Future Technology Devices International Ltd.

Accessing Android Open Accessory Mode with Vinculum-II

Application Note AN_181

Document Reference No.: FT_000502

Version 1.0

Issue Date: 2011-08-22

This application note demonstrates how the VNC2 device can enable the Open Accessory Mode in compatible Android devices and transfer data to and from the Android device over USB.

Future Technology Devices International Ltd (FTDI)

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 (Support): support1@ftdichip.com

1 Introduction

Android Open Accessory Mode is a new feature in Android 3.1 (back-ported to 2.3.4) whereby a USB host device can connect to the Android device to allow data transfer to and from the Android device over USB.

This application note demonstrates how the VNC2 device can enable the Open Accessory Mode in compatible Android devices and transfer data to and from the Android device over USB.

The application note will demonstrate how the VNC2 drivers for Open Accessory Mode (available as of IDE version 1.4.2) are loaded onto the VNC2 and how a simple application accesses them. There will also be a small Android platform app to complete the demonstration.

For development puposes a V2EVAL development board with V2EVAL-64 daughter card was used. http://www.ftdichip.com/Support/Documents/DataSheets/Modules/DS_V2EVAL_Rev2.pdf

for the VNC2 development.

A Motorola Xoom running Android OS version 3.1 was used as the Android target.

The demonstration shows the ability of the VNC2 to enumerate the Xoom device, interrogate the device to determine if it supports Open Accessory Mode, then enable the Open Accessory Mode and re-enumerate the Xoom.

When the USB link has been established the application running on the Xoom tablet will be able to control the LEDs on the V2EVAL platform, while the V2EVAL buttons will be able to control "LEDS" on the Android application GUI.

Full source code and precompiled ROM files are provided on an as is basis.

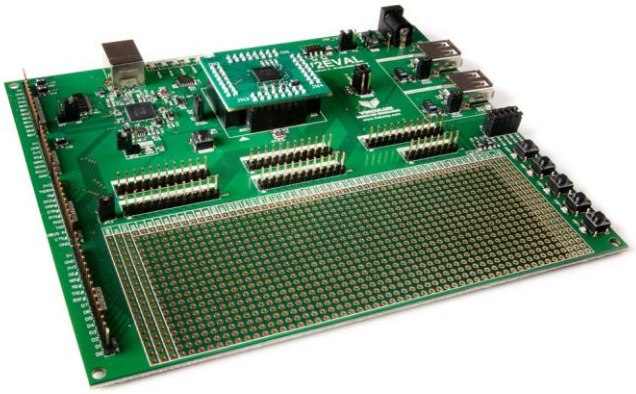

Figure 1.1 – V2-EVAL with daughter card

1.1 VNC2 Devices

VNC2 is the second of FTDI's Vinculum family of embedded dual USB host controller devices. The VNC2 device provides USB Host interfacing capability for a variety of different USB device classes including support for BOMS (bulk only mass storage), Printer and HID (human interface devices). For mass storage devices such as USB Flash drives, VNC2 transparently handles the FAT file structure.

Communication with non USB devices, such as a low cost microcontroller, is accomplished via either UART, SPI or parallel FIFO interfaces. VNC2 provides a new, cost effective solution for providing USB Host capability into products that previously did not have the hardware resources available.

VNC2 allows customers to develop their own firmware using the Vinculum II software development tool suite. These development tools provide compiler, assembler, linker and debugger tools complete within an integrated development environment (IDE).

The Vinculum-II VNC2 family of devices are available in Pb-free (RoHS compliant) 32-lead LQFP, 32-lead QFN, 48-lead LQFP, 48-lead QFN, 64-Lead LQFP and 64-lead QFN packages For more information on the ICs refer to http://www.ftdichip.com/Products/ICs/VNC2.htm

1.2 Motorola Xoom

This application example uses a Motorola Xoom as the Android target. It was chosen for no other reason than it was readily available and already had Android OS 3.1 installed which is required for the Open Accessory Mode. Other Android platforms could be used instead.

See: http://www.motorola.com/Consumers/GB-EN/Consumer-Products-and-Services/ANDROID-TABLETS/MOTOROLA-XOOM-with-Wi-Fi-GB-EN?WT.srch=1&WT.mc_id=EMEA_GB-EN_XOOM_Aug-2011&WT.mc_ev=click

For more information on the Motorola Xoom.

