Future Technology Devices International Ltd.

AN_357

Android Java D2xx Demo Application for FT4222

Document Reference No.:FT_001142

Version 1.0

Issue Date: 2015-01-21

This document provides the application programming interface (API) for the Java D2xx for Android library.

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
E-Mail (Support): support1@ftdichip.com **Web:** http://www.ftdichip.com

Table of Contents

1 Introduction

FTDI provides a proprietary D2XX interface for easy communication with FTxxxx devices. The D2XX API is common across several operating systems supported by FTDI, namely Windows, Windows CE, Linux, Mac OS X and Android. This document explains how to operate the FT4222H demo APK and the functions of all available UI in the FT4222H demo APK.

1.1 Android Support

A Java class library supporting a USB Host is available and applicable to Android v3.2 or any later series. This library requires no special root access privileges.

1.2 Prerequisites

The following is required to install the FTDI D2xx driver:

- An Android device(recommended),
  - A BSP supporting the Android USB Host API corresponding to AOSP 3.2 or later
  - A contemporary Android device running v3.2 or a later OS, with a USB Host or OTG interface. FTDI testing was conducted using a [Google Nexus 7](http://www.google.com/nexus/7/).

An FTDI chip FT4222H based module to test the FTDI D2xx driver

NOTE:

To develop an application using the FTDI D2xx driver for Android, the development machine must have the Eclipse IDE and an up-to-date version of the Android SDK, including the ADB program and Android ADT Plugin installed. The installation and configuration of these tools is not included in this document. For more information, please see (http://developer.Android.com/sdk/index.html).

The Android device should also have USB Debugging enabled to allow access using the ADB utility. To accomplish this, navigate to Settings > Applications > Development and check the USB debugging option.

A summary of the required configuration is provided in the diagram below.

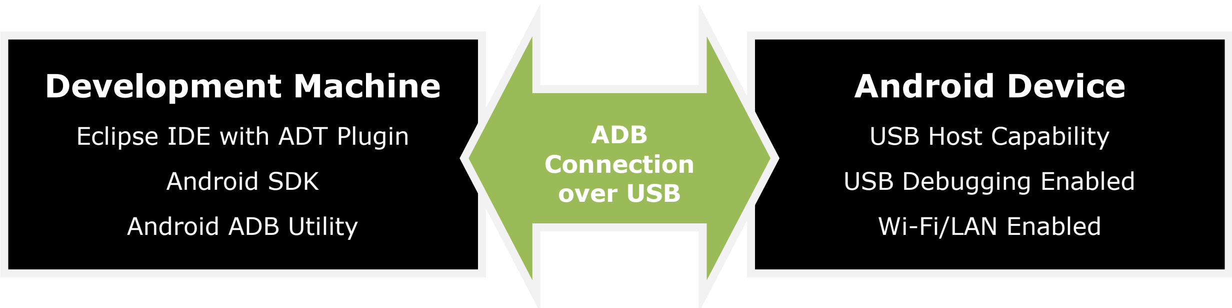

Figure 1: Android Development Configuration

2 Using the FTDI Java D2xx for Android Library

2.1 Introduction and Usage

To support versatile tablet usage scenarios, Google has included a USB Host API in Android since version 3.1. Before version 3.1, an Android application could not access USB devices attached to a system naturally without root access rights. The Android USB Host API removes this limitation, allowing users to utilize the USB gadgets attached to an Android Host or OTG port.

FTDI provides a Java class library that adapts to applications so that the developer can focus on the desired input and output data. The design goal of the class library is to provide access to all the D2XX functions including, EEPROM functions.

The D2xx library can be included in an Android application project in Eclipse. First, copy the library file, d2xx.jar, to the folder of the project, and add it in "Project"–"Properties". Refer to Figure 2 ~ Figure 4, which show how to add the library file from the "\libs" sub-folder of the project folder.

