Application Note

AN_310

FT800 REFRIGERATOR APPLICATION

Document Reference No.: FT_001010

Version 1.0

Issue Date: 2014-03-25

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

Contents

1 INTRODUCTION

This application demonstrates aSmart Refrigerator application using inbuilt fonts, stencil operation and the scissor command implemented on the FT800 platform.

In this application, the commands screensaver and sketch are performed with synchronised audio.

The application coding is simplified by using FT800 widgets and primitives.

1.1 Overview

The document gives the basic understanding for a Smart Refrigerator application using the FT800 command language. All coding is provided as is and is supplied for tutorial purposes.

1.2 Scope

This document will be used by software programmers to develop GUI applications by using the FT800 with any MCU via SPI.

2 Application Flow

2.1 Flowchart

2.1.1 Main Flow

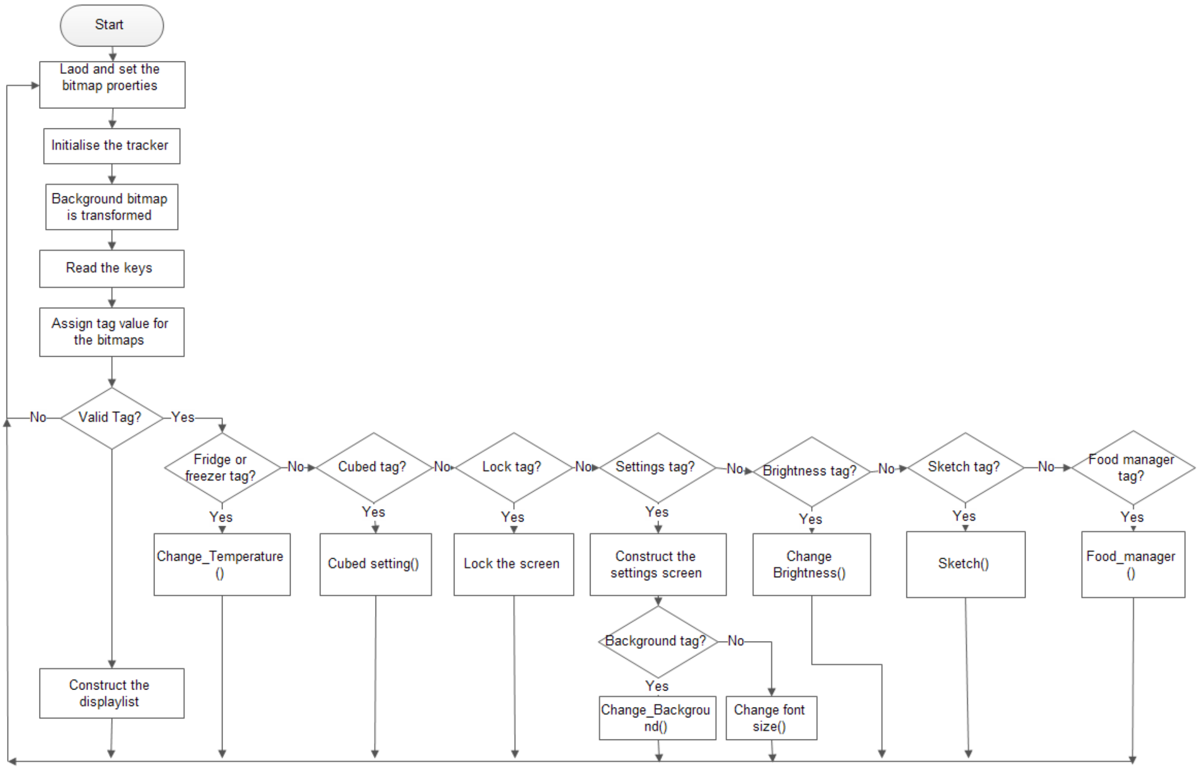

Figure 2-1 Refrigerator main flowchart

This is the main flowchart which explains the main control flow of the application.

