Application Note

AN_268

FT_APP_SIGNALS

Version 1.2

Issue Date: 2018-01-05

This application note is to introduce the Signals 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 the 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 design example demonstrates an interactive user interface that simulates a device where input signals are drawn on the screen, similar to an oscilloscope or heart monitor. Signals are drawn through the use of Strips, Points, Blend function and Sound play based on the FT8XX platform.

In the Signals application, waveforms such as Sine, Triangular, Saw tooth and Electrocardiogram are generated by software using simple math functions. Sound effects are played when the peak values of the signals are reached. Menus are used to select the waveform type and sampling time scale. When not being used, the menu automatically hides.

This application uses all the features of the FT8XX: touch, audio and display.

Loading of the necessary elements to show and manipulate the graphics elements is as follows:

- Draw graphics primitives directly through the display list
- Incorporate display list commands to access sound and touch events through reads and writes of the FT8XX registers.
- Store the display list in DL_RAM

1.1 Overview

The document will provide information on drawing graphics elements through primitives, tagging of audio and 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.

Source code for this application can be found at: http://brtchip.com/SoftwareExamples-eve/.

1.2 Scope

This document can be used as a guide by designers to develop GUI applications by using FT8XX with any MCU via SPI. Note that 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 Waveform display

In the Signals application, there are four waveform types that can be displayed: sine, sawtooth, triangle and "electrocardiogram". Values for the waveforms are calculated as the display lists are generated, although these values could easily be read from an actual sensor in the system.

2.2 Menu

The menu is displayed initially, and then hidden after a few seconds. The touch screen is used to activate (unhide) the menu, to select the desired waveform and to select the rate at which the waveform is displayed.

3 Design Flow

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

The sample code provided 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

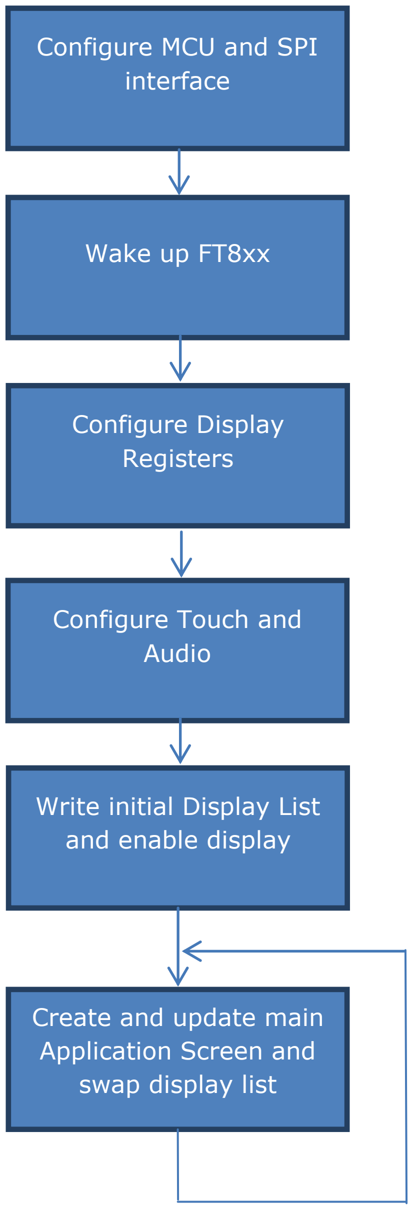

Configure MCU and SPI interface
Wake up FT8xx
Configure Display Registers
Configure Touch and Audio
Write initial Display List and enable display
Create and update main Application Screen and swap display list

Figure 3.1 Generic EVE Design Flow

3.1 Signals Flowchart

The flowchart below is specific to the Signals application. It begins by displaying a horizontal line and the menu contents. A sine wave is displayed by default, then the menu choices determine which waveform is shown and at what rate.

