuperCDEFs

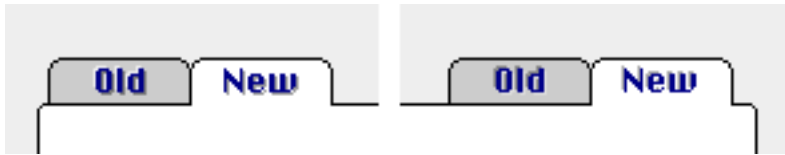
Each of the tab CDEFs included in SuperCDEFs uses the following options in a specific combination:

- Control’s Body: (a) Flat
  - (b) Soft 3D shadows and highlights
  - (c) Bold 3D shadows and highlights
- Control’s Title: (a) Flat
  - (b) Soft 3D shadows and highlights\*
  - (c) Bold 3D shadows and highlights\*

\* a variant code can make the text inset or raised in each case.

Description : Flat control with flat text File Name : Flat Tab/Flat Title	
Description : Flat control with 3D text File Name : Flat Tab/3D Title	 <div style="display: flex; justify-content: space-around; margin-top: 5px;"> <span>Raised title</span> <span>Inset title</span> </div>

Description : Flat control with heavily shadowed 3D text  
File Name : Flat Tab/3D+ Title



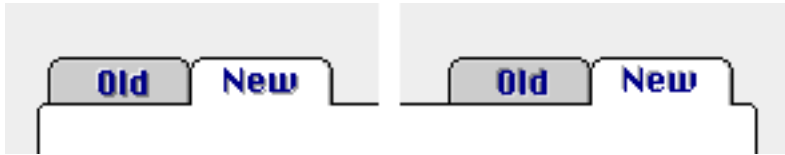
Raised title

Inset title

Description : 3D control with flat text  
File Name : 3D Tab/Flat Title



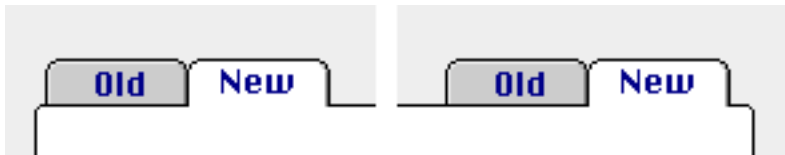
Description : 3D control with heavily shadowed 3D text  
File Name : 3D Tab/3D+ Title



Raised title

Inset title

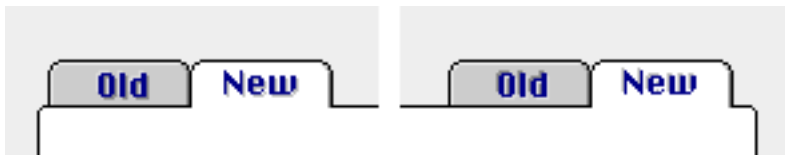
Description : Heavily shadowed 3D control with flat text  
File Name : 3D+ Tab/Flat Title



Raised title

Inset title

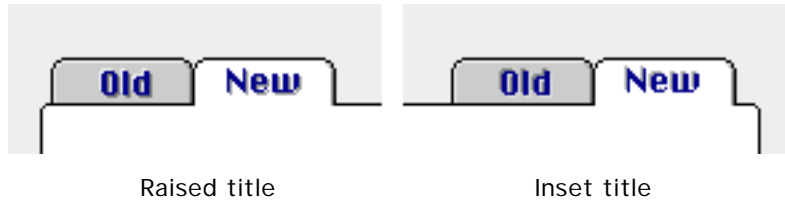
Description : Heavily shadowed 3D control with 3D text  
File Name : 3D+ Tab/3D Title



Raised title

Inset title

Description : Heavily shadowed 3D control with heavily shadowed 3D text  
 File Name : 3D+ Tab/3D+ Title



