APPLICATION NOTE

AN_270

FT_APP_SKETCH

Version 1.2

Document Reference No.: FT_000915

Issue Date: 2014-06-30

This document is to introduce the Sketch Demo Application. The objective of the Demo Application is to enable users to become familiar with the usage of the FT800, the design flow, and display list used to design the desired user interface or visual effect.

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

This design example demonstrates an interactive user interface that provides a drawing, or sketch, area. The touch screen is used to capture user writing inside of the sketch area. The touch events are then drawn within the area interactively. While the sketch area is active, a slider bar is available to change the background colour of the drawing area. A touch button that clears the sketch area is the final element on the screen

1.1 Overview

The document will provide information on drawing graphics elements through primitives, tagging of touch capabilities and the structure of display lists. In addition, this application note outlines the general steps of the system design flow, display list creation and integrating the display list with the system host microcontroller.

This application note should be read in conjunction with the source code provided in section 4 or at:

http://www.ftdichip.com/Support/SoftwareExamples/FT800_Projects.htm

1.2 Scope

This document can be used as a guide by designers to develop GUI applications by using FT80x with any MCU via SPI or I^2C. Note that detailed documentation is available on www.ftdichip.com/EVE.htm, including:

- FT800 datasheet
- FT801 datasheet
- Programming Guide covering EVE command language
- AN_240 FT800 From the Ground Up
- AN_245 VM800CB_SampleApp_PC_Introduction - covering detailed design flow with a PC and USB to SPI bridge cable
- AN_246 VM800CB_SampleApp_Arduino_Introduction – covering detailed design flow in an Arduino platform

2 Display Requirements

This section describes some of the key components of the design.

2.1 Sketch Area

The Sketch area uses the display and touch screen. As the touch screen is written upon, the corresponding pixels in the sketch area are rendered out to simulate drawing on a page. The sketch area only covers most of the screen. Touch events outside of this area are ignored, except for the button and slider features.

2.2 Slider

A second touch area uses a FT800 slider widget. This allows a user to change the background colour of what is drawn in the sketch area. It also changes the colour of the slider itself to match.

2.3 Button

A third and final touch area of the screen is assigned to a button widget with the text "CLR". Tapping this button will generate a touch event called a tag. Upon sensing the tag, the sketch area of the screen is cleared, ready for more drawing.

3 Design Flow

Every EVE design follows the same basic principles as highlighted in Figure 3.1.

Select and configure your host port for controlling the FT800 then wake the device before configuring the display. The creative part then revolves around the generation of the display list, "** APPLICATION DATA **" in the figure below. There will be two lists. The active list and the updated/next list are continually swapped to render the display. Note, header files map the pseudo code of the design file of the display list to the FT800 instruction set, which is sent as the data of the SPI (or I²C) packet (typically <1KB). As a result, with EVE's object oriented approach, the FT800 is operating as an SPI peripheral while providing full display, audio, and touch capabilities.

