# APPLICATION NOTE

# AN_266

# FT_App_Music

**Version 1.1**

**Document Reference No.: FT_000911**

**Issue Date: 2013-11-01**

This document is to introduce the Music Demo Application. The objective of the Demo Application is to enable users to become familiar with the usage of the FT800, 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 design example demonstrates a simple music synthesizer using touch, graphic display and audio playback on a FT800 platform.

In the Music application, different musical instrument sounds are selected from a touch menu. A graphical touch keyboard generates tones. The audio is streamed to a speaker using the audio engine.

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

- Draw graphics primitives directly through a display list
- Incorporate display list commands to access sound and touch events through reads and writes of the FT800 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 computer/microcontroller.

## 1.2 Scope

This document can be used as a guide by designers to develop GUI applications by using FT800 with any MCU via SPI or $I^2C$. Note that detailed documentation is available on www.ftdichip.com/EVE.htm, including:

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

Note source code for the Music application is provided in sections 4 and 5, or at:

http://www.ftdichip.com/Support/SoftwareExamples/FT800_Projects.htm

# 2 Display Requirements

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

## 2.1 Display and Menu

This application demonstrates the usage of built-in widgets such as buttons, keys, and slider. The application draws piano keys on the screen with white and black buttons, a slider to control the volume of the sound, buttons to play different chords and buttons to choose the type of musical instrument to be played.

The application constantly monitors the display for touch input on the piano keys, instrument type, volume control and chord select keys (sound frequency).

## 2.2 Opening Menu

App Music opens with a generic introduction screen that prompts the user to press dots on the touchscreen for calibration. After the FTDI logo appears, pressing the start arrow starts App Music.

# 3 Design Flow

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

Select and configure your host port for controlling the FT800 then wake the device before configuring the display. The creative part then revolves around the generation of the display list, "** APPLICATION DATA **" in the figure below. There will be two lists. The active list and the updated/next list are continually swapped to render the display. Note, header files map the pseudo code of the design file of the display list to the FT800 instruction set, which is sent as the data of the SPI (or I²C) packet (typically <1KB). As a result, with EVE's object oriented approach, the FT800 is operating as an SPI peripheral while providing full display, audio, and touch capabilities.

