This document highlights the enhanced features in the FT81x Series of graphics controller ICs and the changes required to migrate from FT80x to FT81x.

Use of FTDI devices in life support and/or safety applications is entirely at the user's risk, and the user agrees to defend, indemnify and hold FTDI harmless from any and all damages, claims, suits or expense resulting from such use.
Table of Contents

1 Introduction ................................................................. 4
2 FT81x vs FT80x - Features Comparison ......................... 5
3 Application Migration from FT80x to FT81x ............ 6
   3.1 Memory map addresses for RAM/ROM/Registers .......... 6
   3.2 SPI packet structure ................................................. 8
   3.3 Display settings parameters ....................................... 8
   3.4 Default system clock ................................................ 8
   3.5 Extension instructions for GPU primitives ............... 8
   3.6 PALETTED8 Bitmap .................................................... 9
4 FTDI HAL based Migration - FT80x to FT81x .......... 11
5 Recommendations for better performance ............. 12
   5.1 Auxiliary registers for command FIFO ................. 12
   5.2 Media FIFO for image decompression .................... 12
   5.3 VERTEX_FORMAT for pixel / subpixel vertex instructions .13
   5.4 VERTEX_TRANSLATE_X and VERTEX_TRANSLATE_Y ....... 14
   5.5 CMD_SETFONT2 to facilitate easy calculation of address...15
   5.6 CMD_SETBITMAP co-processor command ................ 16
   5.7 DXT1 Usage ............................................................. 16
6 Contact Information ...................................................... 17
Appendix A – References ................................................. 18
   Document References .................................................. 18
   Acronyms and Abbreviations ........................................ 18
Appendix B – List of Tables & Figures ...................... 19
   List of Tables ............................................................ 19
   List of Figures ............................................................ 19
Appendix C – Revision History ............................................. 20
1 Introduction

Building on the success of the first generation of FT80x Embedded Video Engine (EVE) devices with integrated touch and audio functions, the series has now been extended to include a further four devices which offer a plethora of improvements and extended capabilities, the FT81x series. The FT810, FT811, FT812 and FT813 provide increased pixel resolution for sharper imagery, portrait orientation capabilities, increased speed for faster data transfer and image/video loading, and larger memory capacity.
2 FT81x vs FT80x - Features Comparison

This section lists some of the enhanced features available in the FT81x compared to the FT80x series.

- Maximum screen resolution 800x600 pixels (increased from 512 x 512 on FT80x)
- Colour signalling options of 18-bit (262K Colours) or 24-bit RGB (1.6 Million)
- Object memory 1Mbyte (increased from 256kByte on FT80x)
- MCU transfer with SPI or Dual SPI or QSPI interface option
- Available in QFN-48 / QFN-56 package (RoHS compliant)
- Extra-large ROM fonts added
- Orientation switching support added with REG ROTATE and CMD SETROTATE (allows Portrait operation)
- Master clock frequency improvement - REG PCLK can now be 1
- Smooth playback - motion JPG encoded AVI video playback
- Multiple 16/32 bit colour palettes supported with transparency
- L2 format supported for efficient DXT1-style bitmaps
- CMD MEDIAFIFO specifies an area of main memory to use as a FIFO for JPG, AVI and PNG loading
- Multiple numeric formats supported including binary, octal, decimal and hex (FT80x was decimal only)
- Simplified font loading with CMD SETFONT2
- 500-1000 times faster JPG loading
- Pixel clock drawing increased from 4 to 16 pixels per clock
- Quicker Firmware memory operations (e.g. copy, fill, CRC) due to tuned inner loops
- CMD SNAPSHOT hundreds of times faster than FT80x
- Auxiliary registers for cmd fifo for ease of command buffer management.
3 Application Migration from FT80x to FT81x

3.1 Memory map addresses for RAM/ROM/Registers

The register map has changed between the FT80x and FT81x to accommodate the extra object RAM and new registers. The Memory Map addresses for FT80x vs FT81x are given in Table 1 - FT81x Memory map

and Table 2 - FT80x Memory Map

For details of the full register map, refer to the programmers guide for the FT80x and FT81x families.

Header files containing the register address definitions can also be found within the project zip files for the sample projects. These can be included in other software projects to provide definitions for the register map. Please refer to the EVE2 compatible samples on the project samples web page, denoted by (+EVE2).


Note that the EVE2 compatible headers have both FT80x and FT81x definitions, with a #define used to select the required set.