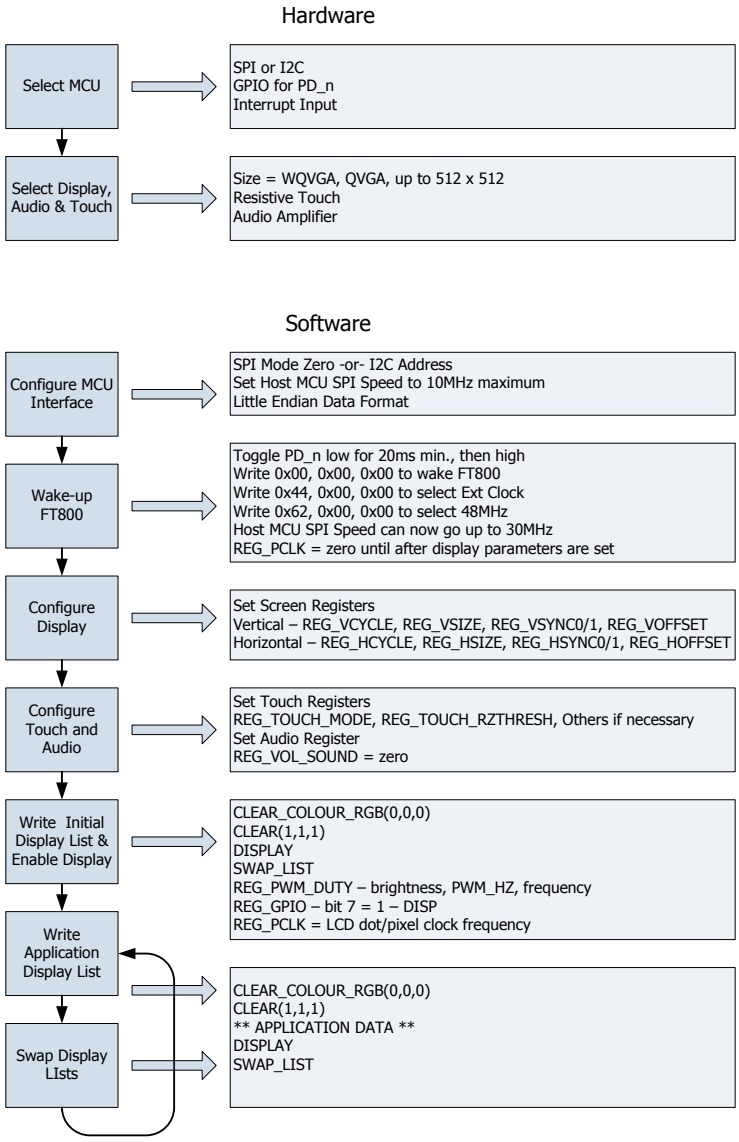

Figure 3.1 Generic EVE Design Flow

3.1 Signature Flowchart

The flowchart below is specific to the Sketch application.

