Application Note

AN_312

FT800 Example with ARM

Version 1.0

Issue Date: 2014-04-01

The FTDI FT800 video controller offers a high quality, high value solution for embedded graphics requirements. In addition to the graphics, resistive touch inputs and an audio output provide a complete human machine interface to the outside world.

This application note will provide a simple example of developing ARM C code to control the FT800 over SPI. The principles demonstrated can then be used to produce more complex applications.

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

The FT800 operates as an SPI or I²Cperipheral to a system processor which provides a high quality, high value, complete, human interface experience by the incorporation of graphics rendering, touch screen sensing and audio capabilities. It is controlled over a low-bandwidth SPI or I^2C interface allowing practically any microcontroller to be used.

For this application, a STM32F4-Discovery board is coupled with the VM800C43A-D, FT800 development kit and the Keil µVision 5 and the STM32CubeMX development environments. The target application demonstrates the use of the standard STM32/ARM libraries to initialize and display different elements on the LCD of the VM800C development system. The design flow follows concepts introduced in application note AN_240 FT800 From the Ground Up.

Once these concepts are mastered, this example can be used as a basis for larger and more complex applications. This example shows the usage of the FT800 with graphics primitives (e.g. lines, shapes, and text), inbuilt widgets (e.g. sliders, switches, and dials), touch events and audio output.

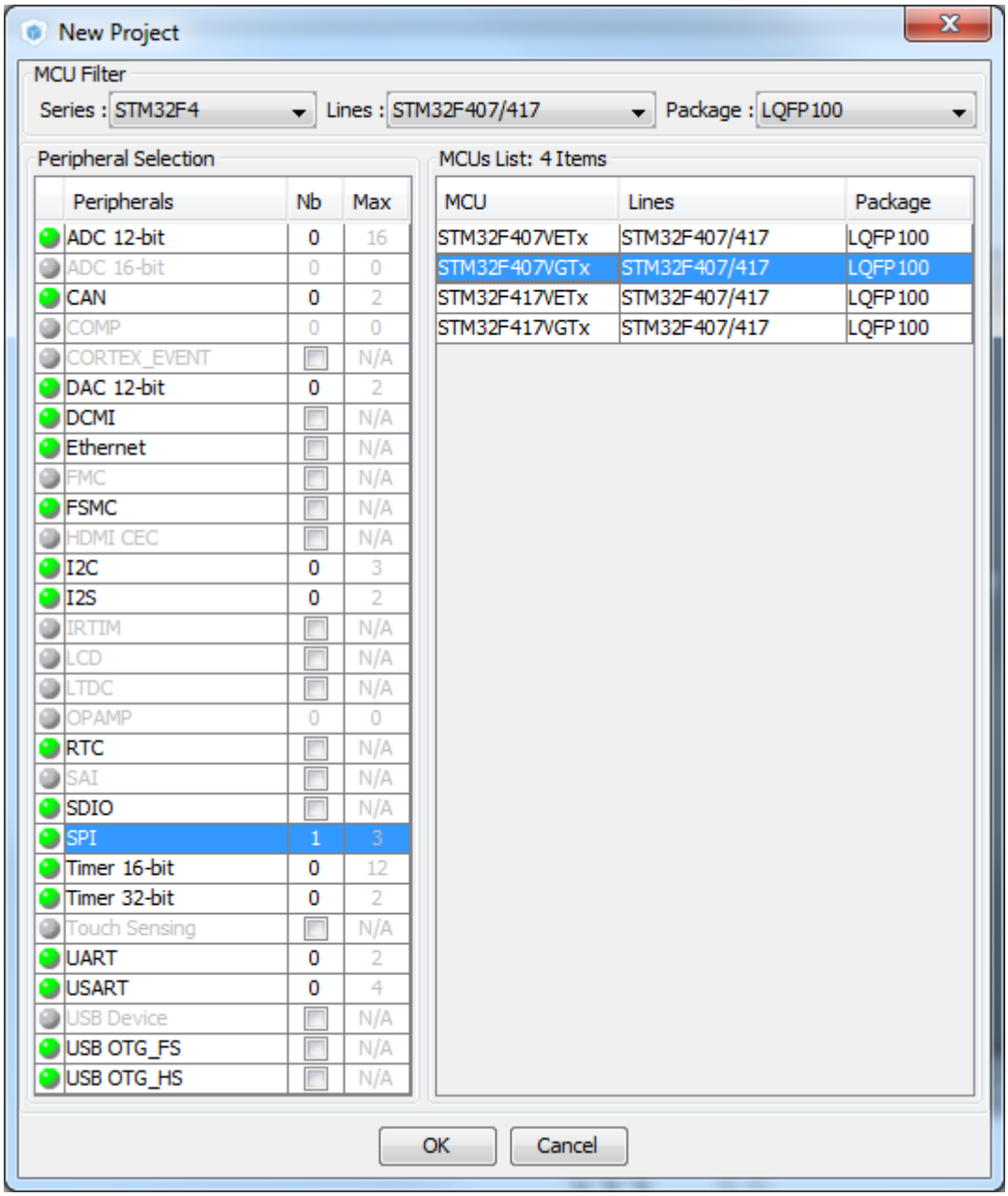
Note: It is recommended to view the code while reading this application note.