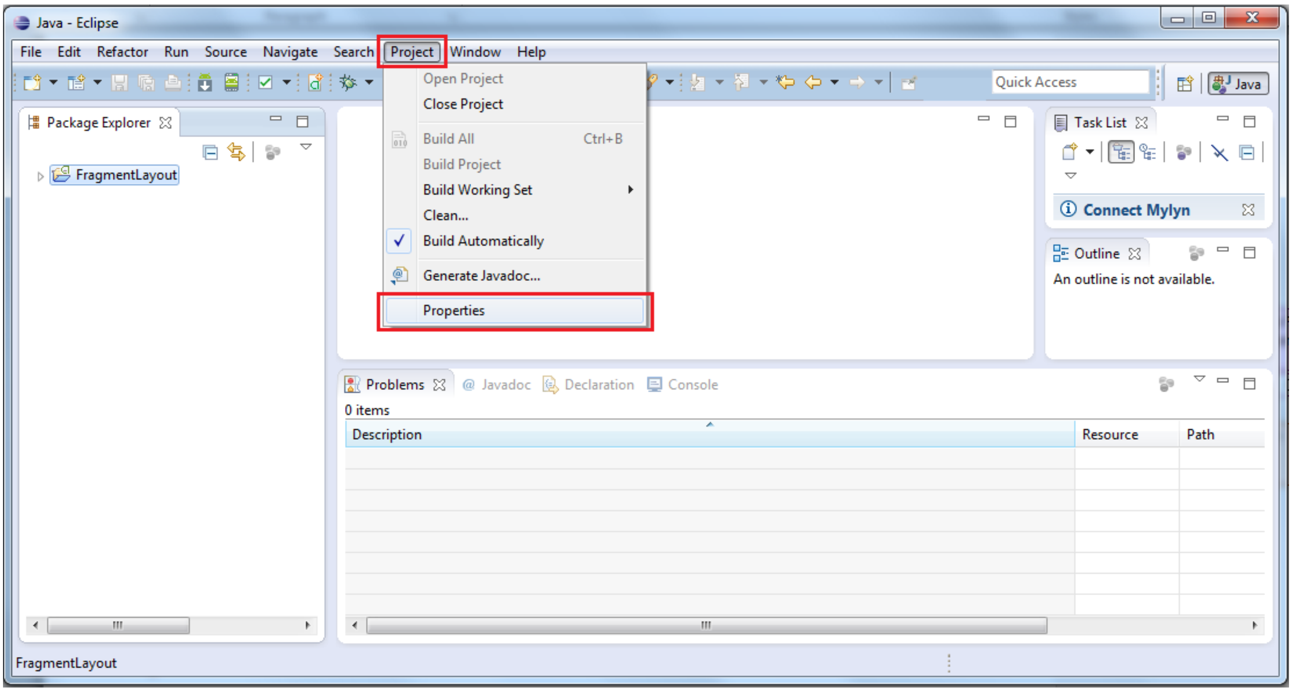

Figure 2: Select "Project" – "Properties"