#### Programming Tips:

- (1) You can create an attractive round-corner layout as seen in the above examples by using an oval size of 7 for the rectangle that the tabs are attached to. Inset the tabs 4 pixels in from the corner of the rectangle.
- (2) The tab overlaps with the main layout's rectangle by 2 pixels on the side where it is attached. For example, if you have tabs along the top of a rectangle and the rectangle's top co-ordinate is 50, the bottom co-ordinate for your tabs should be 52.
- (3) Tabs placed against each other (i.e., several tabs across the top of a layout) overlap by 1 pixel. For example, if the first tab's right co-ordinate is 100, then the tab to the right of it will have a left co-ordinate of 99.
- (4) A good height for top and bottom tabs is 19. For right and left tabs it's 20. These heights assume that you are using Chicago 12pt for your tab's font.
- (5) Use heavily shadowed 3D text only on darker layouts. It may look too overpowering on white or light colors.
- (6) If you are using Tools Plus libraries, you can create and hide all the GUI elements (buttons, fields, list boxes, etc.) associated with all the tabs, then show only those elements that pertain to the selected tab.
- (7) You can get a wide variety of appearances by using different tab CDEFs in combination with different colors. The example at right uses a heavily shadowed 3D control with softly shadowed 3D text. The main page and selected tab use a medium gray (RGB values all set to 52428). The unselected color was automatically darkened by the CDEF because it was also set to the same medium gray. The title and frame are both black and the window's backdrop color is a light gray (RGB values all set to 61166). As you can see, the look is clean, it retains much of Macintosh's unique character, yet it has an exciting "3D look."
- (8) Tools Plus libraries include, among other things, versatile and powerful panels. Here are just some of the things that Tools Plus panels can do:
  - flat or 3D panel (a group box is just a specialized panel)
  - inset or raised 3D effect lets you specify how deep the panel is
  - radio buttons in a group can optionally be deselected when a radio button is selected
  - flat or 3D title (3D title can be raised or inset using soft or bold shadows)
 Tools Plus and SuperCDEFs make perfect partners to create a professional "3D look" interface.



- (9) Advanced programmers: If you are using a 3D tab and it is attached to a 3D panel, you can achieve an enhanced connection between your tab and your panel that preserves the panel's highlight (or shadow) where the two elements connect.

The following example shows you how 3D tabs look when they are connected to a 3D panel. Pay particular attention to the line where the tab connects to the panel. The example on the left was created using tabs with regular color tables. The example on the right was created using tabs with an additional element in their color tables. The additional element is populated with the panel's highlight color or shadow color.



To achieve this subtle difference, allocate a color table to your tab CDEF. Populate the additional color table element with a Value of -2 and an RGB color that is the same as the highlight color or shadow color used by your 3D panel. Use the color found on the side that is attached to the tab, that is the panel's highlight color for tabs that are attached to the top or left side of the panel, and the panel's shadow color for tabs that are attached to the panel's right side or bottom. This little bit of extra effort produces a 3D effect that makes the tab look like it moves back behind the current page when it is unselected, and forward to the current page when it is selected.

## Slider CDEFs

This section details the slider CDEFs that are available in SuperCDEFs. The following items are discussed:

- How our slider CDEFs are similar to Apple's CDEFs
- How our slider CDEFs differ from Apple's CDEFs (extra features)
- Variant codes and other settings
- Slider styles and their options
- Pictures of all slider CDEFs available in SuperCDEFs

How our slider CDEFs are similar to Apple's CDEFs

- Our sliders observe the control's color table just like Apple's CDEFs
- Can be used on color or black and white window records
- Can be used with or without Color QuickDraw
- Colors map perfectly across all monitor settings and multiple monitors
- Apple user interface guidelines were observed in designing and creating the control

How our slider CDEFs differ from Apple's CDEFs (extra features)

- As of this writing, Apple does not have an equivalent control for sliders.
- Our sliders feature an optional 3D look for the control's body and the numbers used for the scale. A soft 3D and bold 3D look are available.
- A variety of styles are available and some sliders feature a variable width indicator (thumb) for even greater diversity.
- Optional tick marks can be displayed on one or both sides of the slider.
- An optional scale can be numbered using the font/size/style of your choice.
- Optional reverse scaling.
- "Smart Scaling" ensures that tick marks and numbers always look their best within the available space.
- Selectable "snap indicator to cursor" or "line up/line down" behavior.
- Unlike a scroll bar, a slider's indicator moves in real time as the user drags it. In contrast, when the user drags the indicator in an Apple scroll bar, a gray outline moves with the user's cursor until the mouse button is released, then the indicator jumps to the new position.
- You can optionally populate an additional element of the color table for complete control over the slider's colors (detailed later).
- The slider's variant codes differ from standard Apple buttons and scroll bars.
- Our sliders are highly optimized for performance and memory efficiency making use of temporary memory outside of your application's heap whenever possible.
- Our CDEF resources are bigger because they run on any Macintosh and any System version. They also provide much more functionality.
- Our CDEFs are available as 680x0 only, PowerMac only, or Fat/Safe Binary format (a single CDEF resource that runs on both 680x0 Macs and on PowerMacs in native mode).