2.1.2 Temperature Change Flowchart

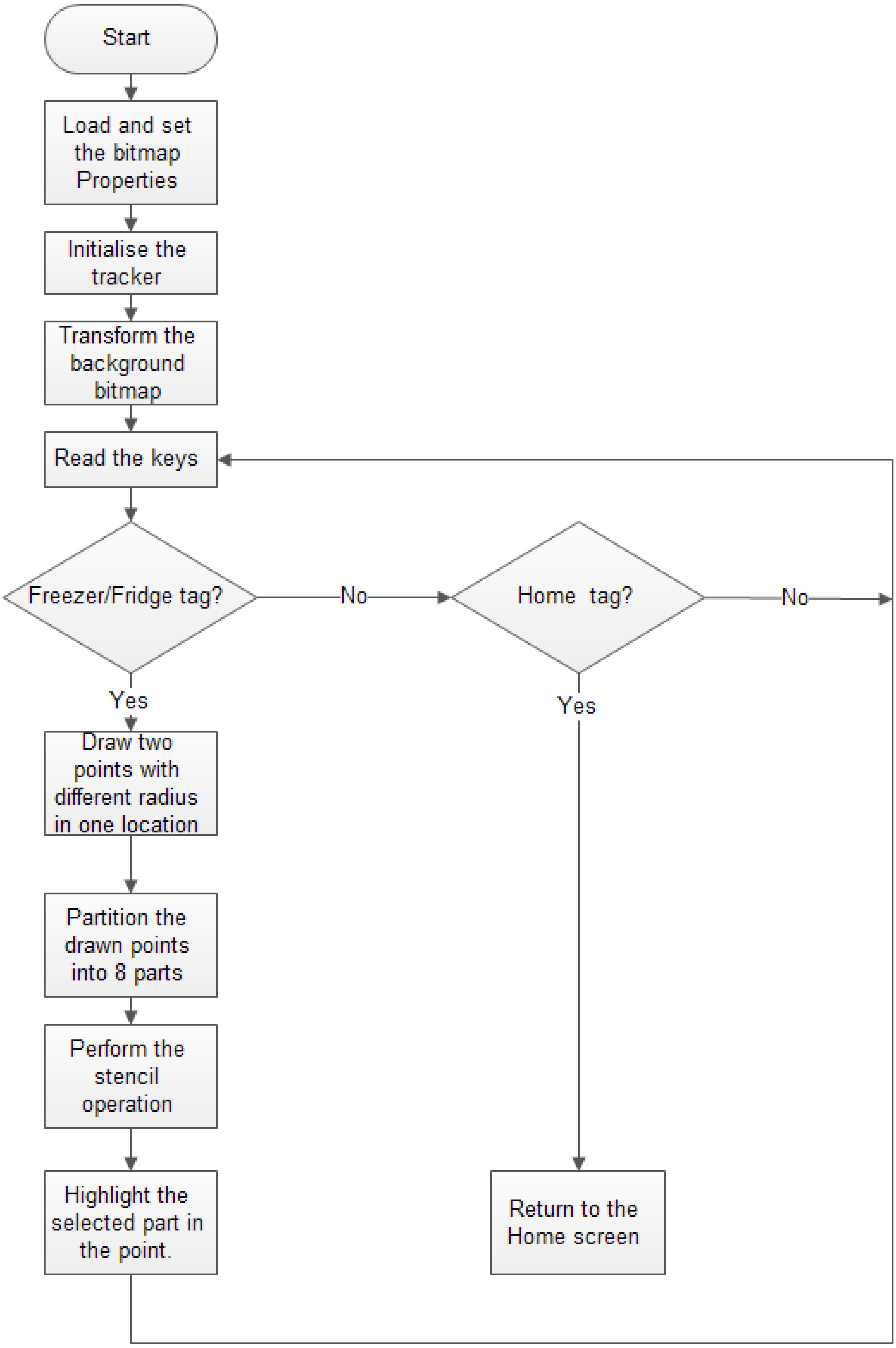

Figure 2-2 Temperature Adjustment flowchart

The above flowchart explains the control flow of the construction of the screenshot in changing the temperature.

2.1.3 Ice Setting Flowchart

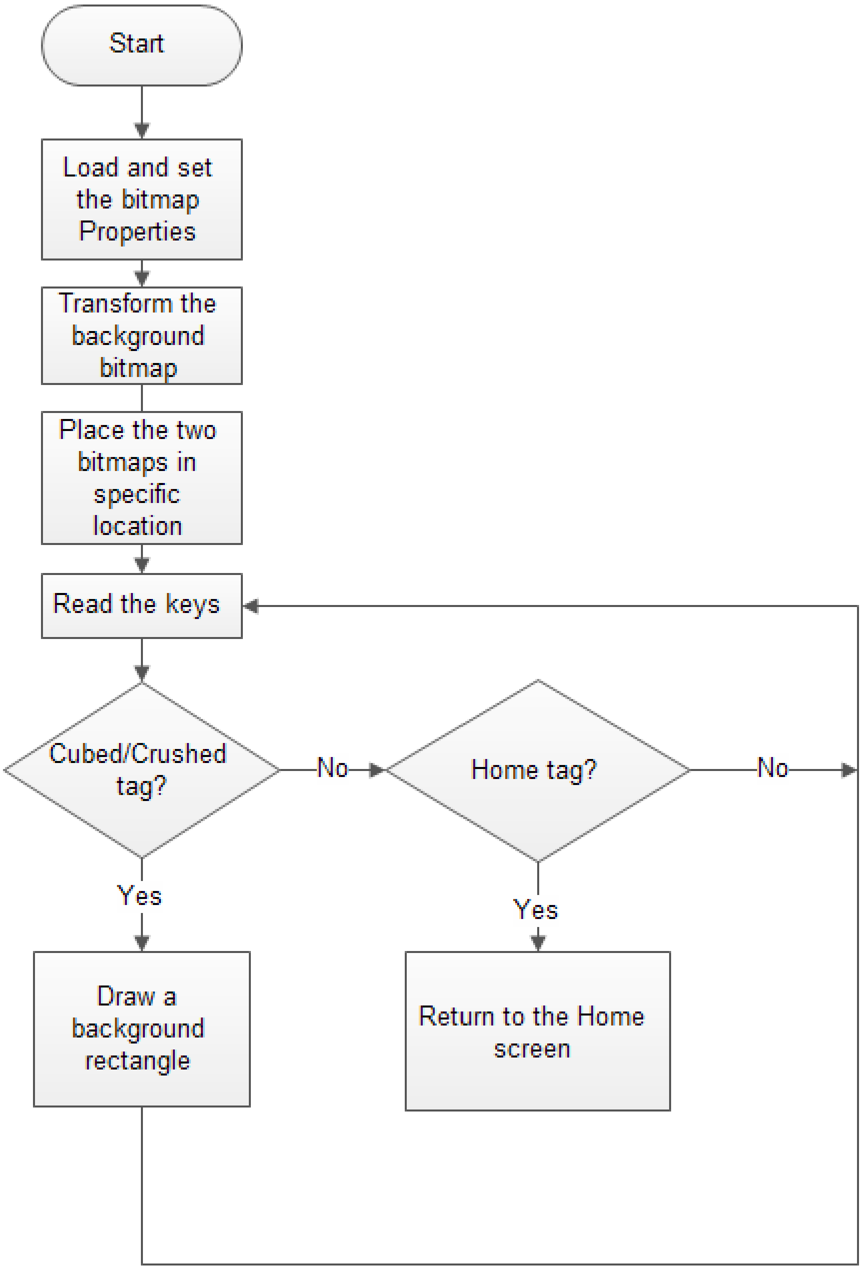

Figure 2-3 Ice Option flowchart

The above flowchart explains the control flow of the ice option to be selected.