<table>
<thead>
<tr>
<th>Start Address</th>
<th>End Address</th>
<th>Size</th>
<th>NAME</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>00 0000h</td>
<td>0F FFFFh</td>
<td>1024 kB</td>
<td>RAM_G</td>
<td>General purpose graphics RAM</td>
</tr>
<tr>
<td>0C0000h</td>
<td>0C0003h</td>
<td>4 B</td>
<td>ROM_CHIPID</td>
<td>FTDI chip identification and revision information:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0C0000h: 08h</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0C0001h: 10h, 11h, 12h (FT810), 13h (FT811),</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>12h (FT812), 13h (FT813)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0C0002h: 01h</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0C0003h: 00h</td>
</tr>
<tr>
<td>1E 0000h</td>
<td>2F FFBh</td>
<td>1152 kB</td>
<td>ROM_FONT</td>
<td>Font table and bitmap</td>
</tr>
<tr>
<td>2F FFFCh</td>
<td>2F FFFFh</td>
<td>4 B</td>
<td>ROM_FONT_ADDR</td>
<td>Font table pointer address</td>
</tr>
<tr>
<td>30 0000h</td>
<td>30 1FFFh</td>
<td>8 kB</td>
<td>RAM_DL</td>
<td>Display List RAM</td>
</tr>
<tr>
<td>30 2000h</td>
<td>30 2FFFh</td>
<td>4 kB</td>
<td>RAM_REG</td>
<td>Registers</td>
</tr>
<tr>
<td>30 8000h</td>
<td>30 8FFFh</td>
<td>4 kB</td>
<td>RAM_CMD</td>
<td>Co-processor command circular buffer</td>
</tr>
</tbody>
</table>

Notes:
The addresses beyond this table are reserved and shall not be read or written unless otherwise specified.

Palette now uses an area within RAM_G instead of the dedicated area in FT80x.

ROM_CHIPID is within the RAM_G memory. These are the reset values. The values can be overwritten and so it is recommended to read them before the application writes to these locations.

Table 1 - FT81x Memory map
<table>
<thead>
<tr>
<th>Start Address</th>
<th>End Address</th>
<th>Size</th>
<th>NAME</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>00 0000h</td>
<td>03 FFFFh</td>
<td>256 kB</td>
<td>RAM_G</td>
<td>Main graphics RAM</td>
</tr>
<tr>
<td>0C 0000h</td>
<td>0C 0003h</td>
<td>4 B</td>
<td>ROM_CHIPID</td>
<td>FT800/801 chip identification and revision information:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0C0000h: 08h 0C0001h: 00h (FT800) 01h (FT801)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0C0002h: 01h 0C0003h: 00h</td>
</tr>
<tr>
<td>0B B23Ch</td>
<td>0F FFFBh</td>
<td>275 kB</td>
<td>ROM_FONT</td>
<td>Font table and bitmap</td>
</tr>
<tr>
<td>0F FFFCh</td>
<td>0F FFFFh</td>
<td>4 B</td>
<td>ROM_FONT_ADDR</td>
<td>Font table pointer address</td>
</tr>
<tr>
<td>10 0000h</td>
<td>10 1FFFh</td>
<td>8 kB</td>
<td>RAM_DL</td>
<td>Display List RAM</td>
</tr>
<tr>
<td>10 2000h</td>
<td>10 23FFh</td>
<td>1 kB</td>
<td>RAM_PAL</td>
<td>Palette RAM</td>
</tr>
<tr>
<td>10 2400h</td>
<td>10 257Fh</td>
<td>380 B</td>
<td>REG_*</td>
<td>Registers</td>
</tr>
<tr>
<td>10 8000h</td>
<td>10 8FFFh</td>
<td>4 kB</td>
<td>RAM_CMD</td>
<td>Graphics Engine Command Buffer</td>
</tr>
<tr>
<td>1C 2000h</td>
<td>1C 27FFh</td>
<td>2 kB</td>
<td>RAM_SCREENSHOT</td>
<td>Screenshot readout buffer</td>
</tr>
</tbody>
</table>

Note: The addresses beyond this table are reserved and shall not be read or written unless otherwise specified.

