

# Designing User Interfaces (“Skins”) for the QCD Player



Designing User Interfaces (“Skins”) for QCD Player.....	2
Overview .....	2
<i>Contents of a Skin</i> .....	3
<i>A Note About Buttons: The Four Button States</i> .....	4
The Bitmap Files.....	5
<i>Map.bmp – The Master Map</i> .....	6
Color codes .....	6
<i>Body.bmp – The Main Body and Extended Areas</i> .....	8
<i>ButtonSet.bmp – The Track Control Buttons</i> .....	8
<i>WindowSet.bmp – Basic Windows Controls</i> .....	9
<i>TrackSet.bmp – The TrackBox Controls</i> .....	10
<i>TimerSet.bmp – The Timer Display</i> .....	13
<i>SliderSet.bmp – Sliders, Progress Bars, and Knobs</i> .....	14
“Tracking” Sliders .....	15
“Progressive” Sliders (“Thermometers”) .....	15
“Rotational” Sliders (“Knobs”) .....	16
<i>EqSet.bmp – Equalizer Controls</i> .....	17
<i>MenuSet.bmp – Menu Title Bars</i> .....	18
<i>BorderSet.bmp – Window Borders</i> .....	19
<i>CharSet.bmp and CharSet.ini – Character Sets</i> .....	20
Text Controls.....	20
CharSet.bmp .....	20
CharSet.ini.....	21
Testing a Skin .....	23
Wrapping up skins .....	24
SkinFamily.ini .....	25
SkinKid.ini.....	26

# Designing User Interfaces (“Skins”) for QCD Player

## Overview

The QCD Player is a fully functional digital audio player application for Windows 95 | 98 | ME, Windows NT 4.x, Windows 2000, and Windows XP. Designed as a lightweight Win32 stand-alone application, it obtains and displays information from the Gracenote CDDB Music Recognition Service if the user is connected (or chooses to connect) to the Internet.

The QCD Player provides a unique skinning implementation that allows a designer to replace the entire graphical user interface, without regard to shape, size, or position of the original QCD user interface.

This feature lets someone with graphics skills create a custom interface for the QCD Player, called a *skin*, without any programming required and with only minimal Quality Assurance before being released to the public. Creating a skin does require a graphic artist skilled in the use of a paint application (such as Adobe Photoshop) to create bitmapped graphics. QCD lets you design a user interface with arbitrary shape and layout, so you do not have to limit your player design to rectangular shapes, or to specific layout or shapes of buttons, slider, displays, and other controls. This gives you tremendous freedom for your player design.

QCD does not require that you include all of its features, buttons, and controls into the design of a skin. In fact, including all features can lead to a player interface that’s difficult to use. However, certain functions (such as Play, Stop, Next Track, etc.) are so basic, a user would consider the player broken if it did not include them. You can also hide some controls in several *extended* areas of the body that only appear if the user clicks the Extend button in the main body of the Player—a common technique used by interface designers to hide more ‘advanced’ functions until the user is ready to deal with them.

## Contents of a Skin

A skin consists of several files. Except for the *CharSet.ini* file, all these are *.bmp* (Windows bitmap) files.

A skin is comprised of the following files.

- *Map.bmp* is the Master Map that defines which controls appear in the player, and where they appear. It also defines the shape of the player and of each of the controls. It also defines what part of the player body is permanently displayed, and which is the *Extended Body*, which is only displayed if the user clicks the Extend button. QCD uses a unique color-coding system for this map, which is what allows it to place the controls of arbitrary shape in an arbitrary layout, including omitting some controls altogether.
- *Body.bmp* defines the overall look of the skin. All other controls appear on top of this body.
- A set of seven *.bmp* files defines the appearance of the various displays, buttons, and sliders. These seven files are *ButtonSet.bmp* (the standard playback buttons, Play, Stop, Eject, etc.); *WindowSet.bmp* (various window controls, such as minimize, close, menu display, and other general widgets); *TrackSet.bmp* (the “TrackBox”, which is an array of buttons corresponding to the individual tracks); *TimerSet.bmp* (the playback timer); the *SliderSet.bmp* (the sliders and knobs such as volume knob or playback progress); the *SkinCtrlSet.bmp* (the various buttons to switch between skin modes); and the *EqSet.bmp* (the sliders and buttons that implement the equalizer). Four of these files (*SliderSet*, *TrackSet*, *TimerSet*, and *EqSet*) use a color coded “Internal Map” that is similar in concept to the *Master Map.bmp* file—in other words it uses color coding to let you construct larger controls from smaller components. The details are defined in the sections for each of these files.
- *MenuSet.bmp* contains the banners that appear on the main menu and the ‘QuickTrack’ menu.
- *BorderSet.bmp* defines the appearance of a resizable border for some of the windows within the QCD Player.
- A *CharSet.ini* and/or *CharSet.bmp* file that define the appearance (typeface and size) and color of text in the player, such as the disc title, artist, and track titles.

If you put these files together in a folder, the QCD Player can load them and display the resulting user interface. This is probably how you will view the skin as you design and test it. However, for purposes of delivering the skin, you should collect the files into a compressed ‘zip’ archive. The QCD Player can unzip the archive and read the files on the fly.

## A Note About Buttons: The Four Button States

Before describing all the bitmap files, it is useful to pause and agree on some general terminology and principles about “button” type controls. While other types of controls, such as sliders and displays, are described in the sections on specific bitmap files, the way buttons behave affects several of these bitmap files.

Buttons behave in one of two ways: A normal button executes a command when the user clicks on it, and then returns to its default appearance. A “toggle” type button changes the behavior of the player (such as changing to loop playback) and stays in a different state of appearance to indicate that the function is active.

In addition, QCD lets each button take one of four appearances depending on the state of the mouse at any given moment. These provide important cues to the user about the function of the button and what will happen if the user clicks or releases the mouse button at that moment. You can make several of these appearances identical, but you must provide an appearance for all four states.

Each button bitmap includes the appearance for four states:

- **Normal:** The normal “up” state of the control.
- **Mouse-over:** The “roll-over” state (the mouse pointer is over or rolling over the control).
- **Depressed with mouse off:** The user most often will see this with a “toggle” type button (such as the Loop or Shuffle buttons) where clicking the button leaves it in a permanently depressed (“down”) state. However, the user might also see this state with a normal button if he has pressed down on the button, but before letting go of the mouse button he changes his mind and moves the pointer off of the control (if he releases the mouse button, nothing will happen.).
- **Depressed with mouse over.** For a “toggle” type button, this state serves as “the roll-over” state when the button is locked in the “depressed” state. For normal buttons, the user will see this as he is clicking a button (he has pressed a button down, but has not released it yet) as a cue that when he releases the mouse button, the control’s function will execute.

Although you must provide an appearance for all four states in the skin bitmaps, the appearance of each state can alter the interaction quite noticeably. For instance, if you give the Normal and Mouse-over states the same appearance (by copying the bitmap), as well as the two depressed states, this effectively creates a two-state button rather than four. Designers typically provide different appearance for each of these states so the user gets more feedback when interacting with the player and one of its controls. The only constraint is that the alternate appearances of a “button” control must all be the same size (in pixels).