## Fonts

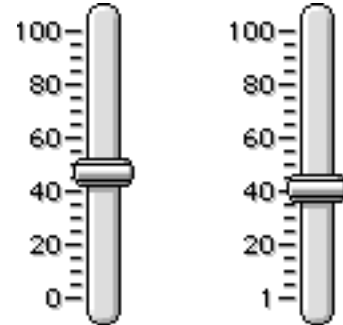
If you display the optional numeric scale for a slider, the numbers are displayed using the window's current font, font size and style as set by the toolbox's TextFont, TextSize and TextFace routines.

If your slider displays a scale, your application must set the window's font, font size and style as required before doing anything with the slider. This includes creating the slider, updating it, setting the minimum or maximum limit or value, showing it after being hidden, or tracking the control.

Tools Plus libraries do all this for you automatically.

## Tick marks and scale

Our sliders can optionally display tick marks along one or both sides of the slider. These tick marks are drawn adjacent to the slider to correspond to each different value the slider can have. If the tick marks get too close together to be aesthetically pleasing, "smart scaling" draws tick marks in intervals (i.e., count by 2's, 5's, 10's, 25's etc.) When this is the case, your slider's indicator may rest between two tick marks. To avoid scale compression, ensure that your slider is long enough to accommodate the required number of ticks along its path.



Sliders can also have numeric values displayed along with the tick marks on the scale. "Smart scaling" ensures that numbers are spaced in a manner that is both logical and attractive. The example above shows two sliders whose scales have been compressed with "smart scaling."

Notice the following automatic features that made this slider look absolutely professional:

- tick marks are incremented by 5's
- numbers are incremented by 20's
- tick marks beside numbers are extended slightly to indicate a major division
- slider's scale can start at 0 or 1 and still create a perfect scale

"Smart scaling" is an exclusive feature found only in SuperCDEFs.

## Implementing Sliders

You should implement sliders in your application as you would a scroll bar. Slider CDEFs default to black text and a black frame on a white background, just like an ordinary button. Unless you specify otherwise, the 3D sliders adopt a color scheme of light grays to give them their 3D effects.

If you are writing your application using Tools Plus libraries, use the NewScrollBar routine to create a slider and to make it work automatically.

**Colors:**

You can assign colors to individual parts of the slider by creating a color table and attaching it to the control. The following code shows you how to create a color table and attach it to the slider:

```
var
    hColorTable: CCTabHandle;      {Handle to slider's color table}
begin
    hColorTable := CCTabHandle(NewHandleClear(SizeOf(CtlCTab)));
    hColorTable^.ctSize := 4;
    hColorTable^.ctTable[0].Value := cFrameColor;
    hColorTable^.ctTable[0].RGB := myFrameRGB;
    hColorTable^.ctTable[1].Value := cBodyColor;
    hColorTable^.ctTable[1].RGB := myBodyRGB;
    hColorTable^.ctTable[2].Value := cTextColor;
    hColorTable^.ctTable[2].RGB := myTextRGB;
    hColorTable^.ctTable[3].Value := cThumbColor;
    hColorTable^.ctTable[3].RGB := myThumbRGB;
    hColorTable^.ctTable[4].Value := -1;
    hColorTable^.ctTable[4].RGB := myBackgroundRGB;
    SetCtlColor(hControl, hColorTable);
```

Notice the last element of the color table does not have a standard Apple value. The value of "-1" is ignored by standard CDEFs but is used by our slider CDEF to specify the slider's background color. If you don't specify a background color, the slider uses the window's background color. This optional element makes it easier to have a number of controls on different colored backgrounds within the same window. If you are using Tools Plus libraries, this entry in the color table corresponds to a scroll bar's (or slider's) "background color."

