Application Note

AN_335

FT801 Graph Application

Document Reference No.:FT_001076

Version 1.0
Issue Date: 2014-07-22

This document introduces the setup of the FT801 Graph Application running on MSVC. The objective of the Graph Application is to enable users to become familiar with the usage of the multi-touch functionality of FT801, 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 application demonstrates zoom-in and zoom-out functionality using the FT801multi-touch capability. The application constructs a power graph on the screen. Based on user touch movement, either-zoom in or zoom-out is performed. This application demonstrates the use of two simultaneous touch inputs from the user.

1.1 Overview

The document will provide an understanding of the FT801 multi-touch functionality, and demonstrate a simple use case.

1.2 Scope

This document will be used by software programmers to develop GUI applications by using the FT801 with any MCU with a SPI master port.

For information on the project file and source code, refer to **AN_264_FT_App_Gradient Application note.**

Note that detailed documentation is available on www.ftdichip.com/EVE.htm , including:

FT801 Datasheet

FT800 Series Programming Guide

2 Application Flow

2.1 Flowchart

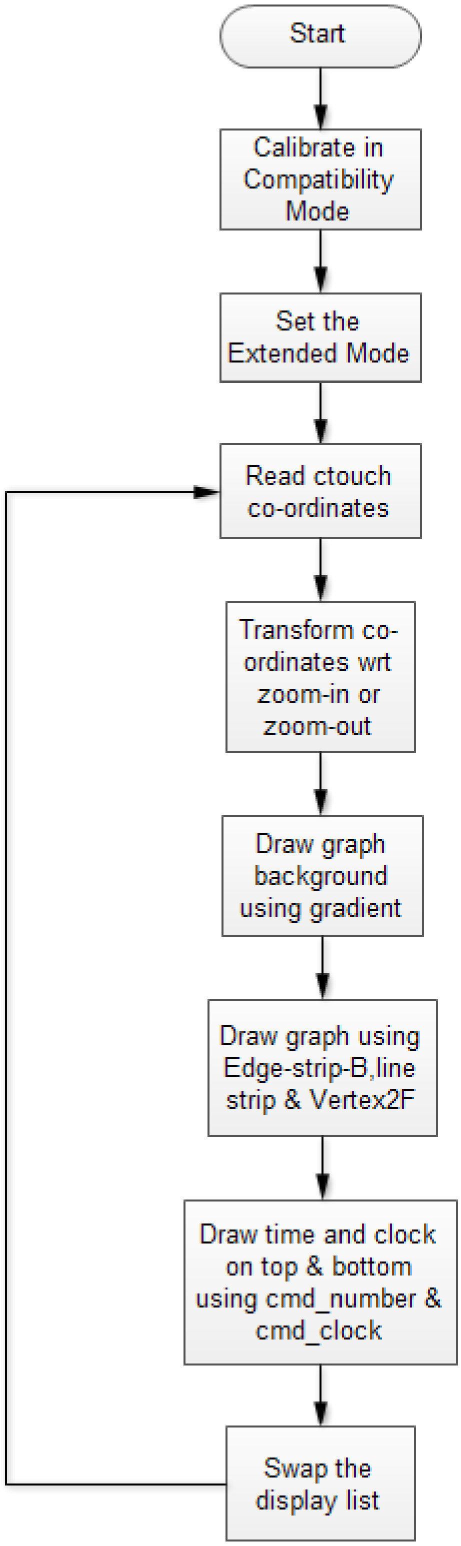

Figure 2-1 Flowchart

3 Description

Parameters needed to be initialized are described below for constructing the display list.

3.1 Intialization

3.1.1 Set Extended mode for multi-touch

By default, the FT801 touch engine works in compatibility mode, and operates much like the resistive touch controller in the FT800. In compatibility mode, only one touch point is detected. In extended mode, the FT801 touch engine can detect up to 5 touch points, simultaneously.

Before entering in extended mode, user needs to do calibration in compatibility mode.

A co-processor command list is started. This command will clear the display parameters.