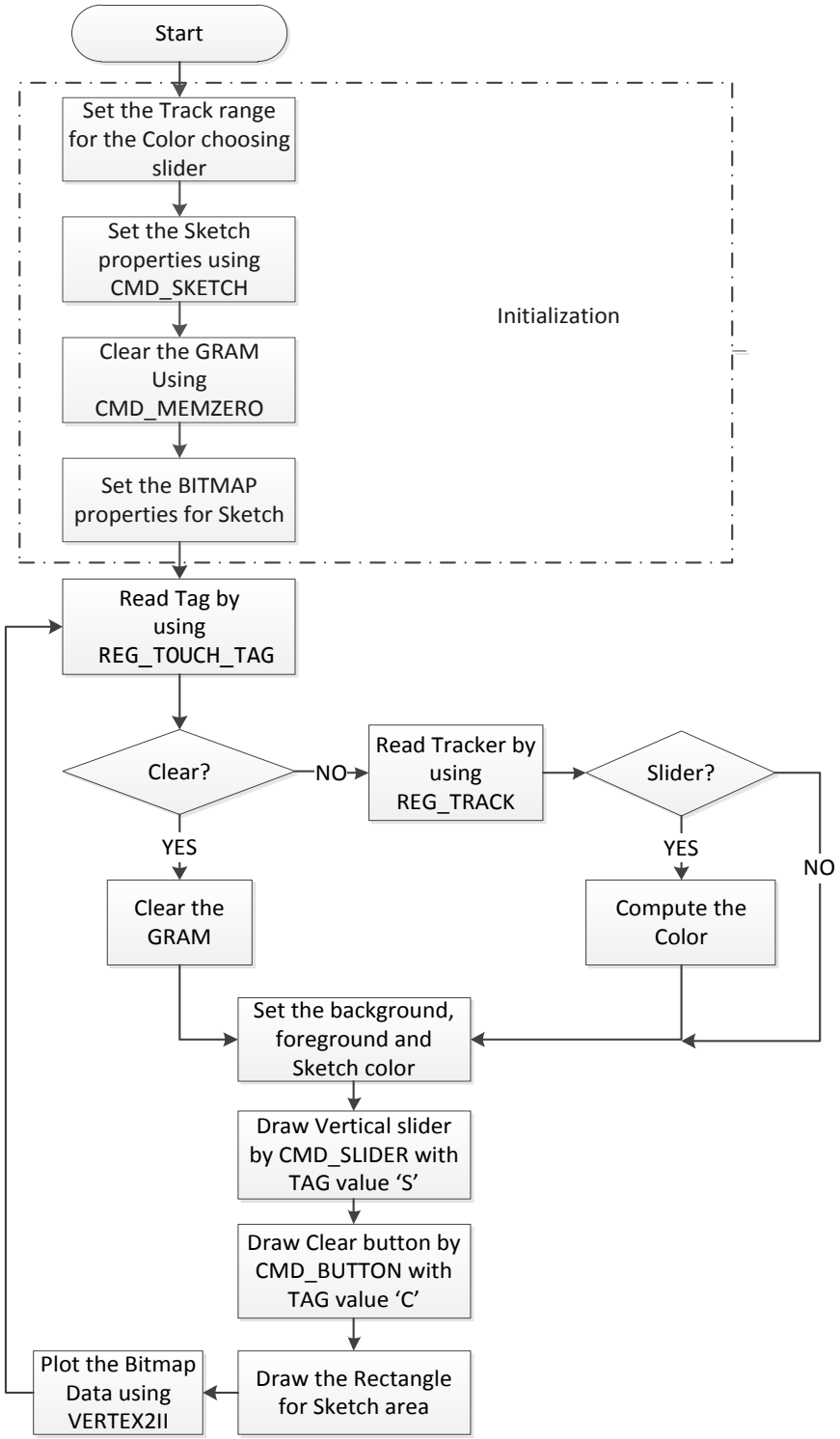

Figure 3.2 Flowchart

4 Description of the Functional Blocks

4.1 System Intialization

Configuration of the SPI master port is unique to each controller – different registers etc, but all will require data to be sent Most Significant Bit (MSB) first with a little endian format.

The function labelled Ft_BootupConfig is generic to all applications and will start by toggling the FT800 PD# pin to perform a power cycle.

```
/* Do a power cycle for safer side */
Ft_Gpu_Hal_Powercycle(phost,FT_TRUE);
Ft_Gpu_Hal_Rd16(phost,RAM_G);

/* Set the clk to external clock */
Ft_Gpu_HostCommand(phost,FT_GPU_EXTERNAL_OSC);
Ft_Gpu_Hal_Sleep(10);

/* Switch PLL output to 48MHz */
Ft_Gpu_HostCommand(phost,FT_GPU_PLL_48M);
Ft_Gpu_Hal_Sleep(10);

/* Do a core reset for safer side */
Ft_Gpu_HostCommand(phost,FT_GPU_CORE_RESET);

/* Access address 0 to wake up the FT800 */
Ft_Gpu_HostCommand(phost,FT_GPU_ACTIVE_M);
```

The internal PLL is then given a prompt by setting the clock register and PLL to 48 MHz.

Note 36MHz is possible but will have a knock on effect for the display timing parameters.

A software reset of the core is performed followed by a dummy read to address 0 to complete the wake up sequence.

The FT800 GPIO lines are also controlled by writing to registers:

```
Ft_Gpu_Hal_Wr8(phost, REG_GPIO_DIR,0x80 | Ft_Gpu_Hal_Rd8(phost,REG_GPIO_DIR));
Ft_Gpu_Hal_Wr8(phost, REG_GPIO,0x080 | Ft_Gpu_Hal_Rd8(phost,REG_GPIO));
```

And these allow the display to be enabled.

To confirm the FT800 is awake and ready to start accepting display list information the identity register is read in a loop until it reports back 0x7C. It will always be 0x7C if everything is awake and functioning correctly.

```
ft_uint8_t chipid;
//Read Register ID to check if FT800 is ready.
chipid = Ft_Gpu_Hal_Rd8(phost, REG_ID);
while(chipid != 0x7C)
        chipid = Ft_Gpu_Hal_Rd8(phost, REG_ID);
```