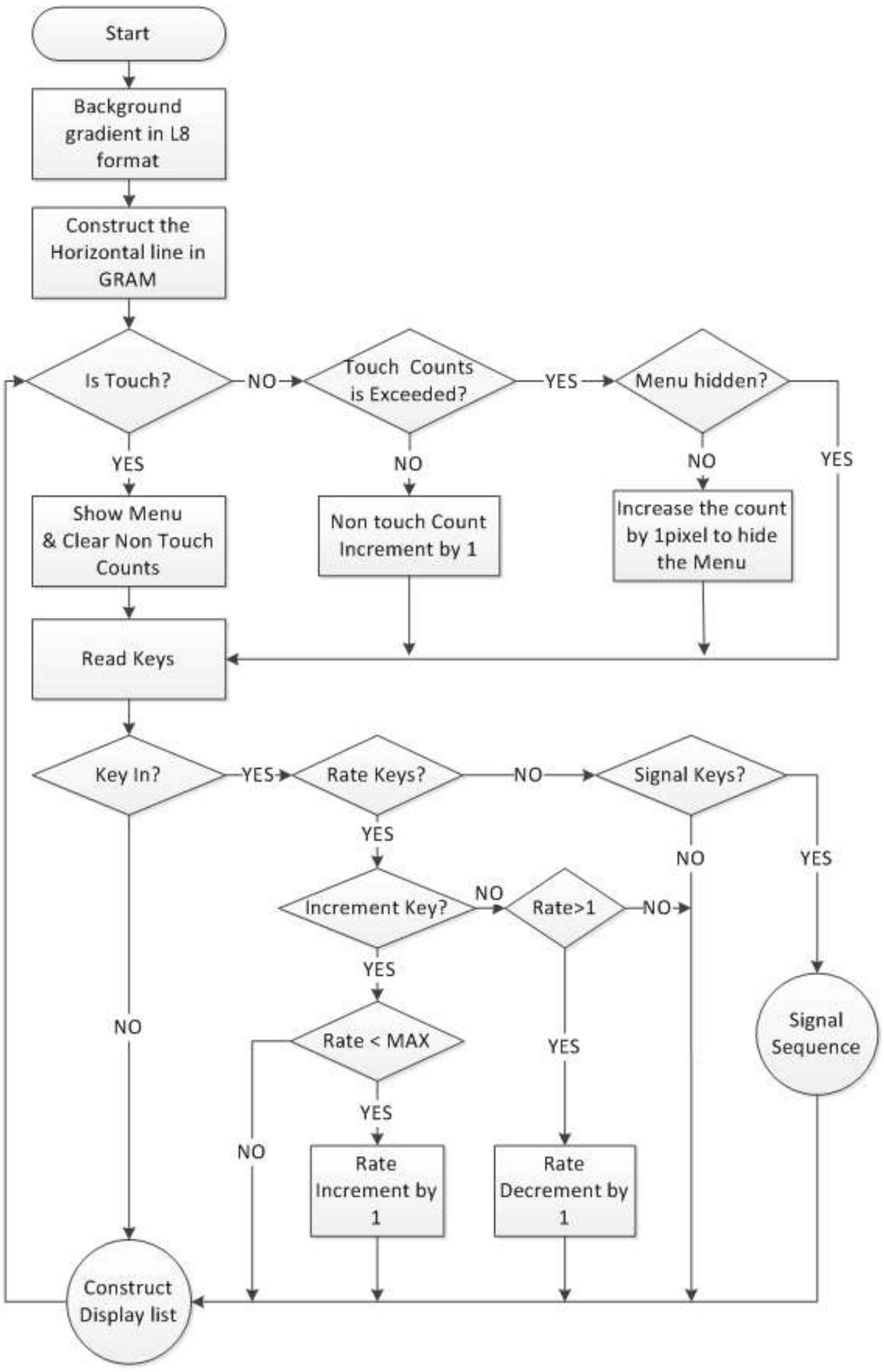
