Application Note

AN_303

FT800 Image File Conversion

Version 1.1

Issue Date: 2015-09-29

This document shows how to change a JPEG or PNG file into the correct format for the FT800 and how to include the edited file in your application.

Future Technology Devices International Limited (FTDI)
Unit 1, 2 Seaward Place, Glasgow G41 1HH, United Kingdom
Tel.: +44 (0) 141 429 2777 Fax: + 44 (0) 141 429 2758
Web Site: http://ftdichip.com

Table of Contents

1 Introduction

Image files which are regular baseline JPEG and non-progressive PNG (FT81X only) format can be loaded into the FT80x's 256kB or FT81x's 1MB graphics RAM by using the CMD_LOADIMAGE co-processor engine command. Alternatively, PNG and baseline JPEG files can be converted to the required format for direct writing into the FT8xx's graphics RAM by using the image conversion utility img_cvt. This utility can also produce compressed (zlib) versions of the image, which can then be processed by the FT8xx series' CMD_INFLATE command to copy the data (in the required format) to the graphics RAM. The utility pngp2pa can be used to create a palletized version of the image from a PNG8 format file. In this case an additional step is required to load the palette look-up table into the palette RAM. Once the image data is stored in graphics RAM (and the palette data, if applicable, is stored in palette RAM), display list or co-processor commands can be used to display and manipulate the image. Alternatively, the FT81x is able to decompress JPEG or PNG image data into an FT81x bitmap in FT81x's 1MB object RAM. PNG files can be converted to several image color formats such as L8, RGB565, PALETTED4444, PALETTED565, and PALETTED8. The utility png2dxt1 can be used to compress a 4-pixel aligned PNG image and save up to 75% of the object RAM space required, compared to other formats. This application note documents the use of the utilities img_cvt.exe, pngp2a.exe and png2dxt1.exe.

1.1 Scope

This document covers the basic file conversion process and how to use the resulting files in an application. Further details on the FT8xx Series are available in the FT8xx Series datasheets and the Programmers Guides.

1.2 Software Required

- img_cvt (EVE Image Converter)
- pngp2pa (PNG8 to Palette Converter)
- png2dxt1 (PNG to 'DXT1' Converter)

At the time of writing the latest version of img_cvt is V0.7, the latest version of pngp2pa is V0.4 and the latest version of png2dxt1 is V0.3.

The latest version of all utilities can be found on the FTDI website utilities page.

1.3 FT8xx Series Image Capability

The FT8xx Series can display the following image formats: ARGB1555, L1, L2 (FT81X only), L4, L8, RGB332, ARGB2, ARGB4, RGB565, PALETTED4444 (FT81x only), PALETTED565 (FT80x only), PALETTED8 (FT81x only) and PALETTED (FT80x only). See Table 1.1 for more information on these formats, Figure 1.1 and Figure 1.2 for examples of each.

Format	Attributes
ARGB1555	**1 bit for alpha (transparency) and 5 bits for each of the primary colours (red, green, blue) to give a total of 32768 colours. Transparency can be either fully on or fully off.**
L1	**1-bit monochrome**

Format	Attributes
L2	**2-bit grayscale (FT81x only)**
L4	**4-bit grayscale**
L8	**8-bit grayscale**
RGB332	**3 bits each for red and green, and 2 bits for blue to give a total of 256 colours.**
ARGB2	**2 bits each for alpha (transparency), red, green and blue to give a total of 64 colours.**
ARGB4	**4 bits each for alpha (transparency), red, green and blue to give a total of 4096 colours.**
RGB565	**5 bits for red, 6 bits for green and 5 bits for blue to give a total of 65536 colours.**
PALETTED4444	**2 bytes bitmap colours are stored in a palette table in RAM_G, 4 bits each for alpha (transparency), red, green and blue. (FT81x only)**
PALETTED565	**2 bytes bitmap colours are stored in a palette table in RAM_G, 5 bits for red, 6 bits for green and 5 bits for blue. (FT81x only)**
PALETTED8	**(FT81x only)**
PALETTED	**Bitmap colours are stored in a palette table in RAM_PAL (FT80x only)**

Table 1.1 FT8xx image formats and attributes

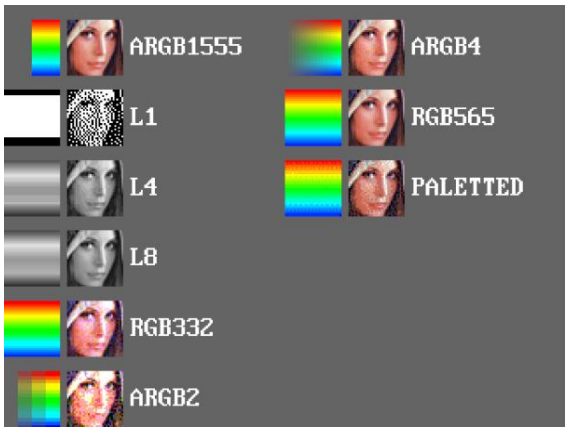

Figure 1.1 **FT80X supported image file formats.**

Figure 1.2 FT81X supported image file formats.