Once the FT800 is awake the display may be configured through 13 register writes according to its resolution. Resolution and timing data should be available in the display datasheet.

```
Ft_Gpu_Hal_Wr16(phost, REG_HCYCLE, FT_DispHCycle);
Ft_Gpu_Hal_Wr16(phost, REG_HOFFSET, FT_DispHOffset);
Ft_Gpu_Hal_Wr16(phost, REG_HSYNC0, FT_DispHSync0);
Ft_Gpu_Hal_Wr16(phost, REG_HSYNC1, FT_DispHSync1);
Ft_Gpu_Hal_Wr16(phost, REG_VCYCLE, FT_DispVCycle);
Ft_Gpu_Hal_Wr16(phost, REG_VOFFSET, FT_DispVOffset);
Ft_Gpu_Hal_Wr16(phost, REG_VSYNC0, FT_DispVSync0);
Ft_Gpu_Hal_Wr16(phost, REG_VSYNC1, FT_DispVSync1);
Ft_Gpu_Hal_Wr8(phost, REG_SWIZZLE, FT_DispSwizzle);
Ft_Gpu_Hal_Wr8(phost, REG_PCLK_POL, FT_DispPCLKPol);
Ft_Gpu_Hal_Wr8(phost, REG_PCLK,FT_DispPCLK);//after this display is visible on the LCD
Ft_Gpu_Hal_Wr16(phost, REG_HSIZE, FT_DispWidth);
Ft_Gpu_Hal_Wr16(phost, REG_VSIZE, FT_DispHeight);
```

To complete the configuration the touch controller should also be calibrated

```
/* Touch configuration - configure the resistance value to 1200 - this value is specific to
customer requirement and derived by experiment */
Ft_Gpu_Hal_Wr16(phost, REG_TOUCH_RZTHRESH,1200);
Ft_Gpu_Hal_Wr8(phost, REG_GPIO_DIR,0xff);
Ft_Gpu_Hal_Wr8(phost, REG_GPIO,0x0ff);
```

An optional step is present in this code to clear the screen so that no artefacts from bootup are displayed.

```
/*It is optional to clear the screen here*/
Ft_Gpu_Hal_WrMem(phost, RAM_DL,(ft_uint8_t *)FT_DLCODE_BOOTUP,sizeof(FT_DLCODE_BOOTUP));
Ft_Gpu_Hal_Wr8(phost, REG_DLSWAP,DLSWAP_FRAME);
```

4.2 Info()

This is a largely informational section of code and it starts by synchronising the physical xy coordinates of the display's touch layer with the display's visual layer.

A display list is started and cleared:

```
Ft_Gpu_CoCmd_Dlstart(phost);
Ft_App_WrCoCmd_Buffer(phost,CLEAR(1,1,1));
Ft_App_WrCoCmd_Buffer(phost,COLOR_RGB(255,255,255));
```

A text instruction is printed on the display followed by the call to the internal calibrate function:

```
Ft_Gpu_CoCmd_Text(phost,FT_DispWidth/2,FT_DispHeight/2,26,OPT_CENTERX|OPT_CENTERY,"Please tap
on a dot");
Ft_Gpu_CoCmd_Calibrate(phost,0);
```

The display list is then terminated and swapped to allow the changes to take effect.

```
Ft_App_WrCoCmd_Buffer(phost,DISPLAY());
Ft_Gpu_CoCmd_Swap(phost);
Ft_App_Flush_Co_Buffer(phost);
Ft_Gpu_Hal_WaitCmdfifo_empty(phost);
```

Next up in the Info() function is the FTDI logo playback:

```
Ft_Gpu_CoCmd_Logo(phost);
Ft_App_Flush_Co_Buffer(phost);
Ft_Gpu_Hal_WaitCmdfifo_empty(phost);
while(0!=Ft_Gpu_Hal_Rd16(phost,REG_CMD_READ));
dloffset = Ft_Gpu_Hal_Rd16(phost,REG_CMD_DL);
```

```
dloffset -=4;
Ft_Gpu_Hal_WrCmd32(phost,CMD_MEMCPY);
Ft_Gpu_Hal_WrCmd32(phost,100000L);
Ft_Gpu_Hal_WrCmd32(phost,RAM_DL);
Ft_Gpu_Hal_WrCmd32(phost,dloffset);
play_setup();
```

