*by Janet Terra - April, 2006*

There are occasions when a programmer needs a bitmap larger than the graphicbox. The user either uses a scrollbar to see more of the graphic, or the program uses the sprite Backgroundxy command to scroll the background. In either case, the Liberty BASIC commands Loadbmp, Drawbmp, and Flush are all that are needed to load a bitmap from file, display to screen, and preserve the entire graphic.

Bitmap files take up a lot of file space, though, and often a bitmap of the desired size can consume 1 or more megabytes of space. In addition, the Getbmp command is unable to grab graphics that aren't actually visible on the screen. This means if you load a bitmap measuring 800 pixels by 400 pixels into a graphicbox measuring 400 pixels by 200 pixels, then Getbmp can only effectively capture 400 pixels by 200 pixels. Attempting to load beyond those borders will result in the pixels of the actual screen and not the bitmap.

One of the more efficient ways to display a background is using tiles. Tiles are small bitmaps that are placed in gridlike patterns to create a larger picture. This is especially useful when patterns are repeated. In this simple demo, a crude house is drawn using native Liberty BASIC turtle graphics. The resulting graphic is captured as a bitmap and loaded into memory using the Getbmp command. The bitmap is drawn and captured in the left graphicbox and then repeatedly drawn in the right graphicbox. Flush preserves the image as the user scrolls back and forth.

- Demo1

It is possible to use an API call to tile this bitmap. This demo assumes you have a working knowledge of Drawing in Memory. If this isn't the case, or you just need a refresher, read Drawing in Memory by Alyce Watson in Newsletter #101. Alyce explains all you need to know about Device Contexts, and the commands GetDC, ReleaseDC, DeleteDC, and SelectObject. You can also read up on the #gdi32 calls in Alyce's Liberty BASIC 4 Companion. Both resources are must-haves for delving into manipulating graphics in memory.

This next demo uses CallDLL #gdi32, "BitBlt" to draw three crude tile images in the second graphicbox.

- Demo2

For practical purposes, this method is less effective than the native Liberty BASIC code used in the first demo. API created bitmaps cannot be flushed with the Flush command. You must follow with Getbmp, Drawbmp, Flush. As discussed, this isn't possible with non-visible graphics.

Fortunately, bitmaps can be constructed in memory. The resulting returned handle of that bitmap can then be loaded, displayed and flushed with the Liberty BASIC commands

```
Loadbmp “myPic”, handlePic
#w.g “Drawbmp myPic 0 0”
#w.g “Flush”
```

See the Liberty BASIC helpfile if you need further explanation loading a bitmap by its handle rather than its title.

To construct a bitmap in memory, the CallDLL #gdi32, “CreateCompatibleBitmap” is used. It is necessary to first create this compatible bitmap that will later contain the finished bitmap. Note that the dimensions of this API created compatible bitmap must be the size of the final desired bitmap.

The steps to using CreateCompatibleBitmap to building one large bitmap by appending several smaller bitmaps are

- Create one graphicbox to display the final bitmap
- Get both the handle of this graphicbox and the handle of its device context
- Create one compatible memory DC for containing the smaller bitmap
- Create a second compatible memory DC for containing the final large bitmap
- Getbmp a screen image of the final large bitmap size
- Obtain the handle of that bitmap
- Select the large bitmap into the second memory device context
- One by one, for the number of images,
- Load or draw a small bitmap and get its handle
- Select the handle into the first memory device context
- Use BitBlt to transfer the small bitmap to the 2nd memory device context in the desired position
- Repeat until all images have been placed on the larger bitmap, replacing the original screen display
- Deselect the bitmap from memory
- Release the device context of the graphicbox
- Release both memory device contexts
- Unload the small bitmaps to free memory
- Loadbmp from the handle number rather than title name
- Drawbmp
- Flush

As always, I'd like to express gratitude to Alyce for posting these steps at LB Forum: Large Scrolling Background. The third demo uses CreateCompatibleBitmap to draw a random series of the three tiles to draw an extended background.

- Demo3

So far, there is no advantage to using Demo3 rather than Demo1. Supposing, though, the final bitmap is being used as a scrolled sprite background. Now you can see how useful CallDLL #gdi,

“CreateCompatibleBitmap” can be. This last demo shows a scrolling background 1200 pixels by 100 pixels using just three small tiled bitmaps. Even if these bitmaps were loaded from disk rather than captured mid program, the resulting saved bytes on disk would still be substantial. This last demo sets the large bitmap as the sprite background, then scrolls the background for an animated effect.

- Demo4

If, like me, you’re not much of an artist, try searching the internet for free tiled graphics. One such site is http://reinerstileset.4players.de/index.html.

Many of the wrapped API functions used in these demos were taken from the Liberty Basic 4 Companion by Alyce Watson. To learn more about the graphic device interface (gdi) DLL's, get your copy of the Liberty Basic 4 Companion at http://alycesrestaurant.com.
In addition to Alyce's LB4 Companion, the information for this article relies heavily upon these earlier Liberty Basic Newsletter articles -

- Drawing in Memory by Alyce Watson, Liberty Basic Newsletter #101
- Bitmaps the Fast Way by Callum Lowcay, Liberty Basic Newsletter #122
- TransparentBLT by Janet Terra, Liberty Basic Newsletter #128 with addendum Liberty Basic Newsletter #129