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

The following commands set the colour and then print a text message to the user which tells them to tap on the dots during the following calibration routine. The FT800's built-in calibration routine is then called.

```
Ft_App_WrCoCmd_Buffer(phost,COLOR_RGB(255,255,255));
Ft_Gpu_CoCmd_Text(phost,FT_DispWidth/2,FT_DispHeight/2,28,OPT_CENTERX|OPT_CENTERY,"Ple
ase 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);
```

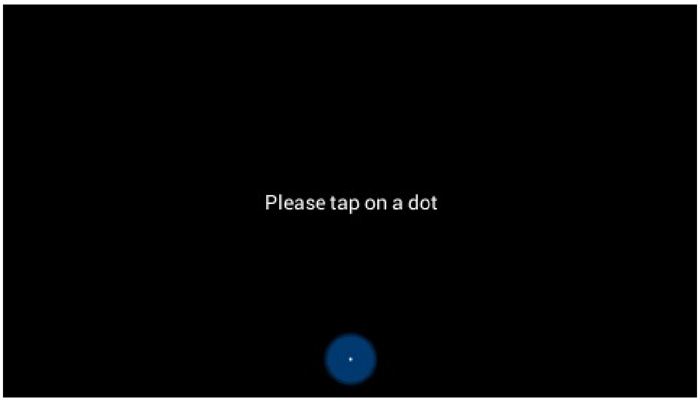

Figure 3-1 Calibration screen

As this application is designed to demonstrate FT801's multi-touch functionality, set the mode to extended. For more information please refer to the FT800 Series Programming Guide

```
Ft_Gpu_Hal_Wr8(phost,REG_CTOUCH_EXTENDED, CTOUCH_MODE_EXTENDED);
```

3.2 Functionality

The Graph Demo is a user interactive demo where the user can touch screen with 2 touch points simultaneously to adjust the zoom level of the displayed image.

3.2.1 Read touch control registers

The FT801 has different touch engine and touch control registers from the FT800. These registers provide coordinates for multiple touch points. The example code below shows the use of the "ctouch" registers.

```
ft_void_t read_extended(ft_int16_t sx[5], ft_int16_t sy[5])
{
  ft_uint32_t sxy0, sxyA, sxyB, sxyC;
  sxy0 = Ft_Gpu_Hal_Rd32(phost,REG_CTOUCH_TOUCH0_XY);
  sxyA = Ft_Gpu_Hal_Rd32(phost, REG_CTOUCH_TOUCH1_XY);
  sxyB = Ft_Gpu_Hal_Rd32(phost, REG_CTOUCH_TOUCH2_XY);
  sxyC = Ft_Gpu_Hal_Rd32(phost, REG_CTOUCH_TOUCH3_XY);

  sx[0] = sxy0 >> 16;
  sy[0] = sxy0;
  sx[1] = sxyA >> 16;
  sy[1] = sxyA;
  sx[2] = sxyB >> 16;
  sy[2] = sxyB;
  sx[3] = sxyC >> 16;
  sy[3] = sxyC;

  sx[4] = Ft_Gpu_Hal_Rd16(phost,REG_CTOUCH_TOUCH4_X);
  sy[4] = Ft_Gpu_Hal_Rd16(phost,REG_CTOUCH_TOUCH4_Y);
}
```

3.2.2 Draw graph

In this application, the FT801 graphic coprocessor command CMD_GRADIENT is used to display the background:

```
Ft_Gpu_CoCmd_Gradient(phost,0, 0, 0x202020, 0, 0x11f, 0x107fff);
```

Figure 3-2 Gradient background

To draw the graph, the application uses the rsin function - sine with radius. It uses two values for radius, 1200 and 700, and different theta values as shown in the code below. Using the rsin function, values are calculated and stored in array y. VERTEX2F uses this y-value as second co-ordinate, and uses multiple values of SUBDIV as the first co-ordinate. The EDGE_STRIP_B primitive and GRADIENT functions are used to draw the graph.

```
Ft_App_WrCoCmd_Buffer(phost,COLOR_A(255));

for (i = 0; i < (YY + 1); i++)
{
       x32 = s2m(SUBDIV * i);
       x2 = (ft_uint16_t)x32 + rsin(7117, x32);
       y[i] = 130 * 16 + rsin(1200, (217 * x32) >> 8) + rsin(700, 3 * x2);
}

Ft_App_WrCoCmd_Buffer(phost,STENCIL_OP(INCR, INCR));
Ft_App_WrCoCmd_Buffer(phost, BEGIN(EDGE_STRIP_B));
for (j = 0; j < (YY + 1); j++)
{
       Ft_App_WrCoCmd_Buffer(phost,VERTEX2F(16 * SUBDIV * j, y[j]));
}
Ft_App_WrCoCmd_Buffer(phost,STENCIL_FUNC(EQUAL, 1, 255));
Ft_App_WrCoCmd_Buffer(phost,STENCIL_OP(KEEP, KEEP));
Ft_Gpu_CoCmd_Gradient(phost,0, 0, 0xf1b608, 0, 220, 0x98473a);
```