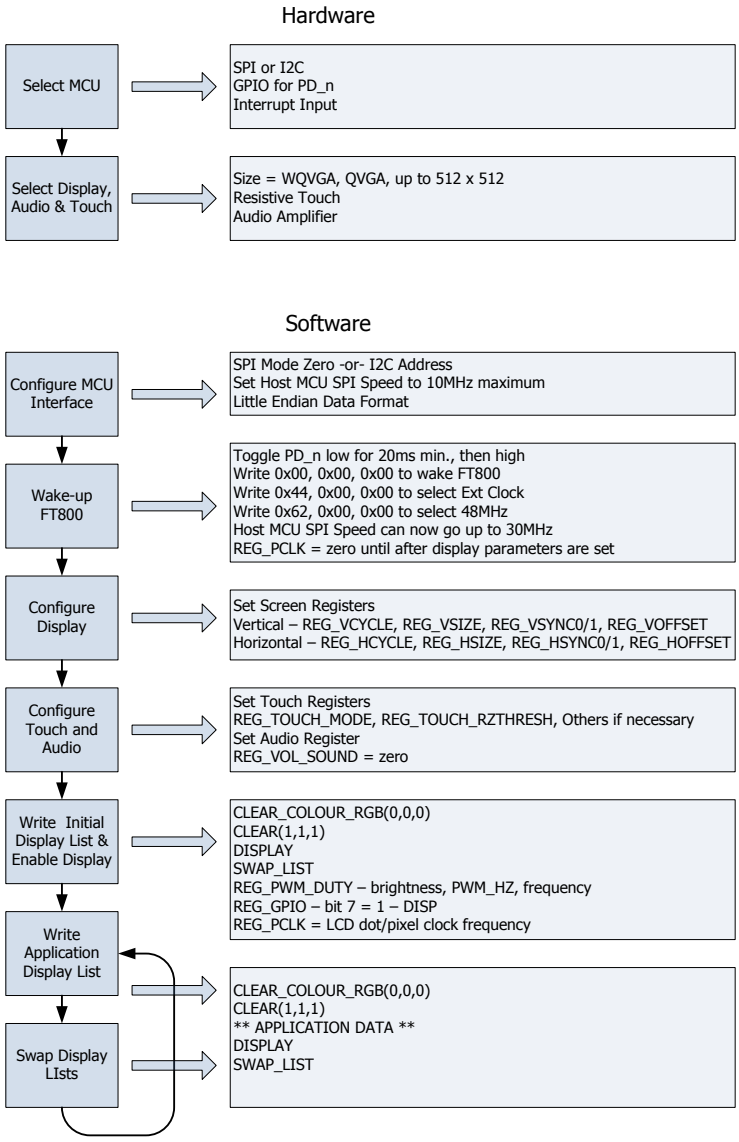


**Figure 3.1 Generic EVE Design Flow**

## 3.1 FT Music Flowchart

The flow chart below is specific to the App Music application. The application uses internal Graphic RAM and SD card storage for storing bitmaps and audio files that may be seen and heard as the application plays