If you are using SuperCDEFs in a dialog ('DLOG' resource), the easiest way to create a color table is to create your slider as a control ('CNTL' resource), and to include a 'cctb' (control color table) resource with the same number as your 'CNTL'. The 'cctb' resource can contain your entire color table. You can also use this strategy to create controls without using a dialog by creating a 'CNTL' resource for your control and including a related 'cctb' resource. The toolbox's GetNewControl is used to load the 'CNTL' resource and to create a control. If you are using Tools Plus libraries, the LoadScrollBar routine performs similarly.

**Control Values:**

Like ordinary scroll bars, the control's `contriMin`, `contriMax` and `contriValue` fields indicate the sliders minimum limit, maximum limit and current value. If the slider's current value is set to a value that is less than the minimum limit or greater than the maximum limit, or if the maximum limit is not greater than the minimum limit, then the slider's thumb is not drawn.



Variant Codes:

The variant codes used by sliders are as follows:

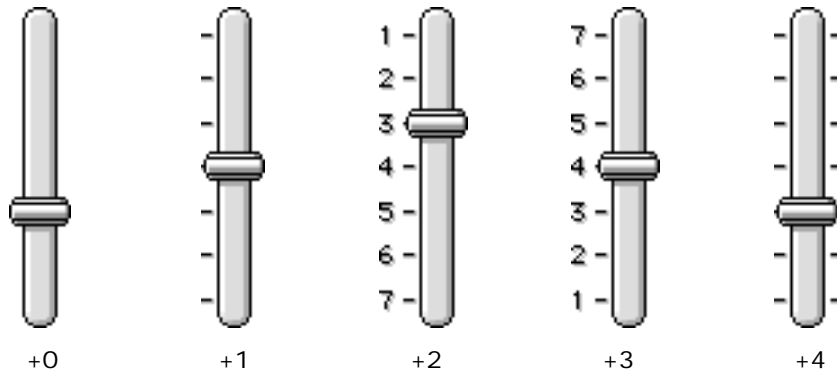
- bit 3 (+8) When the user clicks on either side of the indicator (on the slider’s rail but not on the indicator), an event is generated indicating that the slider’s “line up” or “line down” region was clicked. This is required if you want to let the user increment or decrement the slider without dragging the indicator.

If you do not include this option, the indicator behaves like Apple’s slider in the Sound control panel: when the user clicks on the slider’s rail, the indicator instantly snaps to that position and tracks the mouse until it is released.

NOTE : If you are using this slider in an application that supports “live scrolling” without the use of an action proc (like the live scrolling option in Tools Plus), make sure you include this variant when live scrolling is enabled for this slider.

- bits 0-2 Tick and scale options:
  - (+0) No tick marks, no numeric scale displayed
  - (+1) Tick marks displayed on the left side of vertical sliders and below a horizontal slider
  - (+2) Tick marks with values
  - (+3) Tick marks with values in reverse order
  - (+4) Tick marks on both sides of slider

The samples below show you the effect of each of the five variants:



It’s a good idea to establish a standard set of constants for slider variants and to use them throughout all your applications. The following constants are recommended:

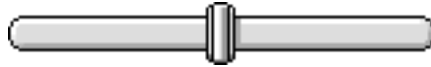
- kNoTickSlider = 32000
- kTickSlider = 32001
- kTickValueSlider = 32002
- kTickReverseValueSlider = 32003
- kDualTickSlider = 32004
- kLineIncrSlider = 8

The above constants assume that your slider CDEF’s resource ID is 2000 (2000 x 16 = 32000 plus your variant code for orientation).

Orientation

Like scroll bars, all sliders can be vertically or horizontally oriented. Their minimum end is always at the top or left, and their maximum is always at the right or bottom. When you create a slider, remember the following:

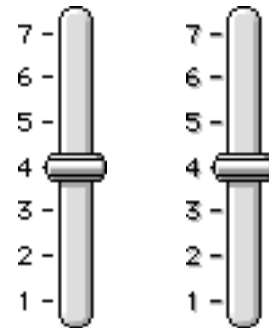
- some sliders' indicators adjust to the sliders' width
- some sliders (like the one seen here) have an indicator whose width is fixed
- the slider's rectangle must be large enough to accommodate tick marks and scale if these items are specified in the control's proclD.