2 Working with the FT8xx image converter 'img_cvt.exe'

The FT8xx image conversion utility is img_cvt. This utility only works with JPEG and PNG image files and runs on the Windows operating system. Please note that it does not edit or resize the original image.

To use the image converter, download and store the file img_cvt.exe. Open a command prompt window and change the working directory to the folder containing the img_cvt.exe file. Copy the file to be converted to this folder. Run img_cvt.exe with the arguments shown below:

Command line format:

```
img_cvt –i  input_filename –f format <ret>
```

Output format options:

0 : ARGB1555 [default]

1 : L1

2 : L4

3 : L8

4 : RGB332

5 : ARGB2

6 : ARGB4

7 : RGB565

8 : PALETTEED [FT80X only]

9 : L2 [FT81X only]

Four files are created for each conversion.

For example, if ARGB1555 is the target format, then for the lenaface40.png file, the command is as follows:

```
img_cvt –i lenaface40.png –f 0 <ret>
```

The text 'image conversion utility for FT8xx Vx.x' will be displayed (based on version at time of writing), followed by 'convert complete!' if successful. A folder called **`lenaface40_argb1555`** will be created which contains the following files:

*.raw Binary format which can be loaded directly into the FT8xx graphics memory.

*.rawh Text representation of the binary file which can then be incorporated into a program and built into the final binary.

*.bin Compressed binary format (using the ZLIB algorithm) which can be loaded into the FT8xx object memory by using the function CMD_INFLATE (see the Programmers Guide for the FT80x or FT81x series).

*.binh Text representation of the compressed .bin file which can be incorporated into a program and built into the final binary. CMD_INFLATE is used to decompress.

*_Converted.png PNG format image file

3 Working with the FT8xx palette tool 'pngp2pa.exe'

The FT8xx palette tool is pngp2pa. This utility only works with image files which are PNG8 in format and runs on the Windows operating system. Please note that it does not edit or resize the original image.

To use the palette tool, download and store the file pngp2pa.exe. Open a command prompt window and change the working directory to the folder containing the pngp2pa.exe file. Copy the file to be converted to this folder. Run pngp2pa.exe with the arguments shown below:

```
pngp2pa –i  input_filename –o output_folder -f format <ret>
```

Supported output formats are:

0 : PALETTED [default, FT80x only]

1 : PALETTED8 [FT81x only]

2 : PALETTED565 [FT81x only]

3 : PALETTED4444 [FT81x only]

For example to convert the file PNG8sample.png use the following command:

```
pngp2pa –i  PNG8sample.png –o PNG8sample <ret> -f 0
```

The text 'PNG to Palette conversion utility for FT8xx V0x.x' will be displayed (x.x is the tool version), followed by 'convert complete!' if successful. A folder labelled **PNG8sample_palette_EVE** will be created which contains the following files:

*_index.raw Binary format which can be loaded directly into the FT8XX graphics memory.

*_index.rawh Text representation of the binary file which can then be incorporated into a program and built into the final binary.

*_index.bin Compressed binary format (using the ZLIB algorithm) which can be loaded into the FT8xx graphics memory by using the function CMD_INFLATE (see FT8xx Programmer Guide).

*_index.binh Text representation of the compressed .bin file which can be incorporated into a program and built into the final binary. CMD_INFLATE is used to decompress.

The **PNG8sample_palette_EVE** folder also contains the subfolder PNG8sample_palette_EVE_LUT which contains the palette look up tables for each file type:

*_lut.raw Binary format which can be loaded directly into the palette RAM for FT80x and into a look up table of graphics RAM for FT81x.

*_lut.rawh Text representation of the binary file which can then be incorporated into a program and built into the final binary.

*_lut.bin Compressed binary format (using the ZLIB algorithm) which can be loaded into the palette RAM for FT80X and into a look up table of graphics RAM of FT81x by using the function CMD_INFLATE (see FT800 Series Programmer Guide).

*_lut.binh Text representation of the compressed .bin file which can be incorporated into a program and built into the final binary. CMD_INFLATE is used to decompress.

The selected lut file needs to be loaded into the 1kB palette RAM (address range 10 2000h to 10 23FFh) for FT80X and into the graphics RAM for FT81x, either directly for the .raw and .rawh versions, or via the co-processor's CMD_INFLATE function for the _lut.bin and _lut.binh files.