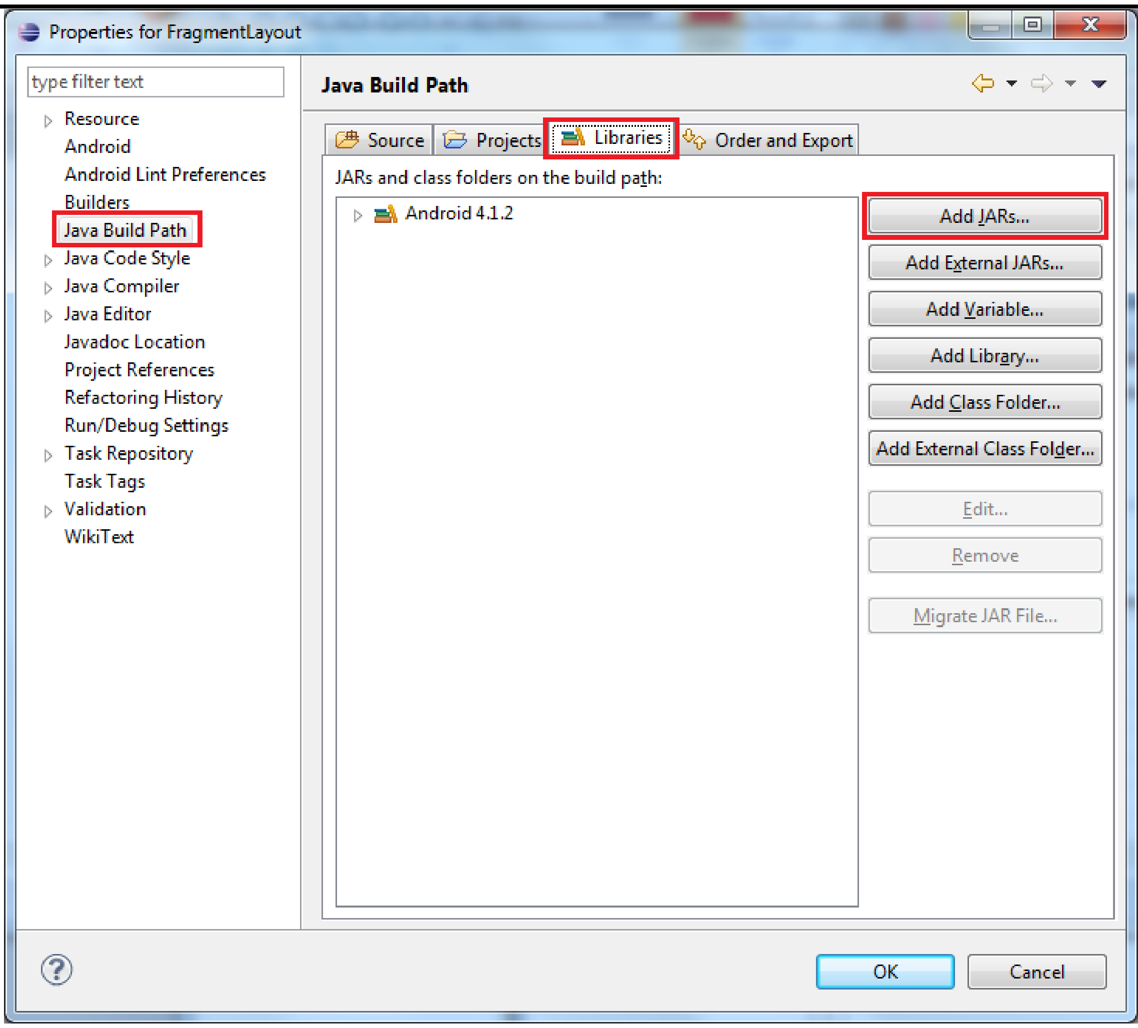

Figure 3: Select "Java Build Path" – "Libraries" – "Add JARs"

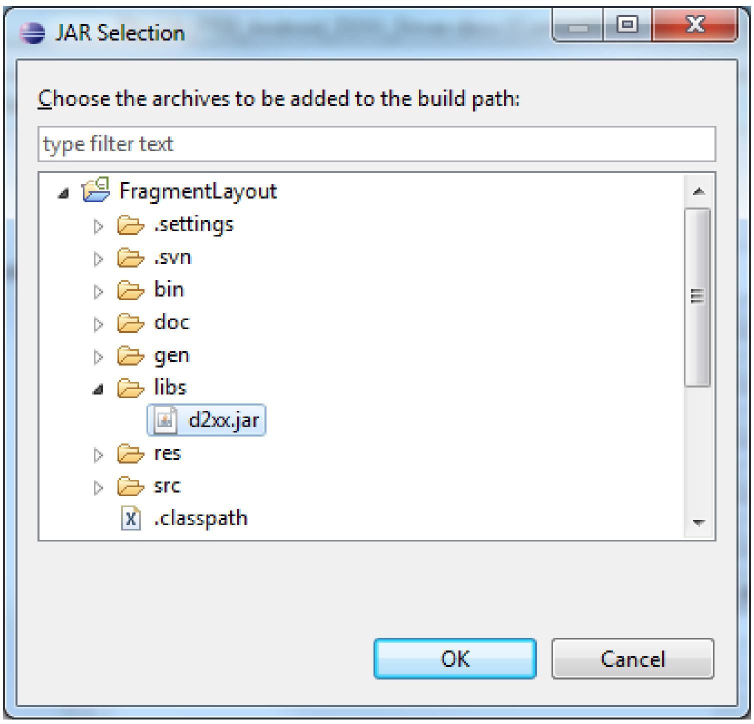

Figure 4: Select the library file – d2xx.jar

The D2xx Java library is fully documented using Javadoc. For information on the D2xx Android library methods, constants and sub-classes, please consult the corresponding Javadoc entry in the sample project's /doc directory.

A sample application demonstrating how to use various methods in the D2xx Java library is shown in section **3 Demo Application**.