Figure 1.2 – Motorola Xoom Tablet

Table of Contents

2 Block Diagram

This block diagram, Figure 2.1, shows the interconnect required for the demonstration.

After the project ROM file is loaded onto the V2EVAL platform the development PC (red block) is not required.

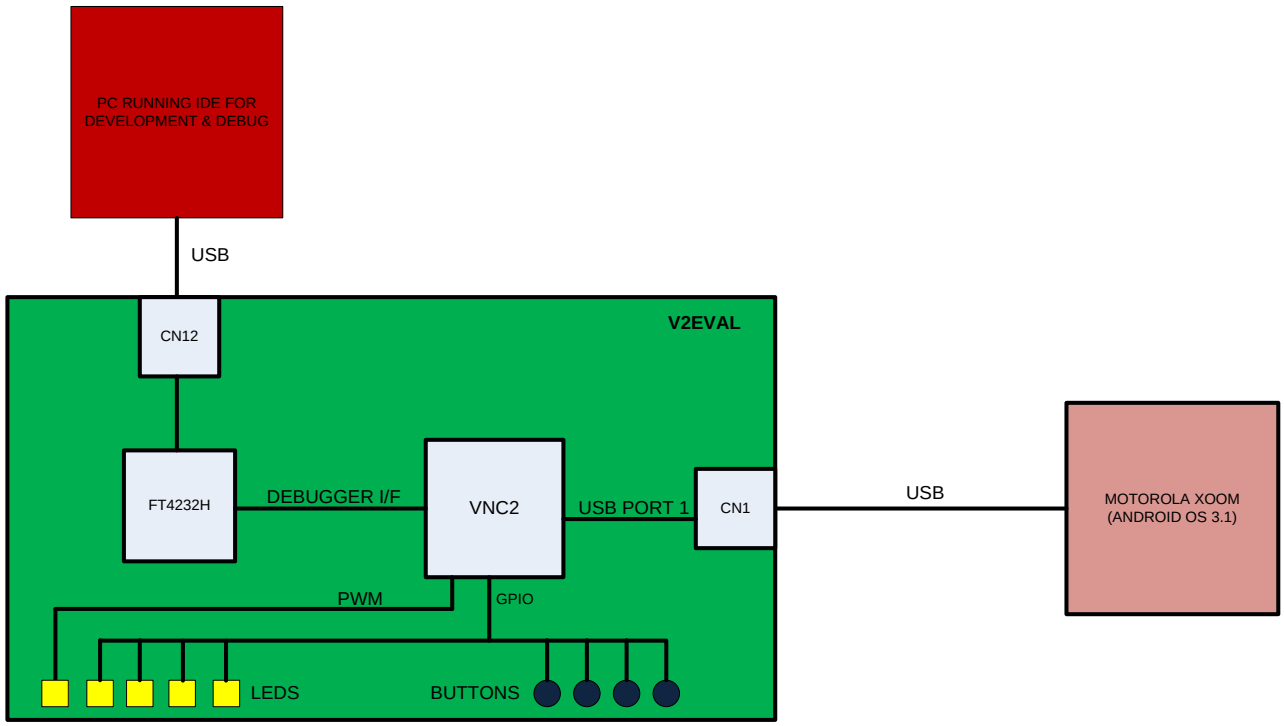

Figure 2.1 – VNC2 open Accessory Mode Demo Block Diagram

The ROM file (and source code) for this project can be viewed in appendix A and is available to download on an as is basis from:

http://www.ftdichip.com/Support/SoftwareExamples/Android/vinco_android_acc.zip

The .rom file can be found inside the "debug" directory of the .zip file

3 Demo

This section describe the demo setup.

3.1 Demo Setup

CN12 of the V2EVAL platform allows the IDE on the developement PC to communicate with the VNC2 debugger port. This is also used to load the Rom file created by the IDE into the VNC2 chip.

CN1 is the VNC2 USB port 1 which must be connected to the Android target (XOOM tablet).

3.2 Demo Description

This demo uses LED1:LED5 LEDs, SW1:SW4 push buttons on VNC2. 4 LEDs and 4 push buttons are created on Android Tablet to duplicate the setup.