Table 2 - FT80x Memory Map
3.2 SPI packet structure

Since the FT81x supports SPI packet structures for dual and quad SPI modes, an extra dummy byte needs to be added. Single channels for the FT81x are similar to the FT80x. For SPI memory read transactions, the host sends two zero bits, followed by the 22-bit address. This is followed by a dummy byte. After the dummy byte, the FT81x responds to each host byte with read data bytes.

3.3 Display settings parameters

Display scan-out parameters may need to be changed to meet the LCD panel requirements. Please refer to the guide below that explains the factors to consider when selecting a display panel and configuring the FT8xx registers accordingly.

Selecting an LCD Display

3.4 Default system clock

The default system clock for the FT80x is 48MHz, and for the FT81x it is 60MHz.

By default the FT81x system clock is 60MHz when the input clock is 12MHz. The host is allowed to switch the system clock to other frequencies (48MHz, 36MHz, and 24MHz) via the host command "CLKSEL". The clock switching command shall be sent in SLEEP mode only.

3.5 Extension instructions for GPU primitives

The maximum resolution for FT80x is 512 x 512 pixels, whereas for FT81x the maximum resolution is enhanced to 800 x 600 pixels. Additional registers with a _H notation accommodate the upper bits of the values (e.g. the higher bits of the stride / width / height).

Like the FT80x, the graphics engine in the FT81x takes the instructions from display list memory RAM_DL in the form of commands. Each command is 4 bytes long and one display list can be filled with up to 2048 commands as the size of RAM_DL is 8K bytes. The graphics engine performs the respective operation according to the definition of commands.

The following set of commands allows setting of the Graphics State –

BITMAP_SIZE_H

Specify the 2 most significant bits of bitmaps dimension for the current handle.

Encoding

<table>
<thead>
<tr>
<th>31</th>
<th>24</th>
<th>23</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x29</td>
<td>reserved</td>
<td>width</td>
<td>height</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Parameters

**Width:** 2 most significant bits of bitmap width. The initial value is zero.

**Height:** 2 most significant bits of bitmap height. The initial value is zero.

**Description**

This command is the extension command of BITMAP_SIZE for bitmap larger than 511 x 511 pixels.

**Graphics context**

None

### BITMAP_SIZE_H

Specify the 2 most significant bits of the source bitmap memory format and layout for the current handle.

**Encoding**

<table>
<thead>
<tr>
<th>31</th>
<th>24</th>
<th>23</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x28</td>
<td>reserved</td>
<td>linestride</td>
<td>height</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Parameters**

**Linestride:** The 2 most significant bits of the 12-bit linestride parameter value specified to BITMAP_LAYOUT.

**Height:** The 2 most significant bits of the 11-bit height parameter value specified to BITMAP_LAYOUT.

**Description**

This command is an extension of BITMAP_LAYOUT for large bitmaps. This command is not needed if the specified linestride parameter value to BITMAP_LAYOUT is less than 1024 and the height parameter value is less than 512.

### 3.6 PALETTED8 Bitmap

The FT80x supports Paletted bitmaps. The new FT81x uses the Paletted8 format instead, which requires some additional display list items in order to display the Paletted8 bitmap in colour. The format can be set as follows:

**BITMAP_LAYOUT**

**Encoding**

<table>
<thead>
<tr>
<th>31</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>9</th>
<th>8</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x07</td>
<td>format</td>
<td>linestride</td>
<td>height</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The table below shows the format options for the FT81x including PALETTED8.

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Bits/pixel</th>
<th>Alpha bits</th>
<th>Red bits</th>
<th>Green bits</th>
<th>Blue bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARGBl555</td>
<td>0</td>
<td>16</td>
<td>1</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>L1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>L4</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>L8</td>
<td>3</td>
<td>8</td>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>RGB332</td>
<td>4</td>
<td>8</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>ARGB2</td>
<td>5</td>
<td>8</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>ARGB4</td>
<td>6</td>
<td>16</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>RGB565</td>
<td>7</td>
<td>16</td>
<td>0</td>
<td>5</td>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td>TEXT8X8</td>
<td>9</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>TEXTVGA</td>
<td>10</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>BARGRAPH</td>
<td>11</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>PALETTED565</td>
<td>14</td>
<td>8</td>
<td>0</td>
<td>5</td>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td>PALETTED4444</td>
<td>15</td>
<td>8</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>PALETTED8</td>
<td>16</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>L2</td>
<td>17</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**Table 3 - BITMAP_LAYOUT Format List**

An example of the PALETTED8 format is shown below.

