Application Note

AN_367

VM800P43/50 Swipe Lists Sample

Version 1.0

Issue Date: 2015-03-24

This document provides a guide for using the Swipe Lists sample code.

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 note documents an example firmware project for the VM800P43A. The source code is available in the "examples\FT_VM800P43_50\Intermediate\Swipe" folder of the FTDI Arduino library.

Refer to the following document for further details of the Arduino libraries: AN_318 Arduino Library For FT800 Series

1.1 Overview

The application implements swipe lists, as found on tablets and mobile phones, and has been written to run on a VM800P43A evaluation module:

- ATMEGA328P MCU (Arduino PRO 5V, 16MHz) processor
- FT800 processor + 4.3" touch panel display
- SD card connector
- RTC with battery backup
- IO expandability via daughter board

Note: The code can be modified to run on other hardware attached to an FT800 enabled display, requiring replacement of the Arduino specific calls to the FT800 with the relevant calls for the platform being developed on.

1.2 Features

The Swipe Lists example has the following features:

- Open source firmware.
- Touch input for control and selection.
- Displays graphics output on the FT800 enabled display via SPI.

1.3 Library dependency

The following FTDI libraries and the Arduino IDE must be installed before running the application. Refer to the library application note in the download location for instruction on how to install the library.

-Arduino Specific library for FT800 (library for the VM800P43/50 and examples)

1.4 Limitations

Code has only been tested on the VM800P43A module.

1.5 Scope

The guide is intended for developers who are creating applications, extending FTDI provided applications or implementing example applications for the VM800P43A module.

The sample application consists of the source code.

2 Swipe Lists Firmware

The firmware included in the example code demonstrates displaying graphics on an FT800 display and detecting touches on the touch screen, when using swipe lists.

2.1 How swipe lists work

Swipe lists are constructed of two main functions, one to display the items in the list and one to track user touches to allow traversal of the list.

2.1.1 Swipe list display

A swipe list is simply a grid of items with a certain number of rows and columns (determined by values supplied by the user) displayed at a certain offset relative to the display area, with the display area acting as a window onto the entire list.