4 Example Code

This example code is based on the FTDI HAL library and it shows the basic steps needed to use the output of the img_cvt converter and palette converter tools.

4.1 Use of .raw file

```
//Load lenaface40.raw into the FT800 graphics RAM
{
	ft_uint8_t imbuff[8192];
	ft_uint16_t filesize;
	ft_uint16_t blocklen;
	ft_uint16_t ram_start=0x00;

	chdir("..\\..\\..\\Test"); //change directory to location (Test) of .raw file
	pfile = fopen("lenaface40.raw","rb");// open file - mode read binary (rb)
	fseek(pfile,0,SEEK_END); //set file position to end of file
	filesize = ftell(pfile); // determine file size
	fseek(pfile,0,SEEK_SET); // return to beginning of file

	while(filesize > 0)
	{
		//copy the .raw file data to imbuff[8192] in 8k blocks
		blocklen = filesize>8192?8192:filesize;
		fread(imbuff,1,blocklen,pfile);
		filesize -= blocklen;
		//write imbuff contents to Graphics RAM at address ram_start = 0x00
		hal_spi_wr8s(phost, ram_start, imbuff, blocklen);
		ram_start = ram_start+8192; //increment ram_start for next 8k block
	}
	fclose(pfile); //close the opened .raw file
}

{
  //load the co-processor commands into a buffer and then write the commands
  //to the FT800 FIFO command buffer
  //clear screen to predefined values
  Ft_App_WrCoCmd_Buffer(phost,CLEAR(1,1,1));
  //start drawing bitmaps
  Ft_App_WrCoCmd_Buffer(phost,BEGIN(BITMAPS));
  //specify the starting address of the bitmap in graphics RAM
  Ft_App_WrCoCmd_Buffer(phost,BITMAP_SOURCE(0L));
  //specify the bitmap format, linestride and height
  Ft_App_WrCoCmd_Buffer(phost,BITMAP_LAYOUT(RGB565,40L*2,40));
  //set filtering, wrapping and on-screen size
  Ft_App_WrCoCmd_Buffer(phost,BITMAP_SIZE(NEAREST,BORDER,BORDER,40,40));
  //set top left corner to (220, 116)
  Ft_App_WrCoCmd_Buffer(phost,VERTEX2F(220*16, 116*16));
  //end the display list (all commands after this ignored)
  Ft_App_WrCoCmd_Buffer(phost,DISPLAY());
  //swap the current display list with the new display list
  Ft_Gpu_CoCmd_Swap(phost);
  //write to the FT800 FIFO command buffer - bitmap will appear after this command
  Ft_App_Flush_Co_Buffer(phost);
}
```

4.2 Use of .rawh file

```
//define structure for bitmap properties and copy .rawh data to the project
typedef struct SAMAPP_Bitmap_header
{
	ft_uint8_t Format;
	ft_int16_t Width;
	ft_int16_t Height;
	ft_int16_t Stride;
	ft_int32_t Arrayoffset;
}SAMAPP_Bitmap_header_t;

const SAMAPP_Bitmap_header_t SAMAPP_Bitmap_RawData_Header[] =
{
	/* format,width,height,stride,arrayoffset for lenaface40 */
	{RGB565, 40, 40, 40*2, 0},
};

//* raw data array – copy .rawh contents to the project */
const ft_uint8_t SAMAPP_Bitmap_RawData[] =
{	/*('file properties: ', 'resolution ', 40, 'x', 40, 'format ', 'RGB565',
'stride ', 80, ' total size ', 3200)*/
72,49,73,57,139,65,204,65,204,57,   //only first 10/3200 bytes shown.
…
};

{
  //Copy raw data array into Graphics RAM, starting location RAM_G
  hal_spi_wr8s(phost,RAM_G, &SAMAPP_Bitmap_RawData[p_bmhdr->Arrayoffset],
  p_bmhdr->Stride*p_bmhdr->Height);

  //load the co-processor commands into a buffer and then write the commands
  //to the FT800 FIFO command buffer

  //clear screen to predefined values
  Ft_App_WrCoCmd_Buffer(phost,CLEAR(1,1,1));
  //start drawing bitmaps
  Ft_App_WrCoCmd_Buffer(phost,BEGIN(BITMAPS));
  //specify the starting address of the bitmap in graphics RAM
  Ft_App_WrCoCmd_Buffer(phost, BITMAP_SOURCE(0L));
  //specify the bitmap format, linestride and height
  Ft_App_WrCoCmd_Buffer(phost,BITMAP_LAYOUT(RGB565,40L*2,40));
  //set filtering, wrapping and on-screen size
  Ft_App_WrCoCmd_Buffer(phost,BITMAP_SIZE(NEAREST,BORDER,BORDER,40,40));
  //set top left corner to (220, 116)
  Ft_App_WrCoCmd_Buffer(phost,VERTEX2F(220*16, 116*16));
  //end the display list (all commands after this ignored)
  Ft_App_WrCoCmd_Buffer(phost,DISPLAY());
  //swap the current display list with the new display list
  Ft_Gpu_CoCmd_Swap(phost);
  //write to the FT800 FIFO command buffer - bitmap will appear after this command
  Ft_App_Flush_Co_Buffer(phost);
}
```