3 Demo Application

3.1 Information

Figure 5 Demo Application Screenshot – Information

This demo application contains a **Refresh** button. Tap **Refresh** to access information about the connected device.

3.2 SPI

3.2.1 SPI Hardware

The diagram below shows how the FT4222H can be connected for use with this demo. Two FT4222H are set to CNFMODE0 and have hardwired SPI master/slave pins. Then a USB Hub is used for connecting both chips to an Android device. Please also refer to the FT4222H datasheet for more information.

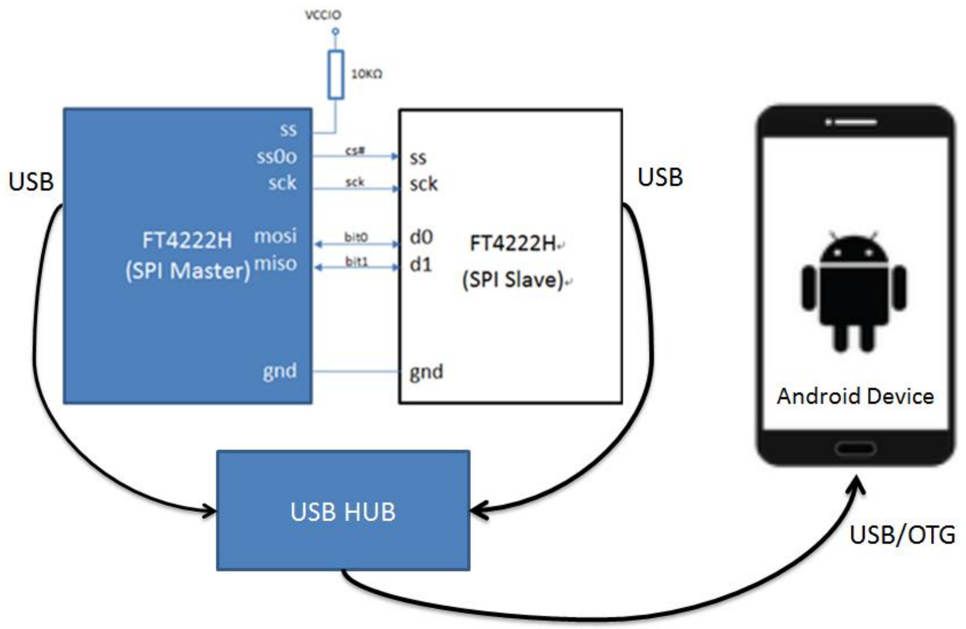

Figure 6 SPI Demo System Diagram

3.2.2 SPI Application

Figure 7 Demo Application Screenshot – FT4222 –SPI

This demo application contains **Config, Master Read, Master Write, Slave Read** and **Slave Write** buttons .

The first row contains a **Config** button to set the SPI configuration with several selectable setting items.

The SPI configuration settings allow the **Polarity / Phase** to be set for CPOL = 0 or 1, CPHA = 0 or 1.

System clock may be set to 24, 48, 60 or 80 Mhz.

Divider may be set to a value between 1/2 and 1/512.These parameters are used to generate the SPI clock and to sample data for SPI read/write functions. For instance, with an 80 MHz system clock and 1/2 divider, the SPI clock will be 40 MHz. The polarity and phase are the sampling trigger to read data from the input pin.

The **Config** button has to be pressed before sending data.

From the second row to the last row are the control fields for transmitting and receiving data; data exchanging functions are divided into Master Write, Master Read, Slave Write, and Slave Read fields.

The Master Write field:

The **Write Data** is an editable text widget which allows any text data to be transmitted to the SPI slave. The **Transmit Bytes** widget is a message box which shows the total number of bytes that the master has successfully transmitted.

The **Master Write** is a button widget for triggering write operation.

The Slave Write field:

The **Write Data** is an editable text widget which allows any text data to be transmitted to the SPI master . The **Transmit Bytes** widget is a message box shows the total number of bytes that the slave has successfully transmitted. The **Slave Write** is a button widget for triggering a write operation.

The Master Read field:

The **Number of Bytes** widget is an editable text field which specifies the total number of bytes that the SPI master is expected to receive. The **Read Data** widget is a message box which shows the content of receiving data. The **Receive Bytes** widget is a message box shows the total number of bytes that the SPI master actually received from the SPI slave. The **Master Read** is a button widget for triggering a read operation and data must transmit from the **Slave Write** in order to obtain the data.

