Using Photoshop to Prepare Graphics for Embedded System Targets

A White Paper

Abstract
This paper explores the process of producing graphics for user interfaces on embedded system displays. It instructs on how to use popular image editing software to prepare and export graphic assets, and gives an overview of next generation tools to streamline and speed this process.

Introduction
Embedded systems range from handheld portable devices such as MP3 players or medical devices, to large stationary systems like traffic lights or factory controllers. As costs of manufacturing continue to come down, more and more embedded systems are using LCD displays as part of their user interface.

From soft keys to touchscreens, from grayscale to full color displays, most embedded system displays today use raster (bitmap) graphics for their user interfaces. Each visual element of the display is comprised of one or more bitmap images, arranged and positioned on-screen by the embedded system’s software.

The graphical user interface elements are typically created by artists or designers on standard desktop computers. Many tools are available to create and prepare bitmap resources for use on embedded systems. Perhaps the most widely used is Adobe Photoshop by Adobe Systems Incorporated. Photoshop has powerful tools for image editing and compositing as well as the ability to read and write a wide range of raster and vector file formats.

This white paper will demonstrate how to use Adobe Photoshop to prepare bitmap graphics for transfer and use by typical embedded systems.

Common Embedded Systems with Graphical Displays
- MP3 Players
- Cellular Phones
- Automotive HVAC / Infotainment
- Medical Devices
- Flight / Navigation Instrumentation
- Automated Teller Machines (ATMs)
- Computer peripherals (printers, etc.)
- Televisions / VCRs / Digital Recorders
- Home Automation / Security Systems
- Videogame Consoles
- Calculators
- Household Appliances (washer/dryers, microwaves, etc.)

Copyright © 2006 Altia, Inc.
www.altia.com
Know Your Embedded System Display Properties
Before attempting to create graphics of any sort for an embedded system, the designer must be aware of the target display’s physical characteristics (Screen resolution, Pixel Aspect Ratio, etc.). Embedded system displays can vary from desktop computer displays in many ways. It is also important to be aware of the target’s graphic processing capabilities (Does it have video acceleration? Does it handle anti-aliasing? What color depth does it support? Does it support transparency?)

Specific display properties are outlined below, along with instructions for using Photoshop to properly prepare images.

Screen Resolution
Screen resolution is the number of pixels in a display. Pixels are usually displayed in a grid-like manner, with a fixed number of rows and columns. For example, a screen with a resolution of 640 x 480 has 640 pixels per horizontal row, and 480 pixels per column, for a total of 307,200 pixels in the display. Common resolutions for handheld embedded systems are 240 x 320 (Quarter VGA) and 480 x 640 (VGA).

Set your image size in Photoshop to the embedded target display’s resolution. This is done when creating a new image, by selecting File/New... The resulting dialog allows various properties to be set for the new image, including width and height (Item 1 in Figure 1).

When dealing with screen based images, it’s best to display the units in pixels, as pixels are the smallest units of an LCD display. Photoshop is set to display in inches by default. You can change this in Preferences if you wish, or whenever a new image is created. Select pixels in the drop down list (Item 2 in Figure 1).
Pixel Aspect Ratio
Pixel aspect ratio is the ratio of an individual pixel’s height to its width. Most desktop computer displays have pixels that are square (a pixel aspect ratio of 1/1, or 1.0). When you draw a horizontal line 100 pixels wide, and a vertical line 100 pixels tall, the lines are the same physical length. Some embedded displays have pixels that are not square. If a designer does not take this into account when designing graphics for the display, the artwork may appear distorted or stretched when shown on the target.

Figure 2 – Pixel Aspect Ratio Example

Graphics on a display with a 1.0 Pixel Aspect Ratio (square pixels).

The same graphics on a display with a 0.8 Pixel Aspect Ratio (pixels are roughly 25% taller than they are wide, resulting in an image that looks “stretched” vertically).

To calculate the pixel aspect ratio of a display, you need the display manufacturer’s dimensions for an individual pixel. (The dimensions are typically in millimeters or fractions of a millimeter. The units do not matter as long as you use the same units for both height and width.) Apply the pixel width and height to the following formula:

$$\text{Pixel Aspect Ratio (PAR)} = \frac{\text{Pixel Width}}{\text{Pixel Height}}$$

It can often be a challenge to the designer to create artwork that looks correct on a given target device’s display. Fortunately, Photoshop has the ability to make this easier in the form of a user-definable Pixel Aspect Ratio. While designing artwork, you can configure Photoshop to show the graphic just as it will appear on the target (See Figure 3).

Select a predefined Pixel Aspect Ratio from the list, or define your own custom PAR by selecting **Image > Pixel Aspect Ratio** in Photoshop.
Color Depth
Nearly all embedded system displays use the RGB (Red, Green, Blue) color model. Each pixel in the display has a supplied red, green, and blue value that combine to display a given color.

Color depth is the number of distinct colors that can be displayed on a particular output device. Color depth is also referred to as bits-per-pixel (bpp), or bit depth, because it is the actual number of bits used to display color for each pixel. A 24-bit video display assigns 8 bits to each channel (red, green, and blue), giving a range of 256 possible values, or intensities, for each hue. This allows the display of 16,777,216 (256 x 256 x 256) unique colors.