2 Software Architecture

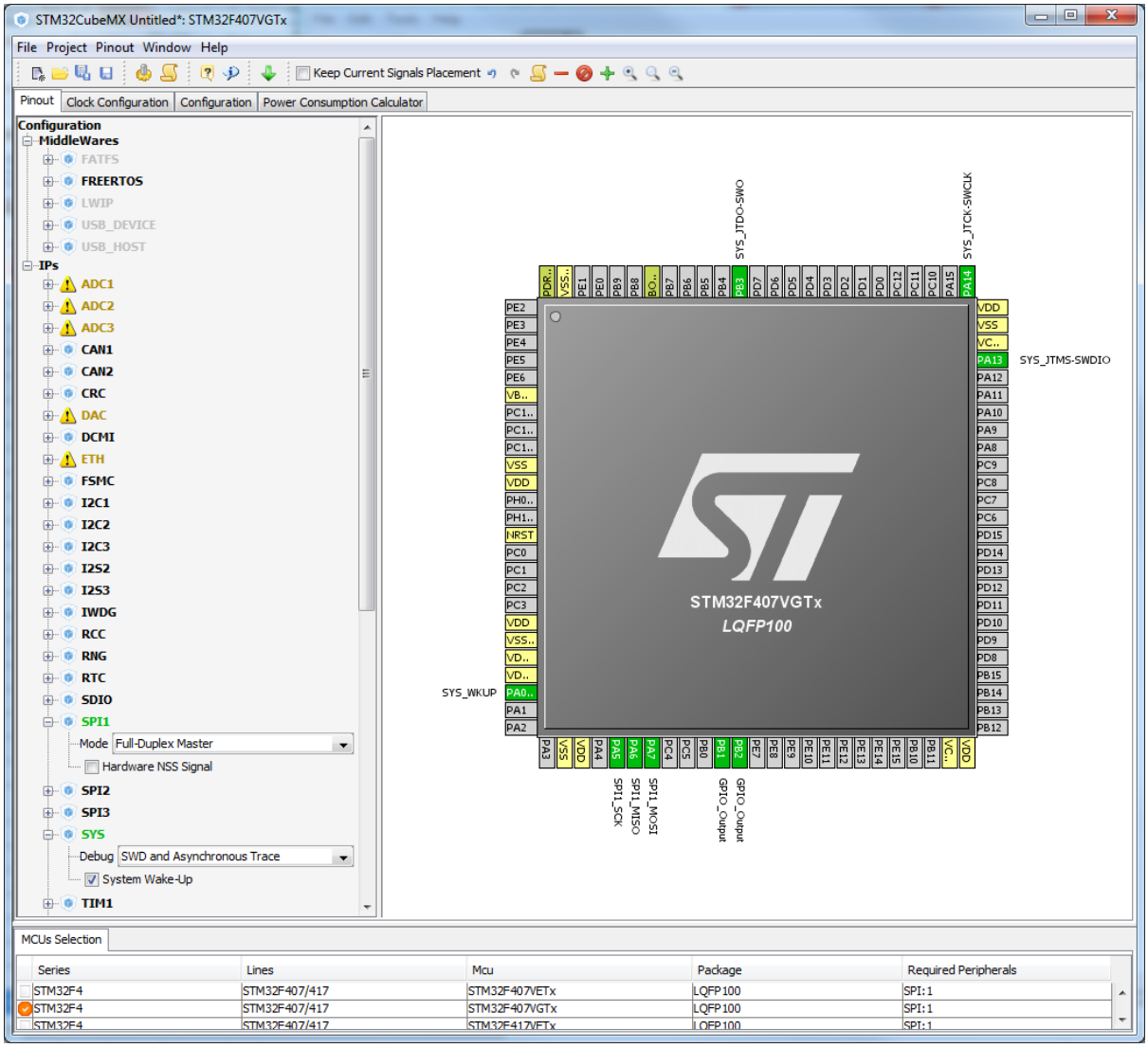
2.1 Project Shell Setup – STM32CubeMX

A new project shell is created through the STM32CubeMX utility available from ST Microelectronics. From the project wizard select the following options and peripherals:

- MCU = STM32F407VGTx
- SPI = Channel 1



The utility will display a pin-map similar to the following:



Inside of the "SPI" tree:

- Select the mode "Full-Duplex Master"
- Untick the "Hardware NSS Signal" option

Inside of the "SYS" tree:

- Select the debug mode "SWD and Asynchronous Trace"
- Tick the "System Wake-Up" option

All other peripherals and options may be left at the default settings.

The "Clock Configuration" tab may be left at the default settings for purposes of this application note.

The "Configuration" tab configures the SPI port parameters. The defaults match the FT800 operation:

- Data size = 8 Bits
- Most Significant Bit first
- Clock prescaler = 2 (for 8Mbps)
- CPOL = Low
- CPHA = 1 (First) Edge
- CRC Calculation = Disabled

- NSS Signal Type = Software

The settings under the other peripheral tabs may be left at the default settings.

This completes the STM32F4 project configuration. Since the Keil compiler is used here, when saving the project, select the "MDK-ARM 4.xx" Toolchain / IDE option.

2.2 Keil µVision 5 IDE

The STM32CubeMX utility creates a project that can be opened directly in Keil µVision versions 4 and 5. For this example, version 5 was used. Both are available from www.keil.com. The demo versions can be used with code sizes up to 32KB. This example is approximately 6KB.

The project contains a shell, which initializes all of the selected peripherals and identifies several areas for "USER CODE". The code to interact with the FT800 will be contained within these user code areas.

Multiple files are created from the STM32CubeMX utility. For the most part, these can be remain as-is. They are the drivers for the various peripherals available on the STM32F407 MCU. The only generated file that requires editing is "main.c". A header file containing FT800-specific details is added to the project as well.

