Application Note

AN_263

FT_App_Gauges

Version 1.3

Issue Date: 2018-01-05

This application note is provided to introduce the Gauges Demo Application. The objective of this Demo Application is to enable users to become familiar with the usage of the FT8XX, the design flow, and display list used to design the desired user interface or visual effect.

Bridgetek Pte Ltd (BRTChip)
178 Paya Lebar Road, #07-03, Singapore 409030
Tel: +65 6547 4827 Fax: +65 6841 6071
Web Site: http://www.brtchip.com

Table of Contents

1 Introduction

This application demonstrates interactive gauges using rectangles, lines and custom fonts for the numeric display offering a display with 2 digit precision on an FT8XX platform.

On WQVGA displays, two gauges are used for demonstration purposes. One gauge displays randomly generated data, while the other displays resistance based on how firmly the touch screen is pressed.

On QVGA displays only one gauge is used to display resistance, based on how firmly the touch screen is pressed.

1.1 Overview

The application will be useful to understand the FT8XX command sets for custom fonts, and FT8XX primitives for lines and rectangles.

The application note should be read in conjunction with the source code, which can be found in section 4 and at http://brtchip.com/SoftwareExamples-eve/.

1.2 Scope

This document can be used as a guide by designers to develop GUI applications using an FT8XX with any MCU via SPI. Note detailed documentation is available on http://brtchip.com/eve/.

2 Display Requirements

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

2.1 BackGround

The display background is set to a dark grey colour which the gauges will contrast against.

2.2 Analogue Gauges

The analogue gauges will display a random value or the resistance in relation to the touch pressure on the screen. The scale markings are coloured green for 0 to 60, yellow for 60 to 80 and red for 80 to 90.

2.3 Numeric Gauges

Numeric gauges will display the same value as the analogue gauges, but as decimal numbers using the custom font loaded at the start of the application.

3 Design Flow

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

The provided sample code includes the full flow but this document focuses on the main application in the final step. The earlier steps are generic to the EVE examples and are covered in the application note AN_391 EVE Platform Guide

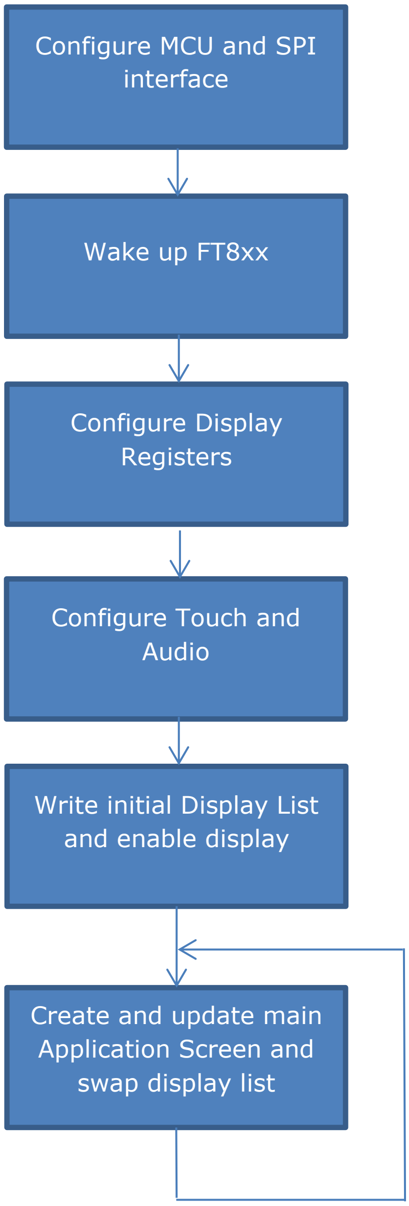

Figure 3.1 Generic EVE Design Flow

3.1 Gauges Flowchart

The flow chart below is specific to the Gauges application. Custom fonts are loaded into the Graphics RAM for use with the numeric display. The background is then generated for the display. Touch pressure is read and converted to a display value on one gauge while the other uses a randomly generated number. The program then operates in a loop to update the display list.

