Application Note

AN_339

Using JPEGs with the FT800 series

Version 1.0

Issue Date: 2014-10-15

This document discusses JPEG files and how to check that they are compatible with the FT800 series. The methods that can be used to display a suitable JPEG image using the FT800 series are also discussed.

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

JPEG commonly refers to the popular image file format which reduces file size by compressing the raw data according to the specifications defined by the Joint Photographic Experts Group (JPEG). File extensions are usually .jpg or .jpeg. The FT800 series can process baseline JPEGs directly and display the images, provided that the amount of processed data is not larger than the capacity of the FT800 series's graphics RAM (256kB). Note that the JPEG specification refers to a range of compression options, not the file structure used to store the data, but the term JPEG is generally applied to both. One such file structure is the JPEG File Interchange Format (JFIF), first published in 1992. Another is Exif which extends the metadata available recorded for an image, for example to include camera type, shutter speed, etc.

1.1 Scope

This document covers the basics of JPEG file use with the FT800 series. It shows how to check the compatibility of a JPEG file and provides an example application, built using Visual Studio Express 2013, to quickly load and display valid JPEG files using the FT800 series. Further details on the FT800 series are available in the FT800 & FT801 datasheets and the FT800 Series Programmer Guide.

1.2 Software Required

Microsoft Visual Studio Express 2013 for Windows Desktop which can be downloaded from:
http://www.microsoft.com/en-gb/download/details.aspx?id=40787

FTDI D2XX driver which can be downloaded from:
http://www.ftdichip.com/Drivers/D2XX.htm

Sample application 'Single JPEG Viewer' which can be downloaded from:
http://www.ftdichip.com/Support/SoftwareExamples/EVE/FT800_Single_JPEG_Viewer.zip

1.3 Hardware Required

PC with Windows OS installed. Visual Studio Express 2013 supports Windows 7 Service Pack 1, Windows 8, Windows 8.1, Windows Server 2008 R2 SP1, Windows Server 2012 & Windows Server 2012 R2. Further PC hardware requirements are stated on the Visual Studio Express 2013 download page.

FT800 or FT801 development module with LCD panel: VM800B or VM800C, with either 4.3" or 5.0" display (WQVGA 480 x 272), or 3.5" display (QVGA).

FTDI MPSSE cable: C232HM-DDHSL-0.

2 JPEG Basics

2.1 Background

The origins of JPEG date back to the early 1980s when the International Standards Organization (ISO) set up the Photographic Experts Group (PEG) to investigate ways to transmit image and graphics data over digital networks. This group later combined with a CCITT subgroup researching a similar area to form the Joint Photographic Experts Group (JPEG). The JPEG standard first appeared in 1991, but this did not specify a file format for the interchange of JPEG data. The JPEG File Interchange Format (JFIF), Version 1.02, was published in September 1992 by Eric Hamilton, then of C-Cube Microsystems. The aim was to provide a means to readily exchange JPEG bitstreams between a range of platforms. The exchangeable image file format (Exif) was first released in 1998. For JPEGs, this adds further metadata, e.g. camera information, exposure time, etc., in comparison to JFIF. The JPEG specification does define a lossless compression technique, but typically JPEGs are lossy to take full advantage of their ability to greatly reduce file size whilst still maintaining an acceptable image quality. JPEG compression is particularly suited to photographic images. Note that the JPEG standard does not specify a single technique, but rather a range of compression options, however most JPEGs encountered will be 'baseline' format which uses Huffman encoding and 8-bit sample data.

2.2 Markers

A JPEG file is comprised of a series of segments. The beginning of a segment is designated by a 'marker' which consists of an 0xFF byte followed by a further byte (which is not equal to 0x00 or 0xFF) which defines the marker type. The first two bytes in a JPEG file are the 'Start of Image' (SOI) marker, 0xFFD8, and the last two bytes are the 'End of Image' (EOI) marker, 0xFFD9. The presence of the S0F0 marker indicates a baseline image. This marker also contains the image height and width. Table 1 below lists some common JPEG markers.