A composite image with the logo and a start arrow is then displayed to allow the user to start the main application

```
do
  {
    Ft_Gpu_CoCmd_Dlstart(phost);
    Ft_Gpu_CoCmd_Append(phost,100000L,dloffset);
    Ft_App_WrCoCmd_Buffer(phost,BITMAP_TRANSFORM_A(256));
    Ft_App_WrCoCmd_Buffer(phost,BITMAP_TRANSFORM_A(256));
    Ft_App_WrCoCmd_Buffer(phost,BITMAP_TRANSFORM_B(0));
    Ft_App_WrCoCmd_Buffer(phost,BITMAP_TRANSFORM_C(0));
    Ft_App_WrCoCmd_Buffer(phost,BITMAP_TRANSFORM_D(0));
    Ft_App_WrCoCmd_Buffer(phost,BITMAP_TRANSFORM_E(256));
    Ft_App_WrCoCmd_Buffer(phost,BITMAP_TRANSFORM_F(0));
    Ft_App_WrCoCmd_Buffer(phost,SAVE_CONTEXT());
    Ft_App_WrCoCmd_Buffer(phost,COLOR_RGB(219,180,150));
    Ft_App_WrCoCmd_Buffer(phost,COLOR_A(220));
    Ft_App_WrCoCmd_Buffer(phost,BEGIN(EDGE_STRIP_A));
    Ft_App_WrCoCmd_Buffer(phost,VERTEX2F(0,FT_DispHeight*16));
    Ft_App_WrCoCmd_Buffer(phost,VERTEX2F(FT_DispWidth*16,FT_DispHeight*16));
    Ft_App_WrCoCmd_Buffer(phost,COLOR_A(255));
    Ft_App_WrCoCmd_Buffer(phost,RESTORE_CONTEXT());
    Ft_App_WrCoCmd_Buffer(phost,COLOR_RGB(0,0,0));
   // INFORMATION
    Ft_Gpu_CoCmd_Text(phost,FT_DispWidth/2,20,28,OPT_CENTERX|OPT_CENTERY,info[0]);
    Ft_Gpu_CoCmd_Text(phost,FT_DispWidth/2,60,26,OPT_CENTERX|OPT_CENTERY,info[1]);
    Ft_Gpu_CoCmd_Text(phost,FT_DispWidth/2,90,26,OPT_CENTERX|OPT_CENTERY,info[2]);
    Ft_Gpu_CoCmd_Text(phost,FT_DispWidth/2,120,26,OPT_CENTERX|OPT_CENTERY,info[3]);
    Ft_Gpu_CoCmd_Text(phost,FT_DispWidth/2,FT_DispHeight-30,26,OPT_CENTERX|OPT_CENTERY,"Click
to play");
    if(sk!='P')
    Ft_App_WrCoCmd_Buffer(phost,COLOR_RGB(255,255,255));
    else
    Ft_App_WrCoCmd_Buffer(phost,COLOR_RGB(100,100,100));
    Ft_App_WrCoCmd_Buffer(phost,BEGIN(FTPOINTS));
    Ft_App_WrCoCmd_Buffer(phost,POINT_SIZE(20*16));
    Ft_App_WrCoCmd_Buffer(phost,TAG('P'));
    Ft_App_WrCoCmd_Buffer(phost,VERTEX2F((FT_DispWidth/2)*16,(FT_DispHeight-60)*16));
    Ft_App_WrCoCmd_Buffer(phost,COLOR_RGB(180,35,35));
    Ft_App_WrCoCmd_Buffer(phost,BEGIN(BITMAPS));
    Ft_App_WrCoCmd_Buffer(phost,VERTEX2II((FT_DispWidth/2)-14,(FT_DispHeight-75),14,0));
    Ft_App_WrCoCmd_Buffer(phost,DISPLAY());
    Ft_Gpu_CoCmd_Swap(phost);
    Ft_App_Flush_Co_Buffer(phost);
    Ft_Gpu_Hal_WaitCmdfifo_empty(phost);
  }while(Read_Keys()!='P');
```

4.3 Sketch()

An initial display list is created to configure a few items:

```
ft_uint32_t  tracker,color=0;
ft_uint16_t  val=32768;
ft_uint8_t tag =0;
Ft_Gpu_CoCmd_Dlstart(phost);
```

Set the foreground color to white:

```
Ft_Gpu_CoCmd_FgColor(phost,0xffffff);        // Set the foreground color
```

Define the area used by the slider bar. This touch area is just to the right of the sketch area, but not all the way to the right edge. This area is tracked so the slider can be moved:

```
Ft_Gpu_CoCmd_Track(phost,(FT_DispWidth-30),40,8,FT_DispHeight-100,1);
```

Start the CMD_SKETCH. This is a large rectangle from 0,0 to 40 pixels in from the right and 30 pixels up from the bottom:

```
Ft_Gpu_CoCmd_Sketch(phost,0,10,FT_DispWidth-40,FT_DispHeight-30,0,L8);
```