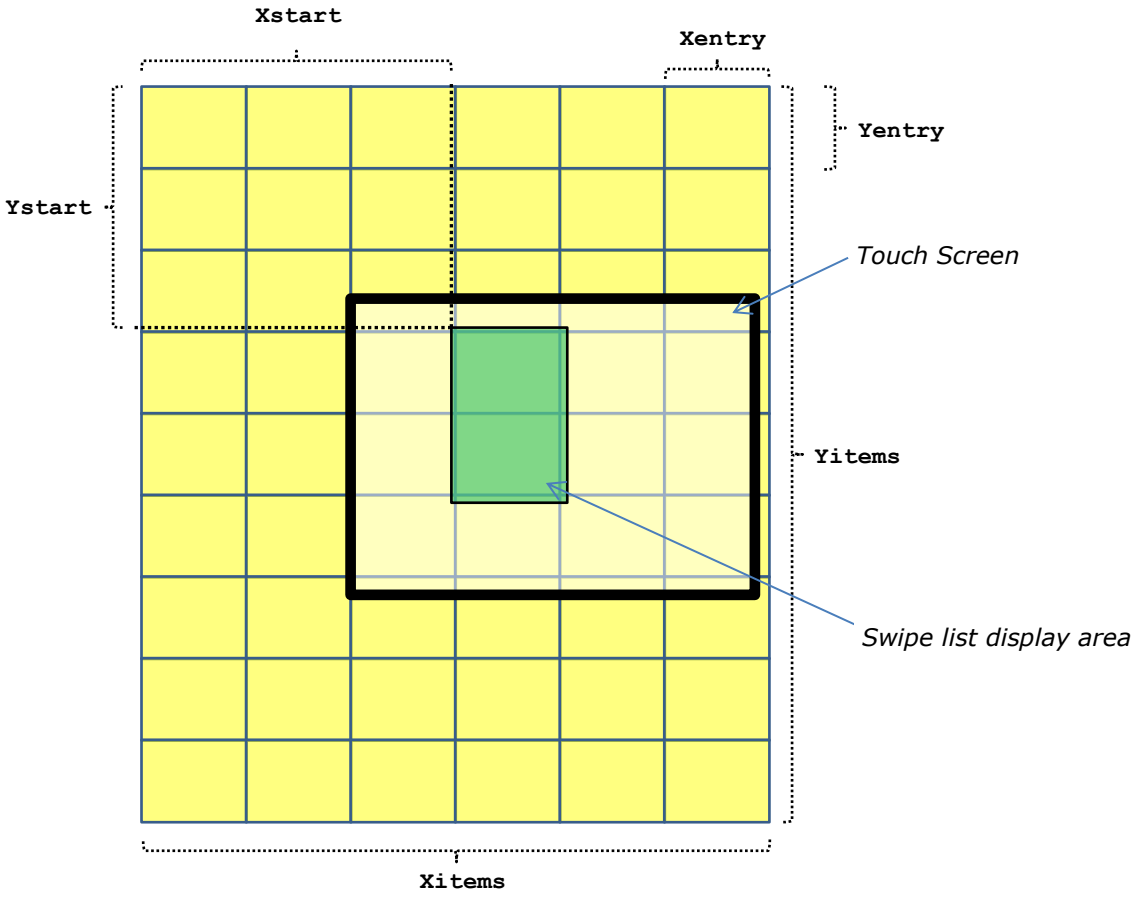

Figure 1: Layout of a swipe list and the variables that define it

The variables are as follows:

- **`Xstart, Ystart`** – Current offset of list relative to top left of swipe list display area
- **`Xentry, Yentry`** – Width and height of each item in the swipe list
- **`Xcoord, Ycoord`** – Position of top left corner of swipe list display area
- **`Xwidth, Ywidth`** – Width and height of swipe list display area

2.1.1.1 Swipe list display constraints

At all times, the offsets of the display list must be kept within the bounds determined by the display area and boundary type of the swipe list.

The display type determines which offsets can be changed and is one of the following:

- **SWIPE_BOTH_WAYS**
  - X offset can change
  - Y offset can change
- **SWIPE_VERTICAL**
  - X offset must not be changed (always 0)
  - Y offset can change
- **SWIPE_HORIZONTAL**
  - X offset can change
  - Y offset must not be changed (always 0)

The boundary type determines the range within which the offsets must be kept and is one of the following:

- **SWIPE_LIMIT**
  - X offset must not be allowed to fall below 0 or be greater than the width of the list minus the display width
  - Y offset must not be allowed to fall below 0 or be greater than the height of the list minus the display height
- **SWIPE_WRAPS**
  - X offset must not be allowed to fall below 0 or be greater than the width of the list
  - Y offset must not be allowed to fall below 0 or be greater than the height of the list

2.1.2 Swipe list touch tracking

An overview of the method of controlling swipe lists is as follows …

1) When a touch is made within the display area of an active swipe list:
    - Make a note of the swipe list touched
    - Make a note of the initial touch coordinates
    - Make a note that this is a selection touch (initially)
    - Reset X and Y speed of list to 0

2) While within the display area of the swipe list touched and still touching:
    - Read the current coordinates
    - Calculate the difference between the initial and current coordinates
    - Display the swipe list offset by the difference in the coordinates
    - If the difference between initial and current touch coordinates exceeds a set limit (at any time) then touch will be treated as a dragging action

3) At this point the user has moved out of the display area or has finished touching the screen:
    - Selection action:
        - Use current display position combined with the last coordinates to determine which item was at the last position touched (if any) and mark it as selected
    - Dragging action:
        - Update the current display coordinates to include difference between initial and last touched coordinates
        - Set X and Y speed of list to obtain the difference between the last two touch positions

2.1.2.1 Swipe list traversal

As mentioned above, traversal of the list is achieved by simply tracking the difference between the initial touch point and where the touch point is moved to, adding this difference to the current X,Y offsets (**`Xstart`** & **`Ystart`**).

- Decreasing the X offset: list moves on at left and moves off at right
- Increasing the X offset: list moves on at right and moves off at left

- Decreasing the Y offset: list moves on at top and moves off at bottom
- Increasing the Y offset: list moves on at bottom and moves off at top

2.1.2.2 Swipe list automatic scrolling

Scrolling the list automatically at speed, due to a fast swipe, is again easy to implement.