Marker	Abbreviation	Description
0xFFD8	**SOI**	**Start of Image**
0xFFFE	**COM**	**Comment**
0xFFDB	**DQT**	**Define Quantisation Table**
0xFFC0	**S0F0**	**Start of Baseline DCT Frame**
0xFFC2	**SOF2**	**Start of Progressive DCT Frame**
0xFFC4	**DHT**	**Define Huffman Table**
0XFFDA	**SOS**	**Start of Scan**
0xDDF9	**EOI**	**End of Image**

Table 1 Common JPEG markers

Some markers are 'stand-alone' (e.g. SOI) and another marker follows directly. Other markers (e.g. S0F0) contain a payload, the size of which is given by the first two bytes following the marker (the size reported includes the two 'size bytes', but not the two marker bytes) . Note that a JPEG file can contain more than one S0F0 (baseline) marker, for example, the file may also contain a thumbnail image which will also have an S0F0 marker. Each marker will contain height and width data for their respective images. Also, a JPEG file may contain both an S0F0 marker and an S0F2 marker, for example, a thumbnail may be stored as a baseline JPEG whilst the main image is progressive.

JPEG files can also contain up to sixteen 'application markers' (APP0 to APP15) which are designated by a 0xFF byte followed by a further byte which takes a value between 0xE0 and 0xEF. The next two bytes indicate the length of the segment corresponding to this marker (similar to the previous markers the length includes the two 'length bytes' themselves, but not the two marker bytes). Decoders can read or skip over these application markers as required.

JFIF format JPEG files include an application marker (APP0), equal to 0xFFE0, which is used to identify it as a JPEG JFIF file. The segment defined by this marker also includes the five byte JFIF identifier (0x4A 46 49 46 00). Exif JPEG files include the 'APP1 marker', equal to 0xFFE1. This segment includes an identifier (the six bytes 0x45 78 69 66 00 00) which can be used to determine if the format is Exif. Figure 2.1 shows an extract from the start of a JPEG (JFIF) file.

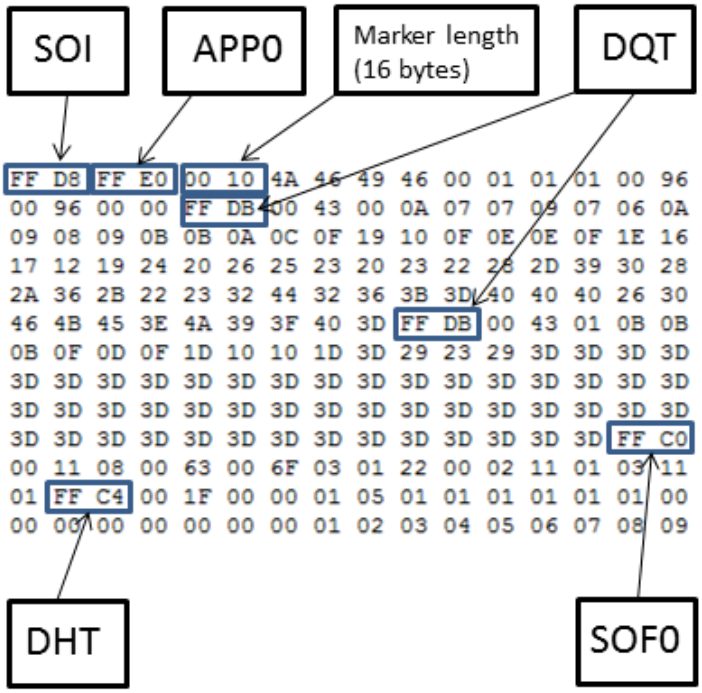

Figure 2.1 Extract from a JPEG (JFIF) file showing markers.

2.3 Baseline v Progressive

Baseline JPEGs are displayed line by line beginning from the top left of the image and finishing at the bottom right. The result is that only a partial image may be seen if downloading a baseline JPEG over a slow internet connection – see figure 2.2.

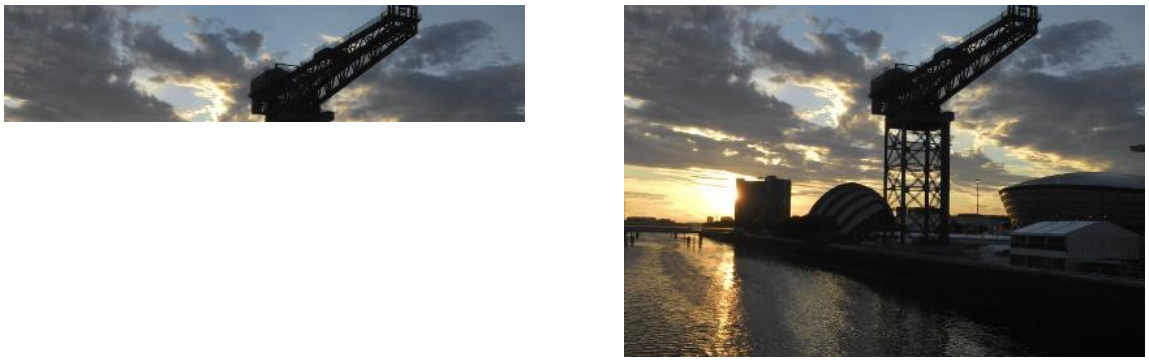

Figure 2.2 Baseline JPEG.