Zero out the GRAM area associated with the sketch area. This clears out any artifacts:

```
Ft_Gpu_CoCmd_MemZero(phost,0L,256*1024L);
Ft_App_WrCoCmd_Buffer(phost,BITMAP_SOURCE(0));
Ft_App_WrCoCmd_Buffer(phost,BITMAP_LAYOUT(L8,FT_DispWidth-40,FT_DispHeight-20));
Ft_App_WrCoCmd_Buffer(phost,BITMAP_SIZE(NEAREST,BORDER,BORDER,(FT_DispWidth-
40),(FT_DispHeight-20)));
```

Show this display list and clear out the buffers for the next one:

```
Ft_Gpu_CoCmd_Swap(phost);
Ft_App_Flush_Co_Buffer(phost);
Ft_Gpu_Hal_WaitCmdfifo_empty(phost);
```

At this point, the CMD_SKETCH command continually translates the X-Y touch coordinates within the sketch area and changes those locations in the GRAM from the foreground colour to background colour. By doing this, any drawing done on the touchscreen in the sketch area is shown on the screen. Once complete, the GRAM corresponding to the sketch area can be read by the MCU and stored as an image. This example does not store any sketch data.

With the visible display initialization complete, the screen elements can now be drawn. A while() loop will create a new display list each time though. The display list checks for any colour changes as a result of moving the slide and whether to clear the sketch area if the button is tapped.

The clear button and slider bar are assigned touch tags. Tags remove the need to manually correlate the X-Y touch coordinates with a drawn element. Instead, the entire element is assigned a tag number. The FT800 determines whether the touch event was inside the element boundaries then assigns the appropriate tag. For the slider, the tag is also assigned with the tracker above. The FT800 automatically sets a value corresponding to the location of the slider. The value is used to assign the background colour. These tags free up considerable host MCU time.

The first item in the while() loop is to see if there are any touch events and whether a tag is assigned. If the CLEAR button is tapped, the signature area is cleared by resetting the GRAM to the original zero values:

```
tracker = Ft_Gpu_Hal_Rd32(phost,REG_TRACKER);
tag = Ft_Gpu_Hal_Rd8(phost,REG_TOUCH_TAG);
if(tag==2)
{
        Ft_Gpu_CoCmd_Dlstart(phost);
        Ft_Gpu_CoCmd_MemZero(phost,0,256*1024L); // Clear the gram for 1024
        Ft_App_Flush_Co_Buffer(phost);
        Ft_Gpu_Hal_WaitCmdfifo_empty(phost);
}
```

If the tag for the slider bar is read, the tracker value is read and a new background colour is assigned:

```
if((tracker&0xff)==1)      // check the tag val
{
        val = (tracker>>16);
}
color = val*255;
```

The display list is then started, buffers flushed, foreground colour set to white and background colour set to correspond to the slider (or an initial value for the first time through):

```
Ft_Gpu_CoCmd_Dlstart(phost);        // start
Ft_App_WrCoCmd_Buffer(phost, CLEAR(1,1,1));
Ft_App_WrCoCmd_Buffer(phost,COLOR_RGB(255,255,255));
Ft_Gpu_CoCmd_BgColor(phost,color);
```

Next, a tag is assigned to the slider and the slider is drawn:

```
Ft_App_WrCoCmd_Buffer(phost,TAG_MASK(1));
Ft_App_WrCoCmd_Buffer(phost,TAG(1));
Ft_Gpu_CoCmd_FgColor(phost,color);
Ft_Gpu_CoCmd_Slider(phost,(FT_DispWidth-30),40,8,(FT_DispHeight-100),0,val,65535);
```

Now another tag is assigned to the "CLR" button and it is drawn:

```
Ft_Gpu_CoCmd_FgColor(phost,(tag==2)?0x0000ff:color);
Ft_App_WrCoCmd_Buffer(phost,TAG(2));          // assign the tag value
Ft_Gpu_CoCmd_Button(phost,(FT_DispWidth-35),(FT_DispHeight-45),35,25,26,0,"CLR");
Ft_App_WrCoCmd_Buffer(phost,TAG_MASK(0));
```