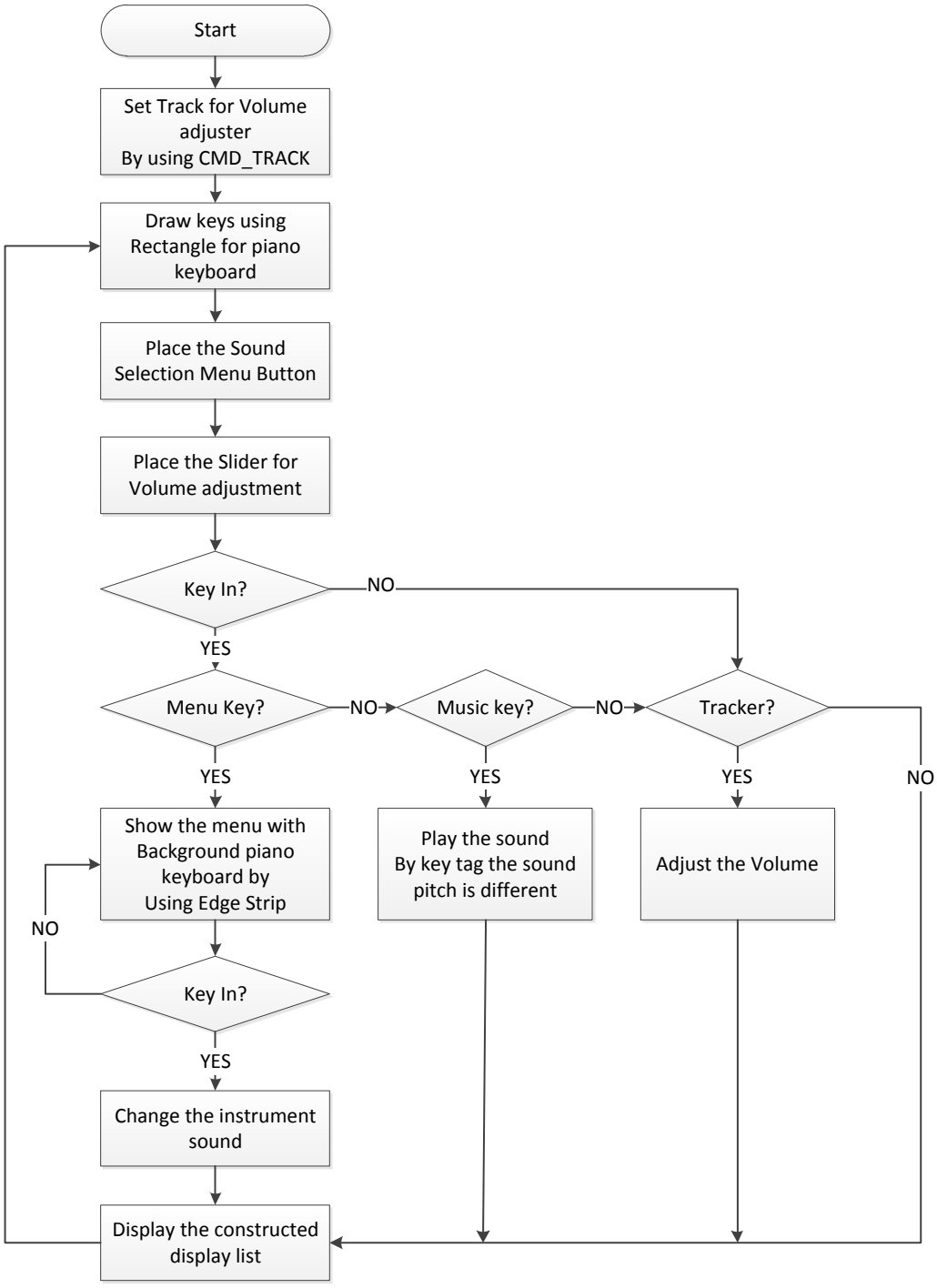


**Figure 3.2 App Music Flowchart**

# 4 Description of the Functional Blocks

## 4.1 System Initilization

Configuration of the SPI master port is unique to each controller – different registers etc, but all will require data to be sent Most Significant Bit (MSB) first with a little endian format.

The function labelled Ft_BootupConfig is generic to all applications and will start by toggling the FT800 PD# pin to perform a power cycle.

```
/* Do a power cycle for safer side */
Ft_Gpu_Hal_Powercycle(phost,FT_TRUE);
Ft_Gpu_Hal_Rd16(phost,RAM_G);

/* Set the clk to external clock */
Ft_Gpu_HostCommand(phost,FT_GPU_EXTERNAL_OSC);
Ft_Gpu_Hal_Sleep(10);

/* Switch PLL output to 48MHz */
Ft_Gpu_HostCommand(phost,FT_GPU_PLL_48M);
Ft_Gpu_Hal_Sleep(10);

/* Do a core reset for safer side */
Ft_Gpu_HostCommand(phost,FT_GPU_CORE_RESET);

/* Access address 0 to wake up the FT800 */
Ft_Gpu_HostCommand(phost,FT_GPU_ACTIVE_M);
```

The internal PLL is then given a prompt by setting the clock register and PLL to 48 MHz.

Note 36MHz is possible but will have a knock on effect for the display timing parameters.

A software reset of the core is performed followed by a dummy read to address 0 to complete the wake up sequence.

The FT800 GPIO lines are also controlled by writing to registers:

```
Ft_Gpu_Hal_Wr8(phost, REG_GPIO_DIR,0x80 | Ft_Gpu_Hal_Rd8(phost,REG_GPIO_DIR));
Ft_Gpu_Hal_Wr8(phost, REG_GPIO,0x080 | Ft_Gpu_Hal_Rd8(phost,REG_GPIO));
```

And these allow the display to be enabled.

To confirm the FT800 is awake and ready to start accepting display list information the identity register is read in a loop until it reports back 0x7C. It will always be 0x7C if everything is awake and functioning correctly.

```
ft_uint8_t chipid;
//Read Register ID to check if FT800 is ready.
chipid = Ft_Gpu_Hal_Rd8(phost, REG_ID);
while(chipid != 0x7C)
        chipid = Ft_Gpu_Hal_Rd8(phost, REG_ID);
```

