How to save 32 bit bitmaps in 24 bit bmp format

```
procedure SaveToFileBMP(const aBmp: TBitmap; aFileName: String);
var
  i, n, m, w: Integer;
  f: File;
  bmfh: BITMAPFILEHEADER;
  bmih: BITMAPINFOHEADER;
  p, p1: Pointer;
  pSrc: PIntArray;
begin
  if ExtractFileExt(aFileName) = '' then
    aFileName := aFileName + '.bmp';
  if GetDeviceCaps(aBmp.Canvas.Handle, BITSPIXEL) <> 32 then
  begin
    aBmp.SaveToFile(aFileName);
    Exit;
  end;
  with bmfh do
  begin
    bfType := Ord('M') shl 8 or Ord('B');
    bfSize := sizeOf(bmfh) + sizeOf(bmih) + aBmp.Width * aBmp.Height * 3;
    bfReserved1 := 0;
    bfReserved2 := 0;
    bfOffBits := sizeOf(bmfh) + sizeOf(bmih);
  end;
  with bmih do
  begin
    biSize := SizeOf(bmih);
    biWidth := aBmp.Width;
    biHeight := aBmp.Height;
    biPlanes := 1;
    biBitCount := 24;
    biCompression := BI_RGB;
    biSizeImage := 0;
    biXPelsPerMeter := 1;  {don't care}
    biYPelsPerMeter := 1;  {don't care}
    biClrUsed := 0;
    biClrImportant := 0;
  end;
  n := aBmp.Width;
  m := n * 3;
  if m mod 4 <> 0 then
    Inc(m, 4 - (m mod 4));
  GetMem(p, m);
  w := abmp.Width;
  BmpToArray(aBmp, Pointer(pSrc));
  AssignFile(f, aFileName);
  Rewrite(f, 1);
  BlockWrite(f, bmfh, SizeOf(bmfh));
  BlockWrite(f, bmih, SizeOf(bmih));
  for i := aBmp.Height - 1 downto 0 do
  {saving from bottom scanline to top because we set positive height value
  biHeight := aBmp.Height}
  begin
    {let Delphi calculate necessary address of current scanline}
    p1 := @pSrc[w * i];
    asm
      {we must preserve all registers we use except EAX, EDX, ECX}
      push esi
      push edi
      {ECX = count of colors in a scanline}
      mov ecx, n
      {ESI = address of source (32 bit) scanline. Format of color 'ARGB'}
      mov esi, p1
      {EDI = address of destination (24 bit) scanline. Format of color 'RGB'}
      mov edi, p
    @L1:
      lodsd  {EAX = source color 'ARGB'}
      stosw  {sending AX register with 'GB' part}
      shr eax, 16  {AX = 'AR'}
      stosb  {sending AL register with 'R' part}
      loop @L1  {decrement counter (ECX) and jump @1 if not zero}
      pop edi  {restoring "spoiled" registers}
      pop esi
      end;
    {while we sent n colors, to file we write m colors, thus doing padding
    values of additional bytes do not matter}
    BlockWrite(f, p^, m);
  end;
  CloseFile(f);
  FreeMem(p);
  FreeMem(pSrc);
end;
```

Original resource:	The Delphi Pool
Author:	Andrew Rybenkov
Added:	2013-01-27
Last updated:	2013-01-27