2.1.4 Settings Flowchart

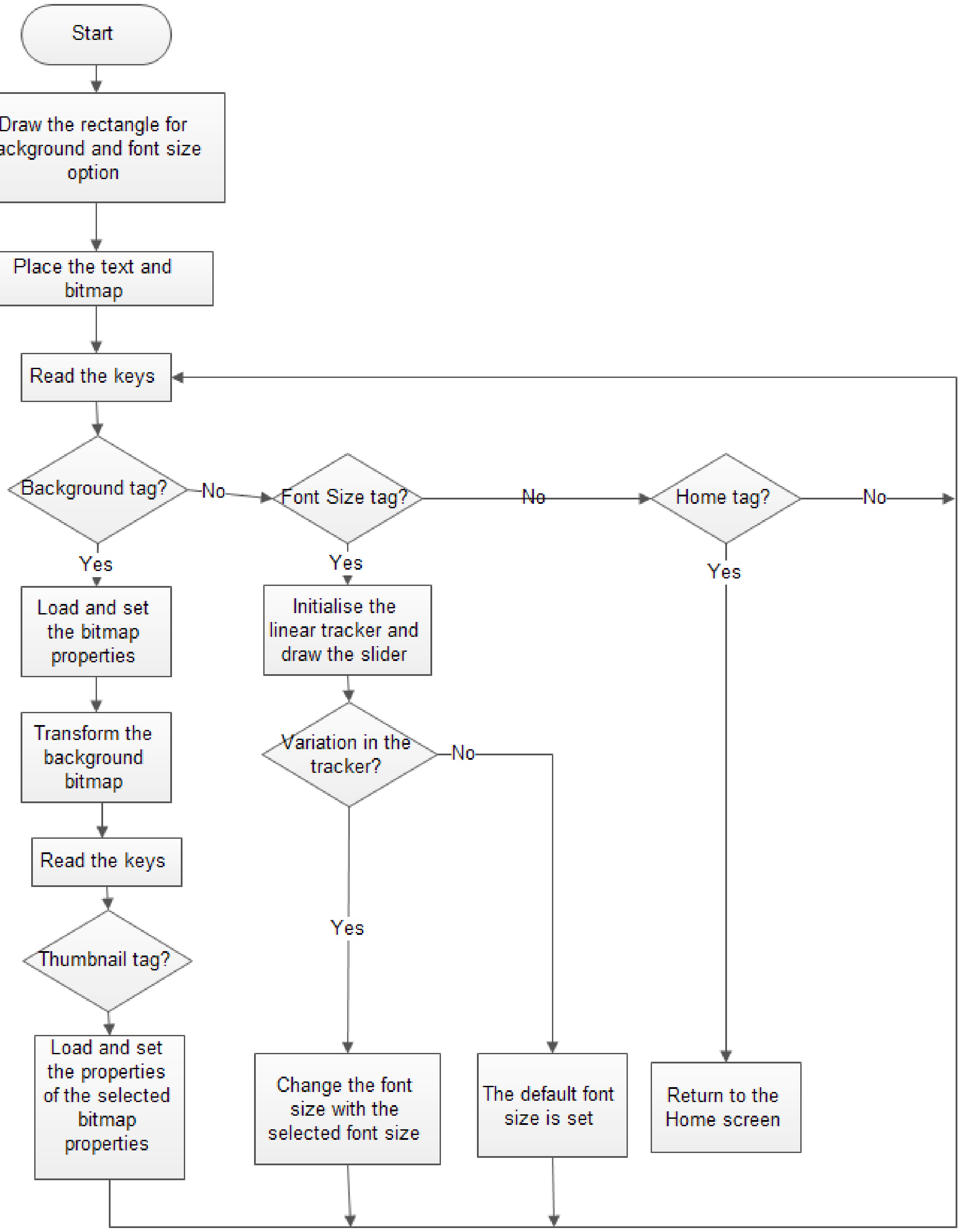

Figure 2-4 Settings flowchart

The above flowchart explains the control flow of font size change and the background change.