The code above generates this initial screen:

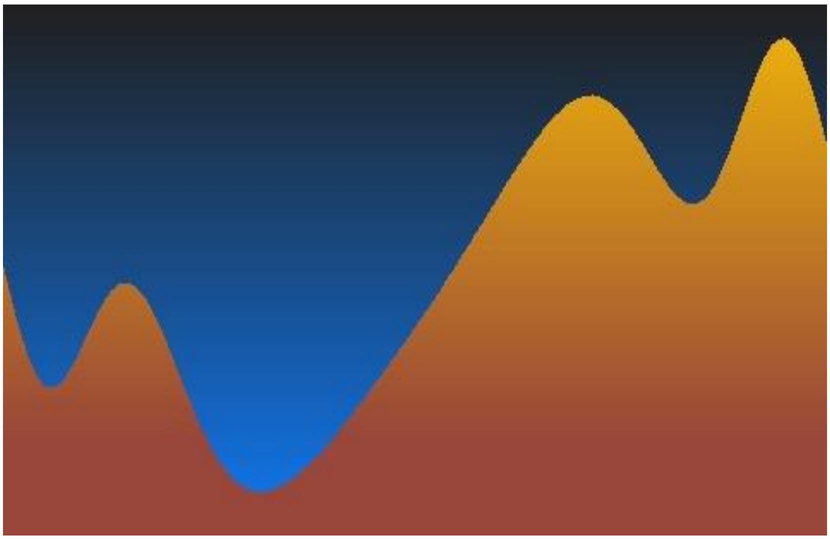

Figure 3-3 Graph drawn using EDGE_STRIP_B

The application then uses LINE_STRIP and VERTEX2F primitives to draw the border of the graph:

```
Ft_App_WrCoCmd_Buffer(phost,STENCIL_FUNC(ALWAYS, 1, 255));
Ft_App_WrCoCmd_Buffer(phost, COLOR_RGB(0xE0,0xE0,0xE0));
Ft_App_WrCoCmd_Buffer(phost, LINE_WIDTH(24));
Ft_App_WrCoCmd_Buffer(phost, BEGIN(LINE_STRIP));

for (j = 0; j < (YY + 1); j++)
{
       Ft_App_WrCoCmd_Buffer(phost,VERTEX2F(16 * SUBDIV * j, y[j]));
}
```

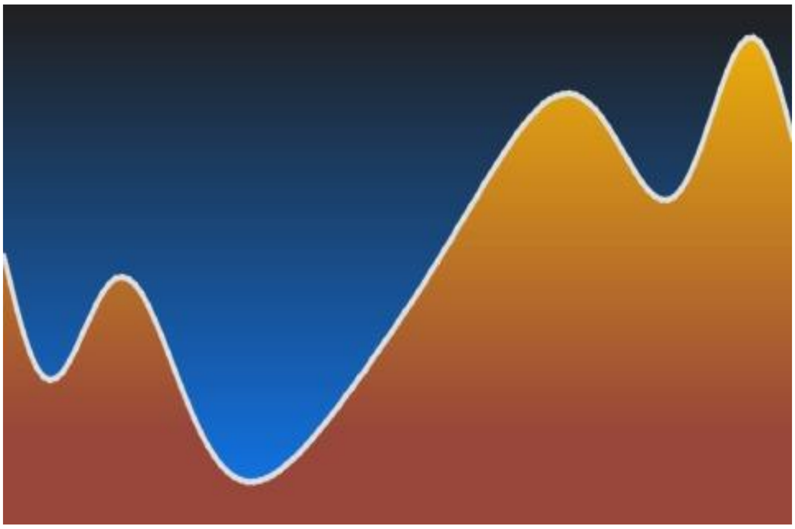

Figure 3-4 Graph border drawn using LINE_STRIP

Next, the application uses LINES and VERTEX2F primitives to draw vertical lines on the background.

