This application note introduces the setup of the FT8XX 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 the FT801, the design flow, and display list used to design the desired user interface or visual effect.
Table of Contents

1 Introduction ................................................................. 3
  1.1 Overview ........................................................................ 3
  1.2 Scope ............................................................................ 3
  1.3 Graph Overview .......................................................... 3

2 Application Flow ............................................................... 4
  2.1 Flowchart ...................................................................... 4

3 Description .......................................................................... 5
  3.1 Application Start Screen ................................................. 5
  3.2 Set Extended mode for multi-touch ................................. 5
  3.3 Functionality ................................................................... 5
    3.3.1 Read touch control registers ..................................... 5
    3.3.2 Draw graph ............................................................. 6

4 Contact Information ............................................................ 14

Appendix A– References ....................................................... 15
  Document References ........................................................ 15
  Acronyms and Abbreviations .............................................. 15

Appendix B – List of Figures & Tables .................................... 16
  List of Figures ................................................................. 16
  List of Tables ................................................................. 16

Appendix C– Revision History ............................................... 17
1 Introduction

This application demonstrates zoom-in and zoom-out functionality using the FT801 multi-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 http://brtchip.com/SoftwareExamples-eve/.

Note that detailed documentation is available on http://brtchip.com/eve/, including:

- FT801 Datasheet
- FT8XX Series Programming Guide

1.3 Graph Overview

![Figure 1.1 Graph Application UI](attachment:image.png)
2 Application Flow

2.1 Flowchart

![Flowchart Diagram]

Figure 2.1 Flowchart
3 Description

Refer to [AN_391 EVE Platform Guide](AN_391 EVE Platform Guide) for information pertaining to platform setup and the necessary development environment.

3.1 Application Start Screen

Upon completing the Setup, the application start screen is displayed.

![Start Screen](image)

**Figure 3.1 - Start Screen**

3.2 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 FT8XX. In compatibility mode, only one touch point is detected. In extended mode, the FT801 touch engine can detect up to 5 touch points, simultaneously.

As this application is designed to demonstrate FT801’s multi-touch functionality, set the mode to extended. For more information please refer to the [FT8XX Series Programming Guide](FT8XX Series Programming Guide).

```c
Gpu_Hal_Wr8(phost, REG_CTOUCH_EXTENDED, CT Touch_MODE_EXTENDED);
```

3.3 Functionality

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

3.3.1 Read touch control registers

The FT801 has different touch engine and touch control registers from the FT8XX. These registers provide coordinates for multiple touch points. The example code below shows the use of the “ctouch” registers.

```c
void read_extended(int16_t sx[5], int16_t sy[5])
{
    uint32_t sxy0, sxyA, sxyB, sxyC;
    sxy0 = Gpu_Hal_Rd32(phost, REG_CTOUCH_TOUCH0_XY);
    sxyA = Gpu_Hal_Rd32(phost, REG_CTOUCH_TOUCH1_XY);
    sxyB = Gpu_Hal_Rd32(phost, REG_CTOUCH_TOUCH2_XY);
    // Process coordinates...
}
```
sxyC = 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] = Gpu_Hal_Rd16(phost, REG_CTOUCH TOUCH4 X);
sy[4] = Gpu_Hal_Rd16(phost, REG_CTOUCH TOUCH4 Y);
}

### 3.3.2 Draw graph

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

```c
Gpu_CoCmd_Gradient(phost, 0, 0, 0x202020, 0, DispHeight, 0x107fff);
```

![Figure 3.2 Gradient Background](image)

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.

```c
App_WrCoCmd_Buffer(phost, COLOR_A(255));
for (i = 0; i < (YY + 1); i++)
{
  x32 = s2m(SUBDIV * i);
  x2 = (uint16_t)x32 + rsin(7117, x32);
  y[i] = 130 * 16 + rsin(1200, (217 * x2) >> 8) + rsin(700, 3 * x2);
}
```
The code above generates this initial screen:

![Initial Screen Image]

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:

```c
App_WrCoCmd_Buffer(phost, STENCIL.Func(ALWAYS, 1, 255));
App_WrCoCmd_Buffer(phost, COLOR_RGB(0xE0, 0xE0, 0xE0));
App_WrCoCmd_Buffer(phost, LINE_WIDTH(24));
App_WrCoCmd_Buffer(phost, BEGIN(LINE_STRIP));

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

![Border Screen Image]

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](image)

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

```c
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 <= (DispWidth+60))) {
        h = 3 * (7 & (m >> 14));
        App_WrCoCmd_Buffer(phost, COLOR_RGB(0,0,0));
        App_WrCoCmd_Buffer(phost, COLOR_A(((h == 0) ? 192 : 64)));
        App_WrCoCmd_Buffer(phost, BEGIN(LINES));
        App_WrCoCmd_Buffer(phost, VERTEX2F(x*16,0));
        App_WrCoCmd_Buffer(phost, VERTEX2F(x*16,DispHeight*16));
        if (fade) {
            x -= 1;
            App_WrCoCmd_Buffer(phost, COLOR_RGB(0xd0,0xd0,0xd0));
            App_WrCoCmd_Buffer(phost, COLOR_A(fade));
            Gpu_CoCmd_Number(phost,x, 0, 30, OPT_RIGHTX | 2, h);
            Gpu_CoCmd_Text(phost,x, 0, 30, 0, "00");
        }
    }
}
```
Now the application uses coprocessor's CMD_CLOCK command to draw a series of analog clocks.