2.1.5 Brightness Flowchart

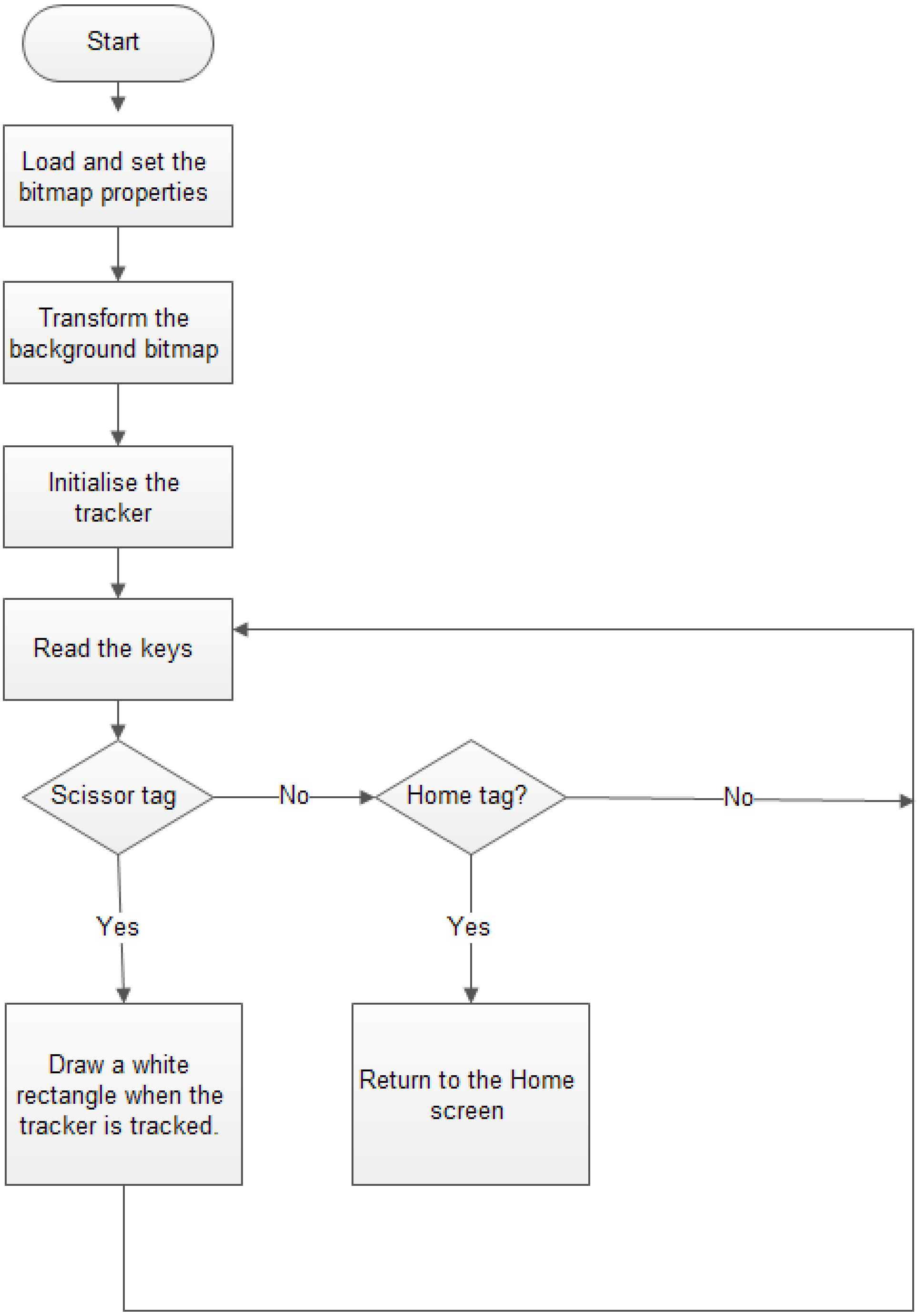

Figure 2-5 Brightness flowchart

The above flowchart explains the control flow of the adjustment of brightness of the display with the PWM vale.

2.1.6 Sketch Flowchart

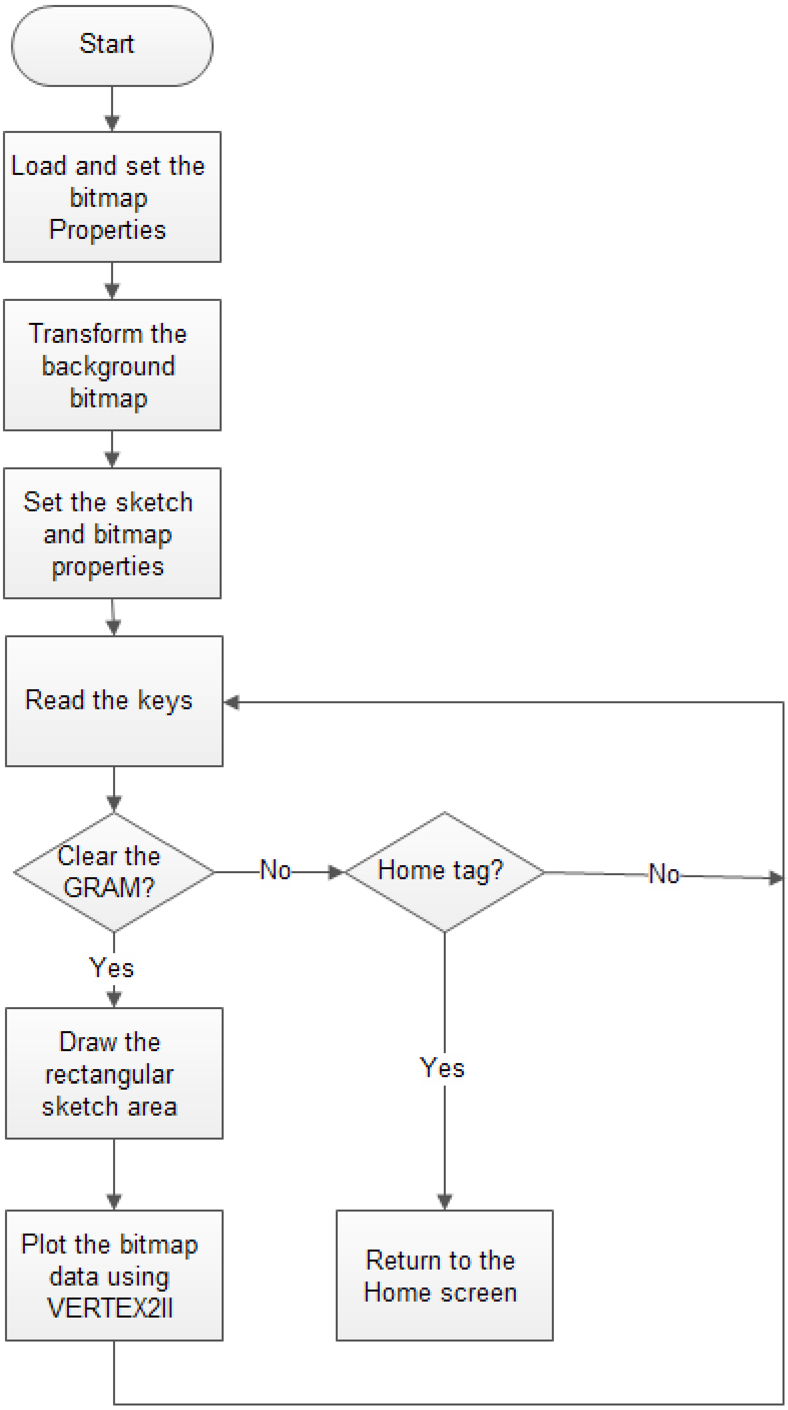

Figure 2-6 Sketch flowchart

The above flowchart explains the control flow of the sketch command implemented in the application.

