Application Note

AN_269

FT_App_Signature

Version 1.3

Issue Date: 2018-01-05

This application note introduces the Signature Demo. The objective of this Demo Application is to enable users to become familiar with the usage of the FT8XX series, 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 design example demonstrates an interactive user interface that provides a signature area, useful for payment terminals. The touch screen is used to capture handwriting inside the signature area. The touch events are then drawn within the signature area interactively. While the signature area is active, an array of stars is drawn in a moving pattern behind the signature area. A button then clears the signature area, ready for another signature.

1.1 Overview

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

The 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 Signature Area

The signature area uses the display and touch screen. As the touch screen is written upon, the corresponding pixels in the signature area are rendered out to simulate drawing on a page. The signature area only covers a portion of the screen. Touch events outside of this area are ignored.

2.2 Button

A second touch area uses an FT8XX button widget. When touched, any information drawn in the signature area is cleared out.

2.3 Background

While the signature and button areas are active, an array of animated stars is drawn behind the active touch areas.

3 Design Flow

Every EVE design follows the same basic principles as highlighted in Figure 3.1. More extensive details on device start-up can be found in AN_391 EVE Platform Guide

Select and configure your host port for controlling the FT8XX 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 FT8XX instruction set, which is sent as the data of the SPI packet (typically <1KB). As a result, with EVE's object oriented approach, the FT8XX is operating as an SPI peripheral while providing full display, audio, and touch capabilities.

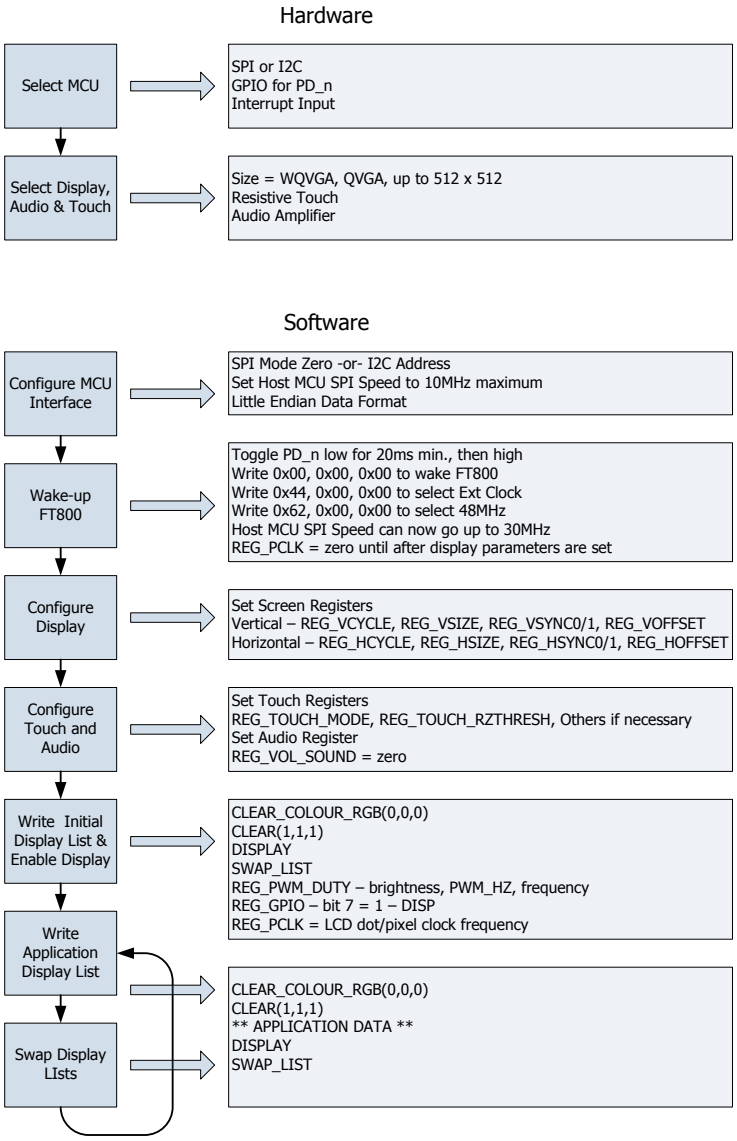

Figure 3.1 Generic EVE Design Flow

3.1 Signature Flowchart

The flowchart below is specific to the Signature application.