It is important to be aware of your target’s supported color depth. If you create an image for the target that contains significantly more colors than the target system can display, the target system will usually attempt to map colors to the closest color in its palette. This almost always results in a moderate to drastic change in the image’s color(s). To avoid this, create an image in Photoshop that is the same color depth as your target system’s display.

### Common Image Color Depths

<table>
<thead>
<tr>
<th>Display Type</th>
<th># of Colors</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 bit</td>
<td>2</td>
</tr>
<tr>
<td>4 bit</td>
<td>16</td>
</tr>
<tr>
<td>8 bit</td>
<td>256</td>
</tr>
<tr>
<td>16 bit</td>
<td>65,535</td>
</tr>
<tr>
<td>24 bit</td>
<td>16.7 million</td>
</tr>
<tr>
<td>48 bit</td>
<td>281.5 trillion</td>
</tr>
</tbody>
</table>
Photoshop allows you to create images in either 1 bit, 24 bit or 48 bit color. If your target display doesn't support one of these color depths, you will need to obtain a 3rd party Photoshop plugin or separate image conversion software.

To set the color depth in Photoshop, be aware that Photoshop refers to color per channel (red, green or blue) rather than total color depth. When creating a 24 bit image in Photoshop, you would select RGB 8 bit as the image type. (This refers to 8 bits of color information for each of the red, green, and blue color channels, for a total of 8 + 8 + 8 = 24 total bits of information.)

To create a 24 bit image in Photoshop, select **File/New** and set the **Color Mode** to RGB color, and choose 8 bit in the next field.

![Creating a new 24 bit image in Photoshop](image)

**Figure 4 – Creating a new 24 bit image in Photoshop**

Remember:
Photoshop displays the number of bits / channel, **NOT** bits / pixel. Therefore, to create a 24 bit image in Photoshop, chose **8 bit** for the color mode. 8 bits Red + 8 Bits Green + 8 bits Blue = 24 bits/pixel

To create a 1 bit image in Photoshop, select **File/New** and set the **Color Mode** to Bitmap color, and choose 1 bit in the next field.

![Creating a new 1 bit image in Photoshop](image)

**Figure 5 – Creating a new 1 bit image in Photoshop**
DPI – Seeing the Physical Size in Photoshop

Printer resolution is measured in ink dots per inch (DPI). DPI is essentially irrelevant for images created in Photoshop (or any graphics editing program) for display on a desktop computer screen or an embedded system’s display. DPI is only relevant when sending the image to an output device such as a laser or inkjet printer.

You can, however, use the DPI setting to make graphics created for an embedded system appear on your desktop display at the actual physical size of the embedded target’s display.

When creating an image in Photoshop, set the image’s DPI to the physical pixels per inch of your target’s display.

\[
\text{Dots or Pixels Per Inch (DPI)} = \frac{\text{Width of Display in Pixels}}{\text{Width of Display in Inches}}
\]

For example, your target display is 240x320 pixels, and the physical dimensions are 1.594 inches x 2.125 inches, with a pixel aspect ratio of 1.0 (square pixels). Divide the width in pixels by the width in inches to get the physical DPI. \(240 \div 1.594 = 150.565\), or roughly 150 DPI.

Now, when you create graphics for your embedded target, set the DPI to 150. To view the image at the target display’s physical size in Photoshop, select View/Print Size (See Figure 6a). To revert back to actual pixel size, select View/Actual Pixels (See Figure 6b).

![Figure 6a – View at Print Size](image)

![Figure 6b – View as Actual Pixels](image)
**Anti-aliasing**

Aliasing is what occurs when you attempt to display analog data on a digital system. A curved line displayed on a grid is an example of this.

![Figure 7 – Analog data to be displayed](image)

Each block in this grid represents one pixel on an LCD display. To draw the line, each grid box (pixel) on the digital display must be completely filled with a single color. You cannot have a partially displayed pixel – each pixel must be completely filled or unfilled. When the graphics are rendered to the display, the result would look similar to this:

![Figure 8 – Analog data rendered digitally](image)

Anti-aliasing uses mathematical algorithms to fool the human eye into seeing smooth edges by blending the color of pixels along the edge of an image.

![Figure 9 – Example of Anti-aliasing](image)

Anti-aliasing “smoothes” the edges by filling adjacent pixels with a similar color to give the illusion of a smooth line.

**NOTE:** Because anti-aliasing uses additional colors to smooth an image, a monochrome (1-bit, or black and white) display **cannot** be anti-aliased.
Embedded systems with more powerful graphics processors may be able to automatically anti-alias images in real time. Many less expensive, less powerful or older embedded graphics processors are not capable of real-time anti-aliasing.

When designing artwork for an embedded display, it’s important to know whether the target has anti-aliasing capabilities. If it does not, a design decision must be made – to use aliased images, or anti-alias images “ahead of time” when the image is created.