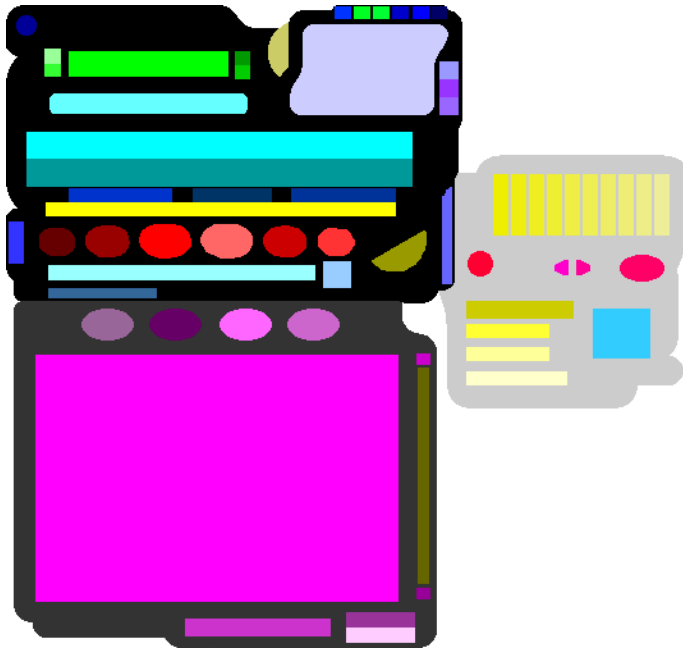
## The Bitmap Files

The following sections describe each of the *.bmp* files that together comprise a skin. The illustrations in the following sections show an example skin designed for skin designers to examine and which illustrates all the various features. When executing, the example skin makes the QCD player look like this.



Note: With all the bitmaps, make sure to crop out any extra white space around the edges of the files, especially the *Body.bmp* file. Getting rid of any unnecessary pixels reduces the size of your skin and thus reduces download time and memory usage.

## Map.bmp – The Master Map



The *Map.bmp* file is the Master Map that defines which controls are displayed in the player as well as the overall geometry and layout of the skin and the shape of the individual controls. The bitmap may be arbitrarily large, although a really large Map may encounter performance issues. In addition to the various code codes for the controls it must include the color codes for the Main Body (black 0x000000) and any of the four Extended Areas you want to implement. Any control that touches an Extended Area region (in other words, is adjoining at least one gray pixel) is considered part of the Extended Area, and will only be visible when the user extends the body. These regions are optional. However, if you choose to include any of the Extended Areas, you must also include the corresponding Extend Button. If you do not include the Extend Button, then QCD always displays the Extended Area.

You cannot have two or more instances of a given control (for example, having two Play buttons). However, you can split a control on the *Map.bmp* file to create different effects. For example, you can put a space in the track name text area to make it look like it was obscured by another control or design element.

























































































**Important Note:** It is very important that you **avoid** any **anti-aliasing** in the *Map.bmp* file. If the edges of a control are anti-aliased, this will introduce colors that QCD does not recognize (which at best, may produce unwanted holes in the player), or which QCD has assigned to other controls. For example, if a light blue control is anti-aliased against a black background, this will introduce pixels that are darker shades of blue, some of which may be assigned to other controls.

### Color codes

The table below defines the color scheme used for each of the controls. These fall into the following main categories:

- |              |  |
|--------------|--|
| • Grays      | Main window body and extended areas.           |
| • Greens     | Timer controls and skin modes                  |
| • Turquoises | Text displays                                  |
| • Reds       | Playback controls, EQ Controls                 |
| • Yellows    | Progress, Volume and Equalizer Slider controls |
| • Blues      | Window controls                                |
| • Violets    | TrackBox and Playlist controls                 |

Each color in the following table is given in both its hexadecimal form, and as an RGB triplet.

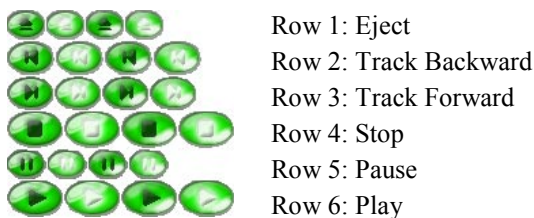
Sample	Hex	RGB		Sample	Hex	RGB	
	#000000	0, 0, 0	Main Window Body		#FF00FF	255, 0, 255	TrackBox Area
	#333333	51, 51, 51	Extended Area 1 (Detachable)		#CC00CC	204, 0, 204	Scroll TrackBox Up
	#CCCCCC	204, 204, 204	Extended Area 2		#990099	153, 0, 153	Scroll TrackBox Down
	#999999	153, 153, 153	Extended Area 3		#660066	102, 0, 102	Delete Tracks Menu
	#666666	102, 102, 102	Extended Area 4		#FF33FF	255, 51, 255	Repeat Playlist Toggle
	#000099	0, 0, 153	Menu		#FF99FF	255, 153, 255	Repeat Track Toggle
	#0000CC	0, 0, 204	Minimize		#993399	153, 51, 153	Repeat Combo Button
	#0000FF	0, 0, 255	Close		#FFCCFF	255, 204, 255	Random Play Toggle
	#000066	0, 0, 102	Help		#FF66FF	255, 102, 255	Save Playlist
	#0033FF	0, 51, 255	Always On Top		#996699	153, 102, 153	Add Tracks Menu
	#0066FF	0, 102, 255	Preferences		#CC66CC	204, 102, 204	Sort Playlist Menu
	#3333FF	51, 51, 255	Gracenote Delivered Links		#CC33CC	204, 51, 204	Search Playlist Control
	#33CCFF	51, 204, 255	QCD Website		#999900	153, 153, 0	Main Volume Slider
	#CCCCFF	204, 204, 255	Visual Effects Area		#CCCC00	204, 204, 0	Soundcard Volume Slider
	#9999FF	153, 153, 255	Next Visual Effect		#FFFF33	255, 255, 51	Soundcard Bass Adjust Slider
	#9966FF	153, 102, 255	External Visual Effects		#FFFF99	255, 255, 153	Soundcard Treble Adjust Slider
	#9933FF	153, 51, 255	Full Screen Visual Effects		#FFFFCC	255, 255, 204	Balance Adjust Slider
	#99CCFF	153, 204, 255	Gracenote Music Browser		#FFFF00	255, 255, 0	Playback Progress Slider
	#003366	0, 51, 102	Edit Track Information		#CCCC66	204, 204, 102	Visual Effects Level Slider
	#003399	0, 51, 153	Extended Information		#666600	102, 102, 0	Playlist Scroll Slider
	#0033CC	0, 51, 204	Launch Library Plugin		#FF0033	255, 0, 51	EQ Enable
	#336699	51, 102, 153	Open Extended Area 1 (dragable)		#FF0066	255, 0, 102	EQ Presets
	#6666FF	102, 102, 255	Open Extended Area 2		#FF0099	255, 0, 153	EQ Next Preset
	#6666CC	102, 102, 204	Open Extended Area 3		#FF00CC	255, 0, 204	EQ Previous Preset
	#666699	102, 102, 153	Open Extended Area 4		#EEEE00	238, 238, 0	EQ Band 1
	#3366FF	51, 102, 255	Close Extended Area 1		#EEEE11	238, 238, 17	EQ Band 2
	#660000	102, 0, 0	Eject Menu		#EEEE22	238, 238, 34	EQ Band 3
	#990000	153, 0, 0	Track Backward		#EEEE33	238, 238, 51	EQ Band 4
	#CC0000	204, 0, 0	Track Forward		#EEEE44	238, 238, 68	EQ Band 5
	#FF0000	255, 0, 0	Stop		#EEEE55	238, 238, 85	EQ Band 6
	#FF3333	255, 51, 51	Pause		#EEEE66	238, 238, 102	EQ Band 7
	#FF6666	255, 102, 102	Play		#EEEE77	238, 238, 119	EQ Band 8
	#CC3333	204, 51, 51	Play/Pause Combo		#EEEE88	238, 238, 136	EQ Band 9
	#CC6666	204, 102, 102	Play/Stop Combo		#EEEE99	238, 238, 153	EQ Band 10
	#33FF33	51, 255, 51	Show Track Time		#33FF00	51, 255, 0	Next Skin Mode
	#99FF99	153, 255, 153	Show Playlist Time		#00FF11	0, 255, 17	Skin Mode 1
	#009900	0, 153, 0	Show Elapsed Time		#00FF22	0, 255, 34	Skin Mode 2
	#00CC00	0, 204, 0	Show Remaining Time		#00FF33	0, 255, 51	Skin Mode 3
	#00FF00	0, 255, 0	Time Display Area		#00FF44	0, 255, 68	Skin Mode 4
	#00FFFF	0, 255, 255	Album Information		#00FF55	0, 255, 85	Skin Mode 5
	#00CCCC	0, 204, 204	Artist Information		#00FF66	0, 255, 102	Skin Mode 6
	#009999	0, 153, 153	Track Information		#00FF77	0, 255, 119	Skin Mode 7
	#66FFFF	102, 255, 255	System Message		#00FF88	0, 255, 136	Skin Mode 8
	#99FFFF	153, 255, 255	Campaign Message (Music Browser)		#00FF99	0, 255, 153	Skin Mode 9