4.3 Use of .bin file

```
//Load lenaface40.bin into Graphics RAM via co-processor

{
	ft_uint8_t imbuff[8192];
	ft_uint16_t blocklen;

	//decompress the .bin file using CMD_INFLATE
	Ft_Gpu_Hal_WrCmd32(phost,CMD_INFLATE);
	//specify starting address in graphics RAM
	Ft_Gpu_Hal_WrCmd32(phost,0L);
	//check filesize and adjust number of bytes to multiple of 4
	chdir("..\\..\\..\\Test"); //change directory to location (Test) of .bin file
	pfile = fopen("lenaface40.bin","rb");
	fseek(pfile,0,SEEK_END); //set file position to end of file
	filesize = ftell(pfile); // determine file size
	fseek(pfile,0,SEEK_SET); // return to beginning of file

	while(filesize > 0)
	{
		//copy the .raw file data to imbuff[8192] in 8k blocks
		 blocklen = filesize>8192?8192:filesize;
		 fread(imbuff,1,blocklen,pfile);

		 filesize -= blocklen;  //reduce filesize by blocklen
		 //write imbuff contents to the FT800 FIFO command buffer
		 Ft_Gpu_Hal_WrCmdBuf(phost,imbuff,blocklen);
	}

	fclose(pfile); /* close the opened .bin file */
}

{
  //clear screen to predefined values
  Ft_App_WrCoCmd_Buffer(phost,CLEAR(1,1,1));
  //start drawing bitmaps
  Ft_App_WrCoCmd_Buffer(phost,BEGIN(BITMAPS));
  //specify the starting address of the bitmap in graphics RAM
  Ft_App_WrCoCmd_Buffer(phost, BITMAP_SOURCE(0L));
  //specify the bitmap format, linestride and height
  Ft_App_WrCoCmd_Buffer(phost,BITMAP_LAYOUT(RGB565,40L*2,40));
  //set filtering, wrapping and on-screen size
  Ft_App_WrCoCmd_Buffer(phost,BITMAP_SIZE(NEAREST,BORDER,BORDER,40,40));
  //set top left corner to (220, 116)
  Ft_App_WrCoCmd_Buffer(phost,VERTEX2F(220*16, 116*16));
  //end the display list (all commands after this ignored)
  Ft_App_WrCoCmd_Buffer(phost,DISPLAY());
  //swap the current display list with the new display list
  Ft_Gpu_CoCmd_Swap(phost);
  //write to the FT800 FIFO command buffer - bitmap will appear after this command
  Ft_App_Flush_Co_Buffer(phost);
  }
```

4.4 Use of .binh file

```
//define structure for bitmap properties and copy .binh data to the project
typedef struct SAMAPP_Bitmap_header
{
	ft_uint8_t Format;
	ft_int16_t Width;
	ft_int16_t Height;
	ft_int16_t Stride;
	ft_int32_t Arrayoffset;
}SAMAPP_Bitmap_header_t;

const SAMAPP_Bitmap_header_t SAMAPP_Bitmap_RawData_Header[] =
{
	/* format,width,height,stride,arrayoffset for lenaface40 */
	{RGB565, 40, 40, 40*2, 0},
};

//* raw data array – copy .rawh contents to the project */
const ft_uint8_t SAMAPP_Bitmap_RawData[] =
{	/*('file properties: ', 'resolution ', 40, 'x', 40, 'format ', 'RGB565',
'stride ', 80, ' total size ', 2628)*/
120,156,85,150,109,80,19,119,30,199, //only first 10/2628 bytes shown

};


{
    //decompress the .binh contents using CMD_INFLATE
    Ft_Gpu_Hal_WrCmd32(phost,CMD_INFLATE);
    //specify starting address in graphics RAM
    Ft_Gpu_Hal_WrCmd32(phost,0L);
    //write the .binh contents to the FT800 FIFO command buffer, filesize=2628
    Ft_Gpu_Hal_WrCmdBuf(phost,&SAMAPP_Bitmap_RawData[0],filesize);
    //clear screen to predefined values
    Ft_App_WrCoCmd_Buffer(phost,CLEAR(1,1,1));
    //start drawing bitmaps
    Ft_App_WrCoCmd_Buffer(phost,BEGIN(BITMAPS));
    //specify the starting address of the bitmap in graphics RAM
    Ft_App_WrCoCmd_Buffer(phost, BITMAP_SOURCE(0L));
    //specify the bitmap format, linestride and height
    Ft_App_WrCoCmd_Buffer(phost,BITMAP_LAYOUT(RGB565,40L*2,40));
    //set filtering, wrapping and on-screen size
    Ft_App_WrCoCmd_Buffer(phost,BITMAP_SIZE(NEAREST,BORDER,BORDER,40,40));
    //set top left corner to (220, 116)
    Ft_App_WrCoCmd_Buffer(phost,VERTEX2F(220*16, 116*16));
    //end the display list (all commands after this ignored)
    Ft_App_WrCoCmd_Buffer(phost,DISPLAY());
    //swap the current display list with the new display list
    Ft_Gpu_CoCmd_Swap(phost);
    //write to the FT800 FIFO command buffer - bitmap will appear after this command
    Ft_App_Flush_Co_Buffer(phost);
}
```