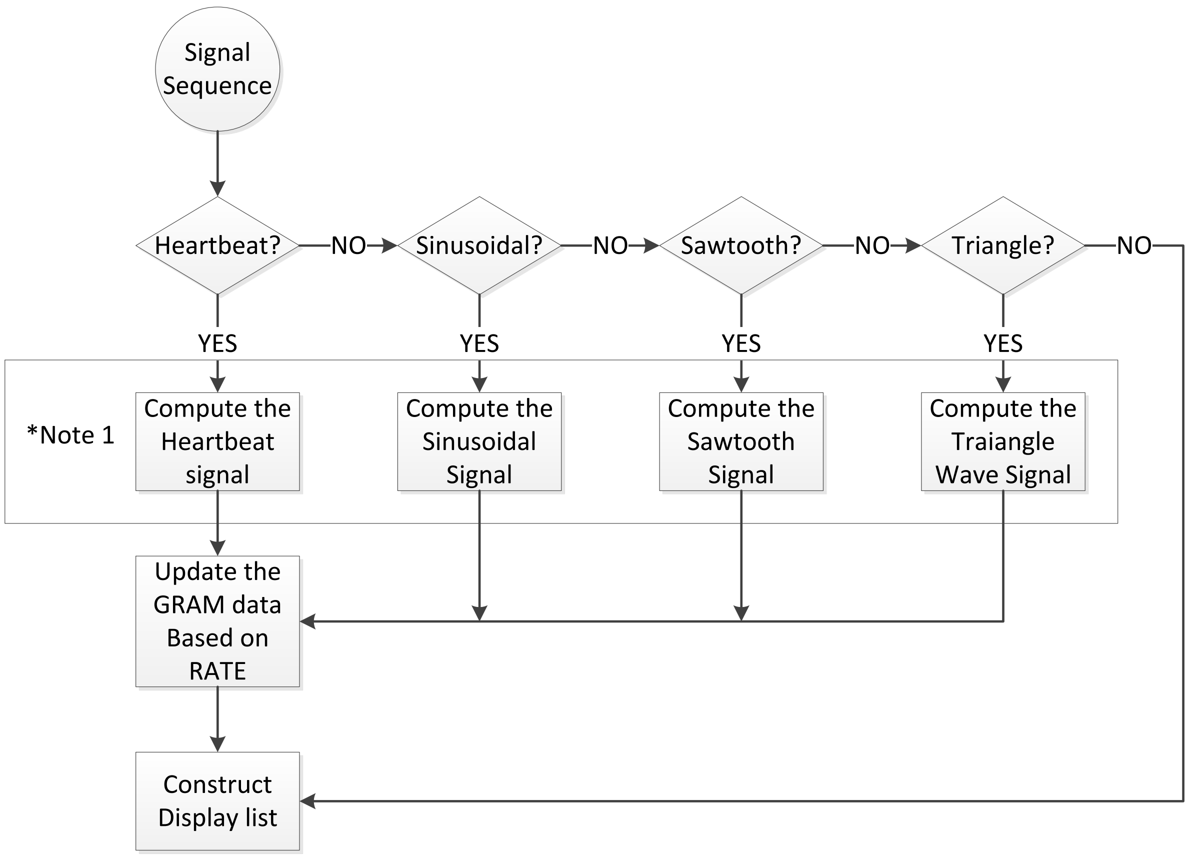