- FT800.h – a header file containing all of the FT800-specific values for memory locations, command values, etc.
- main.c – the main program file

An "abstraction layer" concept is common for ARM projects. This example uses these abstracted drivers without any changes and demonstrates the simplicity of sending and receiving data through the FT800 while producing a graphic output.

Source code for the µVision project used in this application note can be found here:
http://www.ftdichip.com/Support/SoftwareExamples/EVE/AN_312.zip

2.2.1 FT800.h

This header file contains all of the information that is specific to the FT800 and assigns meaningful names to each value or address. The file is separated into the following sections:

- Memory Map – base addresses of the sections of memory within the FT800
  - RAM_CMD = 4KB ring buffer to place Co-Processor commands
  - RAM_DL = 8KB buffer to place display lists
  - RAM_G = 256KB general element memory for images, fonts and audio data
  - RAM_PAL = 1KB color palette
  - RAM_REG = FT800 registers
- Register Addresses
  - Each of the FT800 registers is named and associated with its address. Refer to the FT800 Datasheet for register sizes.
- Graphic Engine Commands
  - Each command associated with the FT800 graphics engine is assigned a value according to the FT800 Programmers Guide. Many of the Co-Processor commands require additional arguments.
- Display List Commands
  - Each display list command is assigned a value according to the FT800 Programmers Guide. All display list commands are 4-bytes in length. The first byte is the actual command. The remaining three bytes contain the necessary arguments.
- Command and register value options
  - Assorted named values useful for the main program.
- Useful Macros
  - Assorted macros to perform basic calculations.