Once the FT800 is awake the display may be configured through 13 register writes according to its resolution. Resolution and timing data should be available in the display datasheet.

```
Ft_Gpu_Hal_Wr16(phost, REG_HCYCLE, FT_DispHCycle);
Ft_Gpu_Hal_Wr16(phost, REG_HOFFSET, FT_DispHOffset);
Ft_Gpu_Hal_Wr16(phost, REG_HSYNC0, FT_DispHSync0);
Ft_Gpu_Hal_Wr16(phost, REG_HSYNC1, FT_DispHSync1);
Ft_Gpu_Hal_Wr16(phost, REG_VCYCLE, FT_DispVCycle);
Ft_Gpu_Hal_Wr16(phost, REG_VOFFSET, FT_DispVOffset);
Ft_Gpu_Hal_Wr16(phost, REG_VSYNC0, FT_DispVSync0);
Ft_Gpu_Hal_Wr16(phost, REG_VSYNC1, FT_DispVSync1);
Ft_Gpu_Hal_Wr8(phost, REG_SWIZZLE, FT_DispSwizzle);
Ft_Gpu_Hal_Wr8(phost, REG_PCLK_POL, FT_DispPCLKPol);
Ft_Gpu_Hal_Wr8(phost, REG_PCLK,FT_DispPCLK);//after this display is visible on the LCD
Ft_Gpu_Hal_Wr16(phost, REG_HSIZE, FT_DispWidth);
Ft_Gpu_Hal_Wr16(phost, REG_VSIZE, FT_DispHeight);
```

To complete the configuration the touch controller should also be calibrated

```
/* Touch configuration - configure the resistance value to 1200 - this value is specific to
customer requirement and derived by experiment */
Ft_Gpu_Hal_Wr16(phost, REG_TOUCH_RZTHRESH,1200);
Ft_Gpu_Hal_Wr8(phost, REG_GPIO_DIR,0xff);
Ft_Gpu_Hal_Wr8(phost, REG_GPIO,0x0ff);
```

An optional step is present in this code to clear the screen so that no artefacts from bootup are displayed.

```
/*It is optional to clear the screen here*/
Ft_Gpu_Hal_WrMem(phost, RAM_DL,(ft_uint8_t *)FT_DLCODE_BOOTUP,sizeof(FT_DLCODE_BOOTUP));
Ft_Gpu_Hal_Wr8(phost, REG_DLSWAP,DLSWAP_FRAME);
```

## 4.2 Info ()

This is a largely informational section of code and it starts by synchronising the physical xy coordinates of the display's touch layer with the display's visual layer.

A display list is started and cleared:

```
Ft_Gpu_CoCmd_Dlstart(phost);
Ft_App_WrCoCmd_Buffer(phost,CLEAR(1,1,1));
Ft_App_WrCoCmd_Buffer(phost,COLOR_RGB(255,255,255));
```

A text instruction is printed on the display followed by the call to the internal calibrate function:

```
Ft_Gpu_CoCmd_Text(phost,FT_DispWidth/2,FT_DispHeight/2,26,OPT_CENTERX|OPT_CENTERY,"Please 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);
```

Next up in the Info() function is the FTDI logo playback:

```
Ft_Gpu_CoCmd_Logo(phost);
Ft_App_Flush_Co_Buffer(phost);
Ft_Gpu_Hal_WaitCmdfifo_empty(phost);
```

```
while(0!=Ft_Gpu_Hal_Rd16(phost,REG_CMD_READ));
dloffset = Ft_Gpu_Hal_Rd16(phost,REG_CMD_DL);
```