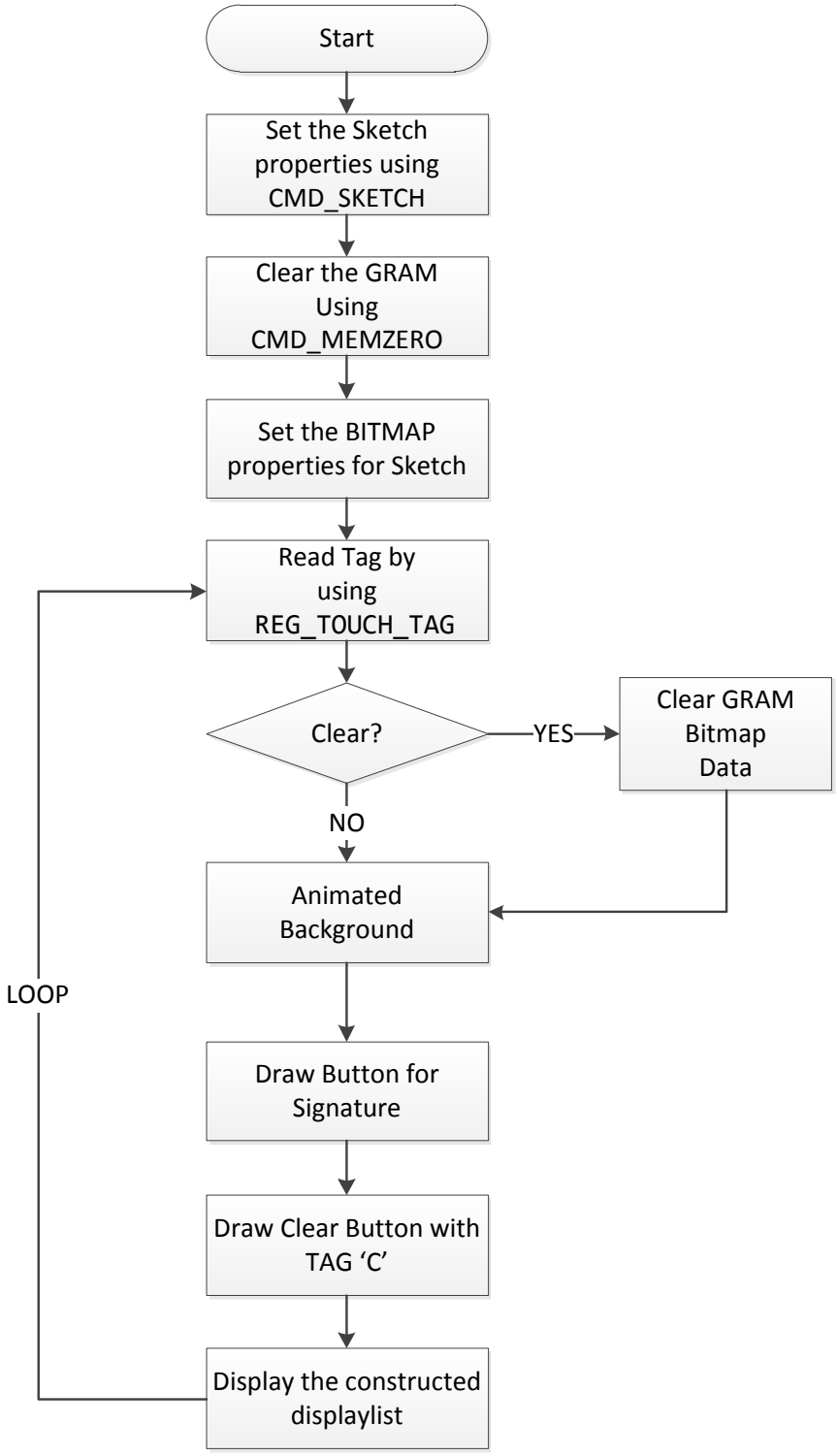

Figure 3.2 Flowchart

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

5 Signature()

The background star image that is rotated in the background is located in line with the code. It is stored as compressed data in the MCU or PC:

```
PROGMEM prog_uchar8_t home_start_icon[SIZE_HOME_START_ICON] = {0x78,0x9C,0xE5,0x94,0xBF,0x4E,0xC2
```