Figure 3.2 Flowchart - Main


Signal Sequence
Heartbeat?
NO
Sinusoidal?
NO
Sawtooth?
NO
Triangle?
NO
YES
YES
YES
YES
*Note 1
Compute the Heartbeat signal
Compute the Sinusoidal Signal
Compute the Sawtooth Signal
Compute the Traiangle Wave Signal
Update the GRAM data Based on RATE
Construct Display list

NOTE 1: All Signals are hard coded and generated by simple mathematical functions.

Figure 3.3 Flowchart – Signal Selection

4 Functional Blocks 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.

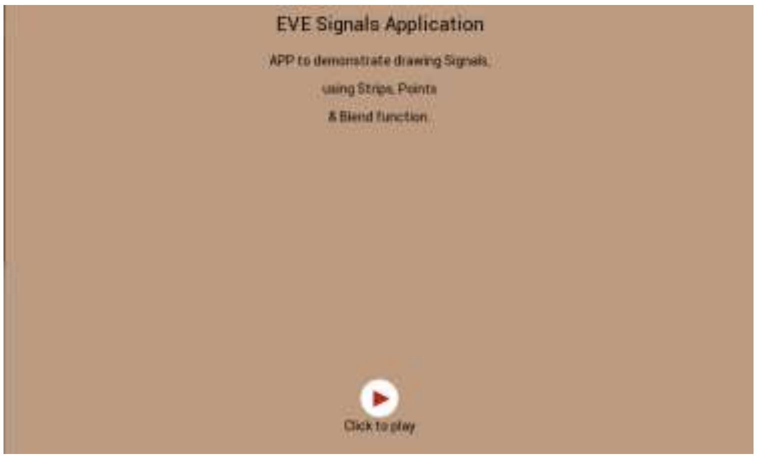


Figure 4.1 Start Screen

4.2 Waves()

Once the user taps the arrow button, a gradient background is computed and displayed as a backdrop for the displayed signals. This gradient background is then loaded into GRAM. For the Gradient, the height of the display is used and the gradient is repeated over the width of the display.

```
        for(tval=0;tval<=DispHeight/2;tval++)
        {
          temp[DispHeight-tval] = temp[tval] = (tval*0.90);
        }
```

Here the gradient is written to the GRAM

```
        Gpu_Hal_WrMem(phost,2048L,temp,sizeof(temp));
```

When the display list is constructed, commands to draw the bitmap are given:

```
        App_WrCoCmd_Buffer(phost,BITMAP_SOURCE(2048L));
        App_WrCoCmd_Buffer(phost,BITMAP_LAYOUT(L8,1,DispHeight));
        App_WrCoCmd_Buffer(phost,BITMAP_SIZE(NEAREST, REPEAT, BORDER, DispWidth, DispHeight));
```

Compute the initial zero-line data for the GRAM and load into GRAM

```
for(tval=0;tval<DispWidth;tval+=rate)
        {
          Gpu_Hal_Wr32(phost,RAM_G+((tval/rate)*4),VERTEX2F(tval*16,y*16));
        }
```

The initial data for the heartbeat is also calculated before entering the loop:

```
        y = DispHeight/2;
        for(tval=0;tval<10;tval++)
        {
          y = y+(beats_Incr[tval]*5);
          beats[tval] = y;
        }
```

Note: Upon configuration, swap the display list and load into the Graphics Memory, the processing should wait for the Command Processor to be Idle by using REG_CMD_WRITE and REG_CMD_READ registers.

4.2.1 Menu

Once the background and initial data are calculated and loaded into GRAM, the right-side menu is constructed. This section checks for any touch screen activity. If there is any touch, it shows the menu, and otherwise hides it after displaying several points of the waveform. The "else" code gives the appearance of sliding the menu off to the right edge.

```
        // ========  Menu  =========================================================
            if(istouch())   fg = 1;
            if(fg){ th_to=0; if(hide_x>0)hide_x=0; else
              fg = 0;  }
            else
            { th_to++;
              if(th_to > 250) {
               if( hide_x < 85) hide_x++;   else
               th_to = 0;     }
            }
```

If there is any touch activity, the "Read_keys()" call will check the FT8XX for what area of the screen was touched. If it corresponds to a specific item, such as the rate increase/decrease buttons or waveform selection, a "tag" is assigned. These tags are convenient methods of determining which drawn object was touched without having to manually calculate the X-Y position of the touch event and verify if it corresponds to the drawn element. The FT8XX, by assigning tags to a drawn element, automatically knows where the item is and sets the tag value. Later processing can then act on the tag event.

The section of code looks at the tag to see if the rate +/- buttons were tapped. If so, increase or decrease the rate:

```
        //=========Option =========================================================
            tag = Read_keys();
            if(tag!=0)
            {
              x = 0;   temp_p = 0;en = 0; temp_x = 0; temp_y = 0; //reset
              if(tag>2)  opt = tag;
              if(tag==1)if(rate>1)rate--;
              if(tag==2)if(rate<6)rate++;
              y = (DispHeight/2);
              for(tval=0;tval<DispWidth;tval+=rate)
              {
                Gpu_Hal_Wr32(phost,RAM_G+((tval/rate)*4),VERTEX2F(tval*16,y*16));
              }
              add2write = 0;
            }
```

If it was not a rate touch, then check whether if it was a waveform selection:

```
        //======== Signals ========================================================
          amp = 100;
          switch(opt)
          {
            case 5:

              Triangle_wave(amp);
            break;
            case 4:
                Sawtooth_wave(amp);
            break;

            case 3:
                  Sine_wave(amp);
            break;

            case 6:
              amp = 50;
              Heartbeat();
            break;
         }
```