The last item to draw is the word "Color" above the slider bar:

```
Ft_Gpu_CoCmd_Text(phost,FT_DispWidth-35,10,26,0,"Color");
```

With everything drawn, the display list now identifies the sketch area and changes the background colour according to the value from the slider:

```
Ft_App_WrCoCmd_Buffer(phost,LINE_WIDTH(1*16));
Ft_App_WrCoCmd_Buffer(phost,BEGIN(RECTS));
Ft_App_WrCoCmd_Buffer(phost,VERTEX2F(0,10*16));
Ft_App_WrCoCmd_Buffer(phost,VERTEX2F((ft_int16_t)(FT_DispWidth-
40)*16,(ft_int16_t)(FT_DispHeight-20)*16));
    Ft_App_WrCoCmd_Buffer(phost,COLOR_RGB((color>>16)&0xff,(color>>8)&0xff,(color)&0xff));
Ft_App_WrCoCmd_Buffer(phost,BEGIN(BITMAPS));
Ft_App_WrCoCmd_Buffer(phost,VERTEX2II(0,10,0,0));
Ft_App_WrCoCmd_Buffer(phost,END());
```

The final section of the while() loop is to swap the display list to make it active, flush the buffers and return to the top of the loop:

```
Ft_App_WrCoCmd_Buffer(phost,DISPLAY());
Ft_Gpu_CoCmd_Swap(phost);
Ft_App_Flush_Co_Buffer(phost);
Ft_Gpu_Hal_WaitCmdfifo_empty(phost);
```

4.3.1 FT_801 CMD_CSKETCH

The FT801 capacitive display uses a different (capacitive) display touch controller compared to the FT800 resistive controller which has a slower sample rate than the resistive display. To ensure smooth interaction between the user touch and the application, CMD_SKETCH is replaced with CMD_CSKETCH. To enable this alternative command in the sample program open the Platform.h file and look for:

```
#define FT_801_ENABLE
```

By default this is undefined (FT800 mode). To switch in the CMD_CSKETCH ensure this line is defined. After making the change, rebuild and run the application.

4.4 Functionality

The function has three user interactive elements:

- Draw in the sketch area
- Clear the sketch using clear Button
- Change the sketch colour by using colour slider.

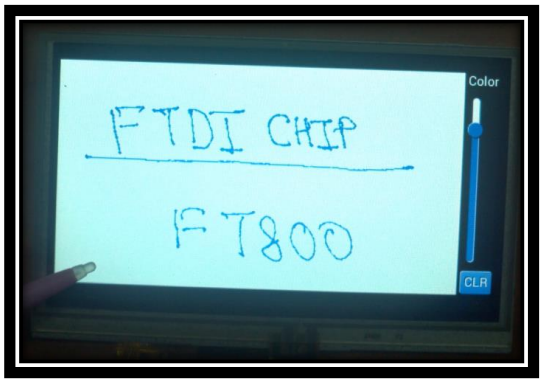

Figure 4.1 Signature Display

5 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

5.1 Document References

1. FT800 datasheet
2. FT801 datasheet
3. Programming Guide covering EVE command language
4. AN_240 FT800 From the Ground Up
5. AN_245 VM800CB_SampleApp_PC_Introduction - covering detailed design flow with a PC and USB to SPI bridge cable
6. AN_246 VM800CB_SampleApp_Arduino_Introduction – covering detailed design flow in an Arduino platform
7. VM800C datasheet
8. VM800B datasheet

5.2 Acronyms and Abbreviations

Terms	Description
Arduino Pro	The open source platform variety based on ATMEL's ATMEGA chipset
EVE	Embedded Video Engine
SPI	Serial Peripheral Interface
UI	User Interface

Appendix B – List of Tables & Figures

5.3 List of Figures

Appendix C– Revision History

Document Title: AN_270 FT_App_Sketch

Document Reference No.: FT_000915

Clearance No.: FTDI# 365

Product Page: http://www.ftdichip.com/EVE.htm

Document Feedback: Send Feedback

Revision	Changes	Date
0.1	Initial draft release	2013-07-18
1.0	Version 1.0 updated wrt review comments	2013-08-21
1.1	Included code discussion	2013-10-15
1.1	Version 1.1	2013-11-01
1.2	Added section 4.3.1	2014-06-30