Figure 1 - PALLETTED8 Format
4 FTDI HAL based Migration - FT80x to FT81x

The migration recommendations mentioned in Section 3 are intended for users who are developing their own source files.

The FTDI website has a range of examples which have been updated to support the FT81x in addition to the original support for FT80x. Please refer to the samples with the (+EVE2) note.


These examples use the #defines FT_80X_ENABLE and FT_81X_ENABLE in order to support both families in the same source files. These defines are used throughout the main application code and supporting c and header files.

To simplify the selection, these defines can be controlled from a single file called platform.h which is present in each of the demo code projects. This file contains a high-level set of defines to select the type of demo board, screen size and host platform. These in turn enable the associated defines throughout the rest of the code.

Please refer to the sample projects from the link above and also the accompanying platform guide for further details.

EVE Platform Guide
5 Recommendations for better performance

The following sections show some of the ways in which the FT81x’s features can be used to improve performance and/or reduce workload on the host MCU.

5.1 Auxiliary registers for command FIFO

To offload work from the MCU for checking the free space in the circular buffer, the FT81x offers two auxiliary registers "REG_CMDB_SPACE" and "REG_CMDB_WRITE" for bulk transfers. It enables the MCU to write commands and data to the co-processor in a bulk transfer, without computing the free space in the circular buffer and increasing the address. As long as the amount of data to be transferred is less than the value in the register “REG_CMDB_SPACE”, the MCU is able to safely write all the data to “REG_CMDB_WRITE” in one write transfer.

5.2 Media FIFO for image decompression

The CMD.MEDIAFIFO command is used to set up a streaming media FIFO in RAM_G.

C prototype

```c
void cmd_mediafifo ( uint32_t ptr, 
                   uint32_t size );
```

Parameters

- **Ptr**: starting address of memory block
- **size**: number of bytes in the source memory block

Command layout

<table>
<thead>
<tr>
<th>Offset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>+0</td>
<td>CMD.MEDIAFIFO (0xffffffff)</td>
</tr>
<tr>
<td>+4</td>
<td>Ptr</td>
</tr>
<tr>
<td>+8</td>
<td>Size</td>
</tr>
</tbody>
</table>

Examples

To set up a 64-Kbyte FIFO at the top of RAM_G for JPEG streaming, and report the initial values of the read and write pointers:

```c
cmd_mediafifo(0x100000 - 65536, 65536); //0x100000 is the top of RAM_G
printf("R=%08xW=%08x\n", rd32(REG.MEDIAFIFO_READ), rd32(REG.MEDIAFIFO_WRITE));
```

prints:

```
000f000 00f000
```
5.3 VERTEX_FORMAT for pixel / subpixel vertex instructions

The FT81x has support for configuring the vertex format. The purpose of using vertex_format is to set the precision of VERTEX2F coordinates. Precisions of 1, 1/2, 1/4, 1/8 and 1/16 can be selected.

Encoding

<table>
<thead>
<tr>
<th>Encoding</th>
<th>Vertex Format</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>31 30 29 28 27 26 25 24</td>
<td>23 22 21 20 19 18 17 16 15 13 12 11 10 9 8</td>
<td>2 1 0</td>
</tr>
<tr>
<td>0x27</td>
<td>RESERVED</td>
<td>Frac</td>
</tr>
</tbody>
</table>

Parameters

- **Frac**: Number of fractional bits in X, Y coordinates. Valid range is from 0 to 4. The initial value is 4.

Description

VERTEX2F uses 15 bit signed numbers for its (X, Y) coordinates. This command controls the interpretation of these numbers by specifying the number of fractional bits.

By varying the format, an application can trade range against precision.

<table>
<thead>
<tr>
<th>frac</th>
<th>Units in pixel precision</th>
<th>VERTEX2F range in pixels</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>-16384 to 16383</td>
</tr>
<tr>
<td>1</td>
<td>1/2</td>
<td>-8192 to 8191</td>
</tr>
<tr>
<td>2</td>
<td>1/4</td>
<td>-4096 to 4095</td>
</tr>
<tr>
<td>3</td>
<td>1/8</td>
<td>-2048 to 2047</td>
</tr>
<tr>
<td>4</td>
<td>1/16</td>
<td>-1024 to 1023</td>
</tr>
</tbody>
</table>

Table 4 - VERTEX_FORMAT and pixel precision

Graphics context

The value of frac is part of the graphics context.
5.4 VERTEX_TRANSLATE_X and VERTEX_TRANSLATE_Y

The VERTEX_TRANSLATE commands are added to ease the movement of a set of objects. These commands allow translation of whole sets of primitives/widgets.