## ***Body.bmp* – The Main Body and Extended Areas**



The *Body.bmp* file provides the graphic background that all of the controls are drawn over. This includes both the main body of the skin as well as the extended areas (if any). Extended areas aren't restricted to just extending the main body. Extended areas can contain the buttons to open other extended areas to create a branch effect. In this case, when the top-level extension is closed, all sub-extensions close automatically.

## ***ButtonSet.bmp* – The Track Control Buttons**

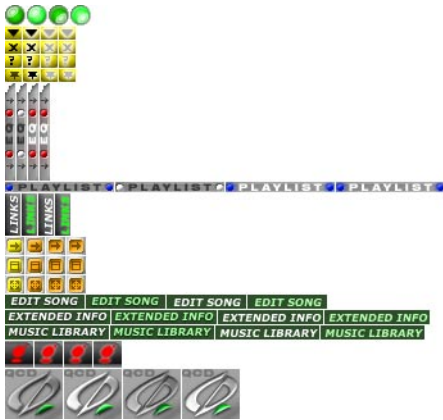


The *ButtonSet.bmp* file defines the basic CD player controls for the skin. Each player Button control must have a graphic element for each of the four states: Normal, Mouse-over (“rollover”), Depressed with mouse-off, and Depressed with mouse-over. Each of the images for the four states for a button is separated on a row by a one-pixel border between graphic elements (vertically) and a one-pixel border between rows (horizontally).

When implementing a Play/Pause combo button or Play/Stop combo button, the layout of this file remains the same. However there is the added necessity that the Play and Pause (or Play and Stop) images must be of the exact same size and shape since they will be occupying the same area on map.



## WindowSet.bmp – Basic Windows Controls



- Row 1: Main Menu (opens the QCD menu)
- Row 2: Minimize button
- Row 3: Close button
- Row 4: Help button
- Row 5: Always On Top
- Row 6: Preferences (in this example, omitted)
- Row 7: Extend Button for Extended Area 1 (in this example, EQ controls)
- Row 8: Extend Button for Extended Area 2 (in this example, omitted)
- Row 9: Extend Button for Extended Area 3 (in this example, omitted)
- Row 10: Extend Button for the detachable Extended Area (in this example, the Playlist)
- Row 11: Close button for detachable Extended Area (in this example, omitted)
- Row 12: Links Menu (displays CDDB-supplied menu of music-related URLs)
- Row 13: Next Visualization (cycles visuals effects to next visualization plug in)
- Row 14: External Visualizations (separates visualization area from main body)
- Row 15: Full screen Visualizations (runs visualizations in full screen mode)
- Row 16: Edit Song Info (brings up window for displaying/editing song or track information)
- Row 17: Extended Info (brings up window for displaying extra song or track information)
- Row 18: Launch Music Library (displays menu for launching a Library plug-in)
- Row 19: Music Browser (displays the CDDB Music Browser)
- Row 20: QCD web site link

The *WindowSet.bmp* file defines the basic window controls (such as Close and Minimize) normally associated with a Windows application, plus some extra controls specific to the QCD Player. The QCD-specific controls include the Extend buttons to display or hide the extended body sections (rows 7-10), Links menu to display a list of music-related dynamically downloaded URLs (row 12), Music Browser button to display the CDDB Music Browser (row 19), and a button that links to the QCD web site (row 20).

Each window control must have a graphic element for each of the four states: Normal, Mouse-over (“rollover”), Depressed with mouse-off, and Depressed with mouse-over. Each of the images for the four states for a button is separated on a row by a one-pixel border between graphic elements (vertically) and a one-pixel border between rows (horizontally).

## TrackSet.bmp – The TrackBox Controls

There are variations in the way *TrackSet.bmp* is laid out, depending on how you want the TrackBox to look and behave, as well as which Repeat control you decide to use.

The following illustration shows *TrackSet.bmp* in two formats. The first layout (example 1) is for a player that uses track titles to label the Track Buttons and the Repeat Combo control. The second layout (example 2) is for a player that labels the Track Buttons with the number of the track and the Repeat Toggle controls. See the text below for more information on the differences.

Example 1:



### Internal Map

0xFF00FF – track item shape  
 0x0000FF – track digit size (omit if using titles, as in example 1)  
 0xFF0000 – track spacing (size of track item with surrounding space)  
 0x00FF00 – transparent track indicator  
 0x00FFFF – bendable trackbox indicator  
 0x777777 – internal map delimiter (end of internal map)

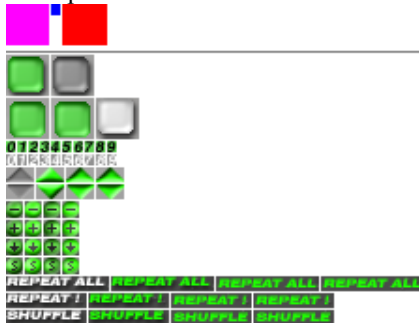
\* when using track titles (example 1)

Row 1: Normal track bitmap  
 Row 2: 'Mouse-over' bitmap  
 Row 3: 'Being dragged' (depressed, mouse-off) bitmap  
 Row 4: 'Track selected' (depressed, mouse-over) bitmap  
 Row 5: 'Track marked' bitmap (track selected)

\* when using track numbers (example 2)

Row 1: Normal track bitmap **and** 'Mouse-over' bitmap  
 Row 2: 'Being dragged' (depressed, mouse-off) bitmap **and** 'Track selected' (depressed, mouse-over) bitmap **and** 'Track marked' bitmap (track selected)  
 Row 3: highlighted set of track number digits  
 Row 4: dimmed set of track number digits  
 Row 5: non-existent

Example 2:



\* rows 6 to 11 same for both layouts

Row 6: Scroll Up button  
 Row 7: Scroll Down button  
 Row 8: Delete Track button  
 Row 9: Add Track button  
 Row 10: Save Playlist button  
 Row 11: Sort Playlist button

\* when using both Loop Playlist and Loop Track buttons (example 2)

Row 12: Repeat Playlist toggle (loop entire playlist)  
 Row 13: Repeat Track toggle (loop current track)  
 Row 14: Shuffle Playlist toggle

\* when using single Repeat Combo button (example 1)

Row 12: Loop Playlist  
 Row 13: Loop Track  
 Row 14: Loop None (off)  
 Row 15: Shuffle Playlist toggle

The *TrackSet.bmp* file defines the controls related to the player's "TrackBox", an array of Track Buttons that the user can click on to jump to a specific track, or can rearrange to create a custom playlist. You can label each Track Button with the number of the track, or with the text of the track title.