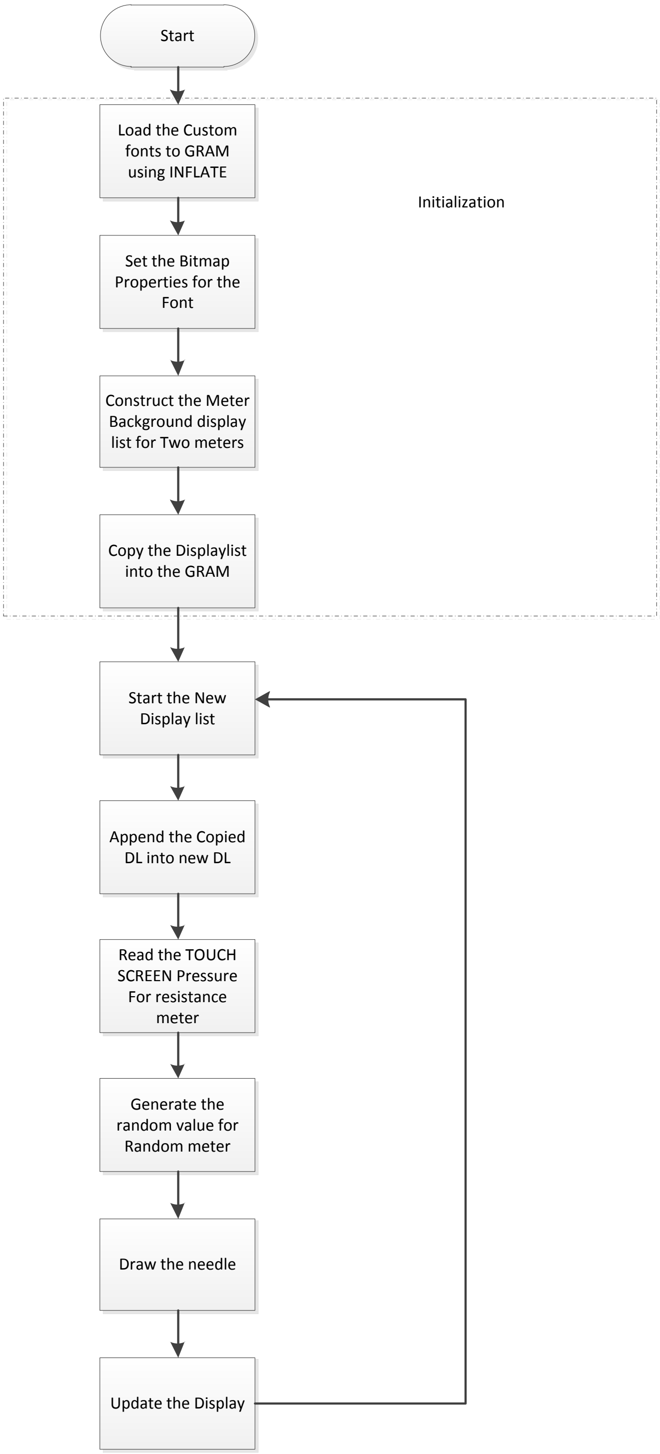

Figure 3.2 Flowchart

4 Description

Refer to AN 391 EVE Platform Guide for information pertaining to platform setup and the necessary development environment.

4.1 Application Start Screen

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

Figure 4.1 Start Screen

4.2 Loading the Font

The font table is hard coded in the application as an array labelled "digits". This array is loaded into the graphics RAM at the start of the "Gauges" function with the CMD_INFLATE command:

```
  Gpu_CoCmd_MemSet(phost,0,0,10*1024);
  App_Flush_Co_Buffer(phost);
  Gpu_Hal_WaitCmdfifo_empty(phost);
  Gpu_Hal_WrCmd32(phost,CMD_INFLATE);
  Gpu_Hal_WrCmd32(phost,0);
#if defined(MSVC_PLATFORM) || defined(MSVC_FT800EMU)
  Gpu_Hal_WrCmdBuf(phost,digits,sizeof(digits));
#else
  Gpu_Hal_WrCmdBufFromFlash(phost,digits,sizeof(digits));
#endif
```

The first 32 Characters are unwanted, so based on the width and height of the font the bitmap source is skipped.

```
  Gpu_CoCmd_Dlstart(phost);
  Gpu_CoCmd_SetFont(phost,13,0);
  App_WrCoCmd_Buffer(phost,BITMAP_HANDLE(13));
  App_WrCoCmd_Buffer(phost,BITMAP_SOURCE(144 - (32L*(54/2)*87)));
  App_WrCoCmd_Buffer(phost,BITMAP_LAYOUT(L4, 54/2,87));
  App_WrCoCmd_Buffer(phost,BITMAP_SIZE(NEAREST, BORDER, BORDER, 54, 87));
  App_WrCoCmd_Buffer(phost,DISPLAY());
  Gpu_CoCmd_Swap(phost);
  App_Flush_Co_Buffer(phost);
  Gpu_Hal_WaitCmdfifo_empty(phost);
```

Note: After these configurations are set, swap the display list and flush into the RAM CMD FIFO. Wait for co-processor to be Idle by using REG_CMD_WRITE and REG_CMD_READ registers.