3D text and tick marks

All styles of sliders (except the pure black and white one) offer two varieties of text and tick marks: flat (sample on left) and 3D (right). They are available as two different CDEFs. When the 3D CDEF is used, the numbers in the scale and the scale's tick marks are drawn with shadows and highlights to make them look raised from the window.

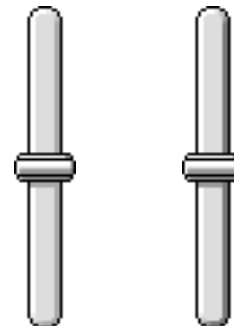
Only a subtle 3D effect is available for text and tick marks, as these items are finely drawn and a more pronounced 3D effect would make them lose their definition and become less legible.



Soft or bold shadows

Some styles of sliders offer two varieties of shadows and highlights: soft (sample on left) and bold (right). They are available as two different CDEFs. These styling variations are applied to the slider's rail and indicator, but not to the tick marks and numbers in the scale.

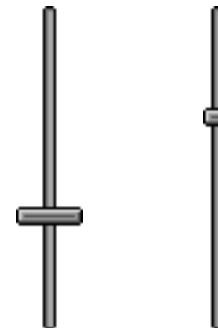
Sliders with bold shadows are best used on dark backgrounds or color schemes. The soft shadow sliders are more typical of Macintosh's subdued and slightly understated nature.



Variable width indicator

Some sliders vary the width of their indicator to fit the control's rectangle if a scale is not drawn. This gives you a large degree of versatility by letting you use one CDEF to create controls that look different.

The two sliders on the right both use the same proclD. The only difference is that the control rectangle for the slider on the left is wider than the one on the right.

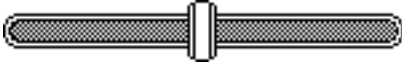
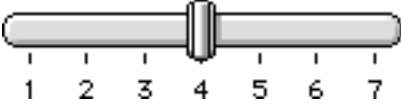

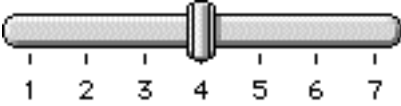
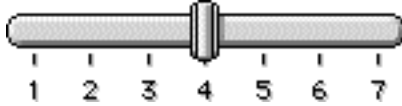
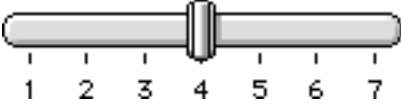

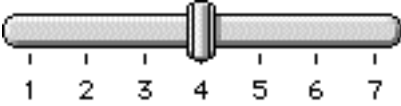
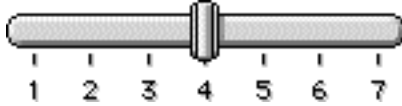
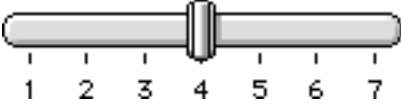

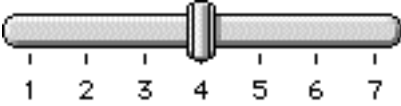
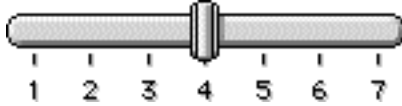
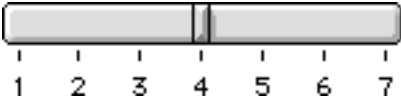
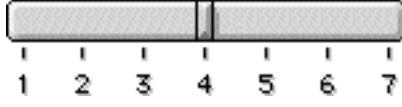


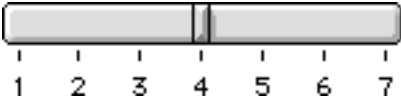
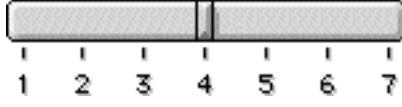


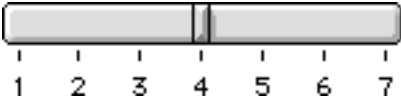
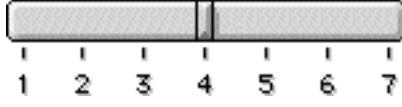




Pictures of the Slider CDEFs available in SuperCDEFs

Each of the sliders included in SuperCDEFs are shown below. Please note that unless otherwise indicated, each slider is available with the following options (each available as a separate CDEF):

- Soft shadows, flat text
- Soft shadows, 3D text
- Strong shadows, flat text
- Strong shadows, 3D text

When combined with the variant codes, each slider style can produce 20 different “looks.”