4.5 Use of palette files

4.5.1 Use of palette files for FT80x

Load the _index.raw, _index.rawh, _index.bin or _index.binh file into graphics RAM using the methods shown in sections 4.1 to 4.4. The additional step with the palletized version is to also load the lut file into the palette RAM. The example below is for the .raw version.

```
//load PNG8_sample_index.raw into the FT800 graphics RAM &
//load PNG8_sample_lut.raw into the FT800 palette RAM
{
	ft_uint8_t imbuff[8192];
	ft_uint8_t palbuff[1024];
	ft_uint16_t filesize;
	ft_uint16_t blocklen;
	ft_uint32_t ram_start=0x00;
	ft_uint32_t ram_start_lut=0x102000;



	chdir("..\\..\\..\\Test"); //change directory to location (Test) of .raw file
	pfile = fopen("PNG8_sample_index.raw","rb");//open file: mode read binary (rb)
	fseek(pfile,0,SEEK_END); //set file position to end of file
	filesize = ftell(pfile); //determine file size
	fseek(pfile,0,SEEK_SET); //return to beginning of file

	while(filesize > 0)
	{
	//copy the _index.raw file data to imbuff[8192] in 8k blocks
	blocklen = filesize>8192?8192:filesize;
	fread(imbuff,1,blocklen,pfile);
	filesize -= blocklen;
	//write imbuff contents to Graphics RAM at address ram_start = 0x00
	hal_spi_wr8s(phost, ram_start, imbuff, blocklen);
	ram_start = ram_start+8192; //increment ram_start for next 8k block
	}
	fclose(pfile); //close the opened _index.raw file



	pfile = fopen("PNG8_sample_lut.raw","rb");// open file: mode read binary (rb)
	fseek(pfile,0,SEEK_END); //set file position to end of file
	filesize = ftell(pfile); // determine file size
	fseek(pfile,0,SEEK_SET); // return to beginning of file

	while(filesize > 0)
	{
	//copy the _lut.raw file data to palbuff[1024]
	blocklen = filesize>1024?1024:filesize;
	fread(palbuff,1,blocklen,pfile);
	filesize -= blocklen;
	//write palbuff contents to palette RAM at address ram_start_lut=0x102000
	hal_spi_wr8s(phost, ram_start_lut, palbuff, blocklen);
	}
	fclose(pfile);	//close the opened _lut.raw file


{
  //load the co-processor commands into a buffer and then write the commands
```

```
  //to the FT800 FIFO command buffer
  //
  //clear screen to predefined values
  Ft_App_WrCoCmd_Buffer(phost,CLEAR(1,1,1));
  //start drawing bitmaps
  Ft_App_WrCoCmd_Buffer(phost,BEGIN(BITMAPS));
  //specify the starting address of the bitmap in graphics RAM
  Ft_App_WrCoCmd_Buffer(phost,BITMAP_SOURCE(0L));
  //specify bit map format, linestride (= width for paletted) and height
  Ft_App_WrCoCmd_Buffer(phost,BITMAP_LAYOUT(PALETTED,200,200));
  //sets filtering, wrapping and on-screen size
  Ft_App_WrCoCmd_Buffer(phost,BITMAP_SIZE(NEAREST,BORDER,BORDER,200,200));
  //set top left corner to (220, 116)
  Ft_App_WrCoCmd_Buffer(phost,VERTEX2F(140*16, 36*16));
  //end the display list (all commands after this ignored)
  Ft_App_WrCoCmd_Buffer(phost,DISPLAY());
  //swap the current display list with new display list
  Ft_Gpu_CoCmd_Swap(phost);
  //write to the FT800 FIFO command buffer - bitmap will appear after this command
  Ft_App_Flush_Co_Buffer(phost);

}
```