2.1.7 Food Manger Flowchart

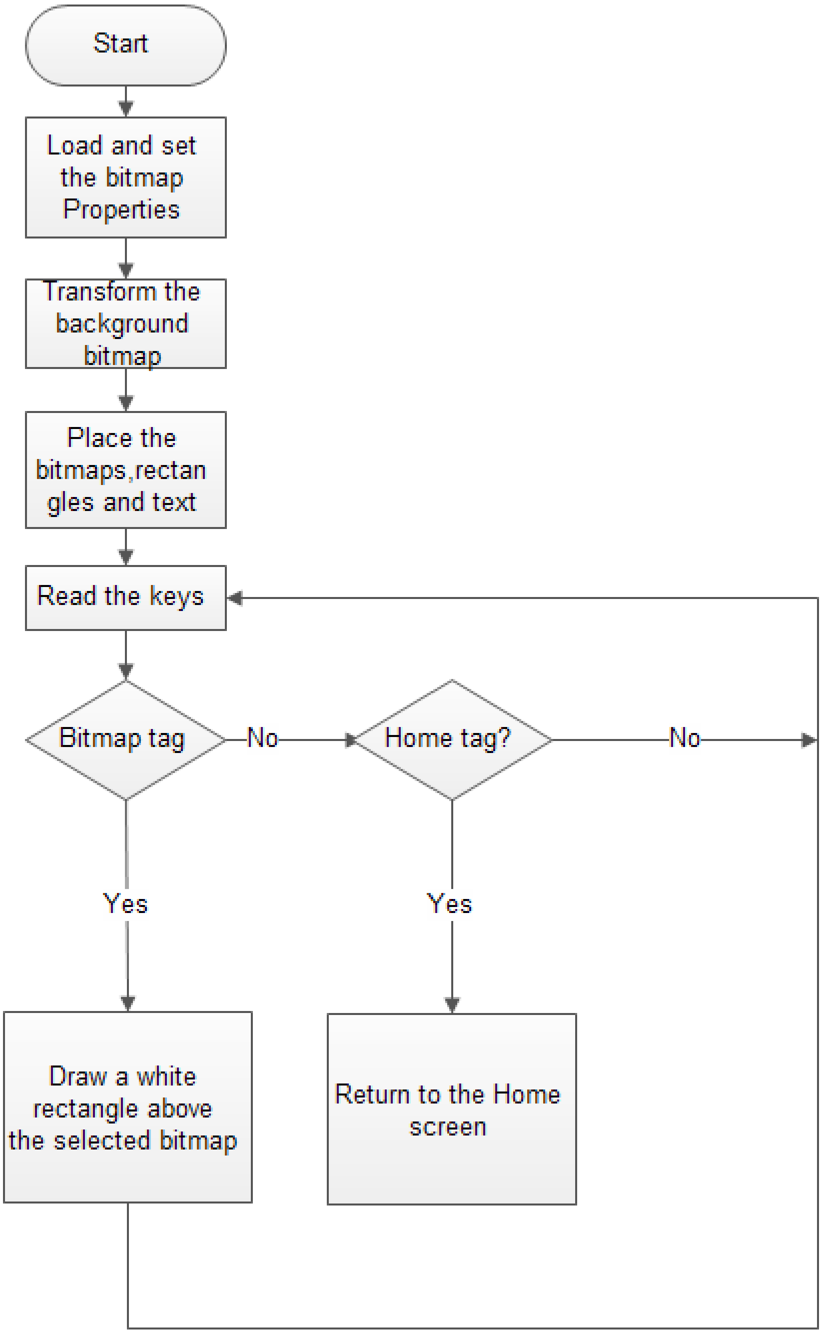

Figure 2-7 Food Manager Flowchart

The above flowchart explains the control flow of arrangement and selection of the bitmaps.

3 Description

3.1 Functionality

In this application, the smart features such as child lock, screensaver, change of themes, adjustment of the brightness, sketch and the food manager are implemented. The features are implemented using the primitives and widgets of the FT800.