The home_setup function is run once prior to displaying the Welcome screen through App_Show_WelcomeScreen(). It takes this data and decompresses it as it's stored in the GRAM:

```
Gpu_Hal_WrCmd32(phost,CMD_INFLATE);
Gpu_Hal_WrCmd32(phost, START_ICON_ADDR);
Gpu_Hal_WrCmdBuf(phost,home_star_icon,sizeof(home_star_icon));
```

The initial display list is started by clearing the buffers and setting the colour to white:

```
Gpu_CoCmd_Dlstart(phost);        // start
App_WrCoCmd_Buffer(phost,CLEAR(1,1,1));
App_WrCoCmd_Buffer(phost,COLOR_RGB(255, 255, 255));
```

The data for the star which was loaded with CMD_INFLATE is now assigned to handle 13 so the images can be displayed and manipulated:

```
App_WrCoCmd_Buffer(phost,BITMAP_HANDLE(13));    // handle for background stars
App_WrCoCmd_Buffer(phost,BITMAP_SOURCE(START_ICON_ADDR));      // Starting
address in gram
App_WrCoCmd_Buffer(phost,BITMAP_LAYOUT(L4, 16, 32));  // format
App_WrCoCmd_Buffer(phost,BITMAP_SIZE(NEAREST, REPEAT, REPEAT, 512, 512  ));
```

The Signature function starts by calculating the size of the signature area based on the screen size:

```
uint16_t w,h,x,y,tag;

int16_t sw = 2 * DispWidth / 3;
int16_t sh = sw / 3;
int16_t ox = ( DispWidth - sw) / 2;
int16_t oy = (2 * DispHeight / 3) - sh;
uint16_t a = 0;

x = DispWidth*0.168;
y = DispHeight*0.317;
w =  DispWidth-(2*x);
h =  DispHeight-(2*y);
```

The display list is started and buffers cleared.

```
Gpu_CoCmd_Dlstart(phost);        // start
App_WrCoCmd_Buffer(phost,CLEAR(1,1,1));
```

Then it clears the GRAM memory covering the size of the display with CMD_MEMZERO, followed by drawing the signature area:

```
Gpu_CoCmd_MemZero(phost,10*1024L,480L*272L); // Clear the gram from 1024
Gpu_CoCmd_Sketch(phost,x,y,w,h,10*1024L,L8);
```

At this point, the signature area is just that – there is nothing drawn on the screen to indicate the boundaries. That's done a bit later in the display list. The CMD_SKETCH command continually translates the X-Y touch coordinates within the sketch area and changes those locations in the GRAM from the foreground colour to background colour. By doing this, the image of the signature

is shown on the screen. Once complete, the GRAM corresponding to the sketch area can be read by the MCU and stored as a signature image. This example does not store any sketch data.

After enabling the signature area, the cleared memory is displayed to ensure a blank screen with no artifacts. The display list is swapped to become active and any remaining command buffer is flushed:

```
App_WrCoCmd_Buffer(phost,BITMAP_HANDLE(1));    // handle for background stars
App_WrCoCmd_Buffer(phost,BITMAP_SOURCE(10*1024L));
App_WrCoCmd_Buffer(phost,BITMAP_LAYOUT(L8,w,h));
App_WrCoCmd_Buffer(phost,BITMAP_SIZE(NEAREST,BORDER,BORDER,w,h));
Gpu_CoCmd_Swap(phost);
App_Flush_Co_Buffer(phost);
Gpu_Hal_WaitCmdfifo_empty(phost);
```

With the visible display initialization complete, the screen elements can now be drawn. A while() loop will create a new display list each time through. The display list will contain the moving stars and display any activity on the touch screen for both the signature area and clear button.

The clear button is assigned a touch tag. Tags remove the need to correspond the X-Y touch coordinates with a drawn element. Instead, the entire element is assigned a tag number. The FT8XX determines whether the touch event was inside the element boundaries then assigns the appropriate tag. This frees up considerable host MCU time.

