APPLICATION NOTE

AN_261

FT_App_Clocks

Version 1.1

Document Reference No.: FT_000906

Issue Date: 2013-11-01

This document describes the Clock 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 application demonstrates interactive clocks using the widget CMD_CLOCK to draw the clock, and CMD_TEXT to display additional text on an FT800 platform.

All the clocks have an individual tracker to adjust the time but the initial reference is SYSTEM time. When the user changes the time in one clock, the time displayed on the other clocks are also changed based on STANDARD TIME of the Country.

To view all the clocks the user can drag the background left or right on the touch screen in a linear fashion.

1.1 Overview

The document will provide information on the image creation, tagging and tracking of displayed objects, and the structure of display lists. Further, the application note will outline the general steps of the design flow, including display list creation, and integrating the display list with the system host micro-controller.

To be read in conjunction with the source code, provided in section 4.3 and in design files located at http://www.ftdichip.com/Support/SoftwareExamples/FT800_Projects.htm 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^{2}C. Note 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

2 Display Requirements

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

2.1 BackGround

The display background is created to show a gradient of light to dark from the bottom to the top of the display.

2.2 Clock Face

The clock face is generated with the FT800 clock widget. It will display a background, ticks for the 12 hours and an hour, minute and second hand.

2.3 Time

Time in the application is initialised to System time from the host controller.

Time adjustment can be done using the tracker for the hour needle and the minute needle is adjusted accordingly. As the tracker is synced with System time, the second hand movement changes accordingly.

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. In essence 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 object oriented approach, the FT800 is operating as an SPI peripheral while providing full display, audio, and touch capabilities.

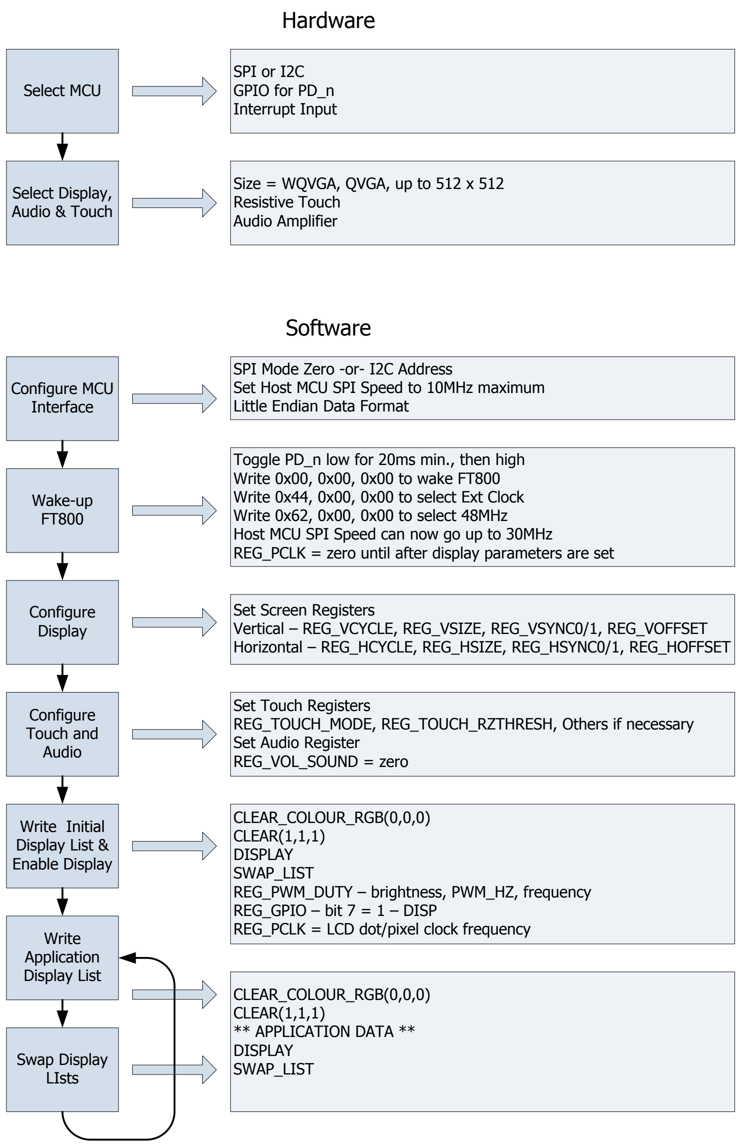
Hardware
Select MCU
SPI or I2C
GPIO for PD_n
Interrupt Input
Select Display, Audio & Touch
Size = WQVGA, QVGA, up to 512 x 512
Resistive Touch
Audio Amplifier
Software
Configure MCU Interface
SPI Mode Zero -or- I2C Address
Set Host MCU SPI Speed to 10MHz maximum
Little Endian Data Format
Wake-up FT800
Toggle PD_n low for 20ms min., then high
Write 0x00, 0x00, 0x00 to wake FT800
Write 0x44, 0x00, 0x00 to select Ext Clock
Write 0x62, 0x00, 0x00 to select 48MHz
Host MCU SPI Speed can now go up to 30MHz
REG_PCLK = zero until after display parameters are set
Configure Display
Set Screen Registers
Vertical – REG_VCYCLE, REG_VSIZE, REG_VSYNC0/1, REG_VOFFSET
Horizontal – REG_HCYCLE, REG_HSIZE, REG_HSYNC0/1, REG_HOFFSET
Configure Touch and Audio
Set Touch Registers
REG_TOUCH_MODE, REG_TOUCH_RZTHRESH, Others if necessary
Set Audio Register
REG_VOL_SOUND = zero
Write Initial Display List & Enable Display
CLEAR_COLOUR_RGB(0,0,0)
CLEAR(1,1,1)
DISPLAY
SWAP_LIST
REG_PWM_DUTY – brightness, PWM_HZ, frequency
REG_GPIO – bit 7 = 1 – DISP
REG_PCLK = LCD dot/pixel clock frequency
Write Application Display List
Swap Display LIsts
CLEAR_COLOUR_RGB(0,0,0)
CLEAR(1,1,1)
** APPLICATION DATA **
DISPLAY
SWAP_LIST