4.5.2 Use of palette files for FT81x

```
//this function demonstrates the usage of the paletted bitmap converted by the FTDI
palette converter
ft_void_t SAMAPP_81X_Paletted_Bitmap() {
       ft_int32_t paletteTbSz = 0, paletteIdxSz = 0;
       ft_uint16_t bitmapHeight = 128, bitmapWidth = 128, bitmapStride = bitmapWidth;
       Ft_Gpu_CoCmd_Dlstart(phost);
       Ft_App_WrCoCmd_Buffer(phost, CLEAR(1, 1, 1));
       Ft_App_WrCoCmd_Buffer(phost, COLOR_RGB(255, 255, 255));
        paletteTbSz = SAMAPP_LoadRawFromFile("..\\..\\..\\Test\\Tomato_lut.raw",
RAM_G);
        paletteIdxSz = SAMAPP_LoadRawFromFile("..\\..\\..\\Test\\Tomato_index.raw",
1024);
        Ft_App_WrCoCmd_Buffer(phost, BEGIN(BITMAPS));
        Ft_App_WrCoCmd_Buffer(phost, PALETTE_SOURCE(RAM_G));
        Ft_App_WrCoCmd_Buffer(phost,BITMAP_SOURCE(1024));
Ft_App_WrCoCmd_Buffer(phost,BITMAP_LAYOUT(PALETTED4444,bitmapWidth,bitmapHeight));

Ft_App_WrCoCmd_Buffer(phost,BITMAP_SIZE(NEAREST,BORDER,BORDER,bitmapWidth,bitmapHeight
));
        Ft_App_WrCoCmd_Buffer(phost,VERTEX2F((FT_DispWidth/2 - bitmapWidth/2) * 16,
(FT_DispHeight/2 - bitmapHeight/2 - bitmapHeight) * 16));
       Ft_App_WrCoCmd_Buffer(phost,DISPLAY());
       Ft_Gpu_CoCmd_Swap(phost);
       Ft_App_Flush_Co_Buffer(phost);
       Ft_Gpu_Hal_WaitCmdfifo_empty(phost);
       SAMAPP_ENABLE_DELAY_VALUE(2000);
}
```

The following code shows a PALETTED8 example for the FT81x. PALETTED8 format is supported indirectly in the FT81x and it is different from the PALETTED format in the FT80x. To render Alpha, Red, Green and Blue channels, multi-pass drawing action is required.

```
//addr_pal is the starting address of palette lookup table in RAM_G
//bitmap source(palette indices) is starting from address 0

dl(BITMAP_HANDLE(0))
dl(BITMAP_LAYOUT(PALETTED8, width, height))
dl(BITMAP_SIZE(NEAREST, BORDER, BORDER, width, height))

dl(BITMAP_SOURCE(0)) //bitmap source(palette indices)

dl(BEGIN(BITMAPS))
dl(BLEND_FUNC(ONE, ZERO))

//Draw Alpha channel
dl(COLOR_MASK(0,0,0,1))
dl(PALETTE_SOURCE(addr_pal+3))
dl(VERTEX2II(0, 0, 0, 0))

//Draw Red channel
dl(BLEND_FUNC(DST_ALPHA, ONE_MINUS_DST_ALPHA))
dl(COLOR_MASK(1,0,0,0))
dl(PALETTE_SOURCE (addr_pal+2))
dl(VERTEX2II (0, 0, 0, 0))

//Draw Green channel
dl(COLOR_MASK(0,1,0,0))
dl(PALETTE_SOURCE(addr_pal + 1))
dl(VERTEX2II(0, 0, 0, 0))

//Draw Blue channel
dl(COLOR_MASK(0,0,1,0))
dl(PALETTE_SOURCE(addr_pal))
dl(VERTEX2II(0, 0, 0, 0))
```

5 Working with the FT8xx PNG to 'DXT1' converter 'png2dxt1.exe'

5.1 Introduction

DXT1 is an image compression algorithm which provides a significant saving in image file size by encoding a 4x4 block of pixels into 64 bits. For RGB565 images (16 bits per pixel) this yields a factor of four reductions in the file size (256 bits are compressed to 16 bits). A drawback of DXT1 is that it is a lossy technique and may therefore result in image quality degradation. The FT8xx cannot support DXT1 directly, but it is possible to emulate DXT1 and achieve a similar reduction in file size.