4.3 Creating the Basic Gauge

Active areas are cut into the background for each gauge with the scissors function. Each gauge image is static, but the needle is continually updated.

```
Gpu_CoCmd_Dlstart(phost);                                                    // dl = 0;
 App_WrCoCmd_Buffer(phost,CLEAR_COLOR_RGB(55,55,55));
 App_WrCoCmd_Buffer(phost,CLEAR(1,1,1));
 App_WrCoCmd_Buffer(phost,CLEAR_COLOR_RGB(0,0,0));

 y = 10;
 for(z=0;z<(DispWidth/w);z++)
 {
        ox = (DispWidth/2)*z;
     px = (DispWidth/(2*NUM_DISPLAY_SCREEN))+(z*(DispWidth/(NUM_DISPLAY_SCREEN)));
#ifdef DISPLAY_RESOLUTION_HVGA_PORTRAIT
   ox = 300*z;
#endif

   App_WrCoCmd_Buffer(phost,SCISSOR_XY(ox+dt,y));


#ifdef DISPLAY_RESOLUTION_HVGA_PORTRAIT
   App_WrCoCmd_Buffer(phost,SCISSOR_SIZE(w,h));
#else
   App_WrCoCmd_Buffer(phost,SCISSOR_SIZE(w,h+30));
#endif
   App_WrCoCmd_Buffer(phost,CLEAR(1,1,1));
   App_WrCoCmd_Buffer(phost,BEGIN(LINES));
   App_WrCoCmd_Buffer(phost,LINE_WIDTH(10));
   for (bi = 0; bi < 81; bi += 10)
   {
     cs(bi);
     for ( i = 2; i < 10; i += 2)

 {
      a = da(bi + i, 45);
      polar(phost, 220, a,px,300);
      polar(phost, 240, a,px,300);
     }
   }

   App_WrCoCmd_Buffer(phost,LINE_WIDTH(16));
   for (i = 0; i < 91; i += 10)
   {
     cs(i);
     a = da(i, 45);
     polar(phost, 220, a,px,300);
     polar(phost, 250, a,px,300);
   }
```

White text to explain the gauge function is also applied.

```
   App_WrCoCmd_Buffer(phost,COLOR_RGB(255,255,255));
   for (i = 0; i < 91; i += 10)
   {
     a = da(i, 45);
     polarxy(260, a, &tx, &ty,px, 300);
     Gpu_CoCmd_Number(phost,tx >> 4, ty >> 4,26,OPT_CENTER, i);
   }
```

```
   if(z==1)Gpu_CoCmd_Text(phost,px,h-10,28,OPT_CENTERX,"Random");
#if defined FT801_ENABLE
   if(z==0)Gpu_CoCmd_Text(phost,px,h-10,28,OPT_CENTERX,"X Position");
#else
   if(z==0)Gpu_CoCmd_Text(phost,px,h-10,28,OPT_CENTERX,"Resistance");
#endif
}
```

4.4 Updating the Display

Updating the gauge displaying “random” data just uses the c code random function.

```
     {
       int d = (tgt - rval) / 16;
       rval += d;
       if (random(60) == 0)
       tgt = random(9000L);
       val = rval;
     }
```

The resistance function relies on reading the touch register REG_TOUCH_RZ which stores the resistance change when the display is touched.

```
       val = Gpu_Hal_Rd16(phost,REG_TOUCH_RZ);
       val = 10*min(899,val);
```