```c
clock_r = min(24, pixels_per_div >> 2);
if (clock_r > 4)
{
    App_WrCoCmd_Buffer(phost, COLOR_A(200));
    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)
            Gpu_CoCmd_Clock(phost, x1, DispHeight-clock_r, clock_r, options, h, 0, 0, 0);
    }
}
```

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

**Figure 3.7 Clocks drawn using cmd_clock**
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:

```c
void set(int32_t x0, int16_t y0,
         int32_t x1, int16_t y1) {
    int32_t xd = x1 - x0;
    int16_t yd = y1 - y0;
    transform_m = yd / (float_t)xd;
    if (transform_m < m_min)
        transform_m = m_min;
    transform_c = y0 - transform_m * x0;
}
void sset(int32_t x0, int16_t y0)
{
    transform_c = (float_t)y0 - transform_m * x0;
}
int16_t m2s(int32_t x)
{
    return (int16_t)(transform_m * x + transform_c);
}
int32_t s2m(int16_t y)
{
    return (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
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 Quarters – Singapore
Bridgetek Pte Ltd
178 Paya Lebar Road, #07-03
Singapore 409030
Tel: +65 6547 4827
Fax: +65 6841 6071
E-mail (Sales) sales.apac@brtchip.com
E-mail (Support) support.apac@brtchip.com

Branch Office – Taipei, Taiwan
Bridgetek Pte Ltd, Taiwan Branch
2 Floor, No. 516, Sec. 1, Nei Hu Road, Nei Hu District
Taipei 114
Taiwan, R.O.C.
Tel: +886 (2) 8797 1330
Fax: +886 (2) 8751 9737
E-mail (Sales) sales.apac@brtchip.com
E-mail (Support) support.apac@brtchip.com

Branch Office – Glasgow, United Kingdom
Bridgetek Pte. Ltd.
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) sales.emea@brtchip.com
E-mail (Support) support.emea@brtchip.com

Branch Office – Vietnam
Bridgetek VietNam Company Limited
Lutaco Tower Building, 5th Floor, 173A Nguyen Van Troi,
Ward 11, Phu Nhuan District,
Ho Chi Minh City, Vietnam
Tel: 08 38453222
Fax: 08 38455222
E-mail (Sales) sales.apac@brtchip.com
E-mail (Support) support.apac@brtchip.com

Web Site
http://brtchip.com/

Distributor and Sales Representatives
Please visit the Sales Network page of the Bridgetek 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 Bridgetek Pte Ltd (BRTChip) 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 Bridgetek devices and other materials) is provided for reference only. While Bridgetek has taken care to assure it is accurate, this information is subject to customer confirmation, and Bridgetek disclaims all liability for system designs and for any applications assistance provided by Bridgetek. Use of Bridgetek 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 Bridgetek 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. Bridgetek Pte Ltd, 178 Paya Lebar Road, #07-03, Singapore 409030. Singapore Registered Company Number: 201542387H.
Appendix A– References

Document References

- FT8XX Series programmer guide
- FT801 Embedded Video Engine Datasheet
- AN_391 EVE Platform Guide
- Graph App

Acronyms and Abbreviations

<table>
<thead>
<tr>
<th>Terms</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arduino Pro</td>
<td>The open source platform variety based on ATMEL’s ATMEGA chipset</td>
</tr>
<tr>
<td>EVE</td>
<td>Embedded Video Engine</td>
</tr>
<tr>
<td>SPI</td>
<td>Serial Peripheral Interface</td>
</tr>
<tr>
<td>UI</td>
<td>User Interface</td>
</tr>
<tr>
<td>USB</td>
<td>Universal Serial Bus</td>
</tr>
</tbody>
</table>
Appendix B – List of Figures & Tables

List of Figures

Figure 1.1 Graph Application UI ................................................................. 3
Figure 2.1 Flowchart .................................................................................. 4
Figure 3.1 - Start Screen ........................................................................... 5
Figure 3.2 Gradient Background ................................................................. 6
Figure 3.3 Graph drawn using EDGE_STRIP_B ........................................ 7
Figure 3.4 Graph border drawn using LINE_STRIP .................................... 7
Figure 3.5 Vertical lines using LINES .......................................................... 8
Figure 3.6 Numbers drawn using cmd_number ......................................... 9
Figure 3.7 Clocks drawn using cmd_clock ............................................... 9
Figure 3.8 Zoom in screenshot-1 ............................................................... 11
Figure 3.9 Zoom in screenshot-2 ............................................................... 11
Figure 3.10 Zoom in screenshot-3 ............................................................. 11
Figure 3.11 Zoom in screenshot-4 ............................................................. 12
Figure 3.12 Zoom in screenshot-5 ............................................................. 12
Figure 3.13 Zoom in screenshot-6 ............................................................. 12
Figure 3.14 Zoom out screenshot-1 ......................................................... 13
Figure 3.15 Zoom out screenshot-2 ........................................................... 13
Figure 3.16 Zoom out screenshot-3 ........................................................... 13

List of Tables

NA
# 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>2014-07-22</td>
</tr>
<tr>
<td>1.1</td>
<td>Document migrated from FTDI to BRT (Updated company logo; copyright info; contact information; hyperlinks)</td>
<td>2018-01-05</td>
</tr>
</tbody>
</table>