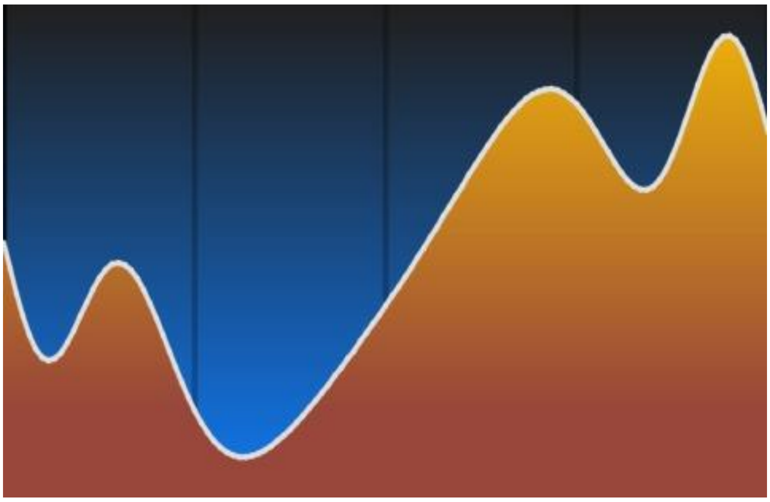

Figure 3-5 Vertical lines using LINES

Now the digits are drawn with CMD_NUMBER. The following code shows the use of LINES and VERTEX2F primitives and the CMD_NUMBER command.

```
Ft_App_WrCoCmd_Buffer(phost, LINE_WIDTH(max(8, pixels_per_div >> 2)));
for (m = mm[0] & ~0x3fff; m <= mm[1]; m += 0x4000)
{
        x = m2s(m);
        if ((-60 <= x) && (x <= 512))
        {
          h = 3 * (7 & (m >> 14));

          Ft_App_WrCoCmd_Buffer(phost, COLOR_RGB(0,0,0));
          Ft_App_WrCoCmd_Buffer(phost,COLOR_A(((h == 0) ? 192 : 64)));
          Ft_App_WrCoCmd_Buffer(phost, BEGIN(LINES));
          Ft_App_WrCoCmd_Buffer(phost,VERTEX2F(x*16,0));
          Ft_App_WrCoCmd_Buffer(phost,VERTEX2F(x*16,272*16));

          if (fadeout)
          {
               x -= 1;
               Ft_App_WrCoCmd_Buffer(phost, COLOR_RGB(0xd0,0xd0,0xd0));
               Ft_App_WrCoCmd_Buffer(phost,COLOR_A(fadeout));
               Ft_Gpu_CoCmd_Number(phost,x, 0, 26, OPT_RIGHTX | 2, h);
               Ft_Gpu_CoCmd_Text(phost,x, 0, 26, 0, ":00");
          }
        }
}
```