Both the graphical and numerical displays are updated.

```
    App_WrCoCmd_Buffer(phost,SCISSOR_XY(ox+dt,10));
    App_WrCoCmd_Buffer(phost,SCISSOR_SIZE(w,120));
    App_WrCoCmd_Buffer(phost,COLOR_RGB(255,255,255));
    App_WrCoCmd_Buffer(phost,BEGIN(LINES));
    App_WrCoCmd_Buffer(phost,LINE_WIDTH(10));

 th = ((uint16_t)val - 4500L)* 32768 / 36000L;
    for (o = -5; o < 6; o++)
    {
      polar(phost, 170, th + (o << 5),px,300);
      polar(phost, 235, th,px,300);
    }
    App_WrCoCmd_Buffer(phost,SCISSOR_XY(ox+dt,y));
    App_WrCoCmd_Buffer(phost,SCISSOR_SIZE(w,(uint16_t)(DispHeight*0.36)));
    App_WrCoCmd_Buffer(phost,CLEAR(1,1,1));
    App_WrCoCmd_Buffer(phost,COLOR_RGB(255,0,0));
#ifdef DISPLAY_RESOLUTION_HVGA_PORTRAIT
    Gpu_CoCmd_Number(phost,ox+dt+50,160,13,2,val/100);
    Gpu_CoCmd_Text(phost,ox+dt+86+50,160,13,0,".");
    Gpu_CoCmd_Number(phost,ox+dt+96+50,160,13,2,val%100);
#else
    Gpu_CoCmd_Number(phost,ox+dt+10,160,13,2,val/100);
    Gpu_CoCmd_Text(phost,ox+dt+96,160,13,0,".");
    Gpu_CoCmd_Number(phost,ox+dt+106,160,13,2,val%100);
#endif
   }
   App_WrCoCmd_Buffer(phost,DISPLAY());
   Gpu_CoCmd_Swap(phost);
   App_Flush_Co_Buffer(phost);
   Gpu_Hal_WaitCmdfifo_empty(phost);
```

4.4.1 FT801 Display

The FT801/FT811/FT813 capacitive display uses a different (capacitive) display touch controller compared to the FT800/FT810/FT812 resistive controller.

As there is no resistance measurement on a capacitive display, the program is altered to show the X coordinate of the touch point. To enable this alternative display 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 to the FT801 mode, ensure this line is defined. After making the change, rebuild and run the application. Note additional FT_8xx_Enable defines are available for other devices and the list may increase with code updates as required.

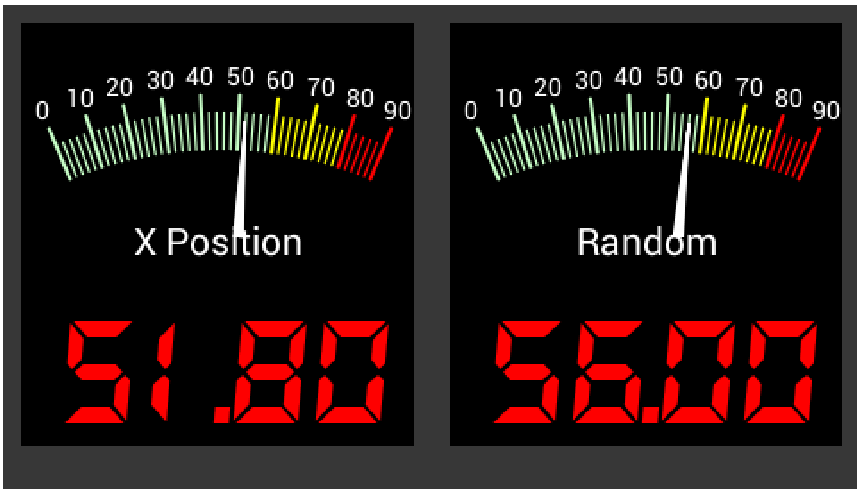

Figure 4.2 FT801 Display

5 Operation

When the user compiles and runs the application code, the first screen will be the calibration screen where the user must tap the screen in 3 places to align the touch and the display layers.

Figure 5.1 Tap Screen

This is followed by the logo and the composite logo/information screen which gives a short description of what the application does.

Figure 5.2 Composite Screen

Upon pressing "Click to Play", the app displays the gauges.

Figure 5.3 Gauges

6 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 5691
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@brtichip.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.

Appendix A– References

Document References

- FT800 datasheet
- FT801 datasheet
- Programming Guide covering EVE command language
- AN_240 FT800 From the Ground Up
- AN_245 VM800CB_SampleApp_PC_Introduction - covers detailed design flow with a PC and USB to SPI bridge cable
- AN_246 VM800CB_SampleApp_Arduino_Introduction - covers detailed design flow in an Arduino platform
- AN_391 EVE Platform Guide
- AN_252 FT800 Audio Primer

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
USB	Universal Serial Bus

Appendix B – List of Figures & Tables

List of Figures

List of Tables

NA

Appendix C– Revision History

Document Title: AN_263 FT_App_Gauges
Document Reference No.: BRT_000203
Clearance No.: BRT#122
Product Page: http://brtchip.com/eve/
Document Feedback: Send Feedback

Revision	Changes	Date
1.0	Initial Release	2013-08-21
1.1	Updated Version	2012-11-01
1.2	Added section 4.5.1	2014-06-30
1.3	Document migrated from FTDI to BRT (Company logo; Copyright information; contact information; hyperlinks updated) Source code and images updated	2018-01-05