LED1:LED4 and 4 LEDs on Android target are controlled by SW1:SW4 as well as 4 Buttons placed on Android Application. A button press, SW1:SW4 or on Android tablets, toggles the corresponding LED on VNC2 board as well as the Android Application.

LED5 is controlled by Volume control bar on the Android Tablet. LED5 demonstrates volume up/down characteristic with varying brightness.

LEDs, LED1:LED4 and push buttons SW1:SW4 are mapped to VNC2 GPIOs. PWM is implemented on LED5.

As all the GPIO wiring is part of the V2EVAL platform the user only needs to provide a USB cable to connect to the developemtn PC and the XOOM tablet.

4 Source code for the VNC2 Application

All VNC2 application firmware follows a similar format and most of the code can be "written" using the IDE application wizard.

The basic steps are:

Initialise device drivers

Define pinouts

Open ports to be used

Configure ports to be used

Read/write data

Close ports

The VNC2 source code for this project can be viewed in appendix A and is available to download on an as is basis from:

http://www.ftdichip.com/Support/SoftwareExamples/Android/vinco_android_acc.zip

Note: A Precompiled ROM file is also downloadable at the same address for users not wishing to rebuild the project. The .rom file can be found inside the "debug" directory of the .zip file.

4.1 Android_ACC.C

Android_ACC.c is the main firmware file. This file is split into multiple functions.

```
void  main();
void iomux_setup(void);
void open_drivers(void);
void close_drivers(void);
unsigned char usbhost_connect_state(VOS_HANDLE hUSB)
VOS_HANDLE android_attach(VOS_HANDLE hUSB, unsigned char devANDACC,
      char *manufacturer, char *model, char *description,
      char *version, char *uri, char *serial)
void android_detach(VOS_HANDLE hANDACC)
void usb_host_processing();
void firmware();
void pwm_processing();
```

4.1.1 main()

Main is where the application starts. It defines the VNC2 core clock speed, loads the drivers to be used and creates the threads to be used in the application. At the very end of main is the call

```
vos_start_scheduler();
```

After this call there can be no further configuration of the device.

4.1.2 Iomux_setup()

Iomux_setup actually refers to the other file in the project, andropid_acc_iomux.c and is used to define the VNC2 pinout. Most functions can be programmed to appear on different pins. The notable exceptions are power, GND and the USB ports.

4.1.3 Open drivers/close drivers

The open drivers function call will provide a handle to each hardware block used in the project and this handle can be used by subsequent commands to control the hardware. Close drivers closes the handle at the end of the project.

4.1.4 USB_Host_Connect_state

USB_Host_Connect_State is a function to check if anything is connected to the USB host.

4.1.5 Android_attach/Android_detach

As the VNC2 uses a layered architecture to control the drivers it is important to attach, detach the correct USB class driver to the USB host port. In this case the android class driver is beibg attached and detached.

4.1.6 Firmware

The firmware function is monitoring/controlling the VNC2 GPIO.

4.1.7 Usb_host_processing

USB_host_Processing is the section that interrogates the Android device as to whether it is capable of supporting open Accessory Mode and if it is, will enable it.

4.1.8 PWM_Processing

As an extra feature the 5th LED can be made to show variable brightness by using the PWM interface of the VNC2.

5 Building and Loading the Firmware into the VNC2

To build the application you simply press the Build button on the IDE ribbon bar under the build tab.

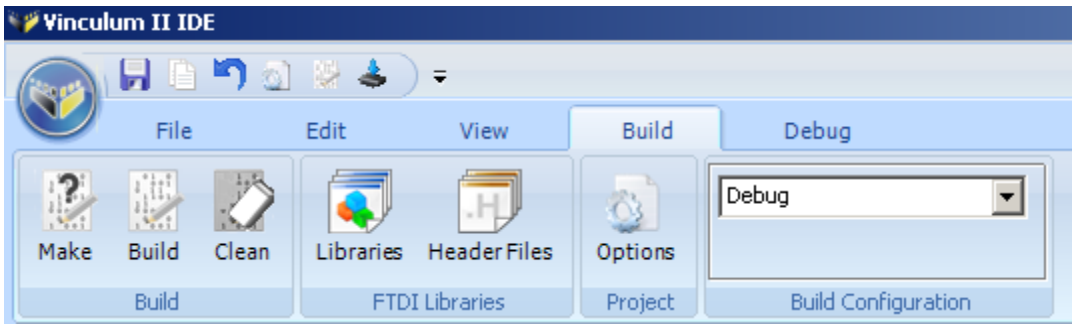

Figure 5.1 – Vinculum II IDE Build Button

Loading the code is equally simple. Just click on the "Flash" button on the ribbon bar under the debug tab.