A composite image with the logo and a start arrow is then displayed to allow the user to start the main application

```
do
  {
    Ft_Gpu_CoCmd_Dlstart(phost);
    Ft_Gpu_CoCmd_Append(phost,100000L,dloffset);
    Ft_App_WrCoCmd_Buffer(phost,BITMAP_TRANSFORM_A(256));
    Ft_App_WrCoCmd_Buffer(phost,BITMAP_TRANSFORM_A(256));
    Ft_App_WrCoCmd_Buffer(phost,BITMAP_TRANSFORM_B(0));
    Ft_App_WrCoCmd_Buffer(phost,BITMAP_TRANSFORM_C(0));
    Ft_App_WrCoCmd_Buffer(phost,BITMAP_TRANSFORM_D(0));
    Ft_App_WrCoCmd_Buffer(phost,BITMAP_TRANSFORM_E(256));
    Ft_App_WrCoCmd_Buffer(phost,BITMAP_TRANSFORM_F(0));
    Ft_App_WrCoCmd_Buffer(phost,SAVE_CONTEXT());
    Ft_App_WrCoCmd_Buffer(phost,COLOR_RGB(219,180,150));
    Ft_App_WrCoCmd_Buffer(phost,COLOR_A(220));
    Ft_App_WrCoCmd_Buffer(phost,BEGIN(EDGE_STRIP_A));
    Ft_App_WrCoCmd_Buffer(phost,VERTEX2F(0,FT_DispHeight*16));
    Ft_App_WrCoCmd_Buffer(phost,VERTEX2F(FT_DispWidth*16,FT_DispHeight*16));
    Ft_App_WrCoCmd_Buffer(phost,COLOR_A(255));
    Ft_App_WrCoCmd_Buffer(phost,RESTORE_CONTEXT());
    Ft_App_WrCoCmd_Buffer(phost,COLOR_RGB(0,0,0));
   // INFORMATION
    Ft_Gpu_CoCmd_Text(phost,FT_DispWidth/2,20,28,OPT_CENTERX|OPT_CENTERY,info[0]);
    Ft_Gpu_CoCmd_Text(phost,FT_DispWidth/2,60,26,OPT_CENTERX|OPT_CENTERY,info[1]);
    Ft_Gpu_CoCmd_Text(phost,FT_DispWidth/2,90,26,OPT_CENTERX|OPT_CENTERY,info[2]);
    Ft_Gpu_CoCmd_Text(phost,FT_DispWidth/2,120,26,OPT_CENTERX|OPT_CENTERY,info[3]);
    Ft_Gpu_CoCmd_Text(phost,FT_DispWidth/2,FT_DispHeight-30,26,OPT_CENTERX|OPT_CENTERY,"Click
to play");
    if(sk!='P')
    Ft_App_WrCoCmd_Buffer(phost,COLOR_RGB(255,255,255));
    else
    Ft_App_WrCoCmd_Buffer(phost,COLOR_RGB(100,100,100));
    Ft_App_WrCoCmd_Buffer(phost,BEGIN(FTPOINTS));
    Ft_App_WrCoCmd_Buffer(phost,POINT_SIZE(20*16));
    Ft_App_WrCoCmd_Buffer(phost,TAG('P'));
    Ft_App_WrCoCmd_Buffer(phost,VERTEX2F((FT_DispWidth/2)*16,(FT_DispHeight-60)*16));
    Ft_App_WrCoCmd_Buffer(phost,COLOR_RGB(180,35,35));
    Ft_App_WrCoCmd_Buffer(phost,BEGIN(BITMAPS));
    Ft_App_WrCoCmd_Buffer(phost,VERTEX2II((FT_DispWidth/2)-14,(FT_DispHeight-75),14,0));
    Ft_App_WrCoCmd_Buffer(phost,DISPLAY());
    Ft_Gpu_CoCmd_Swap(phost);
    Ft_App_Flush_Co_Buffer(phost);
    Ft_Gpu_Hal_WaitCmdfifo_empty(phost);

  }while(Read_Keys()!='P');
```

## 4.3 Memory Loading

The on board graphics RAM (GRAM) is used to store the JPEG data and the audio data.