The first item in the while() loop is to see if there are any touch events and whether a tag is assigned. If the CLEAR button is tapped, the signature area is cleared by resetting the GRAM to the original zero values:

```
tag = Gpu_Hal_Rd8(phost,REG_TOUCH_TAG);
if(tag=='0')
{
        Gpu_CoCmd_Dlstart(phost);
        Gpu_CoCmd_MemZero(phost,10*1024L,480L*272L); // Clear the gram from 1024
        App_Flush_Co_Buffer(phost);
        Gpu_Hal_WaitCmdfifo_empty(phost);
}
```

The display list is started and buffers flushed:

```
Gpu_CoCmd_Dlstart(phost);        // start
App_WrCoCmd_Buffer(phost, CLEAR(1,1,1));
```

Next, the background stars are drawn. This is done by repeating a single star image over the entire screen (actually double the size of the screen – as if a large sheet of printed stars were laid over the screen extending beyond the edges). This is repeated a total of three times – one each for red, green and blue coloured stars. The three layers of stars are made semi-transparent and blended to allow all of the items to be seen even if they're on top of each other. Each time through the while() loop, the three layers are rotated to give the appearance of spinning sheets of stars. In addition to the spinning layers, the star images themselves are rotated.

The display context is saved and restored, similar to a microcontroller stack push and pop. This allows the foreground image to remain static while the background images are changed:

```
App_WrCoCmd_Buffer(phost,SAVE_CONTEXT());
App_WrCoCmd_Buffer(phost,BLEND_FUNC(SRC_ALPHA, ONE));
App_WrCoCmd_Buffer(phost,COLOR_RGB(78, 0, 0));
App_WrCoCmd_Buffer(phost,CMD_LOADIDENTITY);
rotate_around(ox, oy, 47*a);
App_WrCoCmd_Buffer(phost,VERTEX2II(0, 0, 13, 1));
```

```
App_WrCoCmd_Buffer(phost,COLOR_RGB(0, 40, 0));
rotate_around(ox + sw, oy, 53*a);
App_WrCoCmd_Buffer(phost,VERTEX2II(0, 0, 13, 1));

App_WrCoCmd_Buffer(phost,COLOR_RGB(0, 0, 78));
rotate_around(ox, oy + sh, 57*a);
App_WrCoCmd_Buffer(phost,VERTEX2II(0, 0, 13, 1));
App_WrCoCmd_Buffer(phost,RESTORE_CONTEXT());
```

After displaying the new star layers and returning to the foreground, the signature area and CLEAR button are displayed. The CLEAR button is assigned a tag of '0', which is checked at the beginning of the while() loop.

```
Gpu_CoCmd_FgColor(phost,0xffffff);        // Set the fg color
Gpu_CoCmd_Button(phost,x,y,w,h,31,OPT_FLAT,"");

App_WrCoCmd_Buffer(phost,COLOR_RGB(0,0,0));
App_WrCoCmd_Buffer(phost,BEGIN(BITMAPS));
App_WrCoCmd_Buffer(phost,VERTEX2II(x,y,1,0));

if(tag=='0')
      Gpu_CoCmd_FgColor(phost,0x003300);
else
      Gpu_CoCmd_FgColor(phost,0x005500);
App_WrCoCmd_Buffer(phost,COLOR_RGB(255, 255, 255));
App_WrCoCmd_Buffer(phost,TAG('0'));
Gpu_CoCmd_Button(phost,DispWidth / 2 - sw / 4, DispHeight - sh / 2 - 3, sw / 2,
sh / 2,  28, 0, "CLEAR");
```

The final section of the while() loop is to swap the display list to make it active, flush the buffers and return to the top of the loop. The variable 'a' is used for the star layer rotation angle in the next pass:

```
App_WrCoCmd_Buffer(phost,DISPLAY());
Gpu_CoCmd_Swap(phost);
App_Flush_Co_Buffer(phost);
Gpu_Hal_WaitCmdfifo_empty(phost);
a+=1;
```

5.1 FT801 Signature

The FT8X1/FT813 capacitive display uses a different (capacitive) display touch controller compared to the FT8X0/FT812 resistive controller which has a slower sample rate than the resistive display. To ensure smooth interaction between the user touch and the application, CMD_SKETCH is replaced with CMD_CSKETCH. To enable this alternative command in the sample program open the Platform.h file and look for:

```
#define FT801_ENABLE
```

By default this is undefined (FT8XXmode). To switch in the CMD_CSKETCH ensure this line is defined. After making the change, rebuild and run the application.

6 Application Functionality

This application demonstrates the usage of graphics primitives such as bitmaps and inbuilt signature function, text & button widgets.

The application constantly tracks the user touch on the signature area by using sketch function and monitors the user tap on the clear button to clear the signature drawn by the user. The application displays animated star bitmaps in three layers with different colours on the background. These three layers of star bitmaps are rotated with respect to centre axis and rotary rate. The sketch area can be cleared using clear button.