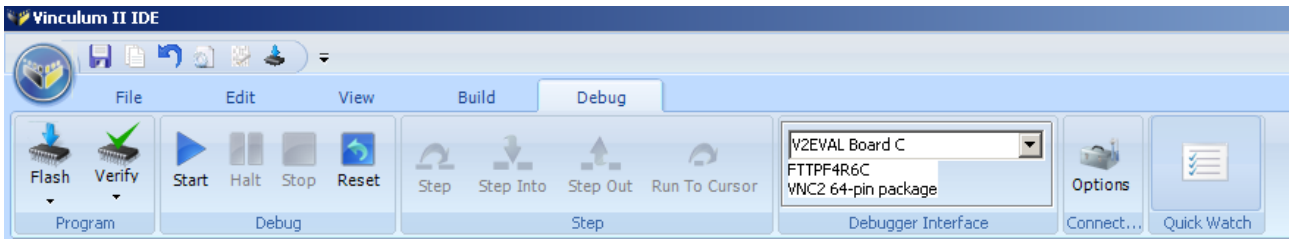

Figure 5.2 – Vinculum II IDE Flash Button

Note the Debugger Interface is listed as V2EVAL Board C. It is important that this box shows a device is connected before attempting to flash a device. Note your debugger may have a different label.

6 The Android Application

To complete the demonstration an application to run on the Android platform is also required. Source code for this can be viewed in Appendix B or downloaded on an as is basis from:

http://www.ftdichip.com/Support/SoftwareExamples/Android/android_acc_appl.zip

The source code is developed in JAVA as this is the standard for Android GUI applications.

The project ".\android_acc_appl\LED" may be built using Eclipse tools (http://www.eclipse.org/jdt/) a free compiler from the web. The resultant ledactivity.apk may then be loaded onto the Android platform (XOOM).

To load the application onto the Xoom tablet ensure the following settings are applied on the XOOM tablet.

Settings-> Applications -> unknown sources is selected.

Settings-> Applications ->Development -> USB Debug is selected to allow the application to be loaded over USB.

Further help on using Eclipse to build Android applications can be found at:

http://developer.android.com/guide/developing/projects/projects-eclipse.html

7 Running the Demo

Power up the V2EVAL board with the code loaded.

Power up the XOOM tablet.

Connect USB port 1 of the VNC2 (V2EVAL CN1) to the XOOM tablet.

The application on the tablet will start automatically.

Pressing SW1 on the V2EVAlL will illuminate/extinguish the left hand LED on the XOOM application.

SW2, 3, 4 correspond to the next LED along.

Pressing the left most button on the XOOM application will illuminate/extinguish LED1 on the V2EVAL PCB.

The other buttons correspond to LED2, 3, 4.

The volume control slider will control the brightness of LED 5 on the PCB.

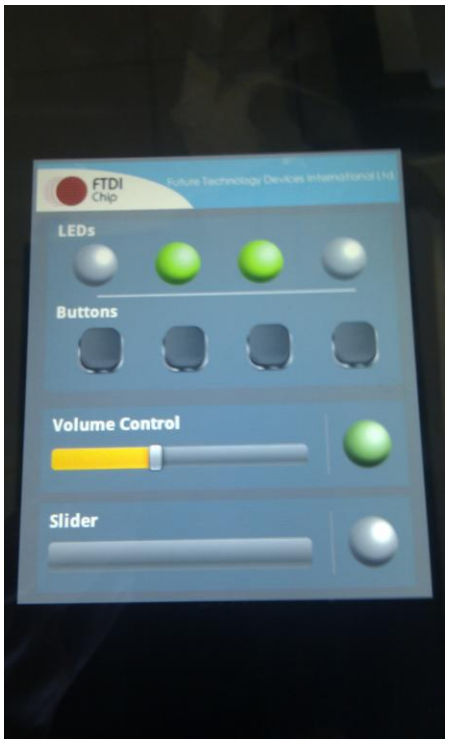

Figure 7.1 – Running the demo

Note the slider does not have any code associated with it at this time.