5.2 Benefit of using the PNG to 'DXT1' converter

The 'img_cvt.exe' utility previously discussed can be used to convert a PNG file for display by the FT8xx. If creating a background image of 480x272 pixels to use on a similar sized display, the RGB565 img_cvt option will result in a .raw file size of 256kB. If a smaller second image is now desired, for example 32x32 pixels, then this will convert to a .raw file size of 3kB, however, because the FT800 object RAM is only 256kB in size, it will not be possible to store (and display) both of these images at the same time. The 'png2dxt1.exe' conversion tool can convert the 480x272 background PNG to four '.raw' files, each of a size 16kB to give a total of 64kB to be stored in the object RAM. As a result sufficient object RAM space is now available to store the smaller image as well as additional images.

5.3 Using the PNG to 'DXT1' converter

This utility only works with image files which are in PNG format and both dimensions must be 4-pixel aligned. The utility runs on the Windows operating system.

To use the PNG to 'DXT1' converter tool, download the latest zip package from the FTDI utilities page. Unzip to a folder on the PC. The package contains three files:

png2dxt1.exe

squishpng.exe

libpng16.dll

Open a command prompt window and change the working directory to the folder containing the PNG to 'DXT1' files. Copy the file to be converted to this folder. Run png2dxt1.exe with the arguments shown below:

```
png2dxt1 -i  input_filename <ret>
```

For example to convert the file paris_480x272.png use the following command:

```
png2dxt1 -i  paris_480x272.png <ret>
```

Text similar to that below will be displayed:

```
Generating raw DXT1 file
Time taken: 0.14 seconds
Splitting DXT1 file into c0,c1,c0,c1 channels
!!Finished!!
```

If successful, four '.raw' files and 4 PNG files will be created:

```
paris_480x272_b0.png
paris_480x272_b0.raw
paris_480x272_b1.png
paris_480x272_b1.raw
paris_480x272_c0.png
paris_480x272_c0.raw
paris_480x272_c1.png
paris_480x272_c1.raw
```

The raw files can now be loaded directly into the FT8xx object RAM. The example code shown in section 5.4 below is then used to display the resulting image on the screen.

5.4 Example Code to display 'DXT1' image

To check the result on the display screen, the converted raw files need to be loaded into the FT8xx graphics RAM. This example code is based on the FTDI HAL library and shows the basic steps needed to use the output of the png2dxt1 converter.

```
ft_void_t SAMAPP_GPU_DXT1()

//c0 is paris_480x272_c0.raw
//c1 is paris_480x272_c1.raw
//b0 is paris_480x272_b0.raw
//b1 is paris_480x272_b1.raw
{
  //load each .raw file into graphics RAM from directory 'test'
  //RAM_G is starting address in graphics RAM, for example 00 0000h
  Ft_App_LoadRawFromFile("..\\..\\..\\Test\\c0.raw", RAM_G);
  Ft_App_LoadRawFromFile("..\\..\\..\\Test\\c1.raw", RAM_G + 16320);
  Ft_App_LoadRawFromFile("..\\..\\..\\Test\\b0.raw", RAM_G + 16320 * 2);
  Ft_App_LoadRawFromFile("..\\..\\..\\Test\\b1.raw", RAM_G + 16320 * 3);

  Ft_App_WrCoCmd_Buffer(phost, CLEAR(1, 1, 1));  // clear screen
  Ft_App_WrCoCmd_Buffer(phost,COLOR_RGB(255,255,255));

  Ft_App_WrCoCmd_Buffer(phost,SAVE_CONTEXT());

  //B0&B1 Handle
  Ft_App_WrCoCmd_Buffer(phost,BITMAP_HANDLE(0));
  Ft_App_WrCoCmd_Buffer(phost,BITMAP_SOURCE(RAM_G + 16320*2));
  Ft_App_WrCoCmd_Buffer(phost,BITMAP_LAYOUT(L1, 60, 272));
  Ft_App_WrCoCmd_Buffer(phost,BITMAP_SIZE(NEAREST, BORDER, BORDER, 480, 272));

  //C0&C1 handle
  Ft_App_WrCoCmd_Buffer(phost,BITMAP_HANDLE(2));
  Ft_App_WrCoCmd_Buffer(phost,BITMAP_SOURCE(RAM_G));
  Ft_App_WrCoCmd_Buffer(phost,BITMAP_LAYOUT(RGB565, 120*2, 68));
  Ft_App_WrCoCmd_Buffer(phost,BITMAP_SIZE(NEAREST, BORDER, BORDER, 480, 272));


  // start drawing bitmaps
  Ft_App_WrCoCmd_Buffer(phost,BEGIN(BITMAPS));

  Ft_App_WrCoCmd_Buffer(phost,BLEND_FUNC(ONE,ZERO));
  Ft_App_WrCoCmd_Buffer(phost,COLOR_A(0x55));
  Ft_App_WrCoCmd_Buffer(phost,VERTEX2II(0, 0, 0, 0));

  Ft_App_WrCoCmd_Buffer(phost,BLEND_FUNC(ONE,ONE));
  Ft_App_WrCoCmd_Buffer(phost,COLOR_A(0xAA));
  Ft_App_WrCoCmd_Buffer(phost,VERTEX2II(0, 0, 0, 1));

  Ft_App_WrCoCmd_Buffer(phost,COLOR_MASK(1,1,1,0));
  Ft_Gpu_CoCmd_Scale(phost, 4*65536,4*65536);
  Ft_Gpu_CoCmd_SetMatrix(phost);

  Ft_App_WrCoCmd_Buffer(phost,BLEND_FUNC(DST_ALPHA,ZERO));
  Ft_App_WrCoCmd_Buffer(phost,VERTEX2II(0, 0, 2, 1));

  Ft_App_WrCoCmd_Buffer(phost,BLEND_FUNC(ONE_MINUS_DST_ALPHA,ONE));
  Ft_App_WrCoCmd_Buffer(phost,VERTEX2II(0, 0, 2, 0));

  Ft_App_WrCoCmd_Buffer(phost, RESTORE_CONTEXT());
```

```
  Ft_App_WrCoCmd_Buffer(phost,END());
  Ft_App_WrCoCmd_Buffer(phost,DISPLAY());

  //swap the current display list with the new display list
  Ft_Gpu_CoCmd_Swap(phost);
  //write to the FT800 FIFO command buffer - bitmap will appear after this command
  Ft_App_Flush_Co_Buffer(phost);

}
```