Simply set the X and Y speeds to the difference between the last two touch points, with the values being added to the X and Y offsets every time the display is updated, while at the same time reducing the speed until it hits zero (at which point the display stops moving).

2.2 Implementing new swipe lists

Everything the user requires for creating and handling swipe lists is provided in the example code.

2.2.1 Creation of a swipe list

Creating a swipe list requires a call to the **`swipeSetupList()`** function, providing the following details:

- **`list`**
  - Used to identify the particular list (an index into an array of swipe lists)
- **`type`**
  - The orientation of the swipe list (**`SWIPE_BOTH_WAYS`**, **`SWIPE_VERTICAL`** or **`SWIPE_HORIZONTAL`**)
- **`line`**
  - For **`SWIPE_BOTH_WAYS`** or **`SWIPE_VERTICAL`** it specifies the number of columns, with the number of rows being derived by dividing the number of items by the number of columns (rounded up to a whole number of rows)
  - For **`SWIPE_HORIZONTAL`** it specifies the number of rows, with the number of columns being derived by dividing the number of items by the number of rows (rounded up to a whole number of columns)
- **`item`**
  - The number of items in the list
- **`xgrd, ygrd`**
  - The X and Y width of each item
- **`xcrd, ycrd`**
  - The X and Y coordinates of the top left corner of the display area
- **`xsiz, ysiz`**
  - The X and Y width of the display area
- **`wrap`**
  - The type of boundary checking performed (**`SWIPE_LIMIT`** or **`SWIPE_WRAPS`**)

An example call to set up a swipe list follows:

```
swipeSetupList(
SWIPE_LIST_V_1C_L,
SWIPE_VERTICAL,
1,
ITEM_TOTAL,
SWIPE_ITEM_XSIZ, SWIPE_ITEM_YSIZ,
190, 8,
SWIPE_ITEM_XSIZ, 220,
SWIPE_LIMIT
);
```

Figure 2: Example call to set up a swipe list

2.2.2 Swipe list control and display

A timer function is called from within the main while loop (which does not exit), and is used to determine the period at which display updates and touch tracking are performed. The function operates by noting the current millisecond time and then checking to see if it has incremented, at which point it decrements the counter `msTickTockDisplay` as shown below:

```
void milliTimer()
{
        msTimeCountWas = msTimeCountNow;
        msTimeCountNow = millis();

        // update count downs if a millisecond (or so) has passed ...
        if (msTimeCountNow != msTimeCountWas)
        {
                if (msTickTockDisplay)
                {
                        msTickTockDisplay--;

                        // Update display if count down reaches 0 ...
                        if (msTickTockDisplay == 0)
                        {
                                // reset timer ...
                                msTickTockDisplay = DISPLAY_REFRESH_RATE;
                                // update display ...
                                renderDisplay();
                        }
                }
        }
}
```

Figure 3: MilliTimer() Function

Upon the counter reaching 0, a call is made to the applications main display rendering function, which itself calls the core swipe list functions to track touches and also render the swipe lists.

The general structure of the main display rendering function is as follows:

```
void renderDisplay()
{
        // Initialise variables

        // Call routine to track all touches on swipe lists
        swipeTouchHandler();

        // Start display list
        FTImpl.DLStart();

        // Render background etc
        …

        // Call routine to render all active swipe lists
        swipeDisplayHandler();


        // End of display list
        FTImpl.DLEnd();
        FTImpl.Finish();
}
```

Figure 4: RenderDisplay() Function

The **`swipeTouchHandler()`** and **`swipeDisplayHandler()`** functions iterate through the list of available swipe lists, tracking touches and rendering displays for all lists that are active.

Note: The timer function is by no means an accurate millisecond timer routine, but it is more than satisfactory for our needs.

2.2.3 Display rendering functions

The user is required to supply a number of functions for each swipe list created, which are as follows:

- OPEN function … responsible for performing any operations which must be done before the main rendering takes place. For example, setting the size and colour of the font.
- ITEM function … responsible for the rendering of a single item. This routine will be automatically called the requisite number of times to render the entire visible area of the swipe list. For example, this could draw a button with text.
- SHUT function … responsible for performing any operations which must be done after the main rendering. For example, playing a sound depending on the item selected.