### ***Trackset.bmp – Internal Map***

The top of area of the *TrackSet.bmp* bitmap image is the Internal Map. This Internal Map is separated from the rest of the *TrackSet.bmp* file by a one-pixel line of gray (0x777777), followed by a one-pixel gap. This allows QCD to stop looking for these reserved colors in the rest of the bitmap so that you don't have to avoid them in your design.

Within the Internal Map the following color regions are defined.

*0xFF00FF [255, 0, 255] (required)*

Represents the size and shape of a single Track Button. Must be present.

*0xFF0000 [255, 0, 0] (optional)*

Represents the size of a Track Button *plus* the space to the right and below the button. Shape is not a factor, only the bounding rectangle. If you omit this region, QCD tiles the buttons with no spacing within the TrackBox area.

*0x0000FF [0, 0, 255] (required for numbered track buttons, omit for text buttons)*

Represents the size of a digit of the track number. All digits must be the same size. The presence of this region also indicates that the Track Buttons are to be numbered.

*0x00FF00 [0, 255, 0] (optional)*

Indicates that the tracks should be drawn with a transparent background. Size and shape of indicator are not factors (1x1 pixel is sufficient). The Track Button bitmap (row 1) must still be present to preserve compatibility with older versions of the player.

*0x00FFFF [0, 255, 255] (optional)*

Indicates that the tracks should follow the contours of the trackbox boundaries instead of lining up vertically. Size and shape of indicator are not factors (1x1 pixel is sufficient).

*0x777777 (required)*

End of Internal Map

The size of the TrackBox in the player is determined by the 0xFF00FF (magenta) region of the *Map.bmp* file. QCD tiles as many whole Track Buttons as it can fit across the width of the TrackBox, and then starts a new row. It then tiles as many whole rows as it can fit into the height of the TrackBox. If there are still more tracks than fit in the TrackBox area, the user can use the TrackBox Scroll Buttons.

**Note:** There are five bitmap images for the Track Button, and these appear on either two lines or five lines depending on whether you are using track numbers or the text of the track title to label the buttons. If you are using numbers to label the buttons (in other words, if you included dark blue pixels in the Internal Map), the bitmaps are on two lines. If you are using text to label the buttons, each of the five bitmaps is on its own line. This is because buttons labeled with text are much wider than buttons labeled with numbers, so this system reduces unnecessary white pixels in the *TrackSet.bmp* file.

The first two button bitmaps define the Normal and Mouse-over (“rollover”) state of the track buttons. Note that when the track is currently playing (and the mouse pointer is not over the TrackBox), the Track Button flashes between these two states. If the transparent background indicator region (0x00FF00) is in the Internal Map, the first bitmap is not used but must still be present in some form. This is to provide compatibility with older player versions.

The second and third track bitmaps define the Depressed with Mouse off, and Depressed with Mouse-on states. When the user is dragging a track (to create a custom play order) the second of these Depressed states is used as the “being dragged” icon. The fifth bitmap provides a graphic that QCD merges on top of a button, using a 50% blend, to indicate that the user has “suppressed” the track from playback by right-

clicking the Track Button. This bitmap appears on the same line with the two depressed states if you are using number-labeled buttons, or on its own line if you are using text-labeled buttons.

If you are using track numbers to label the Track Buttons, QCD constructs the numbers using digits centered within the button. The size of a digit is determined by the 0x0000FF (dark blue) region at the top of the *TrackSet.bmp* file. (It is recommended that the size of the button and the digits be such that you can fit three digits in the Track Button.) Track digits are constant-width and the bitmaps that define the digit characters should appear on two rows following the Track Button bitmaps (rows 3 and 4, example 2 above). The two rows define the highlighted and unhighlighted versions of the digits, starting with digit '0', ending with digit '9'. Each of the ten digits should be separated by a one-pixel gap.

If you are using the text of the track title to label the Track Buttons, omit the dark blue region in the Internal Map, and the two rows of Track Digits. QCD labels the button with the text of the Track Title, truncated if necessary to fit within the button width. QCD uses the type formatting you specify in the *Charset.ini* file (see page 21).

The remaining bitmaps in *TrackSet.bmp* define the appearance of the other controls related to the TrackBox.

The TrackBox Scroll Up and Scroll Down buttons each have the standard button states: Normal, Mouse-over ("rollover"), Mouse-down, mouse-down with mouse-over. Note that there is no "disabled" state for the scroll buttons—they are always enabled, although they may have no effect if there is a short track list. Each of the images for the four states is separated on a row by a one-pixel border between graphic elements (vertically) and a one-pixel border between rows (horizontally).

The Delete Track and Add Track buttons have the standard four button states: Normal, Mouse-over ("rollover"), Mouse-down, mouse-down with mouse-over. A one-pixel border separates each of the images for the four states.

There are two ways to implement the Repeat Playlist / Repeat Track buttons. You can implement the two repeat buttons individually (as example 2 of *TrackSet.bmp*) where each button is a toggle for its respective feature. Or you can have one button that cycles through the 3 repeat states (repeat off, repeat playlist, repeat track) (as example 1 of *TrackSet.bmp*). The type of repeat button is determined by the color of the repeat control defined on *Map.bmp*.

**Important:** you can only have one or the other type of loop button color defined on *Map.bmp*. Having both types will cause errors in the skin.

If implementing two individual repeat buttons, the Repeat Playlist, Repeat Track, and Shuffle Playlist buttons are toggles, and have basically the same standard four states: Normal (the toggle is off and the Mouse is not over the button); Mouse-over (the "rollover" state when the toggle is off); Depressed with mouse off (the normal state when the toggle is on); and Depressed with mouse over (the "rollover" state when the toggle is on).

If implementing the single repeat button that cycles through the states the layout of *TrackSet.bmp* varies somewhat. Rows 12 and 13 are for the Repeat Playlist and Repeat Track buttons respectively, and only the last two states are used but are laid out as if there were the four states (hence the leading white space). Row 14 is for the Repeat Off (disabled) image, but in this case, only the first two state images are used.

## TimerSet.bmp – The Timer Display



Internal Map  
0xFF00FF - Timer digit size  
0x777777 – internal map delimiter (end of internal map)  
Row 1: Timer Digits  
Row 2: Track time toggle  
Row 3: Playlist time toggle  
Row 4: Elapsed time toggle  
Row 5: Remaining time toggle

The *TimerSet.bmp* file defines the appearance of the Timer display and controls. At the least, the timer area should be large enough to host two placeholders for the minutes and seconds (e.g., 12:34). However, the values displayed could require three ‘minute’ placeholders as well as a minus sign for decreasing timers (i.e.: -123:45). QCD will truncate the left side of the number to the available space.

The top area of the *TimerSet.bmp* image is the Internal Map consisting of an area of 0xFF00FF (magenta) pixels defining the size of the Timer Digits. This Internal Map is separated from the rest of *TimerSet.bmp* by a one-pixel line of color 0x777777 (gray), followed by a one-pixel gap.