The VERTEX_TRANSLATE_X is used to specify the vertex transformations X translation. The command is structured as follows:

**Encoding**

<table>
<thead>
<tr>
<th>31 30 29 28 27 26 25 24</th>
<th>23 22 21 20 19 18 17</th>
<th>16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x2B</td>
<td>RESERVED</td>
<td>x</td>
</tr>
</tbody>
</table>

**Parameters**

X: signed x-coordinate in 1/16 pixel. The initial value is 0

**Description**

Specifies the offset added to vertex X coordinates. This command allows drawing to be shifted on the screen.

**Graphics context**

The value of x is part of the graphics context.

The VERTEX_TRANSLATE_Y is used to specify the vertex transformation’s Y translation. The command is structured as follows:

**Encoding**

<table>
<thead>
<tr>
<th>31 30 29 28 27 26 25 24</th>
<th>23 22 21 20 19 18 17</th>
<th>16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x2C</td>
<td>RESERVED</td>
<td>y</td>
</tr>
</tbody>
</table>

**Parameters**

Y: signed y-coordinate in 1/16 pixel. The initial value is 0

**Description**

Specifies the offset added to vertex Y coordinates. This command allows drawing to be shifted on the screen.

**Graphics context**

The value of y is part of the graphics context.
5.5 CMD_SETFONT2 to facilitate easy calculation of address

In FT80x, the CMD_SETFONT was used to register one custom defined bitmap font into the co-processor engine. In FT81x, CMD_SETFONT2, a new co-processor command, is added to setup a custom font. To use a custom font with the co-processor objects, create the font definition in RAM_G and issue CMD_SETFONT2, as described in section 5.5 of the FT81x Programmers Guide.

C prototype

```c
void cmd_setfont2( uint32_t font,
                    uint32_t ptr,
                    uint32_t firstchar );
```

Command layout

<table>
<thead>
<tr>
<th>Offset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>+0</td>
<td>CMD_SETFONT2(0xffffff3b)</td>
</tr>
<tr>
<td>+4</td>
<td>Font</td>
</tr>
<tr>
<td>+8</td>
<td>Ptr</td>
</tr>
<tr>
<td>+12</td>
<td>firstchar</td>
</tr>
</tbody>
</table>

Parameters

- **Font**: The bitmap handle from 0 to 31
- **Ptr**: 32 bit aligned memory address in RAM_G of font metrics block
- **Firstchar**: The ASCII value of the first character in the font.

Examples

With a suitable font metrics block loaded in RAM_G at address 100000, first character’s ASCII value 32, to use it for font 20:

```c
cmd_setfont2(20, 100000, 32);
cmd_button(15, 30, 130, 20, 18, 0, "This is font 18");
cmd_button(15, 60, 130, 20, 20, 0, "This is font 20");
```

**Table 5 - CMD_SETFONT2 Example**
5.6 CMD_SETBITMAP co-processor command

This command facilitates easy construction of a bitmap using a set of GPU instructions. It will generate the corresponding display list commands (BITMAP_SOURCE \ BITMAP_LAYOUT \ BITMAP_SIZE) for a given bitmap information, sparing the effort of writing the display list manually.

The parameters filter / wrapx / wrapy in BITAMP_SIZE is always set to NEAREST / BORDER / BORDER value in the generated display list commands.

**C prototype**

```c
void cmd_setbitmap( uint32_t   addr,
                    uint16_t   fmt,
                    uint16_t   width,
                    uint16_t   height );
```

**Parameters**

- **Addr**: Address of bitmap data in RAM_G.
- **Fmt**: Bitmap format, see the definition in BITMAP_LAYOUT.
- **Width**: bitmap width, in pixels.
- **Height**: bitmap height, in pixels

**Command layout**

<table>
<thead>
<tr>
<th>Offset</th>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>+0</td>
<td>CMD_SETBITMAP(0xffff ff43)</td>
<td></td>
</tr>
<tr>
<td>+4</td>
<td>Addr</td>
<td>Address of bitmap data in RAM_G</td>
</tr>
<tr>
<td>+8</td>
<td>Fmt</td>
<td>Bitmap format, see the definition in BITMAP_LAYOUT</td>
</tr>
<tr>
<td>+10</td>
<td>Width</td>
<td>bitmap width, in pixels</td>
</tr>
<tr>
<td>+12</td>
<td>Height</td>
<td>bitmap height, in pixels</td>
</tr>
</tbody>
</table>