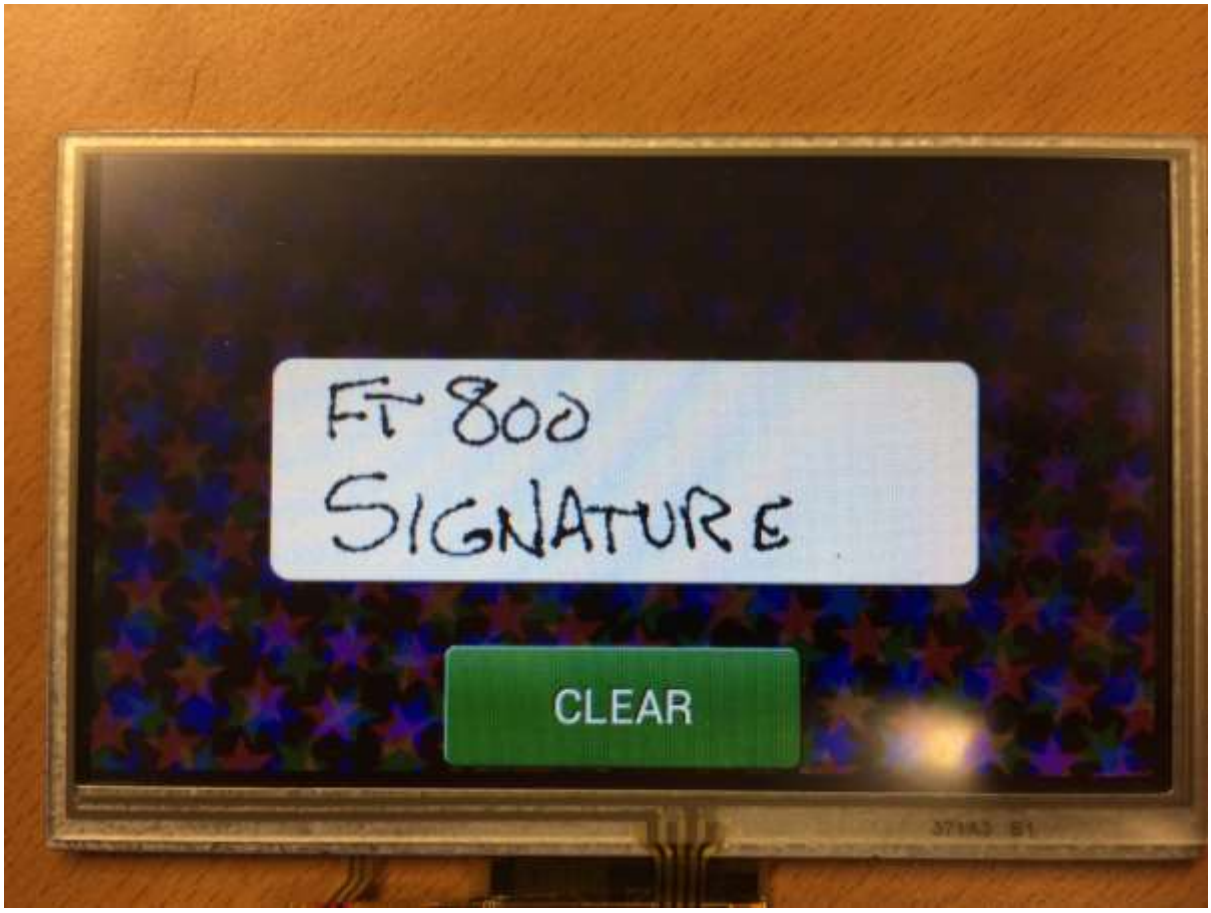

Figure 6.1 Signature Display

7 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_391 EVE Platform Guide
- 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
- VM800C Datasheet
- VM800B Datasheet

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_269 FT_App_Signature
Document Reference No.: BRT_000199
Clearance No.: BRT#118
Product Page: http://brtchip.com/product/
Document Feedback: Send Feedback

Revision	Changes	Date
1.0	Initial Release	2013-08-21
1.1	Updated Version	2013-11-01
1.2	Added section 4.4.1	2014-06-30
1.3	Document migrated from FTDI to BRT (Updated company logo; copyright info; contact information; hyperlinks)	2018-01-05