Figure 3.1 Generic EVE Design Flow

3.1 Clocks Flowchart

The flow chart below is specific to the Clocks application. The application gets the system time and computes this for the defined time zones. The clocks are then drawn on the display. Updates to the time may be made from the rotary tracking.

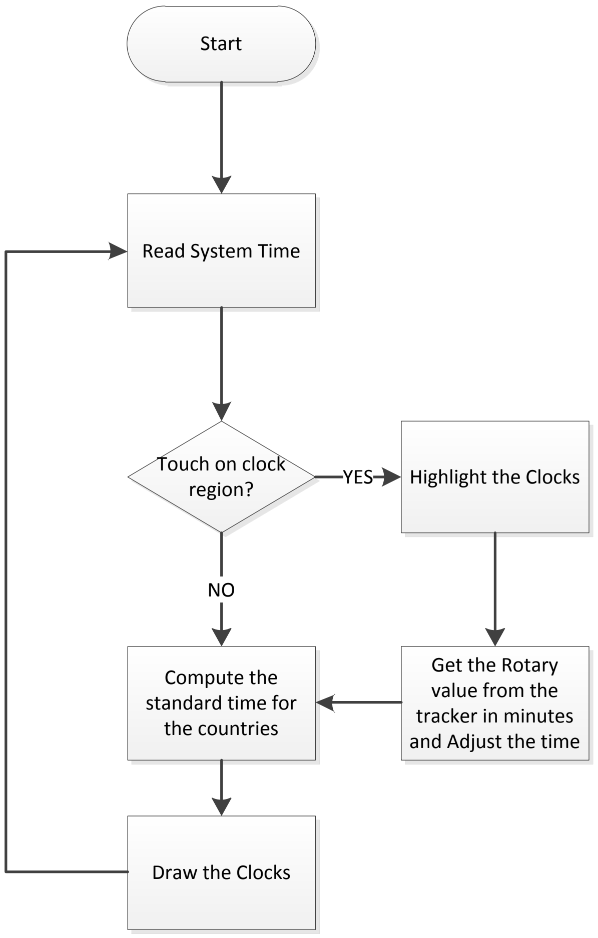
Start
Read System Time
Touch on clock region?
YES
Highlight the Clocks
NO
Compute the standard time for the countries
Get the Rotary value from the tracker in minutes and Adjust the time
Draw the Clocks

Figure 3.2 Flowchart

4 Description of the Functional Blocks

4.1 System Initialisation

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 in this project 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 displays touch layer with the displays 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);
  dloffset -=4;
  Ft_Gpu_Hal_WrCmd32(phost,CMD_MEMCPY);
  Ft_Gpu_Hal_WrCmd32(phost,100000L);
  Ft_Gpu_Hal_WrCmd32(phost,RAM_DL);
  Ft_Gpu_Hal_WrCmd32(phost,dloffset);
  play_setup();
```

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 Draw Clocks

The "clocks" function will apply a TAG mask to the background, the text and the clock faces to allow the display to be scrolled left or right when the display is touched. The clocks faces also have rotary tracking to allow for the time to be adjusted.