The Slave Read field:

The **Number of Bytes** widget is an editable text field which specifies the total number of bytes that the SPI slave is expected to receive. The **Read Data** widget is a message box which shows the content of received data. The **Receive Bytes** widget is a message box which shows the total number of bytes that the SPI slave actually receive from the SPI master. The **Slave Read** is a button widget for triggering a read operation and data must be transmitted from the **Master Write** prior to the read operation

3.3 I²C

3.3.1 I²C Hardware

The diagram below shows how the FT4222H can be connected to be used with this demo. Two FT4222H devices are set to CNFMODE0 and have hardwired I²C master/slave pins. Then a USB Hub is used for connecting both chips to an Android device. More detailed information can be found in FT4222H datasheet.

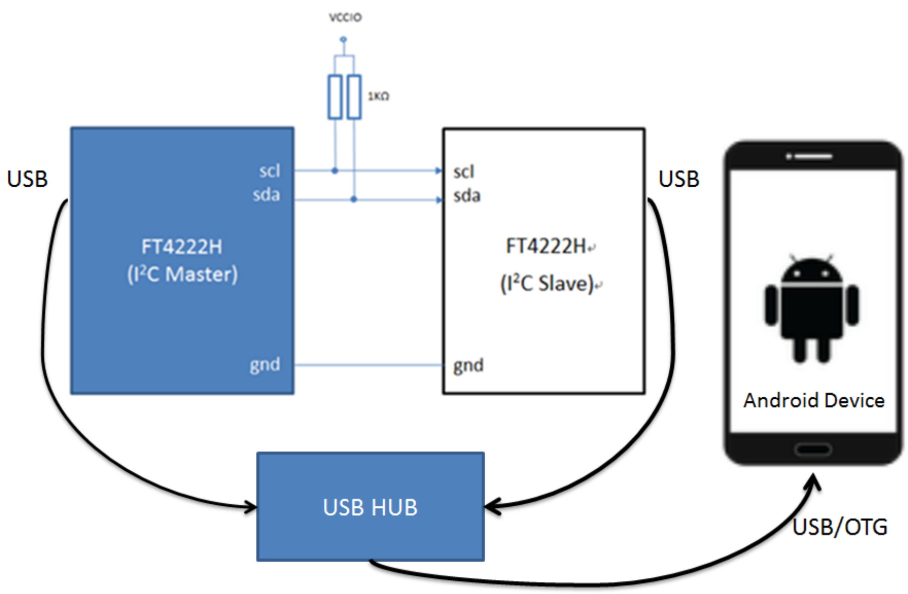

Figure 8 I²C Demo System Diagram

3.2.2 I²C Application

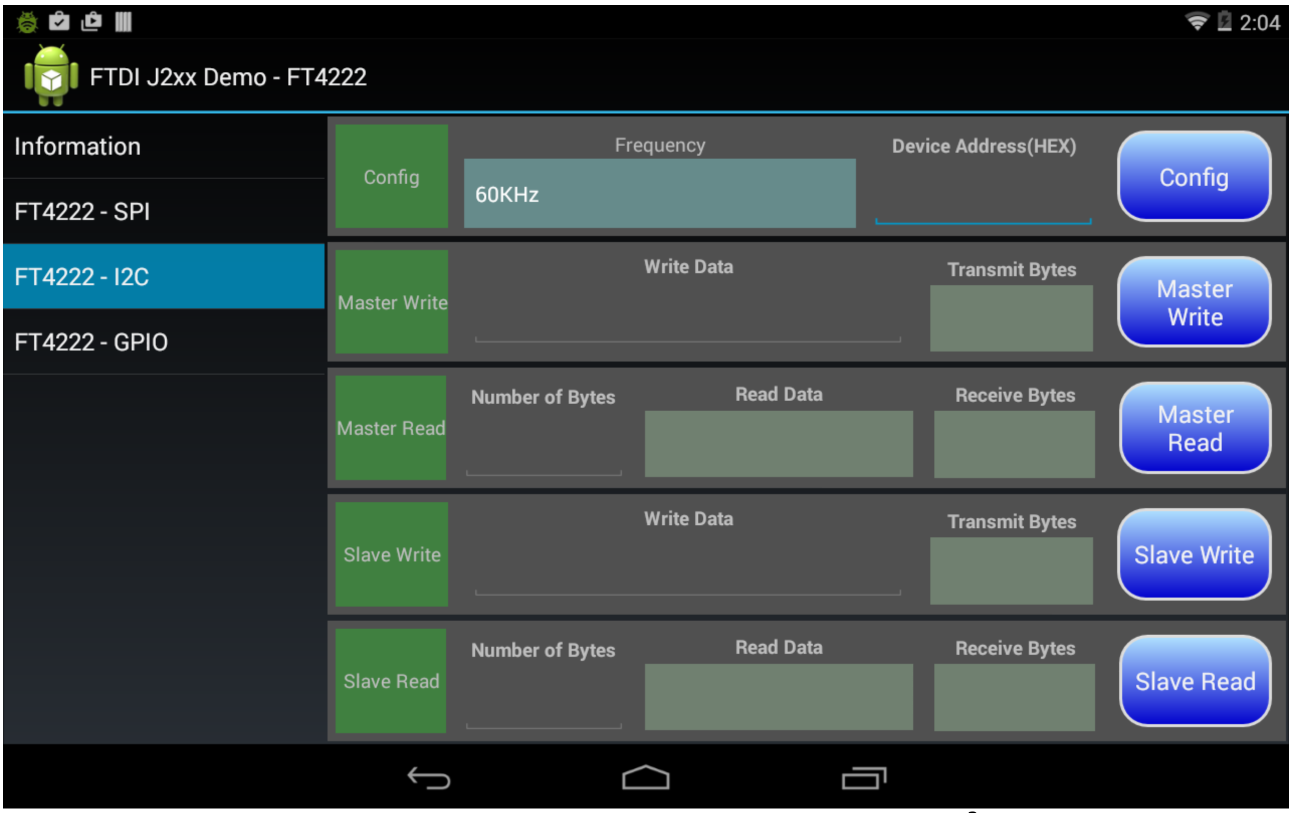

Figure 9 Demo Application Screenshot – FT4222 I²C

This demo application contains **Config, Master Read, Master Write, Slave Read** and **Slave Write** buttons.

The first row contains a **Config** button to set the I^2C configuration with several selectable setting items.

The I^2C configuration settings allow the **Frequency** to be set for 60K, 100K, 400K, 1M or 3.4MHz.

Device Address(HEX) allow the user to input the I^2C address between 0x00~ 0x7F.

The **Config** button has to be pressed before sending data.

From the second row to the last row are the control fields for transmitting and receiving data; data exchanging functions are divided into Master Write, Master Read, Slave Write, and Slave Read fields.

The Master Write field:

The **Write Data** is an editable text widget which allows any text data to be entered for transmitting to the I^2C slave. The **Transmit Bytes** widget is a message box which shows the total number of bytes that the master has successfully transmitted. The **Master Write** is a button widget for triggering the write operation.

The Slave Write field:

The **Write Data** is an editable text widget which allows any text data to be entered for transmitting to the I^2C master. The **Transmit Bytes** widget is a message box which shows the total number of bytes that the slave has successfully transmitted. The **Slave Write** is a button widget for triggering the write operation.

The Master Read field:

The **Number of Bytes** widget is an editable text field which specifies the total number of bytes that the I^2C master expects to receive. The **Read Data** widget is a message box which shows the content of received data. The **Receive Bytes** widget is a message box which shows the total number of bytes that the I^2C master actually receives from I^2C slave. The **Master Read** is a button widget for triggering a read operation and data must transmit from the **Slave Write** at the same time in order to obtain the data.

The Slave Read field:

The **Number of Bytes** widget is an editable text field which specifies the total number of bytes that the I^2C slave expect to receive. The **Read Data** widget is a message box which shows the content of receiving data. The **Receive Bytes** widget is a message box which shows the total number of bytes that the I^2C slave actually received from the I^2C master. The **Slave Read** is a button widget for triggering a read operation and data must transmit from the **Master Write** prior to the read operation.

3.4 GPIO

3.4.1 GPIO Hardware

The diagram below shows how the FT4222H GIPOs can be connected to be used with this demo.