<p>Description : Plain Slider File Name : Plain Slider</p> <ul style="list-style-type: none"> <li>• No 3D affects are available</li> <li>• Provides good performance even on very low-end Macs like the Macintosh Plus.</li> </ul> 					
<p>Description : 3D Slider</p> <table border="0"> <tr> <td>  <p>File Name : 3D Slider/Flat Title 3D body, flat ticks and text</p> </td> <td>  <p>File Name : 3D Slider/3D Title 3D body, 3D ticks and text</p> </td> </tr> <tr> <td>  <p>File Name : 3D+ Slider/Flat Title as above, with bold shadows</p> </td> <td>  <p>File Name : 3D+ Slider/3D Title as above, with bold shadows</p> </td> </tr> </table>		 <p>File Name : 3D Slider/Flat Title 3D body, flat ticks and text</p>	 <p>File Name : 3D Slider/3D Title 3D body, 3D ticks and text</p>	 <p>File Name : 3D+ Slider/Flat Title as above, with bold shadows</p>	 <p>File Name : 3D+ Slider/3D Title as above, with bold shadows</p>
 <p>File Name : 3D Slider/Flat Title 3D body, flat ticks and text</p>	 <p>File Name : 3D Slider/3D Title 3D body, 3D ticks and text</p>				
 <p>File Name : 3D+ Slider/Flat Title as above, with bold shadows</p>	 <p>File Name : 3D+ Slider/3D Title as above, with bold shadows</p>				
<p>Description : 3D Block Slider</p> <table border="0"> <tr> <td>  <p>File Name : 3D BlockSlider/Flat Title 3D body, flat ticks and text</p> </td> <td>  <p>File Name : 3D BlockSlider/3D Title 3D body, 3D ticks and text</p> </td> </tr> <tr> <td>  <p>File Name : 3D+ BlockSlider/Flat Title as above, with bold shadows</p> </td> <td>  <p>File Name : 3D+ BlockSlider/3D Title as above, with bold shadows</p> </td> </tr> </table>		 <p>File Name : 3D BlockSlider/Flat Title 3D body, flat ticks and text</p>	 <p>File Name : 3D BlockSlider/3D Title 3D body, 3D ticks and text</p>	 <p>File Name : 3D+ BlockSlider/Flat Title as above, with bold shadows</p>	 <p>File Name : 3D+ BlockSlider/3D Title as above, with bold shadows</p>
 <p>File Name : 3D BlockSlider/Flat Title 3D body, flat ticks and text</p>	 <p>File Name : 3D BlockSlider/3D Title 3D body, 3D ticks and text</p>				
 <p>File Name : 3D+ BlockSlider/Flat Title as above, with bold shadows</p>	 <p>File Name : 3D+ BlockSlider/3D Title as above, with bold shadows</p>				

Description : 3D Mixer Slider

- This kind of slider resembles a fader on an audio mixer. It looks especially good when several sliders are oriented vertically and placed beside each other.



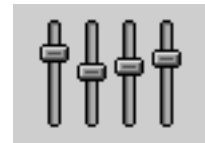
File Name : 3D MixerSlider/Flat Title  
3D body, flat ticks and text



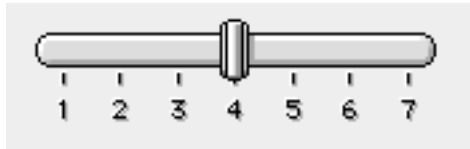
File Name : 3D MixerSlider/3D Title  
3D body, 3D ticks and text

Exploring

Feel free to explore the possibilities with sliders. For example, if you are creating a 4-channel synthesizer, you can use these sliders as volume controls for each channel. With SuperCDEFs, you can create a mixer that looks like one.



Our sliders also have pleasant subtleties you'll discover as you work with them. For example, the effect we apply for 3D text has a highlight as well as a shadow that you can see when the slider is on a colored background. Explore with backgrounds and colors to come up with a combination that's perfect for your application.