4.2.2 Display List

With the rate and waveform selection complete, it’s time to construct the display list.
Clear the screen and set the initial color:

```
Gpu_CoCmd_Dlstart(phost);
App_WrCoCmd_Buffer(phost,CLEAR(1,1,1));
App_WrCoCmd_Buffer(phost,COLOR_RGB(0x12,0x4A,0x26));
```

Draw the background:

```
App_WrCoCmd_Buffer(phost,BITMAP_SOURCE(2048L));
App_WrCoCmd_Buffer(phost,BITMAP_LAYOUT(L8,1,DispHeight));
App_WrCoCmd_Buffer(phost,BITMAP_SIZE(NEAREST, REPEAT, BORDER, DispWidth,
DispHeight));
App_WrCoCmd_Buffer(phost,BEGIN(BITMAPS));
App_WrCoCmd_Buffer(phost,TAG(0));
App_WrCoCmd_Buffer(phost,VERTEX2F(0,0));
App_WrCoCmd_Buffer(phost,COLOR_RGB(0x1B,0xE0,0x67));

App_WrCoCmd_Buffer(phost,COLOR_RGB(0x1B,0xE0,0x67));
App_WrCoCmd_Buffer(phost,LINE_WIDTH(2*16));
```

Begin drawing the waveform

```
App_WrCoCmd_Buffer(phost,BEGIN(LINE_STRIP));
Gpu_CoCmd_Append(phost,RAM_G,(x/rate)*4);
App_WrCoCmd_Buffer(phost,END());

App_WrCoCmd_Buffer(phost,BEGIN(LINE_STRIP));
if((x/rate)<(DispWidth/rate)-(50/rate))
Gpu_CoCmd_Append(phost,RAM_G+(x/rate)*4+((50/rate)*4),((DispWidth/rate)*4)-
((x/rate)*4)-((50/rate)*4));
App_WrCoCmd_Buffer(phost,END());
```

Draw the menu and assign tags to the various buttons. Note that the menu is always drawn. Whether it’s “moved” into position or hidden depends on whether a touch event occurred earlier:

```
App_WrCoCmd_Buffer(phost,POINT_SIZE(6*16));
App_WrCoCmd_Buffer(phost,BEGIN(FTPOINTS));
App_WrCoCmd_Buffer(phost,VERTEX2F(x*16,y*16));
App_WrCoCmd_Buffer(phost,END());
App_WrCoCmd_Buffer(phost,COLOR_RGB(0xff,0xff,0xff));
App_WrCoCmd_Buffer(phost,COLOR_A(100));
App_WrCoCmd_Buffer(phost,BEGIN(EDGE_STRIP_R));
App_WrCoCmd_Buffer(phost,VERTEX2F((hide_x+DispWidth-80)*16,0));
App_WrCoCmd_Buffer(phost,VERTEX2F((hide_x+DispWidth-80)*16,DispHeight*16));
App_WrCoCmd_Buffer(phost,COLOR_A(255));
```

```
Gpu_Radiobutton(hide_x+DispWidth-70,DispHeight-48,0xffffff,0,8,3,opt);

Gpu_Radiobutton(hide_x+DispWidth-70,DispHeight-28,0xffffff,0,8,4,opt);
Gpu_Radiobutton(hide_x+DispWidth-70,DispHeight-8 ,0xffffff,0,8,5,opt);
Gpu_Radiobutton(hide_x+DispWidth-70,DispHeight-68,0xffffff,0,8,6,opt);
Gpu_CoCmd_Text(phost,hide_x+DispWidth-60,DispHeight-48,26,OPT_CENTERY,"Sine");
Gpu_CoCmd_Text(phost,hide_x+DispWidth-60,DispHeight-
28,26,OPT_CENTERY,"Sawtooth");
Gpu_CoCmd_Text(phost,hide_x+DispWidth-60,DispHeight-8
,26,OPT_CENTERY,"Triangle");

Gpu_CoCmd_Text(phost,hide_x+DispWidth-60,DispHeight-68,26,OPT_CENTERY,"ECG");
Gpu_CoCmd_Text(phost,(hide_x+DispWidth-60),20,30,OPT_CENTERY|OPT_CENTERX,"-");
Gpu_CoCmd_Text(phost,(hide_x+DispWidth-20),20,30,OPT_CENTERY|OPT_CENTERX,"+");
Gpu_CoCmd_Text(phost,(hide_x+DispWidth-80),50,28,0,"Rate:");
Gpu_CoCmd_Number(phost,(hide_x+DispWidth-30),50,28,0,rate);
Gpu_CoCmd_Text(phost,(hide_x+DispWidth-80),80,28,0,"Pk:");
Gpu_CoCmd_Number(phost,(hide_x+DispWidth-40),80,28,0,amp);
App_WrCoCmd_Buffer(phost,COLOR_A(50));
App_WrCoCmd_Buffer(phost,POINT_SIZE(15*16));
App_WrCoCmd_Buffer(phost,BEGIN(FTPOINTS));
App_WrCoCmd_Buffer(phost,TAG(1));
App_WrCoCmd_Buffer(phost,VERTEX2F((hide_x+DispWidth-60)*16,20*16));
App_WrCoCmd_Buffer(phost,TAG(2));
App_WrCoCmd_Buffer(phost,VERTEX2F((hide_x+DispWidth-20)*16,20*16));

App_WrCoCmd_Buffer(phost,DISPLAY());
Gpu_CoCmd_Swap(phost);
App_Flush_Co_Buffer(phost);
Gpu_Hal_WaitCmdfifo_empty(phost);
```