3.1.1 Construction and movement of background animation

The background screenshot is constructed by placing the background bitmap of size 240*136 in the display list. The bitmap is then transformed using BITMAP_TRANSFORM. The snowflakes bitmap of size 50*50 for animation is made to move randomly from the bottom of the display in the upward direction. The orientation and the movement of the bitmap is shown in the code below:

```
/* compute the random values at the starting*/
pRefrigeratorSnow = S_RefrigeratorSnowArray;
for(j=0;j<(NumSnowRange*NumSnowEach);j++)
{
        pRefrigeratorSnow->xOffset = ft_random(FT_DispHeight*16);
        pRefrigeratorSnow->yOffset = ft_random(FT_DispWidth*16);
        pRefrigeratorSnow->dx =  ft_random(RandomVal*8) - RandomVal*8;
        pRefrigeratorSnow->dy = -1*ft_random(RandomVal*8);
        pRefrigeratorSnow++;
}
```

```
/* Draw background snow bitmaps */
for(j=0;j<(NumSnowRange*NumSnowEach);j++)
{
        Ft_App_WrCoCmd_Buffer(phost,BITMAP_HANDLE(6));
        if( ( (pRefrigeratorSnow->xOffset > ((FT_DispWidth + 60)*16)) ||
(pRefrigeratorSnow->yOffset > ((FT_DispHeight + 60) *16)) ) ||
(pRefrigeratorSnow->xOffset < (-60*16)) || (pRefrigeratorSnow->yOffset < (-60*16)) ) )
        {
                pRefrigeratorSnow->xOffset = ft_random(FT_DispWidth*16);
                pRefrigeratorSnow->yOffset = FT_DispHeight*16 + ft_random(80*16);
                pRefrigeratorSnow->dx = ft_random(RandomVal*8) - RandomVal*4;
                pRefrigeratorSnow->dy = -1*ft_random(RandomVal*8);
        }
Ft_App_WrCoCmd_Buffer(phost,VERTEX2F(pRefrigeratorSnow->xOffset, pRefrigeratorSnow-
>yOffset));
        pRefrigeratorSnow->xOffset += pRefrigeratorSnow->dx;
        pRefrigeratorSnow->yOffset += pRefrigeratorSnow->dy;
        pRefrigeratorSnow++;
}
```

3.1.2 Home Screen

The home screen is implemented with the display of raw bitmaps, the fonts and the numbers using the text and number command. The tag value is assigned to each of the bitmaps using the register REG_TOUCH_TAG. The bitmap of size 50*50 is randomly made to move in the screen. The background bitmap is of RGB565 format and is transformed using the bitmap transform. The randomly moving bitmap and menu icons are in L4 format whereas the bitmap of the logo is in ARGB4 format. The icons and the logo are placed in position. Whenever a touch is detected on the icon, the specific functions are called to perform its function. The date and time are displayed at the left side of the display with the inbuilt fonts using the windows function.