2.3 main.c

This is the primary program file. Details of the various drivers and configuration can be found in the respective STM32CubeMX and Keil µVision documentation. The user code sections of main.c are:

USER CODE area 0:

- References the FT800.h header file
- Identifies the GPIO Pins used for SPI chip select and Power Down (RESET).
- Selects the display size
- Defines hex values for some default colors
- Declares all of the global variables used throughout the program
- Declares the function prototypes used for reading and writing the FT800

USER CODE area 1:

- Not used

USER CODE area 2:

- Assigns variables for the physical LCD display parameters
- Wakes the FT800
- Configures and identifies the FT800
- Initializes the display, touch and Audio
- Write an initial display list

USER CODE area 3:

- Infinite loop which instructs the FT800 to draw a circle on the screen that alternates between red and white.

USER CODE area 4:

- Contains all of the functions used to read and write the memory and registers of the FT800

3 User Application

The intent of the user application is to render a circle on the LCD. The initial circle is white, then each time through the loop, the color toggles between red and white. Touch and audio are not covered.

3.1 FT800 Graphics Rendering

There are two methods available for rendering graphics elements, playing audio and sensing touch events.

- The first way is to write display list commands directly to the RAM_DL (Display List RAM).
  Note: This method is illustrated when blanking the screen as part of the initialization of the FT800.

- The second way is to write a series of Co-Processor commands or display list commands to the RAM_CMD (Command FIFO). The Co-Processor then creates the display list in RAM_DL based on the commands which it is given in the RAM_CMD FIFO. This method makes it easier to combine the drawing of graphics objects (lines etc.) and Widgets (slider etc.) on the same screen.
  Note: This method is illustrated when creating the main screen.

Although it is in theory possible to mix both methods when creating a new display list (screen) it is recommended that only one of the two methods is used in any given screen. This is because the RAM_DL would be written by both the MCU and the Co-Processor within the FT800.

3.1.1 Display List

With the Display List, the FT800 can draw several primitives (points, lines, edge and line strips, rectangles and images), manipulate the screen LCD parameters, read and write registers, etc. It can also play synthesized sounds and audio files and work with the raw touch screen activity (X-Y coordinates, touch pressure).

There is an 8K Display List buffer. All display list commands are 4-bytes in length, allowing for up to 2K commands. A display list command is constructed by logically combining the command byte with the necessary parameters. For example, this application uses the POINTS primitive. A BEGIN command is used coupled with the type of object associated with following commands - in this case, POINTS. The value for BEGIN is 0x1F, located in bits 31 through 24 of the command. Bits 23 through 4 are not used (reserved). Bits 3 through 0 indicate the primitive type. POINTS is type 2. The full command is then 0x1F000002.

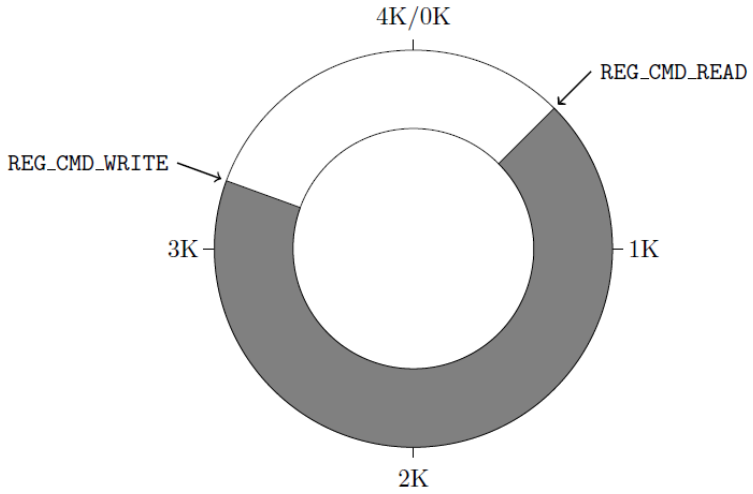
Each display list command is constructed in a similar fashion. When sending these values to the FT800, they are sent in "little-endian" format, or least significant byte first: 0x02, 0x00, 0x00, 0x1F.