Progressive JPEGs are displayed in a series of scans with the picture detail increasing each time. The whole image, albeit blurred, appears first and then becomes clearer – see figure 2.3. In comparison to a baseline JPEG, it is possible to see the whole image earlier when downloading a progressive JPEG over a slow internet connection.

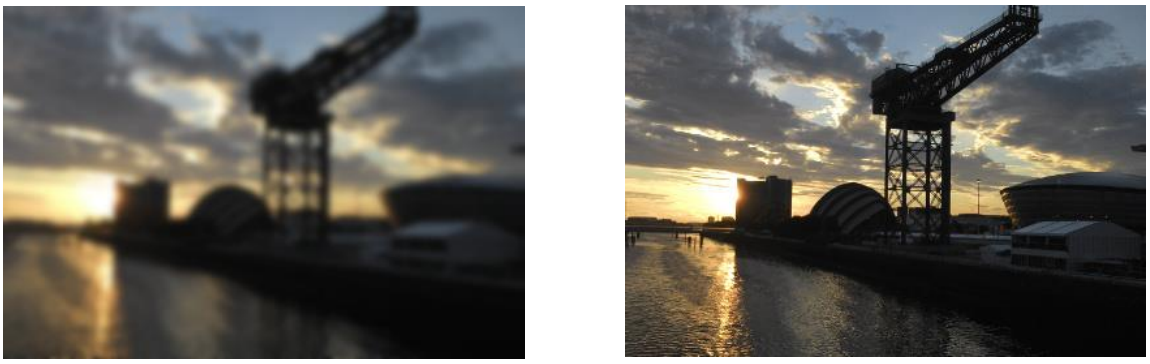

Figure 2.3 Progressive JPEG.

3 Using JPEGs with the FT800 series

3.1 Methods to display a suitable JPEG image

There are two methods to display a suitable JPEG image using the FT800 series. The first method is to use the image conversion utility (img_cvt) to transform the JPEG file into a format which can either be loaded directly (or via the coprocessor CMD_INFLATE in the case of the zlib compressed option) to the FT800 series graphics RAM. This utility allows the conversion format to be specified (e.g. RGB565), therefore this can be optimised to take into account the storage available in graphics RAM and the image quality required. Application note AN 303 has more details on using the img_cvt utility. The second method is to read the JPEG file into the FT800 using the co-processor function CMD_LOADIMAGE. This function will decode the JPEG file, produce either RGB565 or L8 bitmap data and store this in graphics RAM. Figure 3.1 outlines the two JPEG processing options for the FT800 series.

