Application Note

AN_420

FT_App_Keyboard

Version 1.1

Issue Date: 2018-01-05

This application note describes the operation of the Keyboard Demo Application running on an FT81X. The Keyboard example demonstrates how fast and easy it is to use FT81X commands to construct a fully functional keypad GUI.

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 document describes the operation of the Keyboard Demo Application running on an FT81X. The example demonstrates how fast and easily a fully functional keypad GUI can be constructed using FT81X commands. Please also refer to the sample code project provided on the Bridgetek website which demonstrates all the features of FT81X - http://brtchip.com/eve/.

1.1 Overview

This application demonstrates the coprocessor command and touch tracking features of the FT81X.

The example demonstrates that in addition to providing an attractive graphical user interface for an application, the FT81X's tagging features can be used to construct effective, fully interactive GUI operations rapidly.

1.2 Scope

This document can be used by designers to develop GUI applications on a hardware setup that use an FT81X interfaced to a SPI host. The SPI host can be a PC running Visual Studio (C++) with a C232HM cable or an FT900.

The document covers the following topics:

- Brief overview of the Keypad demonstration
- Flow of the program including the FT81X initialisation and keypad code
- Description of the keypad function within the application
- Running the demonstration program

Additional documentation can be found at http://brtchip.com/eve/.

2 Keyboard Overview

This example demonstrates keyboard development capabilities of the FT81X.

The application supports a multiline editor area with an alphanumeric keyboard. The layout with numbers also provides some special character buttons. The button labelled "a^" acts like a CAPS LOCK key and switches key entry to capital (block) letters. The button labelled "12*" switches the keyboard layout to numbers and some special characters. The "<-"backspace key, "Clear" key and "Space" spacebar key are all available in both CAPS LOCK and numeric entry mode. Figures 2.1 and 2.2 show the keyboard layout in alphabets and numbers.

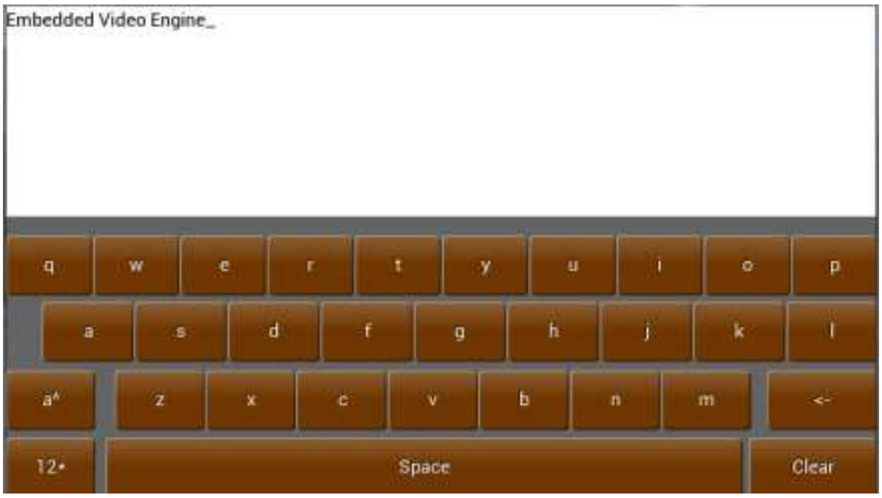

Figure 2.1 The Keyboard example application

Figure 2.2 Keyboard example number layout

This application runs on a host processor which triggers commands for the FT81X over SPI communication. The host processor may be a PC (through an MSVC project application) which uses USB-to-SPI bridge chip from FTDI Chip (e.g. FT232H / FT4222H) or an FT900 which establishes a connection to FT81X over SPI.

The application uses FT81X commands, namely, CMD_KEYS and CMD_BUTTON together with touch controller registers to create this keyboard.

3 Design Flow

3.1 Initialisation

Every EVE design follows the same basic principles as highlighted in Figure 3.1. After configuring the SPI Host itself (such as the PC through the C232HM cable, or an MCU), the application will wake up the FT81X and write to the registers in the FT81X to configure its display, touch and audio settings etc. It then writes an initial display list to clear the screen.

The main application can then create display lists to draw the actual application screens, in this case, the keyboard screen. In essence, there will be two lists; the active list and the edited list which are continually swapped to update the display. Each screen can be created by either writing a display list to the RAM_DL memory in the FT81X or by writing a series of commands to the Co-Processor FIFO in the FT81X (in which case, the Co-Processor will create a display list in RAM_DL based on the commands).

Header files map the pseudo code of the design file of the display list to the FT81X instruction set, which is sent as the data of the SPI packet (typically <1KB). As a result, with EVE's object oriented approach, the FT81X is operating as an SPI peripheral while providing full display, audio, and touch capabilities.