The rendering routine is passed the X and Y coordinates and size of the item to be displayed, together with the item number and currently selected item (allowing a selected item to be highlighted in a manner determined by the user).

```
void swipeRenderOpen_Demo()
{
        // nothing to do!
}

void swipeRenderItem_Demo(int16_t xcrd, int16_t ycrd, uint16_t xsiz, uint16_t ysiz, uint8_t item,
uint8_t pick)
{
        // Highlight item if currently selected ...
        if (item == pick)
        {
                // Open section ...
                FTImpl.Begin(FT_RECTS);
                // Draw rectangle to indicate item is selected ...
                FTImpl.LineWidth(8 * 16);
                FTImpl.ColorRGB(stepperCycleR, stepperCycleG, stepperCycleB);
                FTImpl.Vertex2f((xcrd + 9       ) * 16, (ycrd + 9       ) * 16); // Top left
                FTImpl.Vertex2f((xcrd + xsiz - 9) * 16, (ycrd + ysiz - 9) * 16); // Bottom right
                FTImpl.ColorRGB(0xFF, 0xFF, 0xFF);
                // Shut section ...
                FTImpl.End();
        }

        // Draw button showing item text ...
        FTImpl.Cmd_Button(xcrd + 6, ycrd + 6, xsiz - 13, ysiz - 13, 27, 0, swipeItemText[item]);
}

void swipeRenderShut_Demo()
{
        // nothing to do!
}
```

Figure 5: Open, Item and Shut user supplied functions

It should be noted that the number of user supplied functions required is not necessarily 3x the number of lists. For example, if five lists are required which have no specific requirement for open and shut functions, the following can be used:

- A single empty function which is used by the open/shut operations of all lists
- Five individual functions to perform item rendering for each of the lists

2.2.4 Display rendering constraints

The `swipeDisplayHandler()` function operates in the following way (for each active swipe list):

1) Graphics context is saved before commencing item rendering
2) Scissor rectangle is set to encompass the swipe list display area
3) The user supplied OPEN function is called
4) The user supplied ITEM function is called the relevant number of times to ensure all visible items are rendered
5) The user supplied SHUT function is called
6) Graphics context is restored following completion of item rendering

Due to the use of the scissor rectangle during swipe list rendering, to clip the rendering of items to the swipe list display area, user supplied routines should not normally modify the scissor clip rectangle in order to ensure correct rendering of the swipe list.

However, if the user requires the use of scissor rectangles within their routines, these should be constrained within the swipe list display area.

3 Possible Improvements

The following are possible improvements …

- Add indicator bars to show the position of the (currently) displayed items relative to the size of the list
- Add angular swipe speed tracking for lists which can move in all directions. Currently the X/Y speeds are decreased linearly at the same rate, meaning movement can cease in one direction whilst still moving in the other. If the angle of the swipe was calculated then this would allow the movement of the list to follow the same angle of the swipe, coming to rest in both X and Y directions simultaneously.

4 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

Appendix A – References

Document References

Arduino Library Guide:

http://www.ftdichip.com/Support/Documents/AppNotes/AN_318_Arduino_Library_For_FT800_Series.pdf

FT800 Series Programmers Guide:
http://www.ftdichip.com/Support/Documents/ProgramGuides/FT800%20Programmers%20Guide.pdf

EVE home page:

http://www.ftdichip.com/EVE.htm

Acronyms and Abbreviations

Terms	Description
API	Application Programming Interface
EVE	Embedded Video Engine
RTC	Real Time Clock
SD Card	Secure Digital Card
USB-IF	Universal Serial Bus

Appendix B – List of Tables & Figures

4.1 List of Tables

No tables defined.

4.2 List of Figures

Appendix C – Revision History

Document Title: AN_367 VM800P4350 Swipe Lists Sample
Document Reference No.: FT_001161
Clearance No.: FTDI# 444
Product Page: http://www.ftdichip.com/FTProducts.htm
Document Feedback: Send Feedback

Revision	Changes	Date
0.1	Initial Draft	2014-12-16
1.0	First Release	2015-03-24