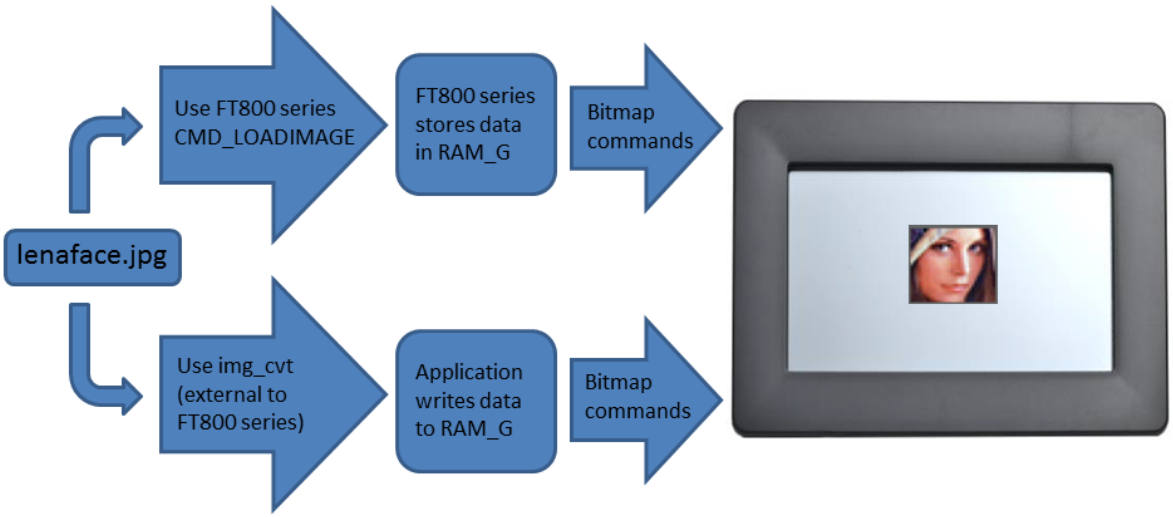

Figure 3.1 FT800 series JPEG processing options

3.2 CMD_LOADIMAGE

The destination starting address in graphics RAM for the converted bitmap and whether the image should be in RGB565 or L8 format is specified in CMD_LOADIMAGE. The JPEG file data stream should then follow directly into the command buffer. An example sequence using the FTDI Hardware Abstraction Language (HAL) is shown below:

```
Ft_Gpu_Hal_WrCmd32(phost, CMD_LOADIMAGE);//Load a JPEG image via command processor
Ft_Gpu_Hal_WrCmd32(phost, 0L);//Destination starting address in Graphics RAM
Ft_Gpu_Hal_WrCmd32(phost, 0);//Options 0 for RGB565, 1 for L8 monochrome,
//2 for RGB565 with no display list commands, 3 for L8 with no display list commands
Ft_Gpu_Hal_WrCmdBuf(phost, jpeg_data, number of bytes);//write data in command buffer
```

CMD_LOADIMAGE also writes commands to the display list to set the source, layout and size of the image (BITMAP_SOURCE, BITMAP_LAYOUT and BITMAP_SIZE). [Note that only a BEGIN and VERTEX2F (or VERTEX2II) display list commands are then required to complete the display list needed to render the image.] The parameter OPT_NODL can be set to suppress the creation of the display list commands by CMD_LOADIMAGE. To invoke this mode, set the options parameter to 2 for RGB565 and 3 for L8 format.

If the number of bytes in the JPEG file to be written to the command buffer is not a multiple of four, then one, two or three bytes (of any value) should be added to ensure four-byte alignment of the next command.

For more details on CMD_LOADIMAGE, please see the FT800 Series Programmer Guide.

3.3 JPEG constraints for the FT800 series

The FT800 series can only process baseline JPEG images. Progressive JPEG or JPEG 2000 images are not supported and if supplied to CMD_LOADIMAGE will cause the FT800 series co-processor engine to stop accepting new commands. For details on how to recover, please see section 5.5 of the FT800 series Programmer Guide. The processed data must not be larger than the FT800's graphic RAM size of 256kB.

4 Using JPEGs with the FT800 series

This sample code is based on the FTDI HAL and will display a suitable JPEG file saved in the folder 'test'. The image will be centred on the display.

Prior to reading the JPEG file into the FT800 using CMD_LOADIMAGE, the image height and width are extracted and checked that they are compatible with the display's dimensions. If the image is a progressive JPEG then it is not passed to the FT800 and an error message is reported.

The default display selection is for a WQVGA (480 x 272) panel, as supplied with the 4.3" and 5.0" basic and credit card modules. The 3.0" QVGA panel can be selected by choosing the appropriate #define in FT_Platform.h.

4.1 Using the Sample Code

Download the sample code from FT800_Single_JPEG_Viewer_1.0.zip and unzip. Copy the JPEG file for display to the folder 'Test'. If the folder contains more than one JPEG, then only the first JPEG (ordered by file name) will be used. Download the latest FTDI driver for Windows. Connect the MPSSE cable's fly leads to the FT800 series module using the connections in Table 1 below. (Note that the cable also supplies power to the module via the 3.3V pin — disconnect this pin if not required.). Now connect the MPSSE cable to the PC. Open the Visual Studio Solution (FT_single_jpg_viewer.sln) located in the directory Project/Msvc_win32/FT_single_jpg_viewer, build and then run the application. The selected JPEG should appear on the module's display. An error message will be reported on the FT800 series display if there are no files in 'Test', the file is not a JPEG, the file is not a baseline JPEG, or the image is the wrong dimensions for the screen.