Figure 3-1 Home screen

3.1.3 Change of Temperature

The construction and movement of the background animation are explained in the section 3.1.1.The change of temperature screen is implemented using points and stencil commands.
The freezer and fridge temperature are implemented in a similar way. First, a point of radius 60 with alpha value 0 is drawn and the point is partitioned into 8 sections with the interval between each partition of 45 degrees. The stencil operation is incremented. The lines are drawn with the polarxy function of radius 60 and 100. The x and y value of the polarxy function is used for drawing lines in between the 8 partitioned sections. The point of radius 100 of light blue color is then drawn. The rotary track is assigned to the point with the assigned tag value. The track value is monitored to get the partition flag which is touch detected. The partition flag is then multiplied with 45 to draw a line on this x any y coordinate of the polarxy function and another line is drawn with the addition of 180 degree of the partitioned section's angle on the x and y coordinate of the polarxy function. The two lines are drawn with width 65 and the stencil operation is performed to color the touch detected partition. Next, the numbers are placed in each partition.
The same commands are applied to the fridge temperature change. The inbuilt fonts are used to display the numbers and text in the headings. The home bitmap exits the screen to the homescreen.
The change in settings for the font size will be applied to the text and numbers used in the screen.

```
/* code snippet showing the construction of the temperature adjustment*/
Ft_App_WrCoCmd_Buffer(phost,COLOR_A(0));
Ft_App_WrCoCmd_Buffer(phost, BEGIN(FTPOINTS));
Ft_App_WrCoCmd_Buffer(phost,POINT_SIZE(60*16));
Ft_App_WrCoCmd_Buffer(phost,STENCIL_OP(INCR,INCR));
Ft_App_WrCoCmd_Buffer(phost,VERTEX2F(xoffset*16,150*16));

Ft_App_WrCoCmd_Buffer(phost, BEGIN(LINES));
Ft_App_WrCoCmd_Buffer(phost,LINE_WIDTH(2 * 16));
for(z = 45; z < 361; z+=45)
{
        polarxy(100,z,&x0,&y0,xoffset,150);
                polarxy(60,z,&x1,&y1,xoffset,150);
```

```
        Ft_App_WrCoCmd_Buffer(phost,VERTEX2F(x0,y0));
        Ft_App_WrCoCmd_Buffer(phost,VERTEX2F(x1,y1));
}

        Ft_App_WrCoCmd_Buffer(phost, STENCIL_OP(KEEP,KEEP));
        Ft_App_WrCoCmd_Buffer(phost, BEGIN(FTPOINTS));
        Ft_App_WrCoCmd_Buffer(phost,COLOR_RGB(102,180,232));
        Ft_App_WrCoCmd_Buffer(phost,POINT_SIZE(100*16));
        Ft_App_WrCoCmd_Buffer(phost,COLOR_A(255));
        Ft_App_WrCoCmd_Buffer(phost,STENCIL_FUNC(EQUAL,0,255
        Ft_App_WrCoCmd_Buffer(phost,VERTEX2F(xoffset*16,150*16));
        Ft_App_WrCoCmd_Buffer(phost,LINE_WIDTH(65*16));
        Ft_App_WrCoCmd_Buffer(phost,STENCIL_OP(INCR,INCR));
        Ft_App_WrCoCmd_Buffer(phost,BEGIN(LINES));
        if(flag >= 0 && flag < 7
        {
               next_flag = flag+1;
               Ft_App_WrCoCmd_Buffer(phost,COLOR_A(0));
               angle = flag*45;
               polarxy(150,angle,&x0,&y0,xoffset,150);
               polarxy(150,angle+180,&x1,&y1,xoffset,150);
               Ft_App_WrCoCmd_Buffer(phost,VERTEX2F(x0 + origin_xy[flag][0]
               *16,y0 + origin_xy[flag][2]*16));
               Ft_App_WrCoCmd_Buffer(phost,VERTEX2F(x1 + origin_xy[flag][0]
               *16,y1 + origin_xy[flag][2] *16));

               polarxy(150,angle+45,&x0,&y0,xoffset,150);
               polarxy(150,angle+45+180,&x1,&y1,xoffset,150);
               Ft_App_WrCoCmd_Buffer(phost,VERTEX2F(x0 + origin_xy[flag][1]
               *16,y0 + origin_xy[flag][3]*16 ));
               Ft_App_WrCoCmd_Buffer(phost,VERTEX2F(x1 + origin_xy[flag][1]
               *16,y1 + origin_xy[flag][3] *16));

               Ft_App_WrCoCmd_Buffer(phost, STENCIL_OP(KEEP,KEEP));

               Ft_App_WrCoCmd_Buffer(phost,BEGIN(FTPOINTS));
               //Ft_App_WrCoCmd_Buffer(phost, TAG(tag));
               Ft_App_WrCoCmd_Buffer(phost,COLOR_RGB(19,43,59));
               Ft_App_WrCoCmd_Buffer(phost,POINT_SIZE(100*16));
               Ft_App_WrCoCmd_Buffer(phost,COLOR_A(255));
               Ft_App_WrCoCmd_Buffer(phost,STENCIL_FUNC(EQUAL,0,255));
               Ft_App_WrCoCmd_Buffer(phost,VERTEX2F(xoffset*16,150*16));

        }

        Ft_App_WrCoCmd_Buffer(phost, STENCIL_OP(KEEP,KEEP));

        Ft_App_WrCoCmd_Buffer(phost, BEGIN(FTPOINTS));
        Ft_App_WrCoCmd_Buffer(phost,COLOR_RGB(19,43,59));
        Ft_App_WrCoCmd_Buffer(phost,POINT_SIZE(100*16));

        Ft_App_WrCoCmd_Buffer(phost,COLOR_A(255));
        Ft_App_WrCoCmd_Buffer(phost,STENCIL_FUNC(EQUAL,0,255));
        Ft_App_WrCoCmd_Buffer(phost,VERTEX2F(xoffset*16,150*16));

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

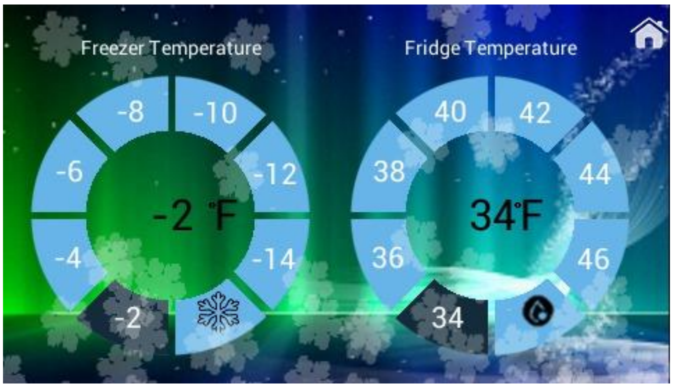

Figure 3-2 Temperature adjustment screen

3.1.4 Ice Option

The background bitmap and the animation construction are explained in section 3.1.1. The ice section of the application is implemented by the display of two raw bitmaps, one for cubed and the other for crushed .The rectangle with the alpha value 50 is drawn on the bitmap when the bitmap is selected. The bitmaps are in ARGB4 format.

Figure 3-3 Ice Option

3.1.5 Settings

The settings section of the application is implemented with the primitives. The settings screen is constructed using the rectangles and the home bitmap at the top.
The change background section is implemented by displaying the thumbnails of the bitmaps in two rows. Each bitmap is assigned a tag value and when the tag value is detected, the selected bitmap is transformed using `BITMAP_TRANSFORM` and is applied to the display list. The bitmaps are in RGB565 format. The thumbnails bitmap of size 100*50 are transformed to bitmap of size 480*272. The selected thumbnail is loaded to the 0th location, transformed to the screen size and the bitmap is assigned to the handle of the background bitmap.
The font size change section of the settings section is implemented using the slider. The slider is tracked using a linear tracker. The tracker movements are divided into four divisions of different font size varying from 26 to 29 and on the ascending track, the font size is applied to the entire application wherever the fonts are used.

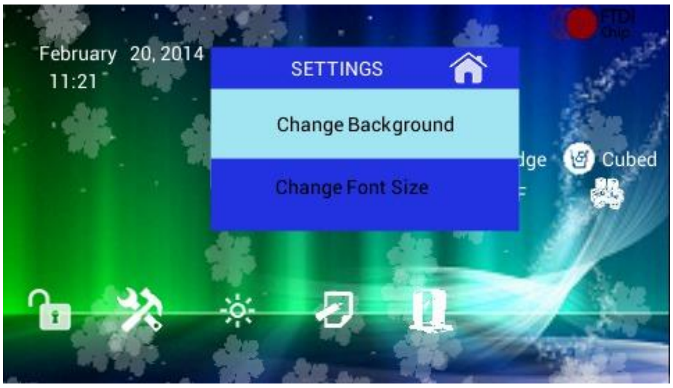

Figure 3-4 Settings screen

Figure 3-5 Background Selection Screen

3.1.6 Brightness

The background bitmap and the animation construction are explained in section 3.1.1 .The brightness section of the application is constructed using the raw bitmap, scissor command and linear tracker. The scissor rectangle is drawn at the bottom of the screen and the gradient is applied to it. The linear tracker is applied to the rectangle. With the change in the track value, a decreasing white coloured rectangle is drawn on the gradient showing the change of the PWM value. The PWM value of the screen varies from 10 to 128.