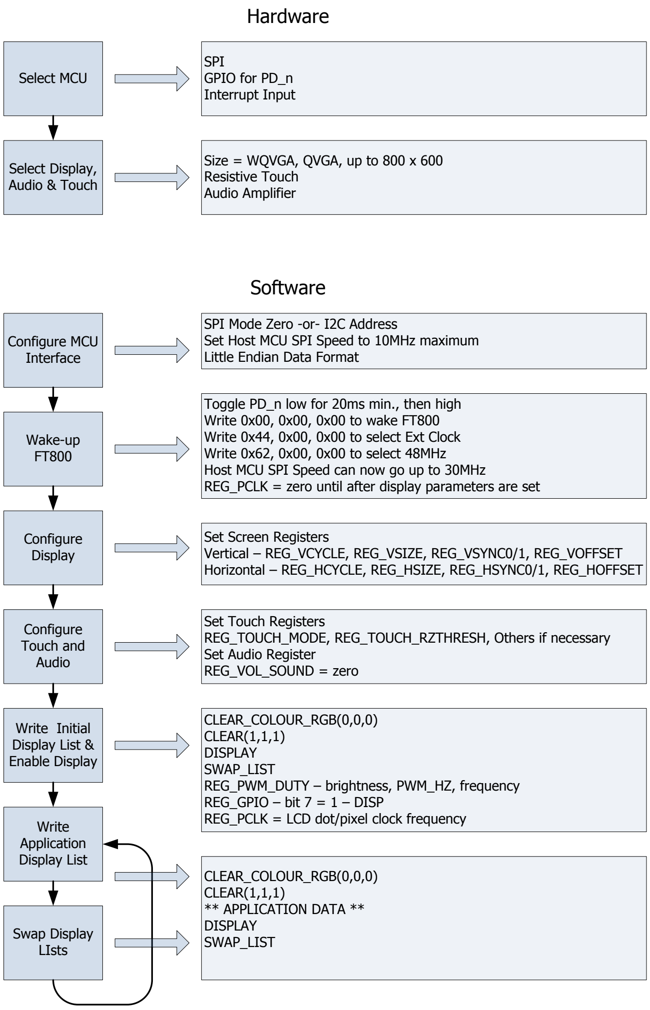

Figure 3.1 Generic EVE Design Flow

3.2 Application Flow

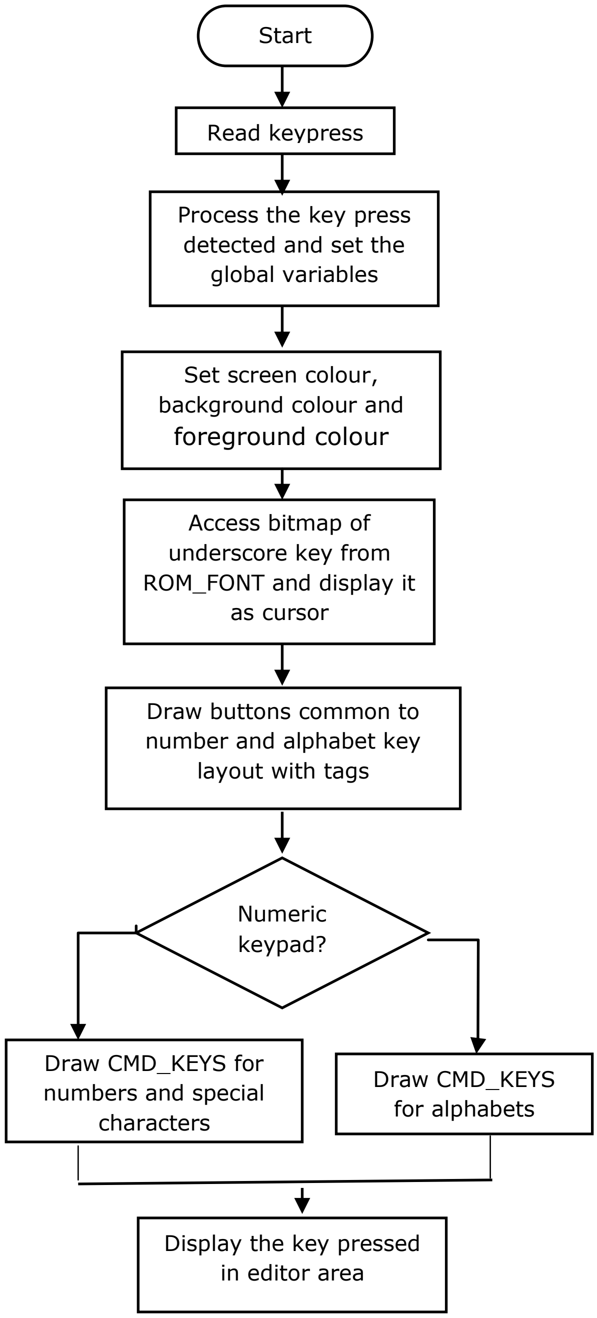

Figure 3.2 Application Flowchart

4 Description

4.1 Application Start Screen

Refer to AN_391 EVE Platform Guide for information pertaining to platform setup and the necessary development environment.