```
do
  {
    Tag = Ft_Gpu_Hal_Rd8(phost,REG_TOUCH_TAG);

    temp_m =  (ist.Mins + (ist.Hrs*60))%720L;
    min_val = Get_TagRotary_Value(Tag,60,12,temp_m);

    if(Tag==0)
    scroller_run();

    drag = scroller.base>>4;
    cts = drag/dx;
    dragth = drag%dx;
    Ft_Gpu_CoCmd_Dlstart(phost);
    Ft_App_WrCoCmd_Buffer(phost,CLEAR(1,1,1));
    Ft_App_WrCoCmd_Buffer(phost,TAG_MASK(1));
    Ft_App_WrCoCmd_Buffer(phost,TAG(0));
    Ft_Gpu_CoCmd_Gradient(phost,0, 135, 0x000000, 0, 272, 0x605040);
    for(i=-1;i<(per_f+2);i++)
    {
        Ox = (clksize+clk_s)+(dx*i);
       Ox-=dragth;
        name = (COUNTRIES+i+cts)%COUNTRIES;
        timer(name);
        if(Tag==name+1) Ft_Gpu_CoCmd_BgColor(phost,0x4040a0); else
        Ft_Gpu_CoCmd_BgColor(phost,0x101040);
        Ft_App_WrCoCmd_Buffer(phost,TAG(name+1));

Ft_Gpu_CoCmd_Clock(phost,Ox,FT_DispHeight/2,clksize,0,utc.Hrs,utc.Mins,utc.Secs,utc.mS
ecs);
        Ft_Gpu_CoCmd_Track(phost,Ox,FT_DispHeight/2,1,1,name+1);
    }
    Ft_App_WrCoCmd_Buffer(phost,TAG_MASK(0));
    for(i=-1;i<(per_f+2);i++)
    {
        Ox = (clksize+clk_s)+(dx*i);
       Ox-=dragth;
        name = (COUNTRIES+i+cts)%COUNTRIES;
        Ft_Gpu_CoCmd_Text(phost,Ox,dy,29,OPT_CENTERX,country[name]);
    }
    Ft_App_WrCoCmd_Buffer(phost,DISPLAY());
    Ft_Gpu_CoCmd_Swap(phost);
    Ft_App_Flush_Co_Buffer(phost);
    Ft_Gpu_Hal_WaitCmdfifo_empty(phost);
  }while(1);
```

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.

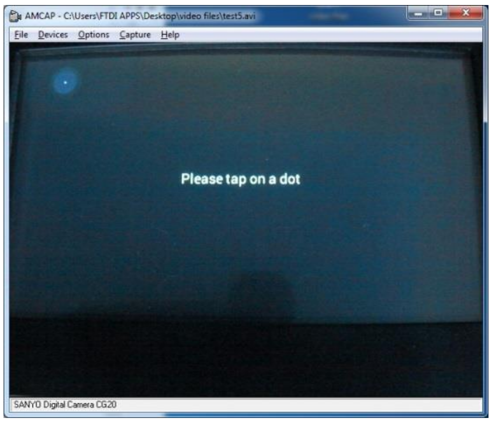
AMCAP - C:\Users\FTDI APPS\Desktop\video files\test5.avi
File Devices Options Capture Help
Please tap on a dot
SANYO Digital Camera CG20

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.

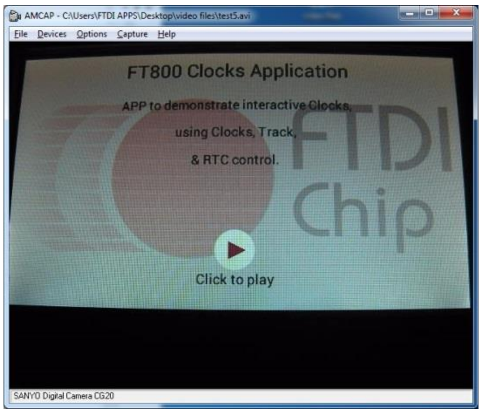
AMCAP - C:\Users\FTDI APPS\Desktop\video files\test5.avi
File Devices Options Capture Help
FT800 Clocks Application
APP to demonstrate interactive Clocks,
using Clocks, Track,
& RTC control.
Click to play
SANYO Digital Camera CG20

Figure 5.2 Composite screen

After pressing "Click to Play" the app displays the clocks.

5.1 Clocks View

There are 12 clock faces in total. Only 3 are visible at a time. They have a dark blue background with white hands and hour markers.

The MCU monitors the user touch constantly. The user can touch or drag the screen linearly to view more clock displays of different countries.

Touching on the clock face will change the colour to light blue.

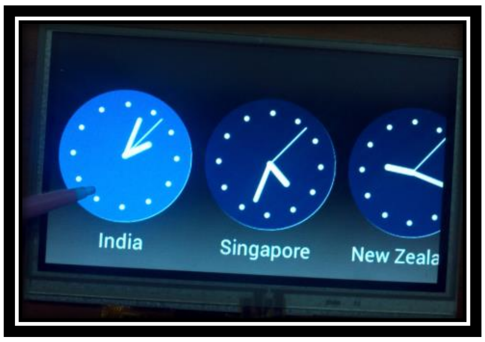

Figure 5.3 Clocks

5.1.1 Time Setting

The user can touch on the clock region to highlight the clock and change the clock time with respect to user touch rotary movement. The time changes with respect to the touch in all the clocks displayed.

5.2 Text

The text under each clock signals the time for that country. Note the time is calculated to be correct for the labelled country.

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_261 FT_App_Clocks
Document Reference No.:	FT_000906
Clearance No.:	FTDI# 356
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	2013-11-01