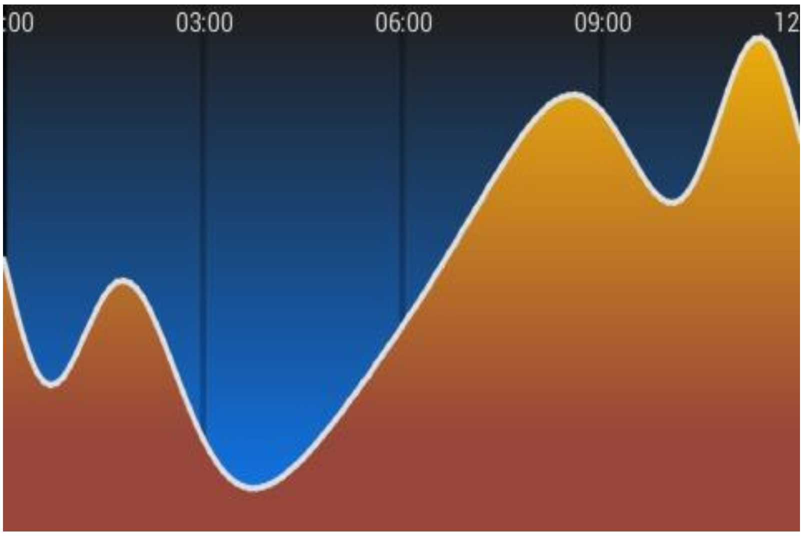

Figure 3-6 Numbers drawn using cmd_number

Now the application uses coprocessor's CMD_CLOCK command to draw a series of analog clocks.

```
clock_r = min(24, pixels_per_div >> 2);

if (clock_r > 4)
{
       Ft_App_WrCoCmd_Buffer(phost,COLOR_A(200));
       Ft_App_WrCoCmd_Buffer(phost, COLOR_RGB(0xff,0xff,0xff));
       options = OPT_NOSECS | OPT_FLAT;
       if (clock_r < 10)
         options |= OPT_NOTICKS;
       for (m = mm[0] & ~0x3fff; m <= mm[1]; m += 0x4000)
       {
         x1 = m2s(m);
         h = 3 * (3 & (m >> 14));
         if(x1 >= -1024)
          Ft_Gpu_CoCmd_Clock(phost,x1, 270 - 24, clock_r, options, h, 0, 0, 0);
       }
}
```

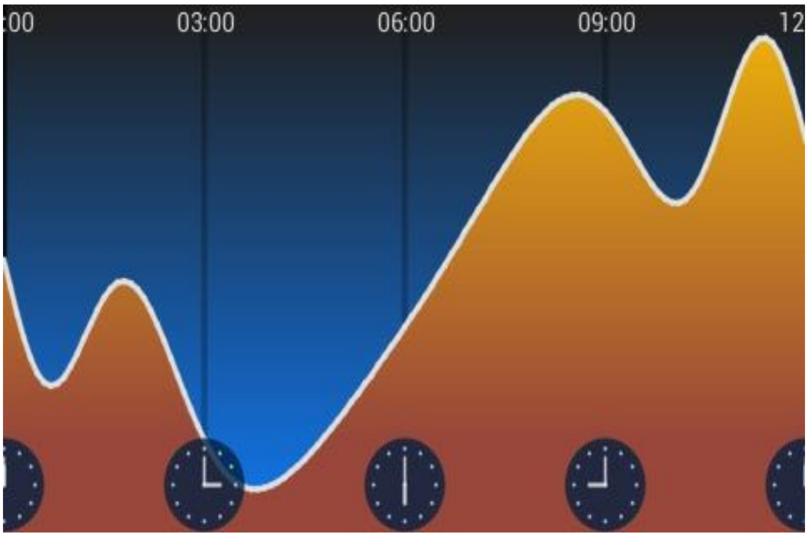

Figure 3-7 Clocks drawn using cmd_clock

The following code is used to zoom the graph in and out, after reading touch co-ordinates.

```
for (i = 0; i < 2; i++)
{
        if (sx[i] > -10 && !down[i])
        {
                down[i] = 1;
                m[i] = s2m(sx[i]);
        }
        if (sx[i] < -10)
                down[i] = 0;
}
if (down[0] && down[1])
{
        if (m[0] != m[1])
                set(m[0], sx[0], m[1], sx[1]);
}
else if (down[0] && !down[1])
        sset(m[0], sx[0]);
else if (!down[0] && down[1])
        sset(m[1], sx[1]);
```

These are the definitions of the above functions :

```
ft_void_t set(ft_int32_t x0, ft_int16_t y0,
           ft_int32_t x1, ft_int16_t y1) {
    ft_int32_t xd = x1 - x0;
    ft_int16_t yd = y1 - y0;
    transform_m = yd / (ft_float_t)xd;
    if (transform_m < m_min)
       transform_m = m_min;
    transform_c = y0 - transform_m * x0;
  }
ft_void_t sset(ft_int32_t x0, ft_int16_t y0)
{
    transform_c = (ft_float_t)y0 - transform_m * x0;
}
ft_int16_t m2s(ft_int32_t x)
{
    return (ft_int16_t)(transform_m * x + transform_c);
}
ft_int32_t s2m(ft_int16_t y)
{
    return (ft_int32_t)(y - transform_c) / transform_m;
}
```

When the user performs a zoom-in or zoom-out, the application reads the touch co-ordinates in a loop. These values are used for calculating the displayed image. In the set function, the difference between x co-ordinates is calculated and according to that smallest division (i.e. transform_m and later transform_c are calculated). These respective values are used in the remaining functions.

In the plot function, the pixel-per-division is calculated using m2s functions which are then used to update the graph.

The sequence of screen shots below demonstrates the use of two simultaneous touch points in a "pinching" action to zoom the graph out.