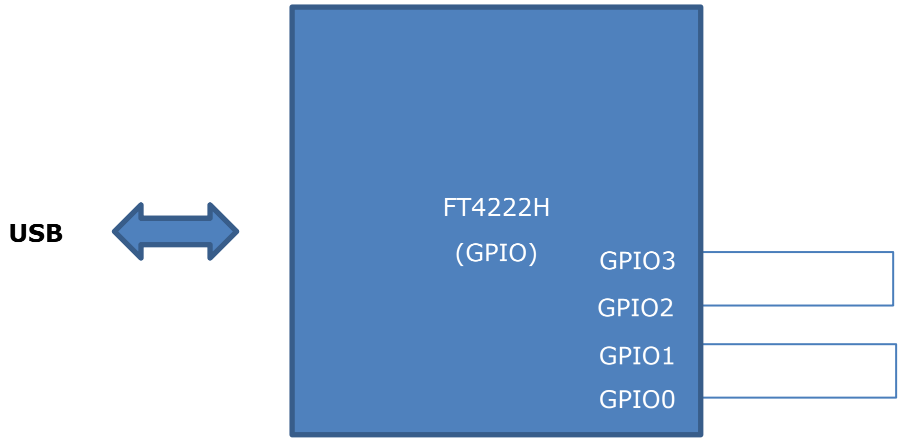

Figure 10 GPIO Block Diagram

3.4.2 GPIO Application

Figure 11 Demo Application Screenshot – FT4222 – GPIO

The GPIO demo application provides four configurable GPIO pins and they are listed in four columns. Each column has **Input**, **Output**, **High** and **Low** fields to control the GPIO's direction and signal level. From figure 8, the demo is hardwired with GPIO0 to GPIO1 and GPIO2 to GPIO3. Based on the hardware setting, GPIO0/GPIO1 GPIO2/GPIO3 directions are mutually exclusive. In other words, changing the GPIO0 to the output direction will automatically set GPIO1 to the input direction and vice versa. In addition, by changing the output port signal level, the input port will automatically respond to the signal level change. With this functional demonstration, GPIOs can be proven to work correctly on triggering bi-directional signals.

4 Appendix A – References

http://developer.Android.com/index.html

http://www.ftdichip.com/Support/Documents/DataSheets/ICs/DS_FT4222H.pdf

http://www.ftdichip.com/Support/Documents/TechnicalNotes/TN_147_Java_D2xx_for_Android.pdf

4.1 Acronyms and Abbreviations

Terms	Description
D2xx	FTDI's proprietary "direct" user space driver interface running on-top of Android USB Host API
OS	Operating System
USB	Universal Serial Bus
BSP	Board Supporting Package
API	Application Programming Interface
OTG	On The Go
SDK	Software Development Kit
ADT	Android Development Tools
IDE	Integrated Development Environment
ADB	Android Debug Bridge
SPI	Serial Peripheral Interface
I^2C	Inter-Integrated Circuit
GPIO	General Purpose Input/Output

5 Appendix B – List of figures

6 Appendix C – Revision History

Document Title: Android D2xx API User Manual

Document Reference No.: FT_001142

Clearance No.: FTDI#

Drivers Page: http://www.ftdichip.com/Drivers/D2XX.htm

Document Feedback: Send Feedback

Revision	Changes	Date
1.0	Initial Release	2015-01-21

7 Contact Information

Head Office – Glasgow, UK

Future Technology Devices International Limited
Unit 1, 2 Seaward Place, Centurion Business Park
Glasgow G411HH
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
Web Site URL	http://www.ftdichip.com
Web Shop URL	http://www.ftdichip.com

Branch Office – Taipei, Taiwan

2F, No. 516, Sec. 1, NeiHu Road
Taipei 114
Taiwan , R.O.C.
Tel: +886 (0) 2 87913570
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
Web Site URL	http://www.ftdichip.com

Branch Office – Tigard, Oregon, 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
Web Site URL	http://www.ftdichip.com

Branch Office – Shanghai, China

No. 666 West Huaihai Road,
Shanghai, 200052
China
Tel: +86 2162351596
Fax: +86 2162351595

E-mail (Sales)	cn.sales@ftdichip.com
E-mail (Support)	cn.support@ftdichip.com
E-mail (General Enquiries)	cn.admin@ftdichip.com
Web Site URL	http://www.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.