

1. Background

When emWin version 5.30 was released a new color format was added. Starting in emWin version 5.48 this new version is now the default format. Existing projects must be updated to be compatible with the new format. In addition some widget behavior has changed that will require changes to the existing projects.

2. Updating to emwin 5.48 and uEZ 2.11

The following section covers some of the GUI changes needed to convert older emWin demos to work with emWin 5.48.

2.1 Verify existing GUI colors

Before beginning the software update FDI **HIGHLY RECOMMENDS** that you record a video of your entire existing GUI making sure to capture all widgets, text, icons, screens and menus in the video. The goal will be to match all of the colors and transparency in the updated GUI.

2.2 Generating updated images

uEZ now ships the full set of NXP licensed emWin development tools in the following directory:
uEZ/Source/Library/GUI/SEGGER/emWin/Libraries/NXP/UtilityTools

All image generation scripts should point to this relative path to use the correct BmpCvtNXP.exe.

Updating generation scripts

The following example is from the project maker showing the relative path call to BmpCvtNXP.exe. To convert multiple images from an older script, just replace the old “BmpCvt.exe” with “%SEGGER_TOOLS_DIR%\BmpCvtNXP.exe” on each line as shown. If a special format is used, the command line argument for that image might need to be changed.

Updated Project Maker ConvertGraphics.bat file for converting the uEZ logo
set SEGGER_TOOLS_DIR=..\..\..\uEZ\Source\Library\GUI\SEGGER\emWin\Libraries\NXP\UtilityTools %SEGGER_TOOLS_DIR%\BmpCvtNXP.exe "logo.png" -convertintobestpalette -saveas"logo",1,7,0 -exit REM Add additional files here.

Example: uEZ logo

Updating the uEZ logo.c in the project maker demo.

BmpCvt.exe older versions

```
static GUI_CONST_STORAGE GUI_COLOR Colorslogo[] = {
    0x000000,0x33268C,0x36298D,0x3A2D90
    ,0x3F3293,0x201B46,0x392F80,0x31296F
    ,0x0F0D21
};
```

BmpCvtNXP.exe version 5.48

```
static GUI_CONST_STORAGE GUI_COLOR _Colorslogo[] = {
#if (GUI_USE_ARGB == 0)
    0x000000, 0x33268C, 0x36298D, 0x3A2D90,
    0x3F3293, 0x201B46, 0x392F80, 0x31296F,
    0x0F0D21
#else
    0xFF000000, 0xFF8C2633, 0xFF8D2936, 0xFF902D3A,
    0xFF93323F, 0xFF461B20, 0xFF802F39, 0xFF6F2931,
    0xFF210D0F
#endif
};
```

Example: GUIDEMO Automotive demo wrong color gauge and text underneath

Update to fix the color of the gauge and the text underneath. Without the 0xFF in the image palette for the gauge the text underneath it will appear as the wrong color due to wrong transparency.

Old version emWin 5.2X

```
static GUI_CONST_STORAGE GUI_COLOR _ColorsScaleR140[] = {
    0x000000, 0x00B400, 0xF5F5F5, 0x0000B4, 0x00F500, 0x494949,
    0xE1E1E1, 0xC0C0C0, 0x7F7F7F, 0x8DD78D, 0x9B9B9B, 0xD7D7D7,
    0xA8A8A8, 0xB4D7B4, 0x5D5D5D, 0xD7F5D7, 0x5DB45D, 0xB4B4D7,
    0x5D5DB4, 0xD7D7F5, 0x8D8DD7, 0xEBEBEB, 0x8DB4B4, 0x8D8D8D,
    0x5D8DB4, 0x8DB48D, 0xB4D7D7, 0x6F6F6F, 0xB4B4B4
};
```