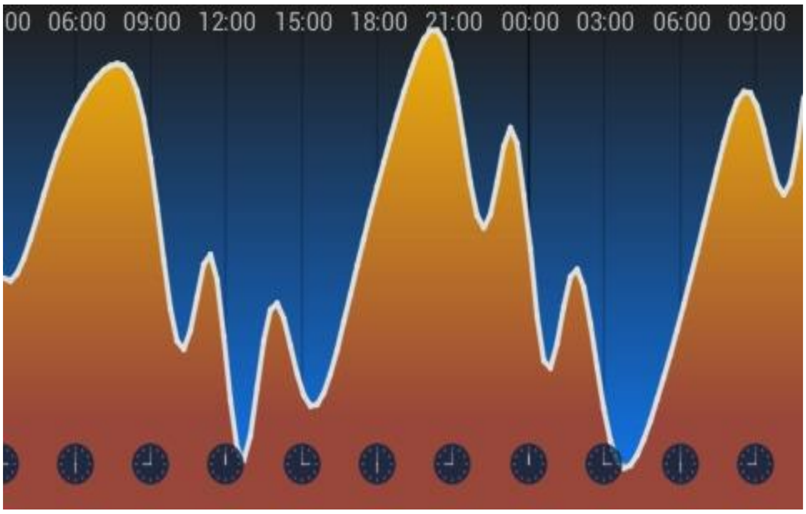

Figure 3-8 Zoom in screenshot-1

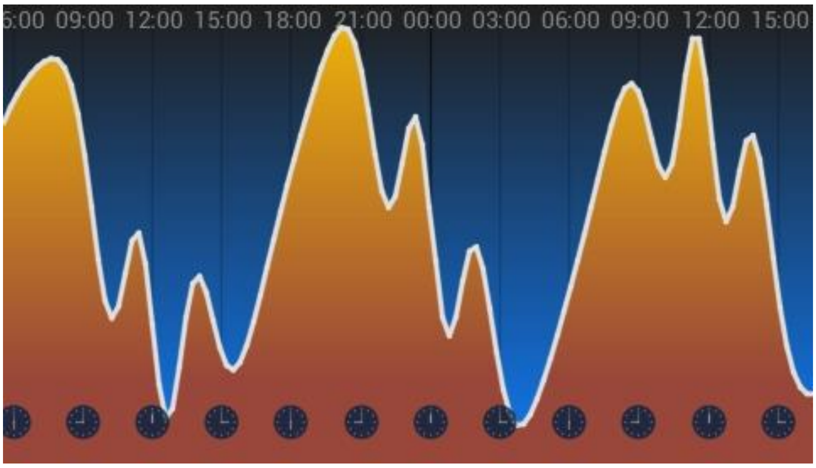

Figure 3-9 Zoom in screenshot-2

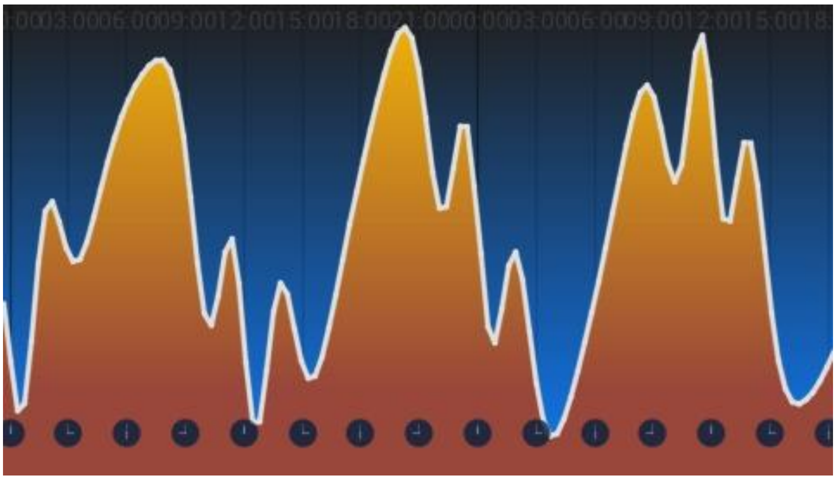

Figure 3-10 Zoom in screenshot-3

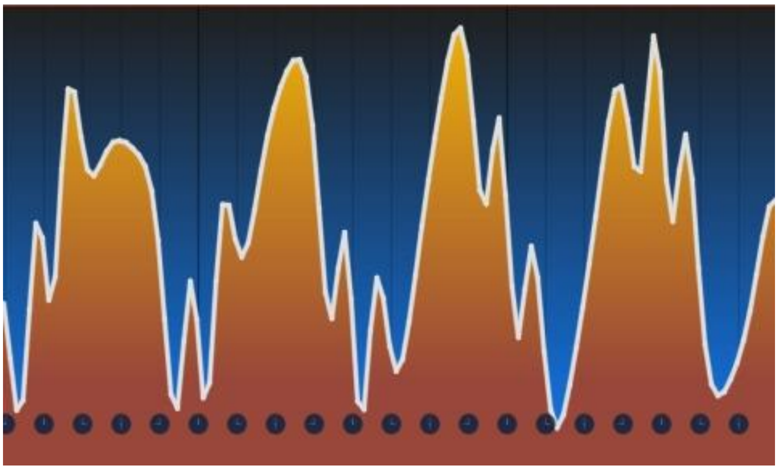

Figure 3-11 Zoom in screenshot-4