Below the Internal Map is a row defining the appearance of the Timer Digits. These are constant-width digits 0 through 9, plus a colon and minus sign. These are *not* separated by a one-pixel boundary.

**Tip:** the colon and minus sign are capable of being skinnier than the digits using the following technique: QCD will calculate the width of the colon + the minus sign as the width between the end of the ‘9’ digit and the right side of the bitmap. The width of each the colon and minus individually are then half that calculated width. The colon and minus are always of equal width, and cannot be wider than the defined digit width.

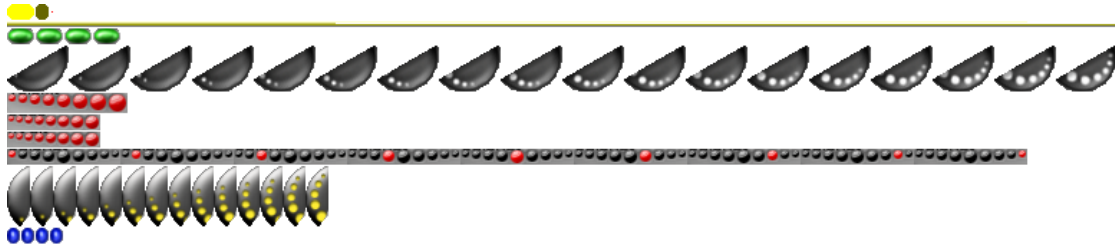
The timer controls are only two-state buttons (toggles) – off and on. Drawing only these two states is required. In the Master Map (*Map.bmp*), you should pair the Elapsed/Remaining Time toggles (place them together). The same applies to the Track vs. Playlist toggles.

### Using *Charset.ini* for Timer digits

As an alternative to using *TimerSet.bmp* digits for the timer you can choose to define a font using the *charset.ini* method described in the Character Sets section below. The existence of a TimerFont in the *charset.ini* file will override the digits in *TimerSet.bmp*.

When using this method you can eliminate the Timer Digit size internal map and the bitmap digits, but the internal map delimiter must remain. However, by including the internal map and bitmap digits the skin will remain backwards compatible with older player versions.

## *SliderSet.bmp* – Sliders, Progress Bars, and Knobs



Internal Map: delimited by gray (0x777777) line. Defines shape and size of each control using the color codes listed below.

Row 1: The Playback Progress Slider	color code: 0xFFFF00 (RGB=[255, 255, 0])
Row 2: The Master Volume Slider	color code: 0x999900 (RGB=[153, 153, 0])
Row 3: Soundcard Volume	color code: 0xCCCC00 (RGB=[204, 204, 0])
Row 4: Bass Control	color code: 0xFFFF33 (RGB=[255, 255, 51])
Row 5: Treble Control	color code: 0xFFFF99 (RGB=[255, 255, 153])
Row 6: Balance Control	color code: 0xFFFFCC (RGB=[255, 255, 204])
Row 7: Visual Effects Level	color code 0xCCCC66 (RGB=[204, 204, 102])
Row 8: Playlist Scrollbar Slider	color code: 0x666600 (RGB=[102,102,0])

The *SliderSet.bmp* file defines the appearance of different types of “slider” controls. In the QCD Player there are eight controls that use sliders (equalizer exempted, see *EqSet.bmp* below): the Playback Progress; Master Volume; Soundcard Volume; Bass; Treble; Balance; Visual Effects Level (a way to control the degree of response of any visualization plug-in displayed in the visualization area of the Player, if your skin has a visualization area), and the Playlist Scrollbar. The user can manipulate any slider by clicking down on the slider then dragging the mouse.

The top of the *SliderSet.bmp* file is the Internal Map. Each of the eight controls has an assigned color code that you use to draw the size and shape of the moving part of the control. The assigned color code in the *SliderSet.bmp* Internal Map is the same color code as used in *Map.bmp* for the same control (these are the yellows on page 6). These color codes are listed again above.

You separate this Internal Map from the body of the *SliderSet.bmp* file using a one-pixel gray (0x777777) line followed by a one-pixel gap. This lets QCD stop looking for reserved color pixels so that you can use these reserved colors in your design of each of the control bitmaps.

Below the Internal Map is a series of rows containing the bitmaps for the eight controls.

You do not have to provide all eight controls, omitting and sliders not required for the skin. The QCD Player knows which controls you are choosing to implement by whether you have included the control in the Master Map (*Map.bmp*). For example, if you wish to provide a Bass control, but not a Balance control, your *Map.bmp* image should have a yellow pixel region of color 0xFFFF33, but no pale yellow pixels of color 0xFFFFCC. If you omit any slider control from the *Map.bmp* file, then you must omit the corresponding row from the *SliderSet.bmp* file as QCD assumes it’s not there.

There are three basic types of sliders that you can use for any of these eight controls:

- A **basic tracking slider** consisting of a “handle” and a “slot” in which the handle moves;
- A **progressive slider (a.k.a. a “thermometer”)** consisting of a graphic that grows;
- A **rotational slider (a.k.a. a “knob”)** consisting of several frames drawn in the same area.

You can use any combination of these three types to implement the eight controls. Note that the basic tracking and progressive (thermometer) sliders can be horizontal, vertical, or diagonal.

### ***“Tracking” Sliders***

A basic “Tracking” slider consists of two parts: a movable “Handle” and a static “Slot” in which the handle appears to move. To implement a “tracking” slider for a control you need to do the following:

- In the *Map.bmp* file, you define the “Slot Size” by specifying a region in which to draw the “slot” image. You do this by using pixels of the appropriate color code for the control you are defining. (For example, for the Bass control, use yellow 0xFFFF33.)
- In the *Body.bmp* file, you include the Slot image.
- In the Internal Map area of the *SliderSet.bmp* file, you define the “Handle Size” using the appropriate color code for the control you are defining. The “Handle Size” must be smaller in width or height (or both) than the “Slot Size” defined above. (If it is not smaller there is no room for the handle to slide anywhere within the slot area. In this case QCD will treat the slider as a “knob”.)
- In the appropriate row of the *SliderSet.bmp* file, you include the Handle image. Be sure to make the Handle image the same pixel dimensions as the “Handle Size.” Otherwise, QCD just truncates your image as it draws it into the designated area.
- By default, there is only one image for the Handle bitmap. To create a Handle that has bitmaps for the four user interaction states, add the color red (0xFF0000) to the internal map section. This color acts as a flag so size and shape are not a factor. With this flag, all tracking sliders must implement the four bitmap states.

QCD draws the Slot image within the designated Slot area of *Body.bmp*. It then draws the Handle image on top of it, with leftmost-lowest position assumed to be the zero position of the control, and the topmost-highest position assumed to be the maximum position of the control. To implement a horizontal slider, just make sure the Handle Size is narrower, but the same pixel height as the Slot Size (i.e. there is room to move left and right in the slot area). To implement a vertical slider, make sure the Handle Size is shorter, but the same width as the Slot Size. To implement a diagonal slider, just arrange it so that there is room for the Handle to move both horizontally and vertically within the Slot area.