The file is copied from the PC to the FT800 GRAM.

e.g. for the audio files, the function Load_afile is called with an address in the 256k Graphics RAM to be written to.

```
void Load_afile(ft_uint32_t add, FILE *afile)
{
  ft_uint8_t pbuff[512],temp[512],tval;
  ft_uint16_t z = 0;
  fread(pbuff,1,512,afile);
  Ft_Gpu_Hal_WrMem(phost,add,pbuff,512L);

if ((add & 2047L) == 0)                          // every 2kb update the bitmap

  {
    for(z=0;z<512L;z++)
    {
      tval = pbuff[z];
      if (tval & 0x80L)                         // 11 bits of data
      tval ^= 0x7fL;
      temp[z] = tval;
    }
    Ft_Gpu_Hal_WrMem(phost,8192L,temp,512L);
  }
}
```

# 5 Working with Widgets and Sound Effects

App Music makes use of built-in widgets such as the buttons, piano keys and a slider for volume as shown in figure 5.1 The application constantly monitors the display for touch input on the piano keys, instrument type, volume control and chord select keys (sound frequency).

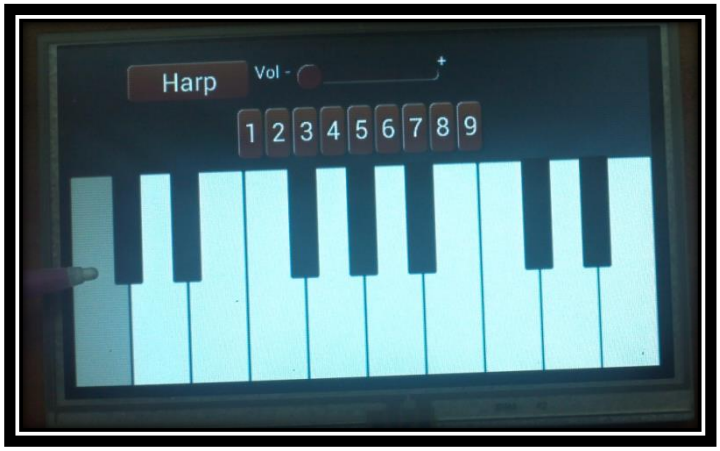


**Figure 5.1 App Music Display**

## 5.1 Volume Slider Widget

You can specify the properties of the volume slider by using the Co-Processor Track widget (CoCmd_Track). Speaker volume is set by the REG_VOL_SOUND register. Volume level can range from 0 to 255.

```
Ft_Gpu_CoCmd_Dlstart(phost);
Ft_Gpu_CoCmd_Track(phost,200,20,100,10,'V');

// where ‘V’ is the Tag of the volume slider

Ft_App_Flush_Co_Buffer(phost);
Ft_Gpu_Hal_WaitCmdfifo_empty(phost);
Ft_Gpu_Hal_Wr8(phost,REG_VOL_SOUND,100);
```

## 5.2 Drawing the Keyboard Keys

The buttons for the keyboard keys (see figure 5.1) are drawn by using the co-processor button widget. (CoCmd Button)

```
void pbuttons(ft_uint8_t no,ft_uint16_t xx,ft_uint16_t y,ft_uint8_t
btspace,ft_uint8_t btw,ft_uint8_t bth,ft_uint8_t tag_no,ft_uint32_t
fgc1,ft_uint32_t fgc2,ft_uint8_t rtag)
{
  ft_uint16_t x,z;
  for(z=0;z<no;z++)
  {
    x = z*(btw+btspace)+xx;
    rtag==(z+tag_no) ? Ft_Gpu_CoCmd_FgColor(phost,fgc1) :
Ft_Gpu_CoCmd_FgColor(phost,fgc2);
    Ft_App_WrCoCmd_Buffer(phost,TAG(z+tag_no));
    Ft_Gpu_CoCmd_Button(phost,x,y,btw,bth,16,OPT_FLAT,"");
  }
```

Button Codes:

| | |
|---|---|
| no | number of keys |
| xx,y | xy co-ordinates |
| btspace | space between keys |
| btw,bth | Button Width and Height |
| fgc1,fgc2 | Foreground colors |
| tag_no | tag for keys |
| rtag | detecting tag |