4.3 Functionality

The signals are drawn using strips and additive blending. The signals are computed using a simple software equation and can be replaced with any stored database values from an actual sensor. The application also draws an option menu on the right side of the screen with options to change the rate of the signal and options to select the type of signal to be displayed.

The application constantly monitors the user click on rate buttons (either increase or decrease). According to the values given by the user, the signal is plotted with sound being played when the signal hits the peak.

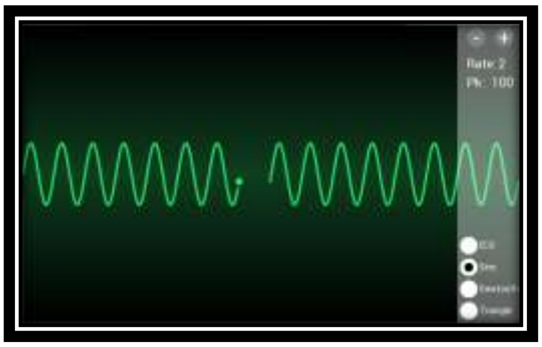


Figure 4.2 Signals Display

4.3.1 Signal Selection

For the demo purpose the signals are generated by using simple functions. The signal selection button is a "Radio Button", drawn by using POINTS. The function App_Read_Tag () will obtain the tag and change the output waveform.

```
tag = App_Read_Tag(phost);
```

4.3.2 Signals

Four example waveforms are calculated on the fly for each point across the screen. The last line in each of the waveform functions is responsible for the new point to be shown.

4.3.2.1 Sine_wave

The sine wave points are calculated using an integer qsin() function based on a stored sine table. If at the peak of the waveform, add in a beep.

```
void Sine_wave(uint8_t amp)
{
  static uint8_t played = 0,change=0;
  x+=rate;
  if(x>DispWidth) x  = 0;
  y = (DispHeight/2) + ((int32_t)amp*qsin(-65536*x/(25*rate))/65536);
  if(played==0 &&  change < y){
  played = 1;
  Play_Sound((108<<8 | 0x10),100); }
  if(change > y)
  played = 0;
  change = y;
  Gpu_Hal_Wr16(phost,RAM_G+(x/rate)*4,VERTEX2F(x*16,y*16));
}
```