## Thermometer CDEF

This section details the thermometer CDEF that is available in SuperCDEFs. The following items are discussed:

- How our thermometer CDEF is similar to Apple's thermometer
- How our thermometer CDEF differs from Apple's (extra features)
- Variant codes
- Pictures of the thermometer CDEF

How our thermometer CDEF is similar to Apple's thermometer

- Looks identical to the Finder's thermometer
- "Standard progress" thermometer as seen when copying files
- "Busy" thermometer (animated barber pole) as seen when searching a large or slow volume in System 7's "Find..." command
- Same use of colors and patterns
- Apple user interface guidelines were observed in designing and creating the control

How our thermometer CDEF differs from Apple's (extra features)

- As of this writing, Apple does not give you access to their progress thermometer.
- Our thermometer CDEF cannot be selected by the user (clicked by the mouse), highlighted or disabled.
- Our CDEF resources are bigger because they run on any Macintosh and any System version.
- Our CDEFs are available as 680x0 only, PowerMac only, or Fat/Safe Binary format (a single CDEF resource that runs on both 680x0 Macs and on PowerMacs in native mode)

Variant Codes

This thermometer CDEF has no variant codes. The low 4 bits of the procID are ignored.

Implementing Thermometers

The thermometer CDEF works similarly to a scroll bar except that it doesn't have a thumb and it does not respond to mouse clicks. The thermometer's progress is normally displayed left to right for horizontal orientation, or bottom up for vertical orientation. If you want progress to be displayed in the opposite direction, assign a negative value to the control's minimum or maximum limit. So instead of running from 0 to 100, you could run from 0 to -100, or from -1 to 99 to have the progress go in the opposite direction.

The thermometer's range is determined by the control's minimum limit (contrlMin field in the ControlRecord) and the control's maximum limit (contrlMax field in the ControlRecord). The control's value (contrlValue field in the ControlRecord) indicates the degree of progress through its range. Once you have created the thermometer control, you can use SetCtlValue to change the control's value and thereby progress through its range.

Pictures of the Thermometer CDEF available in SuperCDEFs

The example below shows you how the thermometer CDEF looks when running under System 7. It takes on the appearance of a standard System 6 (or System 5) thermometer when running under those systems. If you are using Aaron, a system extension that runs under System 7 to give you the desk top icons, windows and controls as seen in System 8 (also known as "Copland"), set the display rectangle to be either 11 or 12 pixels high to make Aaron draw the thermometer using a System 8 style.



The thermometer CDEF can also display a "busy thermometer" indicating that the application is doing something but it does not have enough information to determine a percentage complete. You can see this in System 7's Finder by selecting "Find...", choosing "more options" and selecting a large or slow volume with the "all at once" option on. In this example, the thermometer is busy (no progress shown) while the Finder scans a CD-ROM for file names containing the word "pictures." After the Finder has found 80 files, it shows its progress using a standard thermometer while those items are displayed.

To display a "busy" thermometer, set the control's minimum limit to -32768. When you are finished "being busy" and you have determined the amount of work that needs to be done, set the control's minimum limit and value to zero (0) and the maximum limit to the value you have determined (the amount of work to be done.)

You can make the "busy thermometer" scroll like a barber pole by changing the control's value. Only a range of 0 through 15 is required to scroll the thermometer's pattern, but it also works perfectly with any value greater than zero as it progresses upward to the control's maximum limit.



## Hierarchy Triangle CDEF

This section details the hierarchy triangle CDEF that is available in SuperCDEFs. The following items are discussed:

- How our hierarchy triangle CDEF is similar to Apple's hierarchy triangle
- How our hierarchy triangle CDEF differs from Apple's (extra features)
- Variant codes
- Pictures of the hierarchy triangle CDEF

How our hierarchy triangle CDEF is similar to Apple's hierarchy triangle

- Looks identical to the Finder's hierarchy triangle
- Same use of colors and patterns
- Apple user interface guidelines were observed in designing and creating the control

How our hierarchy triangle CDEF differs from Apple's (extra features)

- As of this writing, Apple does not give you access to their hierarchy triangle.
- Our CDEF resources are bigger because they run on any Macintosh and any System version.
- Our CDEFs are available as 680x0 only, PowerMac only, or Fat/Safe Binary format (a single CDEF resource that runs on both 680x0 Macs and on PowerMacs in native mode)

Variant Codes

This hierarchy triangle CDEF has no variant codes. The low 4 bits of the procID are ignored.