```
	/*Code snippet which shows the construction of the screen*/
Ft_App_WrCoCmd_Buffer(phost,TAG(12));
Ft_App_WrCoCmd_Buffer(phost,SCISSOR_XY(20, 200)); //Scissor rectangle bottom left at
(20, 200)
Ft_App_WrCoCmd_Buffer(phost,SCISSOR_SIZE(420, 40)); // Scissor rectangle is 420 x 40
pixels
Ft_Gpu_CoCmd_Gradient(phost, 20,0,0x0000ff,440,0,0xff0000);
Ft_App_WrCoCmd_Buffer(phost,CLEAR_COLOR_RGB(255, 255, 255));

Ft_App_WrCoCmd_Buffer(phost,COLOR_A(255));

if(track_val)
val = 10+(track_val/3);
Ft_Gpu_Hal_Wr8(phost,REG_PWM_DUTY,val);

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

```
Ft_App_WrCoCmd_Buffer(phost,BEGIN(RECTS));

Ft_App_WrCoCmd_Buffer(phost,VERTEX2F((20 + track_val)*16,190*16));
Ft_App_WrCoCmd_Buffer(phost,VERTEX2F(440*16,250*16));
```

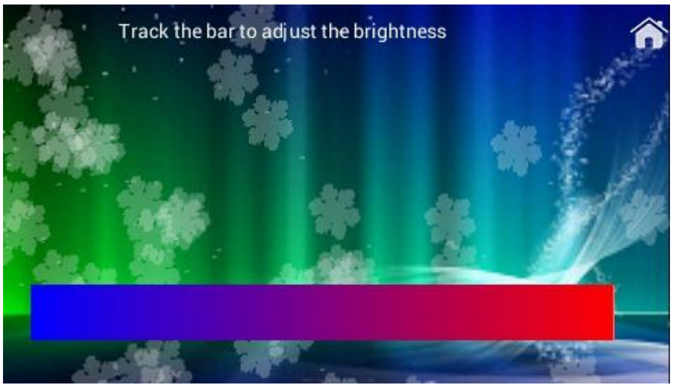

Figure 3-6 Brightness Adjustment screen

3.1.7 Sketch

The background bitmap and the animation construction are explained in section 3.1.1 .The sketch section of the application is constructed using the bitmap on the background, the sketch command and a button to clear the sketch. The sketch is of L8 format.

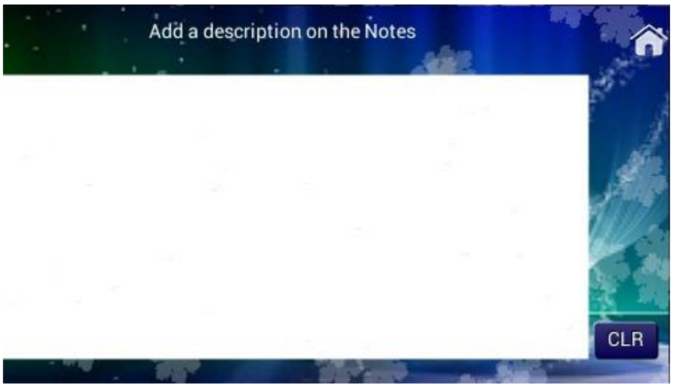

Figure 3-7 Sketch area

3.1.8 Food Manger

The background bitmap and the animation construction are explained in section 3.1.1 .The food manager section is constructed with the bitmaps, rectangles and inbuilt fonts. The bitmaps placed are in ARGB4 format. The bitmaps are assigned a tag value. When the bitmap's tag value is detected, a rectangle of alpha value 255 is dawn over the bitmap.

Figure 3-8 Food Manager

3.1.9 Screensaver

When the unlock bitmap's tag value is detected, the bitmap changes to lock bitmap. After a delay of 100ms, the logo raw bitmap is displayed using the command `CMD_SCREENSAVER`.

Figure 3-9 Screensaver

3.1.10 Synchronisation of Audio

The synthesised sound is played when pen up or pen down or when icon is selected.

4 Contact Information

Head Office – Glasgow, UK

Future Technology Devices International Limited
Unit 1,2 Seaward Place, Centurion Business Park
Glasgow G411HH
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
Web Site URL	http://www.ftdichip.com
Web Shop URL	http://www.ftdichip.com

Branch Office – Tigard, Oregon, USA

7130 SW Fir Loop
Tigard, OR 97223
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
Web Site URL	http://www.ftdichip.com

Branch Office – Taipei, Taiwan

2F, No. 516, Sec. 1, NeiHu Road
Taipei 114
Taiwan , R.O.C.
Tel: +886 (0) 2 87971330
Fax: +886 (0) 2 87519737

E-mail (Sales)	tw.sales1@ftdichip.com
E-mail (Support)	tw.support1@ftdichip.com
E-mail (General Enquiries)	tw.admin1@ftdichip.com
Web Site URL	http://www.ftdichip.com

Branch Office – Shanghai, China

Room 408, 317Xianxia Road,
Shanghai, 200051
China
Tel: +86 2162351596
Fax: +86 2162351595

E-mail (Sales)	cn.sales@ftdichip.com
E-mail (Support)	cn.support@ftdichip.com
E-mail (General Enquiries)	cn.admin@ftdichip.com
Web Site URL	http://www.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.

5 Appendix A– References

5.1 Document References

1. datasheet for VM800C
2. datasheet for VM800B
3. FT800 programmer guide FT_000793.
4. FT800 Embedded Video Engine Datasheet FT_000792

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
USB	Universal Serial Bus

6 Appendix B – List of Tables & Figures

List of Figures

Appendix C– Revision History

Document Title: AN_001010 FT800 REFRIGERATOR APPLICATION

Document Reference No.: FT_310

Clearance No.: FTDI#380

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

Document Feedback: Send Feedback

Revision	Changes	Date
1.0	First Release	