### ***“Progressive” Sliders (“Thermometers”)***

A “Progressive” slider, otherwise known as a “thermometer”-style slider, consists of a single bitmap that grows (or more correctly, is *revealed*) within a background area. To implement a “progressive” thermometer-style slider for a control, you need to do the following:

- In the *Map.bmp* file, you define the region in which to draw the “thermometer” image. You do this by using pixels of the appropriate shade of yellow for the control you are defining (see page 6). (For example, for the Playback Progress control, use yellow 0xFFFF00.)
- In the Internal Map area of the *SliderSet.bmp* file, do not include any pixels in the color corresponding to the control you are defining. For example, if you are defining the Playback Progress control as a thermometer, omit any pixels of color 0xFFFF00. This is how QCD knows to implement this as a “progressive” thermometer-type slider rather than a “tracking” slider with a separate moving Handle. Incidentally, if you are using all thermometer-style sliders, so that the Internal Map is empty, you still need to draw the one-pixel gray line at the top of the file indicating the end of the Internal Map.
- In the appropriate row of the *SliderSet.bmp* file, you must have one image: the image that you want to grow into the designated region on *Map.bmp*.
- Note that the background of the “thermometer” is defined by the background in *Body.bmp* (the Main Body of the player, or the Extended Body if this control is part of the extended area).

QCD draws the control by revealing the slider bitmap within the designated area of *Map.bmp*. When the control is in the zero position, the slider bitmap is completely hidden. As the control progresses, or as the

user drags within the designated area, more of the bitmap is revealed, from left to right and from bottom to top, until in the maximum position the slider is completely displayed within the designated area.

### **“Rotational” Sliders (“Knobs”)**

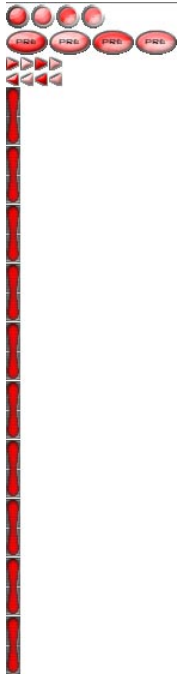
A “Rotational” slider, otherwise known as a “knob”, consists of a series of bitmaps representing frames showing the control at various stages from zero position to maximum position. You can also use this to implement standard sliders with special effects, such as a tracking slider that changes color. The user experiences each frame as a “detent”, so for smoother animation, you need more frames, although you should bear in mind that this of course makes your skin bitmaps larger. Note that for some controls where a center position is meaningful, such as a Balance knob, it is advisable to have an odd number of frames so that there is a center detent. To implement a “knob” for a control, you need to do the following:

- In the *Map.bmp* file, you define the “Knob Size” by specifying a region in which to draw the “slot” image. You do this by using pixels of the appropriate shade of yellow for the control you are defining (see page 6). For example, for the Master Volume control, use dark yellow 0x999900.
- In the Internal Map area of the *SliderSet.bmp* file, you define the width of the collection of frames by drawing a one-pixel-wide line using the appropriate reserved color for the control you are defining. Note that it is the fact that this is wider than the designated control region in the *Map.bmp* file that tells QCD that this is a “knob” and not a slider. Note also that if you have several “knob” controls, you need a different one-pixel-wide line for each (in its appropriate color). This is because each knob may have different sizes, or different number of frames, or both, and so QCD needs you to tell it how wide the total collection of frames is.
- In the appropriate row of the *SliderSet.bmp* file, you draw the individual frames of the knob, from its zero state at left to its maximum state at right. Unlike images in other files, you **do not separate** the frames using a one-pixel gap. (This is to make it easier for you to determine the total width of the frames for drawing the line in the Internal Map.) Note that each frame should be exactly the same pixel dimensions and shape as the designated region in the *Map.bmp* file.

QCD draws the control by drawing one of the frames within the designated region. When the control is in the zero position, QCD draws the first frame. When the control is in the maximum position, QCD draws the last frame. All other frames are drawn appropriately for the intervening states.



## *EqSet.bmp* – Equalizer Controls



Internal Map: delimited by gray (0x777777) line. Defines shape and size of each control using the color codes listed below.

- Row 1: Equalizer Enable (On / Off)
- Row 2: Equalizer Presets (Load / Save)
- Row 3: Next Equalizer Preset
- Row 4: Previous Equalizer Preset
  
- Row 5 - 14: Ten Equalizer bands

The *EqSet.bmp* file defines the appearance of the different equalizer controls. The equalizer consists of 4 buttons (for On/Off, Presets, and two buttons for moving between presets), and 10 sliders for the 10 bands of the equalizer.

The *EqSet.bmp* file follows the exact same rules as *SliderSet.bmp* (see above) for the 10 sliders making up the equalizer. The assigned color code in the *EqSet.bmp* Internal Map is the same color code as used in *Map.bmp* for the same control (these are the yellows on page 6). Again, you separate this Internal Map from the body of the *EqSet.bmp* file using a one-pixel gray (0x777777) line followed by a one-pixel gap.

There is a difference with *EqSet.bmp* in that the first four rows after the Internal Map are for the various buttons, followed then by the 10 sliders. Each slider is independent and can be of a different slider type, but for consistency in the interface they all should be of the same type.

### *SkinCtrlSet.bmp* – Skin Control Buttons



The *SkinCtrlSet.bmp* file defines the appearance of the buttons for changing between skin modes (if any). You can create a skin “family” consisting of up to nine skin “kids”, which the user experiences as different modes of the same skin. (See “Wrapping up skins” on page 24.) Note that end users can also create skin “families” by grouping together skin “kids” using the QCD Skin Browser. So even if you are only creating a single skin “kid”, it’s a good idea to include the first button, “Next Skin Mode”, in your skin.

Each of the buttons in *SkinCtrlSet.bmp* draws in the area indicated by the corresponding color in *Map.bmp*. If no pixels of the color are in *Map.bmp*, QCD skips the corresponding row of *SkinCtrlSet.bmp*.

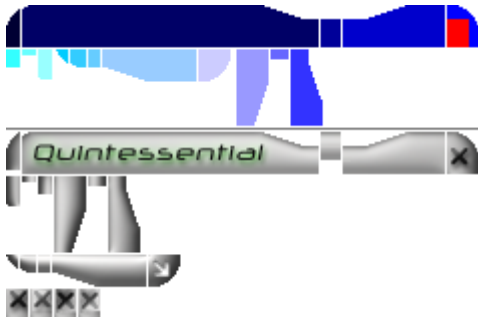
Row 1: Next Skin Mode	color code: 0x33FF00 (RGB=[51, 255, 0])
Row 2: Go to skin mode 1	color code: 0x00FF11 (RGB=[0, 255, 17])
Row 3: Go to skin mode 2	color code: 0x00FF22 (RGB=[0, 255, 34])
Row 4: Go to skin mode 3	color code: 0x00FF33 (RGB=[0, 255, 51])
Row 5: Go to skin mode 4	color code: 0x00FF44 (RGB=[0, 255, 68])
Row 6: Go to skin mode 5	color code: 0x00FF55 (RGB=[0, 255, 85])
Row 7: Go to skin mode 6	color code: 0x00FF66 (RGB=[0, 255, 102])
Row 8: Go to skin mode 7	color code: 0x00FF77 (RGB=[0, 255, 119])
Row 9: Go to skin mode 8	color code: 0x00FF88 (RGB=[0, 255, 136])
Row 10: Go to skin mode 9	color code: 0x00FF99 (RGB=[0, 255, 153])

### *MenuSet.bmp* – Menu Title Bars



The *MenuSet.bmp* file defines the banner image used as the title bar on the left side of the two QCD menus: the QuickTrack menu, and the QCD main menu. The width of each banner is fixed at 23 pixels. The two banners can be any height, but both must be the same height. If the menu gets longer than the banner bitmap, QCD tiles the bottom 20 pixels onto the bottom of the bitmap.