MPSSE cable Signal Name	Colour
SCK	ORANGE
MOSI	YELLOW
MISO	GREEN
CS#	BROWN
INT#	PURPLE
PD#	BLUE
3.3V	RED
GND	BLACK

Table 2 MPSSE cable to FT800 series connections

4.2 Main Functions of Sample Code

```
//*******************************
  ft_uint16_t Ft_List_Out_files()
//*******************************
// Find *.jp* file & ensure file size is a multiple of 4 bytes
{
//ft_uint16_t nooffiles=0;
char *path = "..\\..\\..\\Test";  //Look for *.jp* files in Test directory
struct _finddata_t Files;
long file_spec;
if (!_chdir(path))
{
if ((file_spec = _findfirst("*.jp*", &Files)) == -1L)
{
return nooffiles;
}
else
{
ft_strcpy_P(Image_prop[nooffiles].name, Files.name);
//ensure file size is a multiple of 4 bytes
Image_prop[nooffiles].file_size = (Files.size + 3)&~3;
nooffiles++;
}
}
return nooffiles;
}


//*********************
  ft_void_t Load_JPEG()
//*********************

{
ft_uint8_t imbuff[8192];
ft_uint16_t blocklen;
ft_uint8_t error_code = 0;
ft_uint16_t xcoord;
ft_uint16_t ycoord;

ft_uint8_t markerID[2];
ft_uint8_t marker_info[8];
ft_uint8_t marker_next_byte[8];
ft_uint16_t size[8] = { 0 };

//read first 2 bytes to check for JPEG SOI marker 0xFFD8
fseek(pfile, 0, SEEK_SET);  // Beginning of file
fread(&markerID, 1, 2, pfile); //read first 2 bytes

if (((markerID[0] << 8) + markerID[1]) != 0xFFD8)
{
error_code = 1; //No SOI marker (0xFFD8) detected
}
else
{
while (fread(&markerID, 1, 1, pfile) > 0)
{
if (markerID[0] == 0xFF)
{
```

```
fread(&marker_next_byte, 1, 1, pfile);
if (marker_next_byte[0] >> 4 == 0xE){  //APPn marker found if true
fread(&marker_info, 1, 2, pfile); //read APPn marker length
fseek(pfile, ((marker_info[0]*256) + marker_info[1]-2), SEEK_CUR); //skip APPn marker
}
if (marker_next_byte[0] == 0xC0){  //baseline JPEG marker found
fread(&marker_info, 1, 8, pfile); //read next 8 bytes & extract height & width
image_height = marker_info[3]*256 + marker_info[4];
image_width = marker_info[5]*256 + marker_info[6];
printf("image_height = %d\n", image_height);
if (image_height > FT_DispHeight) error_code = 2; //Max is 272 for WQVGA, 240 for QVGA
printf("image_width = %d\n", image_width);
if (image_width > FT_DispWidth) error_code = 2;  //Max is 480 for WQVGA, 320 for QV
}
if (marker_next_byte[0] == 0xC2){  //check if progressive JPEG
error_code = 3;
}
}
}

}
if (error_code != 0) { //error flagged, display message
fclose(pfile);       /* close the opened JPEG file */
Ft_CmdBuffer_Index = 0;
Ft_Gpu_CoCmd_Dlstart(phost);
Ft_App_WrCoCmd_Buffer(phost, CLEAR(1, 1, 1));
Ft_App_WrCoCmd_Buffer(phost, COLOR_RGB(255, 255, 255));
if (error_code == 1) Ft_Gpu_CoCmd_Text(phost, FT_DispWidth / 2, FT_DispHeight / 2, 26,
OPT_CENTERX | OPT_CENTERY, "Error: Selected file is not a JPEG!");
if (error_code == 2) Ft_Gpu_CoCmd_Text(phost, FT_DispWidth / 2, FT_DispHeight / 2, 26,
OPT_CENTERX | OPT_CENTERY, "Error: Image size is not compatible!");
if (error_code == 3) Ft_Gpu_CoCmd_Text(phost, FT_DispWidth / 2, FT_DispHeight / 2, 26,
OPT_CENTERX | OPT_CENTERY, "Error: Selected file is a progressive JPEG!");
Ft_App_WrCoCmd_Buffer(phost, DISPLAY());
Ft_Gpu_CoCmd_Swap(phost);
Ft_App_Flush_Co_Buffer(phost);
Ft_Gpu_Hal_WaitCmdfifo_empty(phost);
}
else
{
//no error, therefore load image into FT800 via command processor
Ft_Gpu_Hal_WrCmd32(phost, CMD_LOADIMAGE); //Load a JPEG image via command processor
Ft_Gpu_Hal_WrCmd32(phost, 0L); //Destination starting address in Graphics RAM
Ft_Gpu_Hal_WrCmd32(phost, 0); //Options 0 for RGB565, 1 for L8 monochrome,
//2 for RGB565 with no display list commands, 3 for L8 with no display list commands

fseek(pfile, 0, SEEK_SET);
while (filesize > 0)
{
blocklen = filesize > 8192 ? 8192 : filesize;
/*copy the data into imbuff and then transfter it to command buffer*/
fread(imbuff, 1, blocklen, pfile);
filesize -= blocklen; //reduce filesize by blocklen
Ft_Gpu_Hal_WrCmdBuf(phost, imbuff, blocklen); //write JPEG file to command buffer
}

fclose(pfile);       /* close the opened JPEG file */
```

```
//centre image on display
xcoord = (FT_DispWidth/2) - (0.5*image_width);
ycoord = (FT_DispHeight/2) - (0.5*image_height);

Ft_CmdBuffer_Index = 0;
Ft_App_WrCoCmd_Buffer2(phost, CLEAR(1, 1, 1)); //clear screen to predefined values

/*Display list commands in this section are generated by CMD_LOADIMAGE unless option
//is set to 2 or 3
Ft_App_WrCoCmd_Buffer(phost, BITMAP_SOURCE(0L)); //specify the source address of
//bitmap in RAM_G
//specify bit map format, linestride and height for RGB565
Ft_App_WrCoCmd_Buffer(phost, BITMAP_LAYOUT(RGB565, image_width * 2, image_height));
//specify bit map format, linestride and height for L8
Ft_App_WrCoCmd_Buffer(phost, BITMAP_LAYOUT(L8, image_width * 1, image_height));
// controls drawing of bitmap
Ft_App_WrCoCmd_Buffer(phost, BITMAP_SIZE(NEAREST, BORDER, BORDER, image_width,
image_height));*/

Ft_App_WrCoCmd_Buffer(phost, BEGIN(BITMAPS));
Ft_App_WrCoCmd_Buffer(phost, VERTEX2II(xcoord, ycoord, 0, 0));
Ft_App_WrCoCmd_Buffer(phost, DISPLAY());//ends the display,commands after this ignored
Ft_Gpu_CoCmd_Swap(phost);
Ft_App_Flush_Co_Buffer(phost);
}
}
```