6 Conclusion

The conversion procedures described in this application note can be used to create image files for display by the FT8xx Series. Non-progressive PNG and baseline JPG files can be converted into various formats for direct loading into the FT8xx graphics memory. The 'PNG to DXT1' tool significantly reduces the amount of graphics RAM needed to store a 4-pixel aligned image. Therefore, FT80x's 256KB graphics RAM can accommodate up to four 480x272 images, or a mixture of 480x272 and smaller images, and even more in the FT81x's 1MB graphics RAM.

7 Contact Information

Head Office – Glasgow, UK

Future Technology Devices International Limited
Unit 1, 2 Seaward Place, Centurion Business Park
Glasgow G41 1HH
United Kingdom
Tel: +44 (0) 141 429 2777
Fax: +44 (0) 141 429 2758

E-mail (Sales) sales1@ftdichip.com
E-mail (Support) support1@ftdichip.com
E-mail (General Enquiries) admin1@ftdichip.com

Branch Office – Tigard, Oregon, USA

Future Technology Devices International Limited
(USA)
7130 SW Fir Loop
Tigard, OR 97223-8160
USA
Tel: +1 (503) 547 0988
Fax: +1 (503) 547 0987

E-Mail (Sales) us.sales@ftdichip.com
E-Mail (Support) us.support@ftdichip.com
E-Mail (General Enquiries) us.admin@ftdichip.com

Branch Office – Taipei, Taiwan

Future Technology Devices International Limited
(Taiwan)
2F, No. 516, Sec. 1, NeiHu Road
Taipei 114
Taiwan , R.O.C.
Tel: +886 (0) 2 8791 3570
Fax: +886 (0) 2 8791 3576

E-mail (Sales) tw.sales1@ftdichip.com
E-mail (Support) tw.support1@ftdichip.com
E-mail (General Enquiries) tw.admin1@ftdichip.com

Branch Office – Shanghai, China

Future Technology Devices International Limited
(China)
Room 1103, No. 666 West Huaihai Road,
Shanghai, 200052
China
Tel: +86 21 62351596
Fax: +86 21 62351595

E-mail (Sales) cn.sales@ftdichip.com
E-mail (Support) cn.support@ftdichip.com
E-mail (General Enquiries) cn.admin@ftdichip.com

Web Site

http://ftdichip.com

Distributor and Sales Representatives

Please visit the Sales Network page of the FTDI Web site for the contact details of our distributor(s) and sales representative(s) in your country.

Appendix A – References

Document References

FT800 Programmer Guide

FT800 Datasheet

Acronyms and Abbreviations

Terms	Description
ARGB	Alpha Red Green Blue
RGB	Red Green Blue
DXT1	Compression algorithm which stores a block of 4x4 pixels as 64 bits

Appendix B – List of Tables & Figures

List of Tables

List of Figures

Appendix C – Revision History

Document Title: AN_303 FT800 Image File Conversion
Document Reference No.: FT_001070
Clearance No.: FTDI# 390
Product Page: http://www.ftdichip.com/FTProducts.htm
Document Feedback: Send Feedback

Revision	Changes	Date
1.0	Initial Release	2014-06-06
1.1	Add information for FT81X	2015-09-29