## ***BorderSet.bmp* – Window Borders**



The *BorderSet.bmp* file defines the borders of three resizable windows displayed by the QCD player:

- The CDDB Music Browser, a small web browser window with content provided by the CDDB service;
- The Visualization area (when detached from the player);
- The Preview Skin window (displayed by the user in Preferences:Download Skins:Preview Skin).

As with other *.bmp* files in the skin, the upper area is the Internal Map, which uses color codes to define the size of the bitmaps in the lower area. The Internal Map is separated from the bitmaps using a one-pixel gray line (0x777777).

- The border is separated into 16 segments: four corners and three segments for each of the four sides.
- Each side is divided into three segments: two attach to the corners and do not scale, and one is the “joiner” and QCD stretches this bitmap either horizontally or vertically to fill the width or height of the window. Thus you must design the “joiner” bitmap so that it joins to the other two segments, and design these other two segments to connect to the corners.
- For the color codes of the 16 segments, see the example *BorderSet.bmp* file.
- The order and layout of the colored regions in the Internal Map is not important. You can put them in any order and they do not have to be top-aligned. (QCD knows which is which by the color coding.) So you can rearrange these to occupy less space and make the size of the *.bmp* file smaller.
- The order and layout of the bitmaps in the lower bitmap area *is* important. This order is as follows: The bitmaps must be on three rows: the top border (including top corners), the sides, and the bottom border (including the bottom corners). The top row starts with the top-left corner, followed by the three bitmaps for the top edge—the left segment, “joiner”, and right segment—and ends with the top-right corner. The bottom row starts with the bottom-left corner, followed by the three bitmaps for the bottom edge, and ends with the bottom-right corner. The middle row has the three bitmaps for the left side—the bottom segment, the “joiner”, and the top segment—followed by the three bitmaps for the right side.
- Each bitmap in the lower area must be top-aligned to the top of its row, and separated from other bitmaps on the row by a one-pixel line.

### Close Button

- Optionally, a four state close button may be included within the top-right corner, indicated by the red color (0xFF0000) within the top-right corner section of the internal map (see the example *BorderSet.bmp* file). You can leave this red area out and the entire top-right area will function as the close button.
- If you include the four state close button, the graphic elements for the button are aligned in the fourth row (as shown in the example *BorderSet.bmp* file).
- Note: the red area for the close button must not extend beyond the left or bottom of the top-right corner area.



0x0) in the *CharSet.bmp* file, QCD will use this color to render all the characters. Note that as a non-white character, this also serves as the boundary delimiter for the character below it (ASCII 32, the space character). If you set this first pixel to white (0xFFFFFFFF), QCD makes an exception to the boundary rule and counts this first pixel as the boundary from the first character (the space character). This is to let you specify absolute white characters.

The second way to specify the color of the text is using the *CharSet.ini* file as described in the next section. A color specification in the *CharSet.ini* file overrides the first-pixel (position 0x0) color in *CharSet.bmp*.

QCD uses *CharSet.bmp* for all Text controls, unless you specify differently using *CharSet.ini*. You can have more than one *CharSet.bmp* file if you want different character bitmaps for different text controls.

### ***CharSet.ini***

The *CharSet.ini* file allows increased flexibility when defining what bitmaps and fonts to use for your skin. With the *CharSet.ini* file you can:

- specify multiple *CharSet.bmp* files. In other words, you could make a file named *CharSetDisc.bmp*, and a *CharSetTrack.bmp* and link them to Disc text and Track text respectively using the *CharSet.ini* file.
- specify multiple colors using the same *CharSet.bmp* file. In other words you can use the same *CharSet.bmp* file for any text areas, and use a different color for each.
- save from creating a *CharSet.bmp* bitmap by simply specifying a Windows font.
- specify a Windows fonts to better support international character sets

The *CharSet.ini* file is optional. If you do not provide it, QCD will use the *CharSet.bmp* file for the font and color specifications of the text for the Disc, Track and Artist text controls. For the Status, TrackButton, and Campaign text controls QCD will use the *SmallCharSet.bmp* file if it exists, otherwise those text controls will default to *CharSet.bmp* as well.

*CharSet.ini* has a section for each text control: **DiscFont**, **TrackFont**, **ArtistFont**, **StatusFont**, **CampaignFont**, **SearchFont** and **TimerFont**. There are also up to 6 sections for describing the fonts used on track buttons (see **TrackButtonFont** below). The format of *CharSet.ini* has a section header appearing in brackets (e.g., [DiscFont], followed by key/value combinations describing the font for that section).

The **SearchFont** section is to describe the font used in the Playlist Search control. If the search control exists on the skin, this section must be included since only Window's fonts will be used for the search control.

The **TimerFont** section is to describe the font used for the Timer control. Without the TimerFont section the timer control will revert to it's bitmap skinning method. See the *TimerSet.bmp* section for more information.

The **TrackButtonFont** section is to describe the font to be used on the track buttons in the TrackBox control (See *TrackSet.bmp* above). If you want to specify a different font for any of the different states a track button can have (Current, Selected, Pressed, Current+Selected, Pressed+Selected) then include the appropriate sections from the following:

**TrackCurrentFont**, **TrackSelectedFont**, **TrackPressedFont**, **TrackCurrentSelFont**,  
**TrackPressedSelFont**

Any section missing will default to the TrackButtonFont settings.

Valid key/value combinations for each section are:

**Height**=10      height in pixels  
**Width**=8      average width in pixels  
**Escapement**=0      specifies the angle, in tenths of degrees, between the escapement vector and the x-axis  
**Orientation**=0      specifies the angle, in tenths of degrees, between each character's base line and the x-axis  
**Weight**=400      weight of the font (0 – 1000, 400 is default)  
**Italic**=0      italic font if set to 1  
**Underline**=0      underline font if set to 1  
**Strikeout**=0      strikeout font if set to 1  
**FaceName**=Arial      string that specifies the typeface name of the font (e.g.: 'Arial', or 'Courier')

**Color**=255,255,255      Text color. Red,Green,Blue triple (valid values are from 0 – 255)  
**BgColor**=255,255,255      Background color. Red,Green,Blue triple (valid values are from 0 – 255) (**for SearchFont section only**)  
**FontFile**=filename      filename of font file (\*.fon; \*.ttf) included with skin to be used for this font  
**Bitmap**=filename      filename of character bitmap included with skin (same layout as *CharSet.bmp*)

**Tip:** If your skin uses a font file that may not be found on some systems, you can include it with the skin and have it loaded by QCD automatically. Use the *FontFile=filename* key/value to reference the included file.

*CharSet.ini* should always include Windows font settings even if a bitmap is being used since those values will be used to create a Windows font when the user enables international support or the 'Windows Fonts Only' option.