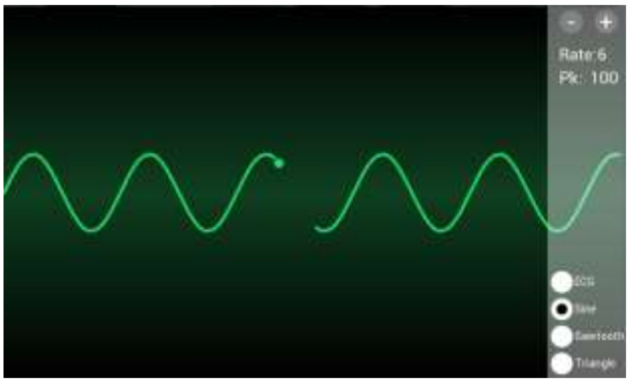


Figure 4.3 Sine Wave

4.3.2.2 Sawtooth_wave

The sawtooth wave is calculated as a simple repeating ramp. As with the sine wave, play a beep at the peak.

```
void Sawtooth_wave(uint8_t amp)
{
  static uint16_t temp=0;
```

```
     static uint8_t pk = 0;
     x+=rate;
     if(x>DispWidth){ x  = 0;}
     temp+=2; if(temp>65535L) temp = 0;
     y = (temp % amp);
     pk = y/(amp-2);
     if(pk) Play_Sound((108<<8 | 0x10),100);
     y = (DispHeight/2)-y;
     Gpu_Hal_Wr16(phost,RAM_G+(x/rate)*4,VERTEX2F(x*16,y*16));
    }
```

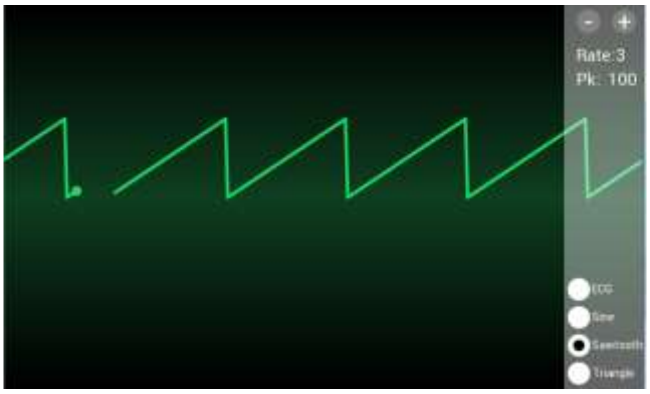


Figure 4.4 Sawtooth Wave

4.3.2.3 Triangle_wave

The Triangle wave linearly increases amplitude to a maximum, plays a beep, and then decreases the amplitude.

```
    void Triangle_wave(uint8_t amp)
    {
     static uint16_t temp=0;
     static uint8_t pk = 0,dc=0,p=0;
     x+=rate;
     if(x>DispWidth){ x  = 0;}
     temp+=2; if(temp>65535L) temp = 0;
     y = (temp % amp);
     pk = (y/(amp-2))%2;
     dc = (temp / amp)%2;
     if(pk) { if(p==0){ p=1; Play_Sound((108<<8 | 0x10),100); } else  p=0;}
     if(dc) y = (DispHeight/2) -(amp-y);  else
     y = (DispHeight/2) - y;
     Gpu_Hal_Wr16(phost,RAM_G+(x/rate)*4,VERTEX2F(x*16,y*16));
    }
```