5 Conclusion

The FT800 series can work with compatible JPEG files directly, or via the image converter utility available from the FTDI website. In each case the JPEG data is converted into a bitmap which is stored in the graphics RAM of the FT800 series. This bitmap can then be displayed using either display list or co-processor commands.

6 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

Appendix A – References

Document References

FT800_Single_JPEG_Viewer.zip

FT800 Series Programmer Guide

FT800 Datasheet

FT801 Datasheet

img_cvt

C232HM-DDHSL-0 MPSSE Cable Datasheet

Latest FTDI drivers

AN 303

Acronyms and Abbreviations

Terms	Description
CCITT	The International Telegraph and Telephone Consultative Committee
Exif	Exchangeable image file format
ISO	International Standards Organization (now the International Organization for Standards)
JFIF	JPEG file interchange format
JPEG	Joint Photographic Experts Group
L8	8-bit grayscale image
MPSSE	Multi-Protocol Synchronous Serial Engine
QVGA	Quarter Video Graphics Array
RAM_G	FT800 series graphics RAM
RGB	Red Green Blue
RGB565	Bitmap with 5 bits for red, 6 bits for green and 5 bits for blue, for each pixel, to give a total of 65536 colours.
WQVGA	Wide Quarter Video Graphics Array

Appendix B – List of Tables & Figures

List of Tables

List of Figures

Appendix C – Revision History

Document Title: AN_339 Using JPEGs with the FT800 series
Document Reference No.: FT_001106
Clearance No.: FTDI#415
Product Page: http://www.ftdichip.com/FTProducts.htm
Document Feedback: Send Feedback

Revision	Changes	Date
1.0	Initial Release	2014-10-15