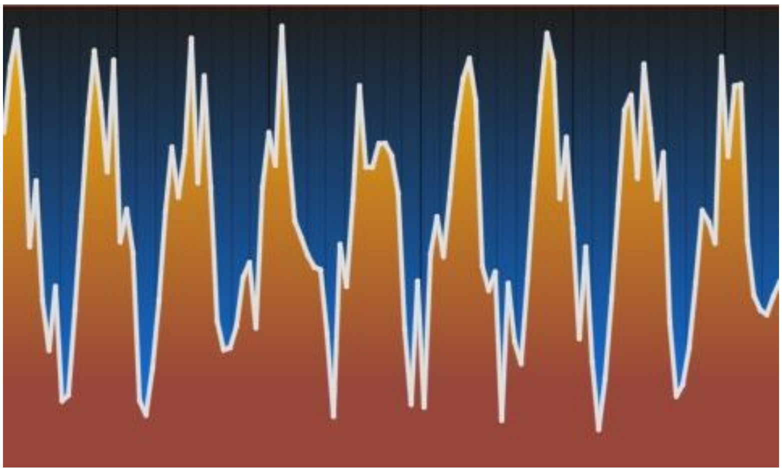

Figure 3-12 Zoom in screenshot-5

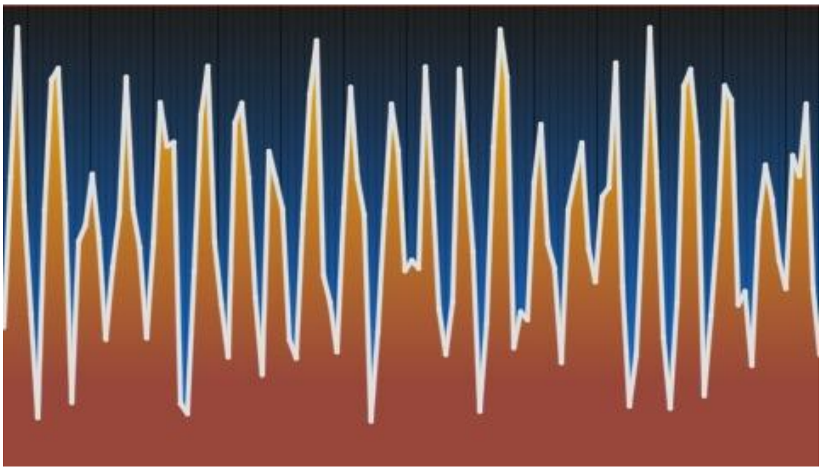

Figure 3-13 Zoom in screenshot-6

In a similar fashion, the next sequence of screen shots demonstrate the use of two simultaneous touch points in an "expanding" action to zoom the graph out.