## 5.3 Drawing the Instrument Select Buttons

The following code configures the instrument select menu shown in figure 5.2. The co-processor Dlstart and Track widgets are used.

```
  const char *ins[] =
{"Harp","Xylophone","Tuba","Glockenspiel","Organ","Trumpet","Piano","Chimes",
"Musicbox","Ding" };
// Intial stage
  instr = 0x40;
  Volume = 100;
  button_space = 2;                                    // space between buttons

  button_w = (FT_DispWidth / 10)-button_space;              // Button width

  button_h = (FT_DispHeight - (FT_DispHeight/3));           // Button Height

  b1w = button_w/2;                                       // button width one
  b1h = button_h/2;
  b1s = b1w+button_space;

  Ft_Gpu_CoCmd_Dlstart(phost);
  Ft_Gpu_CoCmd_Track(phost,200,20,100,10,'V');
```

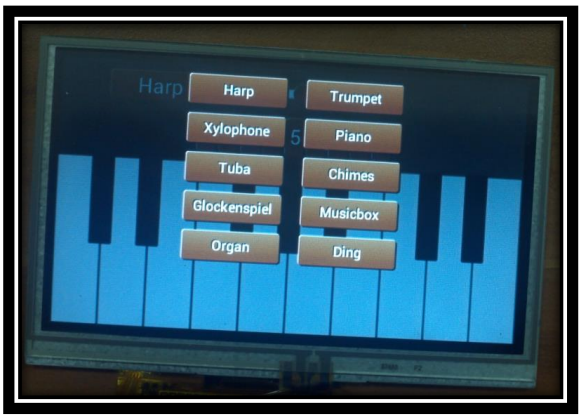


**Figure 5.2 Instrument Select Menu**

## 5.4 Sound Synthesis and Playback

The sounds of musical instruments are synthesized by using the REG_SOUND, REG_PLAY and REG_VOLUME_SOUND registers.

```
  Mkeys = Ft_Gpu_Hal_Rd8(phost,REG_TOUCH_TAG);
    Skeys = Read_Keys();
    if(Skeys=='I'){Flag.sound  = 1;
    Mkeys = 0;}
    if(Skeys>=100 && Flag.sound==1)          //Tagval from 100
    {
      Flag.sound  = 0;
      pk = (Skeys-100);              // assign the selection
      instr = pk|0x40;
    }
    if(Mkeys > '0' && temp_keys!=Mkeys)
    {
//    freq = Mkeys;
      Ft_Gpu_Hal_Wr16(phost,REG_SOUND, 0x50 + (Mkeys-'0'));
      Ft_Gpu_Hal_Wr8(phost,REG_PLAY,1);
    }

if(Mkeys !=0 &&  Mkeys < 16 && temp_keys!=Mkeys)
    {
      Ft_Gpu_Hal_Wr16(phost,REG_SOUND,((Mkeys|50) << 8) | instr);
      Ft_Gpu_Hal_Wr8(phost,REG_PLAY,1);
    }


Read_Tag_val = Ft_Gpu_Hal_Rd32(phost,REG_TRACKER);
    if((Read_Tag_val&0xff)=='V' && Flag.sound==0)
    {
       Volume = (Read_Tag_val>>24);
       Ft_Gpu_Hal_Wr8(phost,REG_VOL_SOUND,Volume);
    }
    temp_keys=Mkeys;
```

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

## 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 Tables & Figures

## List of Figures

# Appendix C– Revision History

| | |
|---|---|
| Document Title: | AN_266 FT_App_Music |
| Document Reference No.: | FT_000911 |
| Clearance No.: | FTDI# 361 |
| Product Page: | http://www.ftdichip.com/EVE.htm |
| Document Feedback: | Send Feedback |

| Revision | Changes | Date |
|---|---|---|
| 0.1 | Initial draft release | 2013-07-18 |
| 1.0 | Version 1.0 updated wrt review comments | 2013-08-21 |
| 1.1 | Version 1.1 | 2012-11-01 |
| | | |
| | | |
| | | |