Updated version:

```
static GUI_CONST_STORAGE GUI_COLOR _ColorsScaleR140[] = {
#if (GUI_USE_ARGB == 0) // 5.30
    0x000000, 0x00B400, 0xF5F5F5, 0x0000B4, 0x00F500, 0x494949,
    0xE1E1E1, 0xC0C0C0, 0x7F7F7F, 0x8DD78D, 0x9B9B9B, 0xD7D7D7,
    0xA8A8A8, 0xB4D7B4, 0x5D5D5D, 0xD7F5D7, 0x5DB45D, 0xB4B4D7,
    0x5D5DB4, 0xD7D7F5, 0x8D8DD7, 0xEBEBEB, 0x8DB4B4, 0x8D8D8D,
    0x5D8DB4, 0x8DB48D, 0xB4D7D7, 0x6F6F6F, 0xB4B4B4
#else //5.48
    0xFF000000, 0xFF00B400, 0xFFFF5F5F5, 0xFFB40000, 0xFF00F500, 0xFF494949,
    0xFFE1E1E1, 0xFFC0C0C0, 0xFF7F7F7F, 0xFF8DD78D, 0xFF9B9B9B, 0xFFD7D7D7,
    0xFFA8A8A8, 0xFFB4D7B4, 0xFF5D5D5D, 0xFFD7F5D7, 0xFF5DB45D, 0xFFD7B4B4,
    0xFFB45D5D, 0xFFFF5D7D7, 0xFFD78D8D, 0xFFEBEBEB, 0xFFB4B48D, 0xFF8D8D8D,
    0xFFB48D5D, 0xFF8DB48D, 0xFFD7D7B4, 0xFF6F6F6F, 0xFFB4B4B4
#endif
};
```

2.3 General software changes need for new emWin release

In addition to regenerating images some other problems might arise during the emWin update that will require changes to application code.

Change all GUI_SetColor() to use GUI_MAKE_COLOR()

In some cases there will be transparency related color mismatch issues in the GUI unless the color setting functions are updated to use the new format. Therefore FDI recommends to change ALL GUI colors in the application software to use the following format: **GUI_MAKE_COLOR(0xAARRGGBB)**. For previous 24-bit color assignments the alpha component should most likely be 0xFF to retain the same behavior as before.

Example: GUIDEMO Background color

To fix the background colors in the GUI demo the following had to be changed in GUIDEMO.h.

Old version:	
#define BK_COLOR_0	GUI_MAKE_COLOR(0xFF5555)
#define BK_COLOR_1	GUI_MAKE_COLOR(0x880000)
Updated version:	
#define BK_COLOR_0	GUI_MAKE_COLOR(0xFFFF5555)
#define BK_COLOR_1	GUI_MAKE_COLOR(0xFF880000)

Invalidation needed when changing some widgets

In some cases where a widget is updated the software will no longer automatically clear some UI elements from the buffer, such as the AXIS text on a graph. This can lead to seeing “double” where the old text and new text will be shown at the same time.

Example: Graphing Demo Axis problem

The graphing demo had a problem where the old axis would show up on the screen after the transition to the new graphs. The result was two Y axis labels overlayed on each other.

```
for (i = 0; i < GUI_COUNTOF(_aWave); i++) {
    GUIDEMO_SetInfoText(_aWave[i].pName);
    _DataAdjust = GUIDEMO_ShiftRight(Data_ySize * _aWave[i].DataVOff, GRAPH_DIV);
    GRAPH_SetGridOffY (hGraph, GUIDEMO_ShiftRight(Data_ySize * _aWave[i].GridVOff, GRAPH_DIV));
    GRAPH_SCALE_SetOff(_hScaleV, GUIDEMO_ShiftRight((Data_ySize - BORDER_BOTTOM) * _aWave[i].ScaleVOff,
    GRAPH_DIV));
    _ShowGraph(hGraph, hData, _aWave[i].NumWaves, _aWave[i].pfAddData);
    WM_InvalidateWindow(WM_HBKWIN); // make sure that we clear the old vertical scale numbers before drawing new ones
}
```

As shown above to fix this you must invalidate the part of the window where the axis would be visible. While the example above shows invalidating the whole window, in a real application it would be preferable to just invalidate the coordinates of the axis.

Memory device behavior changed

Any GUIs that use a memory device in emWin, similar to the emWin speedometer demo, will most likely need some changes made to the creation of the memory device. At this time FDI does not have an out of the box working example demo or setup example. But when running some older emWin memory device demos on emWin 5.48 it was observed that many of the colors in the application changed, even when a non-memory device demo was running.

Example: Antialiased text demo alpha problems

In this demo the intro text would still be present on the screen during the drawing of the demo. In addition the previous demo's text did not get cleared before starting the demo.

The solution was to make sure to enable alpha mode after the intro text had finished displaying.