Commands are sent to successive locations in the RAM_DL (0x100000) memory of the FT800. The last Display List command of every list is "DISPLAY" to instruct the FT800 to process the commands and draw the screen. Processing is started when the register REG_DLSWAP is written with a value of 1 (swap after the current line) or 2 (swap after the current screen).

At this point, the list is made "active", and a new list can now be written starting at RAM_DL again.

3.1.2 Co-Processor

While the Display List covers basic screen manipulation, the Co-Processor allows more advanced rendering using inbuilt widgets, memory management and touch tags. The command memory for the Co-Processor is located at RAM_CMD (0x108000). While the display list always starts at the beginning of RAM_DL, the Co-Processor uses a 4K FIFO ring buffer.



Note: This FIFO is mapped at FT800 memory addresses 108000h (RAM_CMD) to 108FFFh (RAM_CMD + 4095)

Figure 3.1 - Co-Processor Ring Buffer

The index of the last command executed is held in the register REG_CMD_READ, while the index of the last command written is in REG_CMD_WRITE. The process for adding commands is:

1) Read REG_CMD_READ and REG_CMD_WRITE. Loop here until they're equal.
2) Write new commands starting at REG_CMD_WRITE.
3) Update REG_CMD_WRITE with the next address following the last command in the list (4-byte aligned).
4) With the new value in REG_CMD_WRITE, the Co-Processor will start executing each command and updating REG_CMD_READ until it reaches REG_CMD_WRITE.

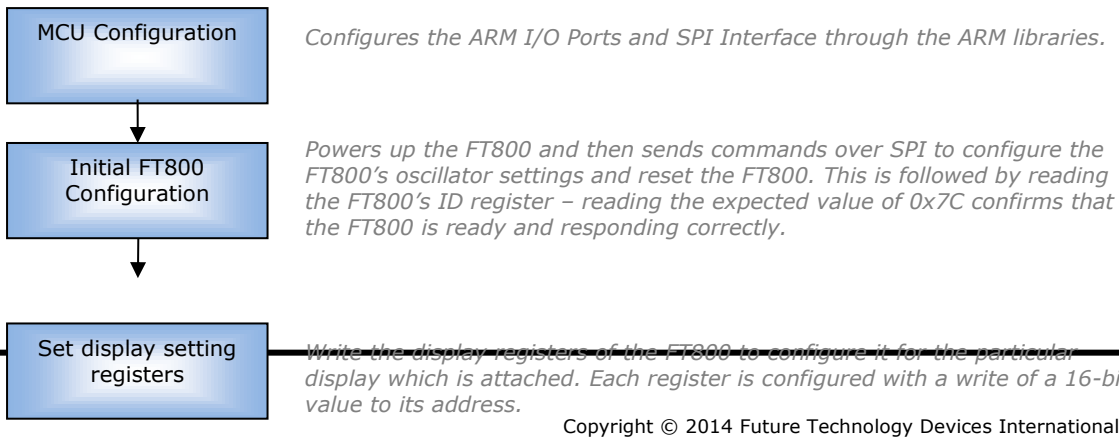
While only one type of rendering is recommended; display list commands can be embedded into the command list. This allows mixing of the primitive graphics elements with widgets allowing the full capabilities of the FT800 to be utilized. The while() loop function utilizes this method of embedding display list commands into the command list.

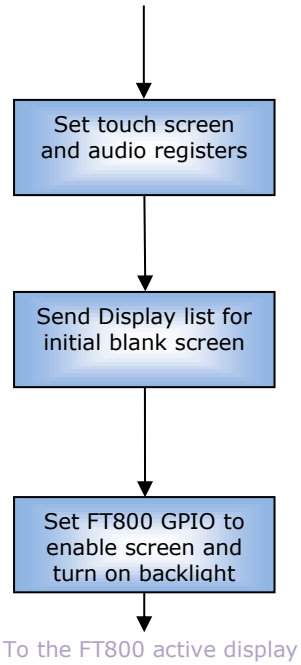
3.2 ARM code

As noted above, the output from the STM32CubeMX utility generates the background code required to access the various peripherals.