The following is an example *CharSet.ini* file:

```
; Describe a font for each section used on skin
;

; You can just supply the individual
; keys that matter (non zero)
;
; BTW: color is R,G,B
;

; this one is using the Window's 'Tahoma' Font
[TrackFont]
Height=17
Weight=900
FaceName=Tahoma
Color=0,0,255

; this one is using the custom font bitmaps in 'mycharset.bmp'
[ArtistFont]
Bitmap=mycharset.bmp
Color=0,0,255

; this one is using the Window's 'CoolFont' Font
; and the font file is included to be used incase
; it doesn't already exist on the system
[StatusFont]
Height=17
Weight=900
FaceName=CoolFont
FontFile=CoolFont.ttf
Color=0,0,255
```

## Testing a Skin

Once you have created all the *.bmp* files, the process for testing a target player using QCD is simple. Simply copy all of the *.bmp* files for a single skin into a directory. The next time you run QCD the new skin will be available to you by selecting it from the list of skins invoked by opening the **Skin Browser** (from the main window menu, or by hitting Alt+S). If your skin directory is not listed in the selection window, change where QCD looks for skins by clicking **Select Skins Folder**. Select the parent directory of your skin directory. The assumption is that the parent skin directory will contain several skins. Select the new skin (its name is taken from the directory name that contains the *.bmp* files) and verify that the user interface functions as you expect. If you find problems or inconsistencies you will need to edit the *.bmp* files and repeat the testing process. A skin can be reloaded without restarting QCD, by hitting F5.

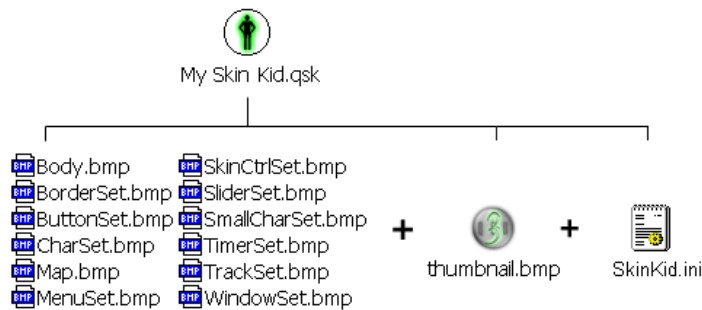
## Wrapping up skins

When you are ready to deliver one or more skins, you can deliver skins either as a Skin “Family”, or as individual Skin “Kids.” You can use a Skin Family to give the experience of several “modes” of the same skin. You can have up to nine modes in a Skin Family.

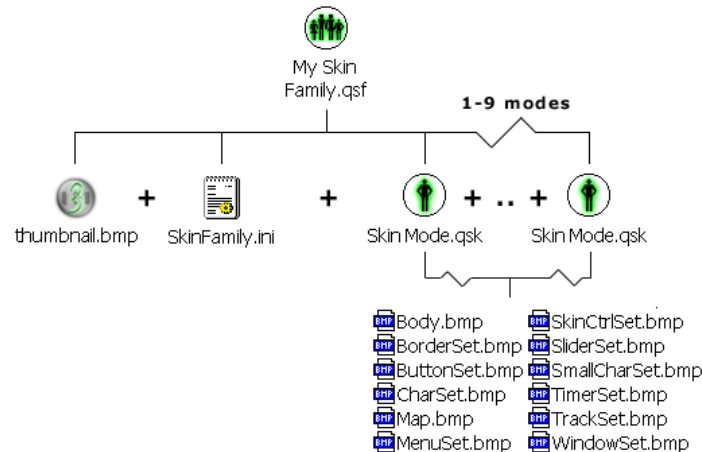


In either case, you need to create a Thumbnail image to represent your skin in the QCD Skin Browser. This is just a 60x60 pixel, RGB *.bmp* file that represents your skin. This does not have to be a screenshot of the skin, but can be an icon or logo.

To create a Skin Kid, use WinZip to combine all the skin bitmap files and text files into a single *zip* archive and then change the extension from *.zip* to *.qsk* (a Quintessential Skin Kid). You can also include an optional text file that follows the *SkinKid.ini* format described below. Including the *SkinKid.ini* file allows you to set a name for the skin independent of the skin filename and to name any extensions on the skin.



To create a Skin Family, use WinZip to create individual Skin Kid (*.qsk*) files for each of the skins as described above (except without the thumbnail). Then create a text file that follows the *SkinFamily.ini* format described on the next page. (Note: *SkinKid.ini* files within the *.qsk* files are not used in a family skin). Finally, use WinZip to create a master zip archive containing all the individual *.qsk* files, the *thumbnail.bmp* file, and the *SkinFamily.ini* file. Change the extension from *.zip* to *.qsf* (a Quintessential Skin Family).



To submit your *.qsk* or *.qsf* files to Quinnware, follow instructions found at <http://www.quinnware.com/skinz.html>.



## *SkinFamily.ini*

The *SkinFamily.ini* file defines how a set of individual skins (Skin Kids) can display as “modes” of the same Skin Family. There can be up to nine modes defined in a Skin Family. The *SkinFamily.ini* file defines the filenames, display names and relationships of the modes.

The format of the *SkinFamily.ini* file is as follows:

- Under the [family] section, the ‘**name=**’ key defines the name of the Skin Family as it appears in the QCD skin browser.
- The ‘**notes=**’ key defines the text that appears in the notes field in the Skin Browser.
- There can be up to nine [modeN] sections ([mode1] ... [mode9])
- Under each of the [modeN] sections, the ‘**name=**’ key sets the name of each skin mode.
- The ‘**file=**’ key defines the exact filename of the Skin Kid file containing the skin mode. Note, if you renamed the .zip archive containing the skin bitmaps and text files from .zip to .qsk, the ‘**file=**’ key should end in .qsk.
- The ‘**extensionN=**’ key sets the name of any of the extensions found on that mode. There are up to four extensions, with *extension1* being the moveable extension.

### Mode Switch Forced Extensions

To indicate a relationship between modes and extension states, use the ‘**msfe=**’ key for each mode. The **msfe** key lets you indicate which extensions must be opened, must be closed, or must remain the same as the current state, when switching to a new mode.

To indicate an extension is to be forced open, set msfe=N+, where N is the number of the extension.

To indicate an extension is to be forced closed, set msfe=N-.

And to indicate an extension is to remain in the same state as the last mode, set msfe=N=

Multiple extension states can be part of the **msfe** setting separated by commas.

Example: msfe=1=,2+,4-

Means set extension 1 as it stands in the current mode (open if open, closed if closed), set extension 2 as always opened, set extension 4 as always closed. Extension 3 remains unforced and will have the same state as when last using the mode.

The following is an example *SkinFamily.ini* file.

```
[family]
name=Family Values
notes=A basic skin suitable for the whole family \n Use '\n' for line breaks

[mode1]
name=Large-Mode Fancy
file=Large Mode Fancy.qsk
extension1=name of extension 1 (moveable extension)
extension2=name of extension 2
extension3=name of extension 3
extension4=name of extension 4
msfe=1=,2+,4-

[mode2]
name=Mid-Mode Fancy
file=Mid Mode Fancy.qsk

[mode3]
name=Mini-Mode Fancy
file=Mini Mode Fancy.qsk

[mode4]
name=Mini-Mode Simple
file=Mini Mode Simple.qsk
```

## ***SkinKid.ini***

The *SkinKid.ini* file gives a Skin Kid more customization and control over naming. The format of the *SkinKid.ini* file is as follows:

- Under the [settings] section, the “name=” key defines the name of the Skin Family as it appears in the QCD skin browser.
- The “notes=” key defines the text that appears in the Status area of QCD.
- The “extensionN=” key sets the name of any of the extensions found on that mode. There are up to 4 extensions, with extension1 being the moveable extension.

The following is an example *SkinKid.ini* file.

```
[settings]
name=The best kid skin
notes=A simple skin where one mode is enough \n Use '\n' for line breaks
extension1=name of extension 1 (moveable extension)
extension2=name of extension 2
extension3=name of extension 3
extension4=name of extension 4
```