Old version:

```
static void _DrawScreen(void) {
    GUI_RECT Rect; int    TitleSize; int    xSize; int    ySize; int    xOff; int    yOff;
    unsigned OldAlphaState = GUI_EnableAlpha(1);
    xSize = LCD_GetXSize();
    ySize = LCD_GetYSize();

    GUIDEMO_ConfigureDemo("Antialiased text", "Output antialiased text\nnon different backgrounds.",
    GUIDEMO_SHOW_CURSOR | GUIDEMO_SHOW_CONTROL);
    GUIDEMO_DrawBk();

    GUIDEMO_DispTitle("Antialiased text");
    TitleSize = GUIDEMO_GetTitleSizeY();
    xOff    = (xSize - XSIZE_MIN) / 2;
    yOff    = ((ySize - TitleSize) - (YSIZE_MIN - TitleSize)) / 2;

    // 4 bit anti-aliasing sample
    Rect.x0 = xOff;
    Rect.y0 = TitleSize + yOff;
    Rect.x1 = xSize - xOff;
    Rect.y1 = TitleSize + (ySize - TitleSize) / 2 - 1;
    _DrawSample(Rect, &GUI_FontAA4_32, "Antialiased text\n(4 bpp)");

    // 2 bit anti-aliasing sample
    Rect.y0 = Rect.y1 + 1;
    Rect.y1 = ySize - yOff;
    _DrawSample(Rect, &GUI_FontAA2_32, "Antialiased text\n(2 bpp)");
    GUIDEMO_Wait(4000);

    GUI_EnableAlpha(OldAlphaState); // set previous alpha state
}
```

Updated version:

```

static void _DrawScreen(void) {
    GUI_RECT Rect; int    TitleSize; int    xSize; int    ySize; int    xOff; int    yOff;
    unsigned OldAlphaState;
    xSize = LCD_GetXSize();
    ySize = LCD_GetYSize();
    GUIDEMO_ConfigureDemo("Antialiased text", "Output antialiased text\nnon different backgrounds.",
    GUIDEMO_SHOW_CURSOR | GUIDEMO_SHOW_CONTROL);
    GUIDEMO_DrawBk();
    OldAlphaState = GUI_EnableAlpha(1); // Don't enable this until after the demo title text was shown!

    GUIDEMO_DispTitle("Antialiased text");
    TitleSize = GUIDEMO_GetTitleSizeY();
    xOff    = (xSize - XSIZE_MIN) / 2;
    yOff    = ((ySize - TitleSize) - (YSIZE_MIN - TitleSize)) / 2;

    // 4 bit anti-aliasing sample
    Rect.x0 = xOff;
    Rect.y0 = TitleSize + yOff;
    Rect.x1 = xSize - xOff;
    Rect.y1 = TitleSize + (ySize - TitleSize) / 2 - 1;
    _DrawSample(Rect, &GUI_FontAA4_32, "Antialiased text\n(4 bpp)");

    // 2 bit anti-aliasing sample
    Rect.y0 = Rect.y1 + 1;
    Rect.y1 = ySize - yOff;
    _DrawSample(Rect, &GUI_FontAA2_32, "Antialiased text\n(2 bpp)");
    GUIDEMO_Wait(4000);

    GUI_EnableAlpha(OldAlphaState); // set previous alpha state
}

```

2.4 Verifying color changes in GUI

Once done updating the GUI to emWin 5.48 verify the colors against the previously recorded video to ensure correctness.

3. Excerpts from emWin documentation

uEZ now includes the matching emWin manual for the included version of emWin. Refer to the included UM03001_emWin_v5.48.pdf manual as a reference for this document and when doing general emWin development with uEZ on NXP. Note that SEGGER's website only lists the newest manual which does not match the NXP provided version. Some useful excerpts from the manual are included below:

3.1 Page 222-227 – Bitmap converter command line usage

Here are the following command line options of the bitmap converter:

-convertintobw	Converts image into BW
-convertintogray4	Converts image into Gray4
-convertintogray16	Converts image into Gray16
-convertintogray64	Converts image into Gray64
-convertintogray256	Converts image into Gray256
-convertinto111	Converts image into 111
-convertinto222	Converts image into 222
-convertinto233	Converts image into 233
-convertinto323	Converts image into 323
-convertinto332	Converts image into 332
-convertinto8666	Converts image into 8666
-convertintorgb	Converts image into RGB
-convertintobestpalette	Converts image into best palette
-convertintotranspalette	Converts image into best palette + transparency
-convertintocustompalette	<FNAME> Converts image into custom palette
-fliph	- FNAME: Name of palette file to be used
-flipv	Flips image horizontally
-hide	Flips image vertically
-rotate90ccw	Hides the application window
-rotate90ccw	Rotates image by 90 degrees CW
-rotate180	Rotates image by 90 degrees CCW
-invertindices	Rotates image by 180 degrees
-invertpalette	Inverts indices
-saveas	Inverts palette entries
	<FNAME> <TYPE> [<FORMAT> [<NOPALETTE>]]
	Save image file
	- FNAME: Filename to be used
	- TYPE: 1 - C-bitmap file 2 - Windows Bitmap file 3 - C-stream 4 - GIF file
	- FORMAT: [C or C-stream]
	1 - 1bpp 2 - 2bpp 4 - 4bpp 5 - 8bpp 6 - RLE4 compression 7 - RLE8 compression 8 - High color (565) 9 - High color (565), RB swap 10 - High color (555) 11 - High color (555), RB swap 12 - High color (565), compressed 13 - High color (565), RB swap, compr. 15 - True color with alpha, compr. 16 - True color with alpha 18 - Alpha channel, compr. 19 - 16bpp (444_12) 20 - 16bpp (444_12), RB swap 21 - 16bpp (444_12, 1) 22 - 16bpp (444_12, 1), RB swap 23 - 16bpp (444_16) 24 - 16bpp (444_16), RB swap 25 - High color with alpha (555) 26 - High color with alpha (555), RB swap 27 - High color with alpha (565) 28 - High color with alpha (565), RB swap 29 - True color with alpha, RB swap, alpha inverted 99 - True color with alpha, both variants
	- NOPALETTE: 0 - Save with palette (default) 1 - Save without palette
-scale	<WIDTH[%]> <HEIGHT[%]> Scale image to desired size
	- WIDTH/HEIGHT: Size in pixels to be used [%] Size in percent
-reducecolors	<NUM_COLORS> Reduce number of colors
	- NUM_COLORS: Number of colors to be used
-transparency	<COLOR> Sets image transparency
	- COLOR: Color to be used as transparent color
-exit	Terminates the Bitmap Converter
-help	Displays this box
-?	Displays this box

3.2 Page 318-319 – Logical Colors

15.2 Logical colors

A logical color contains 8 bits for each color component and 8 bits for the alpha channel. Since emWin V5.30 emWin supports 2 logical color formats:

Logical color format ABGR

Alpha	Blue	Green	Red
0x00 - opaque 0xFF - transparent			
DB31 - DB24	DB23 - DB16	DB15 - DB08	DB07 - DB00

For a long period of time the above format was the only supported logical color format. That implies that the used logical color format of all applications using emWin written within that period is also the same.

Logical color format ARGB(default)

Alpha	Red	Green	Blue
0x00 - transparent 0xFF - opaque			
DB31 - DB24	DB23 - DB16	DB15 - DB08	DB07 - DB00

Because of more and more hardware platforms using a slightly different pixel format we decided to add the option of using ARGB as logical color format to be able to improve performance significantly under certain circumstances. Please note that the alpha definition of the ARGB format is also different to the ABGR format.







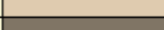



This format is the default setting since version 5.48 of emWin. To use the ABGR format add the following to your GUIConf.h:

```
#define GUI_USE_ARGB 0
```

3.3 Page 321 – Predefined colors

15.4 Predefined colors

In addition to self-defined colors, some standard colors are predefined in emWin, as shown in the following table:

GUI_BLUE		0xFF0000
GUI_GREEN		0x00FF00
GUI_RED		0x0000FF
GUI_CYAN		0xFFFF00
GUI_MAGENTA		0xFF00FF
GUI_YELLOW		0x00FFFF
GUI_LIGHTBLUE		0xFF8080
GUI_LIGHTGREEN		0x80FF80
GUI_LIGHTRED		0x8080FF
GUI_LIGHTCYAN		0xFFFF80
GUI_LIGHTMAGENTA		0xFF80FF
GUI_LIGHTYELLOW		0x80FFFF
GUI_DARKBLUE		0x800080
GUI_DARKGREEN		0x008000
GUI_DARKRED		0x000080
GUI_DARKCYAN		0x808000
GUI_DARKMAGENTA		0x800080
GUI_DARKYELLOW		0x008080
GUI_WHITE		0FFFFFFF
GUI_LIGHTGRAY		0xD3D3D3
GUI_GRAY		0x808080
GUI_DARKGRAY		0x404040
GUI_BLACK		0x000000
GUI_BROWN		0x2A2AA5
GUI_ORANGE		0x00A5FF

3.4 Page 319-320 – Switching to ARGB

15.3 Switching to ARGB

Using that logical color format could make sense if a display controller is used which supports exactly that color format as index value. In that case the performance could be improved significantly, for example when using an on chip LCD controller with hardware acceleration.