Photoshop’s default setting is to anti-alias anything you draw. This is set on a tool-by-tool basis on the Tool Options bar when drawing in Photoshop.

![Adobe Photoshop interface with anti-aliasing options](image)

**Figure 10 – Anti-aliasing options can be turned on or off on a tool by tool basis on the Tool Options bar**

**Transparency / Alpha Channels**
All raster (bitmap) images are rectangular. To display a non-rectangular image in a rectangular bitmap, some of the pixels must be defined as transparent.

Without some form of transparency support, images that are rendered to a display will always be rectangular. Here’s an example of a round button saved as a bitmap.

![Round button bitmap](image)

**Figure 11 – Round button bitmap**

If this button bitmap is rendered to a display, the result is not as intended – the white portions of the image are still visible.

![Non-transparent round button bitmap](image)

**Figure 12 – Non-transparent round button bitmap**
Some image file formats support transparency in the image file itself (PNG and GIF for example). These image file formats store a separate 1 or 8 bit transparency channel, which allows you to define which pixels in the image are partially or fully transparent. Some embedded systems may directly support these file formats, making the designer’s job much simpler.

The simplest example of using transparency in Photoshop is with .PNG files. If your embedded target supports PNG’s, simply save your Photoshop artwork with no background layer. You will see a “checkerboard” effect which represents the transparent regions of your image.

When this image is imported and displayed on the embedded display, the “checkerboard” regions become transparent.

Another common option that embedded systems use is the concept of a pre-determined “transparent color.” Any image with that specific pre-determined color in it will have that color appear as transparent when rendered to the display.

For example, an embedded system’s graphics processor has magenta (RGB 255,0,255) as the pre-determined transparent color. Any magenta pixel in an image will become transparent.
If the embedded graphics processor does not support any form of transparency, a final less optimal option is also available. If you include the known background as part of the image itself, you can achieve the illusion of transparency.

![Round button bitmap with background detail included](pause_button.png)

Figure 15 – Round button bitmap with background detail included

When this image is placed at the exact location, the effect of transparency is achieved. If, however, the image is moved to another location, the resulting effect appears incorrect.

![Moving an image with included background detail may produce undesired results](pause_button.png)

Figure 16 – Moving an image with included background detail may produce undesired results
Preparing the Artwork for Output

Once the artwork is completed in Photoshop, it then becomes necessary to extract all the various individual elements out of the Photoshop file for use by the embedded system. Designers typically will manually “cut out” individual objects or elements and save them as separate image files. These files are then passed on to system engineers or programmers for inclusion in the embedded system code.

Software programming tools for desktop and embedded systems typically support a wide range of image file formats. Common image formats include Bitmap (BMP), Targa (TGA), Jpeg (JPG), and Portable Network Graphics (PNG). When exporting your images for transfer to an embedded system, if the image file format is not one of the commonly used formats, a conversion tool or process is often available.

Making Exporting of Images Faster using the Slice Tool

This manual exporting process can be very time consuming or confusing, as there may be literally hundreds of individual files that are exported. One option to alleviate some of this tedium is the Slice Tool in Photoshop.

![Slice Tool](image)

The slice tool in Photoshop is typically used when designing web pages. A designer can define rectangular regions of an image to divide, or slice it, into multiple smaller images, which are automatically saved as separate image files (See figure 17).

![Image divided into multiple slices using the Slice Tool](image)
A major disadvantage of using slices is that if elements of a source design are repositioned, the slices must also be manually repositioned — an often time consuming process.

For more information on slices and using the slice tool, refer to the Adobe Photoshop User Guide.

**Exporting the Slices**

Once you have created and positioned your slices, you are now ready to export them to multiple images. Each rectangular slice region will be saved as a separate file of the image format you specify.

To start this process, select **File / Save for Web...**
You will then be presented with the Save for Web window, where you can specify various settings for each slice in your image (See figure 20).

Selecting **Save** will prompt you for a location to save the exported images. An HTML file will also be created, so you can view your design in a browser.

![Save for Web Window](image)

**Figure 20 – The "Save for Web..." Window**

**Summary**

There are many considerations to take into account when designing graphics for display on today’s embedded systems. A graphics designer needs knowledge of physical properties as well as the specific capabilities of the embedded system’s display and graphics processor. Designing around the target embedded system’s graphics capabilities and limitations is becoming easier with current desktop graphics applications like Adobe Photoshop.
**Altia Contact Information**
5030 Corporate Plaza Drive
Colorado Springs, CO 80919

Phone: (719) 598-4299
Fax: (719) 598-4392
Email: info@altia.com
Technical Support: support@altia.com

**COPYRIGHT NOTICE**
Copyright © 1991-2006 Altia, Inc., All Rights Reserved.

**TRADEMARK INFORMATION**
The trademarks, logos and service marks ("Marks") contained herein are the property of Altia or other third parties. You are not permitted to use the Marks without the prior written consent of Altia or such third party which may own the Marks.

Altia® and the Altia logo are registered trademarks of Altia Incorporated.

Photoshop® is the registered trademark of Adobe Systems Incorporated.

All other brand and product names are trademarks, registered trademarks or service marks of their respective holders.