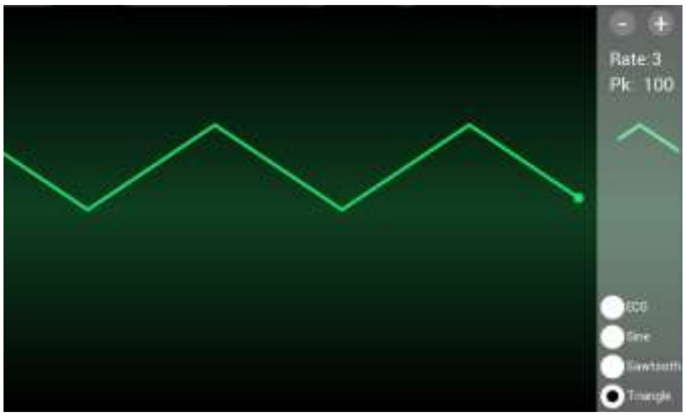


Figure 4.5 Triangle Wave

4.3.2.4 Heartbeat

The heartbeat is a simulation of an ECG waveform. As with the other waveforms, play a beep when the peak is reached.

```
void Heartbeat()
{
    x+=rate; if(x>DispWidth){ x  = 0;temp_p = 0;temp_y=0;
                                y=DispHeight/2; en=0;temp_x=0;}
   tx = 5*rate;
   tx = ((temp_p+1)*tx) + temp_p*temp_x;
   if(tx<=x){ if(0==en)      en = 1;}
  if(en==1){
  if(y!=beats[temp_y])
  {
    y += beats_Incr[temp_y] * 5;
    if(y==(DispHeight/2)+beats_Incr[4] * 5)
              Play_Sound((108<<8 | 0x10),100);
  }
  else
  {
    temp_y++;
    if(temp_y>9) {
    temp_y = 0;   temp_p++;
    en = 0;  temp_x = x - tx;}
  }}
  Gpu_Hal_Wr32(phost,RAM_G+(x/rate)*4,VERTEX2F(x*16,y*16));
 }
```

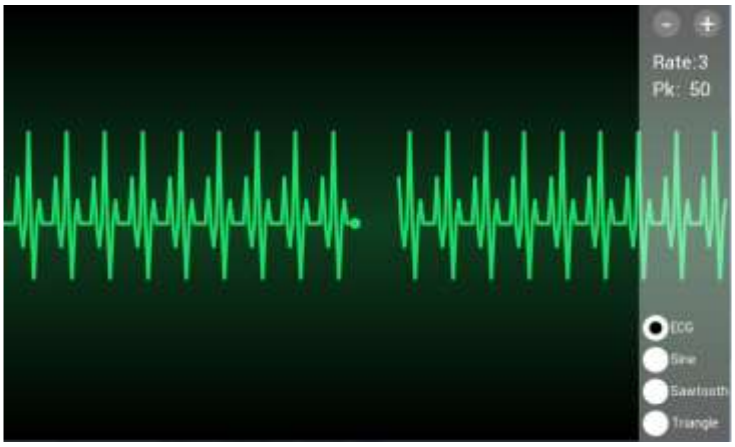


Figure 4.6 Heartbeat

4.3.3 Signal Plotting

With each waveform point calculated as X increases across the screen, the commands of LINE_STRIP and CMD_APPEND are used to actually display the wave. There is also a gap between the end of the previous wave and new points of the current one.

```
App_WrCoCmd_Buffer(phost,LINE_WIDTH(2*16));
App_WrCoCmd_Buffer(phost,BEGIN(LINE_STRIP));
Gpu_CoCmd_Append(phost,RAM_G,(x/rate)*4);
App_WrCoCmd_Buffer(phost,END());

App_WrCoCmd_Buffer(phost,BEGIN(LINE_STRIP));
if((x/rate)<(DispWidth/rate)-(50/rate))
Gpu_CoCmd_Append(phost,RAM_G+(x/rate)*4+((50/rate)*4),((DispWidth/rate)*4)-
((x/rate)*4)-((50/rate)*4));
```

4.3.4 Menu Hiding

The MCU continuously watches the REG_TOUCH_RAW_X. If the touch is not detected, the MCU starts the count to hide the menu. If the count is exceeded, the menu bar is slowly moved pixel by pixel.

```
// ========  Menu  ==========================================================
    if(istouch())   fg = 1;
    if(fg){ th_to=0; if(hide_x>0)hide_x=0; else
      fg = 0;  }
    else
    { th_to++;
      if(th_to > 250) {
       if( hide_x < 85) hide_x++;   else
       th_to = 0;     }
    }
```

5 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 Embedded Video Engine datasheet
- FT8XX Series Programmer's Guide
- AN_391 EVE Platform Guide
- Datasheet for VM800C
- Datasheet for VM800B

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_268 FT_App_Signals
Document Reference No.:	BRT_000200
Clearance No.:	BRT#119
Product Page:	http://brtchip.com/product/
Document Feedback:	Send Feedback

Revision	Changes	Date
1.0	Initial Release	2013-08-21
1.1	Updated release	2013-11-01
1.2	Document migrated from FTDI to BRT (Updated company logo; copyright info; contact information; hyperlinks Updated/added section 4.2; 4.3	2018-01-05