Upon setting up the platform, the application start screen is displayed –

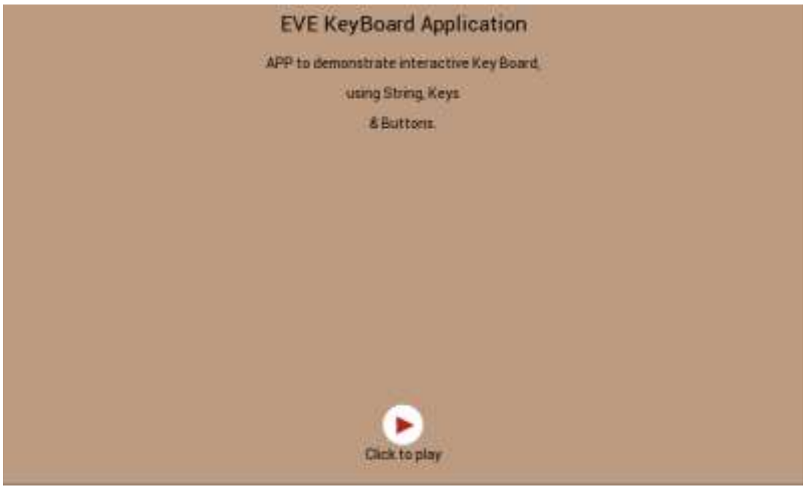

Figure 4.1 Start screen

4.2 Keypad Function

This is the main function in which the application will now remain. After initialization, the code will run in a continuous loop where it will perform the following functions.

- Check the Tag registers and determine the keypress
- Draw the background, and clear an area with the scissor functions
- Draw those buttons common for both number and alphabet keypad layout with labels and tags
- Draw the keypad layout by determining whether number keypad or alphabet with labels and set tags
- Display the key pressed

Following are the crucial global variables and macros which are controlling the keypad operation.

1. `uint8_t Read_Keypad()` is the function to read a keypress. Debounce logic is also included in it. The registers `REG_TOUCH_TAG` and `REG_TOUCH_RAW_XY` are used to detect the key presses.

2. `void Notepad(void)` is the function which holds the entire keyboard functioning logic.

3. The multiline editor area content is held in a global variable called Buffer which is a structure type as shown below.

```
struct Notepad_buffer
{
        char8_t *temp;
        char8_t  notepad[MAX_LINES][80];
}Buffer;
```

4. The following variable Flag maintains the information about the keypad layout to be drawn.

```
struct
{
        uint8_t Key_Detect :1;
        uint8_t Caps :1;
        uint8_t Numeric : 1;
        uint8_t Exit : 1;
}Flag;
```

5. To display the next character input position on the editor area, the current character width is read from the font metric block and added to the current display position thereby computing the next character input position.

```
Disp_pos += Gpu_Rom_Font_WH(Buffer.notepad[Line][0],Font);     // Update
the Disp_Pos
noofchars+=1;
```

6. The detected key press is processed in an if/else block based on whether the key pressed is a special function or normal character/number key press. The following special functions are defined for this purpose.

```
#define SPECIAL_FUN      251
#define BACK_SPACE    251                         // Back space
#define CAPS_LOCK     252                         // Caps Lock
#define NUMBER_LOCK   253                         // Number Lock
#define BACK          254                         // Exit
```

7. The TAG display list command along with CMD_BUTTON and CMD_KEYS co-processor commands are used to draw the complete keyboard. The pressed keys are displayed in editor area using CMD_TEXT co-processor command.

5 Running the Demonstration Code

This package has project solutions to build and execute the keypad application using the following IDEs.

- <...>\App_Keyboard\Project\Msvc\ has a Visual Studio (C++) project in which a PC is used as an SPI host to send FT81X commands. This setup requires a USB-to-SPI bridge chip to establish communication from PC to FT81X. The development modules like ME812AU and ME813AU have this bridge chip mounted on the PCB.

- <...>\App_Keyboard\Project\FT90x has an Eclipse project in which an FT900 is used as the SPI host. The FT90x Toolchain can be downloaded from Bridgetek website.

- <...>\App_Keyboard\Project\Msvc_Emulator has a Visual Studio (C++) project for an Emulated version of the FT81X. This project doesn't need any hardware to launch the application.

The user of this application needs to open the project in one of the IDE listed above, followed by enabling the platform macros in the Platform.h header file. Depending on the underlying hardware, one of the following platform configurations must be chosen.

```
//#define VM800B43_50                          (1)
//#define VM800B35                             (1)
//#define VM801B43_50                          (1)
//#define VM810C50                             (1)
#define ME812AU_WH50R                          (1)
//#define ME813AU_WH50C                        (1)
```

Now the project should be able to build and run.

Figure 5.1 - Keyboard

6 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

- Datasheet for VM800C
- Datasheet for VM800B
- AN_391 EVE Platform Guide
- FT8XX Series Programmer Guide
- 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 Figures & Tables

List of Figures

List of Tables

NA

Appendix C– Revision History

Document Title: AN_420 FT_App_Keyboard

Document Reference No.: BRT_000054

Clearance No.: BRT#048

Product Page: http://brtchip.com/product/

Document Feedback: Send Feedback

Revision	Changes	Date
1.0	Initial release	2016-11-08
1.1	Document migrated from Dual Branding to BRT (Updated company logo; copyright info; contact information; hyperlinks) Updated source code and pictures	2018-01-05