**Examples**

NA

**Note**: Two bytes need to be appended after last parameter to provide 4 bytes alignment as required by the co-processor

5.7 DXT1 Usage

The FT81x series provides support for the L2 bitmap format. If an FT80x application currently uses DXT1, it could use the L2 bitmap format feature instead of the two L1 bitmaps supported in the FT80x series.
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

**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.

System and equipment manufacturers and designers are responsible to ensure that their systems, and any Future Technology Devices International Ltd (FTDI) devices incorporated in their systems, meet all applicable safety, regulatory and system-level performance requirements. All application-related information in this document (including application descriptions, suggested FTDI devices and other materials) is provided for reference only. While FTDI has taken care to assure it is accurate, this information is subject to customer confirmation, and FTDI disclaims all liability for system designs and for any applications assistance provided by FTDI. Use of FTDI devices in life support and/or safety applications is entirely at the user's risk, and the user agrees to defend, indemnify and hold harmless FTDI from any and all damages, claims, suits or expense resulting from such use. This document is subject to change without notice. No freedom to use patents or other intellectual property rights is implied by the publication of this document. Neither the whole nor any part of the information contained in, or the product described in this document, may be adapted or reproduced in any material or electronic form without the prior written consent of the copyright holder. Future Technology Devices International Ltd, Unit 1, 2 Seaward Place, Centurion Business Park, Glasgow G41 1HH, United Kingdom. Scotland Registered Company Number: SC136640
## Appendix A – References

### Document References
- FT81x Series Programmer's Guide
- FT800 Series Programmer's Guide
- FT81x EVE Datasheet
- FT800 EVE Datasheet

### Acronyms and Abbreviations

<table>
<thead>
<tr>
<th>Terms</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>AVI</td>
<td>Audio Video Interleave</td>
</tr>
<tr>
<td>FIFO</td>
<td>First In First Out</td>
</tr>
<tr>
<td>GPU</td>
<td>Graphics Processor Unit</td>
</tr>
<tr>
<td>HSYNC</td>
<td>Horizontal Synchronization</td>
</tr>
<tr>
<td>IO</td>
<td>Input / Output</td>
</tr>
<tr>
<td>JPG</td>
<td>Joint Photographic Group</td>
</tr>
<tr>
<td>LCD</td>
<td>Liquid Crystal Display</td>
</tr>
<tr>
<td>MCU</td>
<td>Micro Controller Unit</td>
</tr>
<tr>
<td>MPU</td>
<td>Memory Protection Unit</td>
</tr>
<tr>
<td>PNG</td>
<td>Portable Network Graphics</td>
</tr>
<tr>
<td>QFN</td>
<td>Quad Flat No leads</td>
</tr>
<tr>
<td>QSPI</td>
<td>Quad Serial Peripheral Interface</td>
</tr>
<tr>
<td>RAM</td>
<td>Random Access Memory</td>
</tr>
<tr>
<td>RoHS</td>
<td>Restriction of Hazardous Substances Directive</td>
</tr>
<tr>
<td>ROM</td>
<td>Read only Memory</td>
</tr>
<tr>
<td>VSYNC</td>
<td>Vertical Synchronization</td>
</tr>
</tbody>
</table>
Appendix B – List of Tables & Figures

List of Tables
Table 1 - FT81x Memory map .......................................................... 6
Table 2 - FT80x Memory Map .......................................................... 7
Table 3 - BITMAP_LAYOUT Format List ........................................... 10
Table 6 - VERTEX_FORMAT and pixel precision ............................. 13
Table 5 - CMD_SETFONT2 Example ................................................ 15

List of Figures
Figure 1 - PALLETED8 Format ......................................................... 10
## Appendix C – Revision History

<table>
<thead>
<tr>
<th>Revision</th>
<th>Changes</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>Initial Release</td>
<td>2015-09-30</td>
</tr>
</tbody>
</table>

Document Title: AN_390 FT80x to FT81x Migration Guide
Document Reference No.: FT_001246
Clearance No.: FTDI#471
Document Feedback: [Send Feedback](mailto:ftdichip@ftdichip.com)