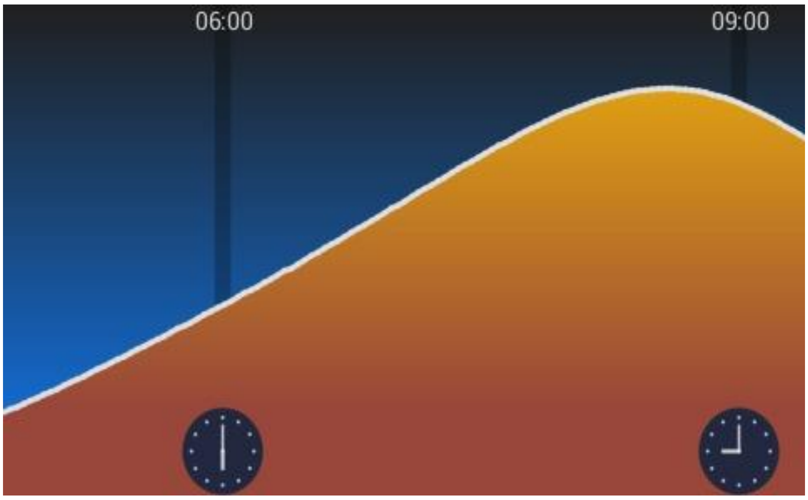

Figure 3-14 Zoom out screenshot-1

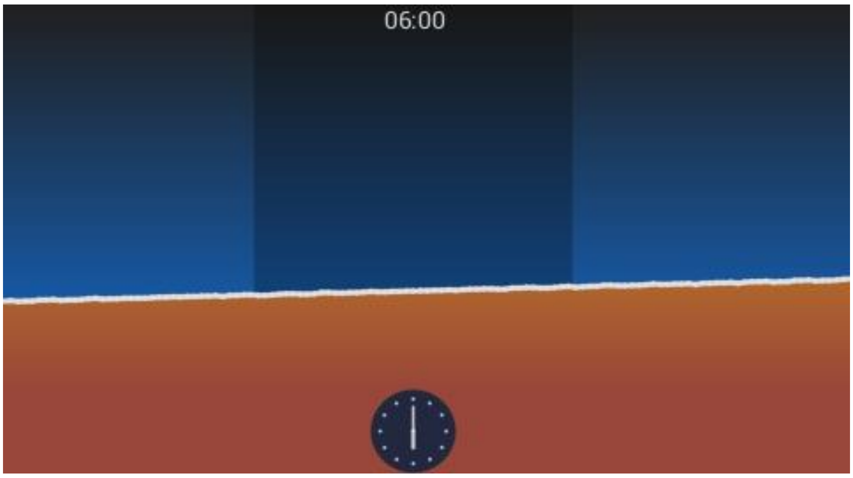

Figure 3-15 Zoom out screenshot-2

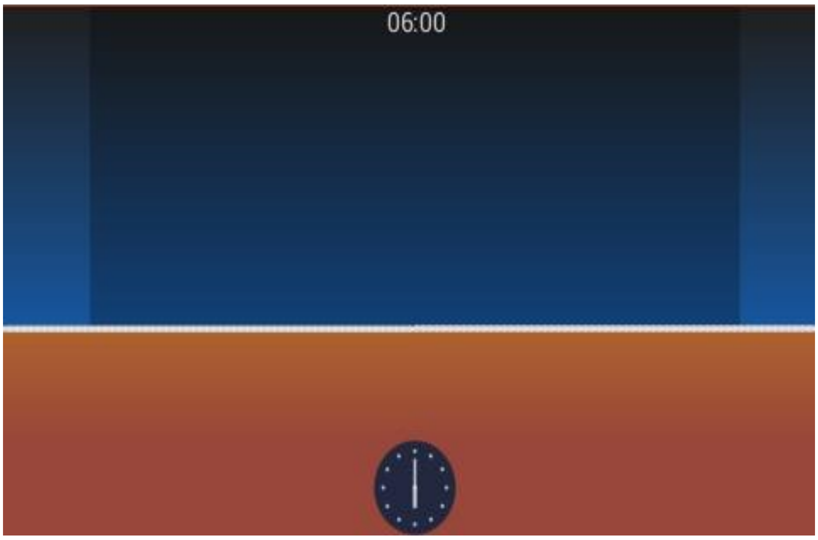

Figure 3-16 Zoom out screenshot-3

4 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

1. FT800 Series programmer guide
2. FT801 Embedded Video Engine Datasheet
3. Graph App

Acronyms and Abbreviations

Terms	Description
SPI	Serial Peripheral Interface
GUI	Graphical User Interface
MSVC	Microsoft Visual C

Appendix B – List of Tables & Figures

Appendix C– Revision History

Document Title: AN_335 FT801 Graph Application

Document Reference No.: FT_001076

Clearance No.: FTDI#403

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

Document Feedback: Send Feedback

Revision	Changes	Date
1.0	Initial release	2014-07-22