15.3.1 Configuration

In previous versions of emWin it was required to configure emWin to use the ARGB format. Since version 5.48 this format is the default and no changes to the GUIConf.h are required any longer.

Under certain circumstances it might be necessary to switch back to the 'old' configuration. If this is necessary just add the following to your GUIConf.h:

```
#define GUI_USE_ARGB 0
```

FDI does not recommend using the "GUI_USE_ARGB" option. In particular this option will not "fix" all GUI elements back to using the same color display on the newer release. In addition this option might disappear from a future release of emWin.

3.5 Page 320 – Required color changes in existing applications

15.3.2 Required changes in existing applications

When switching from ABGR to ARGB or vice versa some things have to be considered.

Colors

Wherever colors are defined as hexadecimal values in the application the values have to be changed or even better a conversion macro has to be used. The following table shows the use of the same color with ARGB, ABGR and conversion macro:

File	Description
ARGB	<code>GUI_SetColor(0xA02010);</code>
ABGR	<code>GUI_SetColor(0xFF1020A0);</code>
Conversion macro	<code>GUI_SetColor(GUI_MAKE_COLOR(0xFF1020A0));</code>

Table 15.1: Changes for color usage in existing applications

Of course all predefined color values will be changed automatically.

32 bpp Memory devices

emWin contains a couple of functions to be used with 32 bpp memory devices only. Those functions expect a determined memory device format. When working in ABGR mode that format is `GUICC_8888`. When switching to ARGB that format is `GUICC_M8888I`.

DIB bitmaps

Palette based bitmaps created by the bitmap converter contains an array of palette colors. Example:

```
static GUI_CONST_STORAGE GUI_COLOR _Colors8x1[] = {
    0x000000, 0xC04040, 0x40C020, 0xC0A000,
    0x4020E0, 0xC040A0, 0x00FFFF, 0xFFFFFFFF
};
```

All existing bitmaps need to be changed:

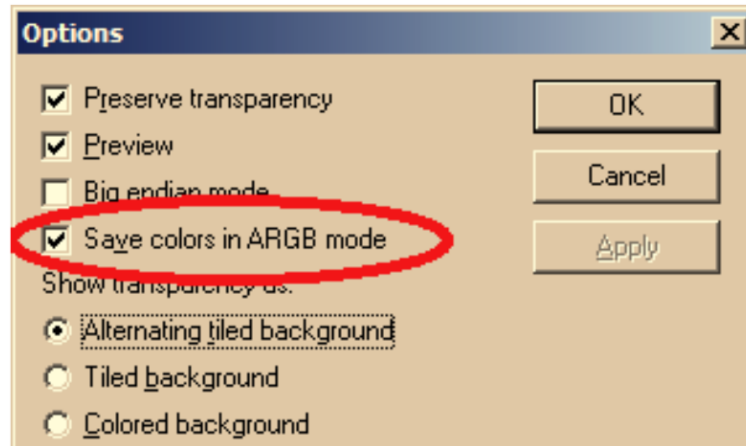
```
static GUI_CONST_STORAGE GUI_COLOR _Colors8x1[] = {
    0xFF000000, 0xFF4040C0, 0xFF20C040, 0xFF00A0C0,
    0xFFE02040, 0xFFA040C0, 0xFFFFF00, 0xFFFFFFFF
};
```

If an application contains a large number of bitmaps a better way would be to convert the bitmaps again with new settings for the bitmap converter.

3.6 Page 321 – Configuring Bitmap Converter for manual usage

15.3.3 Configuring the BitmapConverter

In order to configure the bitmap converter to save colors directly in ARGB instead of ABGR the option available under 'Options\Save colors in ARGB mode' should be activated:



4. Appendix

While uEZ and NXP's reference software is based around a customized NXP licensed version of emWin, the standard emWin webpages and support sections are a good reference for emWin. NXP occasionally also releases documentation with their emWin releases.

- uEZ/UM03001_emWin_v5.48.pdf
- <https://www.segger.com/products/user-interface/emwin/tools/tools-overview/>
- <https://www.segger.com/products/user-interface/emwin/#package-content>
- <https://www.segger.com/products/user-interface/emwin/>
- https://www.nxp.com/design/software/embedded-software/nxp-emwin-libraries:EMWIN-GRAPHICS-LIBRARY?tab=Design_Tools_Tab