3.2.1 FT800 Setup

All FT800 and ARM initialization and configuration requirements are handled in USER CODE areas 0 through 3:





The touch screen threshold is set here. The touch screen and audio functionality is not used in this application note but some the later code examples will use these features.

Create an initial display list which simply blanks the screen. This code writes three 4-byte commands to successive locations in the Display list RAM.

- [RAM_DL + 0] Specify the color which will be used when the screen is cleared
- [RAM_DL + 4] Specify that the Color, Stencil and Tag buffers should be cleared
- [RAM_DL + 8] Display command which signifies the end of the Display List
- Writing to the DL_Swap register then tells the FT800 to render the above display

The FT800 has its own GPIO port which can be controlled by writing to the FT800's GPIO_DIR and GPIO registers over SPI. This part of the code writes to these registers to assert the display's enable pin which is connected to the FT800 GPIO. The FT800 backlight PWM is also ramped up to turn on the LCD panel's LED backlight.

3.2.2 FT800 active display

The generated images are shown while executing the while(1) loop.

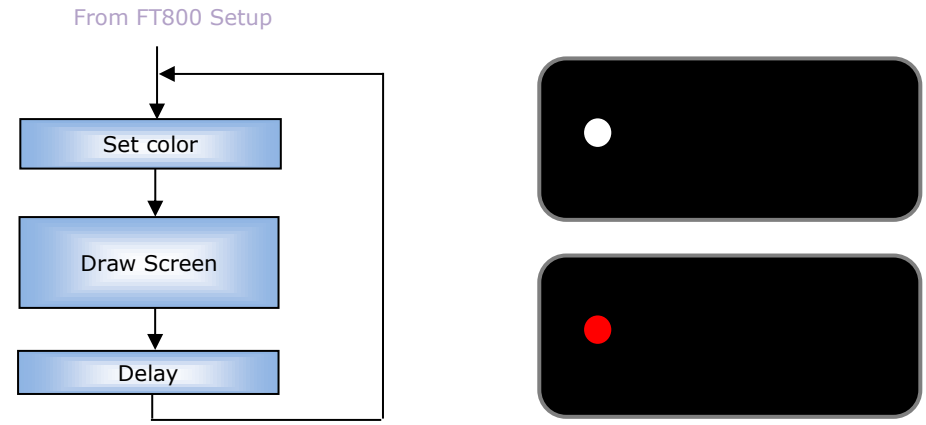


Figure 3.2 - Program Output

The Draw Screen block above performs the following steps:

1. Check the Co-Processor command buffer pointers. Wait until they're equal.
2. Alternate active colors between white and red (start with white).
3. Start the display list.
4. Clear the screen to black to eliminate any artifacts from the previous screen.
5. Clear the color, stencil and tag buffers.
6. Set the active color.
7. Define a point location and size.
8. Display the screen.
9. Swap display lists.

3.2.3 Other functions

Several functions provide basic read and write capabilities for accessing the FT800:

- `void` **`ft800memWrite8`**`(unsigned long ftAddress, unsigned char ftData8)`
- `void` **`ft800memWrite16`**`(unsigned long ftAddress, unsigned int ftData16)`
- `void` **`ft800memWrite32`**`(unsigned long ftAddress, unsigned long ftData32)`
  - This function writes a value from the ARM through the SPI port to the FT800.
  - Available in three "sizes": 8-bits (unsigned char), 16-bits (unsigned int) and 32-bits (unsigned long).
  - The sequence of events is:
    - Set CS# active
    - Send the MEM_WRITE command combined with the 3-byte address
    - Send the data, least significant byte first
    - Set CS# inactive
- `unsigned char` **`ft800memRead8`**`(ftAddress)`
- `unsigned int` **`ft800memRead16`**`(ftAddress)`
- `unsigned long` **`ft800memRead32`**`(ftAddress)`
  - This function reads a value from the FT800 through the SPI port to the ARM.
  - Available in three "sizes": 8-bits (unsigned char), 16-bits (unsigned int) and 32-bits (unsigned long).
  - The sequence of events is:
    - Set CS# active
    - Send the MEM_READ command combined with the 3-byte address
    - Send one dummy byte
    - Read the data, least significant byte first and construct the return value
    - Set CS# inactive
    - Return the read data
- `void` **`ft800cmdWrite`**`(unsigned char ftCommand)`
  - This function sends specific commands from the ARM through the SPI port to the FT800.
  - A command is one of 8 3-byte values as noted in section 4 of the FT800 Datasheet. Since the second and third bytes are always zero, an 8-bit value is passed to the function.
  - The sequence of events is:
    - Set CS# active
    - Send the command byte
    - Send two bytes with the value 0x00
    - Set CS# inactive
- `void` **`incCMDOffset`**`(unsigned int currentOffset, unsigned char commandSize)`
  - This function ensures the command offset of the graphics processor command buffer rolls over within the 4KB ring buffer size.