Implementing Hierarchy Triangles

The hierarchy triangle CDEF works similarly to a check box, in that it has a deselected state (arrow points right) and a selected state (arrow points down). The control's value (contrlValue field in the ControlRecord) indicates if the triangle is selected. A zero value indicates an unselected state, and a non-zero value indicates it is selected. The hierarchy triangle fits perfectly into an 11 x 11 pixel square, but it will center vertically and horizontally in a control with larger co-ordinates.

If you are using Tools Plus libraries, then the button-related routines you normally use for check boxes can be used to create and maintain hierarchy triangles.

**Colors:**

The hierarchy triangle CDEF uses a control color table to display the control with non-default colors. By default, it draws just like the Finder's triangles with a black outline and filled with a "gray" pattern that is made up of alternating violet and white pixels. The hierarchy triangle is drawn on a background that is the same color as the window's "content" color table entry, which defaults to white.

You can assign colors to individual parts of the hierarchy triangle by creating a color table and attaching it to the control. The following code shows you how to create a color table and attach it to the hierarchy triangle :

```
var
  hColorTable: CColorTable;      {Handle to triangle's color table}
begin
  hColorTable := CColorTable(NewHandleClear(SizeOf(CColorTable)));
  hColorTable^.ctSize := 3;
  hColorTable^.ctTable[0].Value := cFrameColor;
  hColorTable^.ctTable[0].RGB := myFrameRGB;
  hColorTable^.ctTable[1].Value := cBodyColor;
  hColorTable^.ctTable[1].RGB := myBodyRGB;
  hColorTable^.ctTable[2].Value := -1;
  hColorTable^.ctTable[2].RGB := myBackgroundRGB;
  hColorTable^.ctTable[3].Value := -2;
  hColorTable^.ctTable[3].RGB := myPatternBackRGB;
  SetCtlColor(hControl, hColorTable);
```

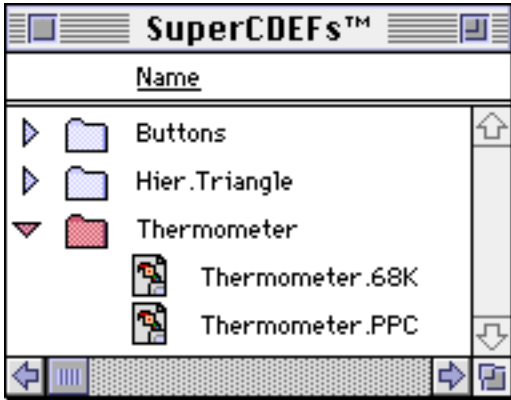
The hierarchy triangle ignores the text color and thumb color entries if they are present in the color table. Notice the last two elements of the color table do not have standard Apple values. The values "-1" and "-2" are ignored by standard CDEFs. The "-1" entry specifies the control's background color. This optional element makes it easier to have a number of controls on different colored backgrounds within the same window. If you are using Tools Plus libraries, this entry in the color table corresponds to a button's "background color." The "-2" entry specifies a color that is used instead of white inside the control's body. This too is optional.

If you are using SuperCDEFs in a dialog ('DLOG' resource), the easiest way to create a color table is to create your hierarchy triangle as a control ('CNTL' resource), and to include a 'cctb' (control color table) resource with the same number as your 'CNTL'. The 'cctb' resource can contain your entire color table. You can also use this strategy to create controls without using a dialog by creating a 'CNTL' resource for your control and including a related 'cctb' resource. The toolbox's GetNewControl is used to load the 'CNTL' resource and to create a control. If you are using Tools Plus libraries, the LoadButton routine performs similarly.



Pictures of the Hierarchy Triangle CDEF available in SuperCDEFs

The example at right shows you how the hierarchy triangle CDEF looks in the unselected state (top) and selected state (bottom). The leftmost triangle is using the default colors, whereas the others are using the body color from a related color table.



The most common place you'll see hierarchy triangles is in the Finder, as seen at left. They are the standard Macintosh user interface element used to indicate that a hierarchy is present (a folder inside a folder), and to expand and collapse the hierarchy. In the example at left, the "Thermometer" folder is expanded to display its contents hierarchically.