4 Hardware

The FT800 LCD "Basic" or "Credit Card" module kits can be used with this application. The STM32F4-Discovery board pins are connected to the FT800 kit SPI header with standard 0.025" square post jumper wires.

This application uses compiler directives to select the platform type and LCD size.

```
// Set LCD display resolution here
//#define LCD_QVGA  // QVGA  = 320 x 240 (VM800B/C 3.5")
#define LCD_WQVGA   // WQVGA = 480 x 272 (VM800B/C 4.3" and 5.0")
```

The connection between the STM32F4-Discovery board and either the VM800B or VM800C is made with Seven wires. The STM32F4-Discovery board is powered by a USB connection to a host PC which also provides a debug interface. The FT800 module is powered, in turn, from the STM32 board:

Signal	STM32F4-Discovery	VM800B/ VM800C
SCLK	PA5	1 (SCLK)
MOSI	PA7	2 (MOSI)
MISO	PA6	3 (MISO)
CS#	PB1	4 (CS#)
PD#	PB0	6 (PD#)
+5V	5V	5V
Ground	GND	GND
INT#	Not assigned	Not assigned

Table 4.1 – STM32F4-Discovery to VM800B/C Pin Definitions

5 Conclusion

This application note presented a simple example utilizing a readily available ARM platform with the standard ARM development tools and libraries to initialize the FT800. This was followed by the creation of different displays using the graphics processor commands. Low-level SPI function calls were created to allow convenient methods of sending and receiving data to the FT800.

Other display screens that contain other graphics objects, images and widgets, as well as audio output and touch events can be implemented through changing the code within the while() loop function in USER CODE area 3.

The SPI transfer calls could also be expanded to include more complex, multi-byte transfers useful for image and audio file transfers to and from the FT800 memory. Interrupts can also be enabled to indicate touch events, completion of display lists, etc. Any ARM-compatible circuit can be used.

Note a complete Programming Guide (FT800 Programming Guide) is available that details the complete command language for the EVE Series of display, audio, and touch controllers.

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

Appendix A – References

Document References

EVE Product Page

FT800 Datasheet

FT800 Programmers Guide

VM800C Datasheet – Credit Card sized development board with FT800

VM800B Datasheet – Bezel-mounted Display with FT800

AN_240 EVE From the Ground Up

AN_259 FT800 Example with 8-bit MCU

AN_275 FT800 Example with 8-bit Arduino

STM32F4 Discovery Kit

STM32CubeMX Initialization Code Generator (UM1718)

Keil MDK-ARM Microcontroller Development Kit

Project source code

Acronyms and Abbreviations

Terms	Description
CS#	Chip Select
GND	Ground
HMI	Human-Machine Interface
I^2C	Inter-Integrated Circuit
INT#	Interrupt
LCD	Liquid Crystal Display
MISO	Master In / Slave Out
MOSI	Master Out / Slave In
PD#	Power Down
SCLK	Synchronous Clock
SPI	Synchronous Peripheral Interface

Appendix B – List of Tables & Figures

List of Tables

List of Figures

Appendix C – Revision History

Document Title: AN_312 FT800 Example with ARM
Document Reference No.: FT_001015
Clearance No.: FTDI# 384
Product Page: http://www.ftdichip.com/EVE.htm
Document Feedback: Send Feedback

Revision	Changes	Date
1.0	Initial Release	2014-04-01