Figure 7.2 – Running the demo 2

8 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
Web Site URL http://www.ftdichip.com
Web Shop URL http://www.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
Web Site URL http://www.ftdichip.com

Branch Office – Hillsboro, Oregon, USA

Future Technology Devices International Limited (USA)
7235 NW Evergreen Parkway, Suite 600
Hillsboro, OR 97123-5803
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

Future Technology Devices International Limited (China)
Room 408, 317 Xianxia Road,
ChangNing District,
ShangHai, 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.admin1@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.

Appendix A – VNC2 Application Code

Source code for this project is provided on an "as is" basis and functionality is neither guaranteed or supported. The code has been verified as running on a Motorola Xoom platform, with Android OS 3.1.

Also available from http://www.ftdichip.com/Support/SoftwareExamples/Android/vinco_android_acc.zip

Android_Acc.c contents

```
/*
** Filename: android_acc.c
**
** Automatically created by Application Wizard 1.4.2
**
** Part of solution android_acc in project android_acc
**
** Comments:
**
** Important: Sections between markers "FTDI:S*" and "FTDI:E*" will be
overwritten by
** the Application Wizard
*/

#include "android_acc.h"

/* FTDI:STP Thread Prototypes */
vos_tcb_t *tcbusbFIRMWARE;
vos_tcb_t *tcbpwmFIRMWARE;
vos_tcb_t *tcbgpioFIRMWARE;

void firmware();
void open_drivers(void);
void usb_host_processing();
void pwm_processing();

/* FTDI:SDH Driver Handles */
VOS_HANDLE hUSBHOST_1; // USB Host Port 1
VOS_HANDLE hANDROID_ACCESSORY; // Android Open Accessory Class Driver
VOS_HANDLE hGPIO_PORT_A; // GPIO Port A Driver
VOS_HANDLE hGPIO_PORT_B; // GPIO Port B Driver
VOS_HANDLE hPWM; // PWM Driver
/* FTDI:EDH */

/*global variables*/
#define PWM_MAX_DUTY_CYCLE          45
#define PWM_MIN_DUTY_CYCLE          3
#define PWM_MAX_DUTY_COUNT          50

unsigned char u8PwmDutyCycle = 0;
unsigned char u8Changed = 0;
unsigned char u8AccesoryConnected = 0;

/*accessory packet*/
android_accessory_packet gstAccPacketWrite;
android_accessory_packet gstAccPacketRead;

/* Declaration for IOMUx setup function */
void iomux_setup(void);

/* Main code - entry point to firmware */
void main(void)
```

```
{
	/* FTDI:SDD Driver Declarations */
	// GPIO Port A configuration context
	gpio_context_t gpioContextA;
	// GPIO Port B configuration context
	gpio_context_t gpioContextB;
	// USB Host configuration context
	usbhost_context_t usbhostContext;
	/* FTDI:EDD */

	/* FTDI:SKI Kernel Initialisation */
	vos_init(50, VOS_TICK_INTERVAL, VOS_NUMBER_DEVICES);
	vos_set_clock_frequency(VOS_48MHZ_CLOCK_FREQUENCY);
	vos_set_idle_thread_tcb_size(512);
	/* FTDI:EKI */

	iomux_setup();

	/* FTDI:SDI Driver Initialisation */
	// Initialise GPIO A
	gpioContextA.port_identifier = GPIO_PORT_A;
	gpio_init(VOS_DEV_GPIO_PORT_A,&gpioContextA);

	// Initialise GPIO B
	gpioContextB.port_identifier = GPIO_PORT_B;
	gpio_init(VOS_DEV_GPIO_PORT_B,&gpioContextB);

	// Initialise PWM
	pwm_init(VOS_DEV_PWM);

	// Initialise USB Host
	usbhostContext.if_count = 8;
	usbhostContext.ep_count = 16;
	usbhostContext.xfer_count = 2;
	usbhostContext.iso_xfer_count = 2;
	usbhost_init(VOS_DEV_USBHOST_1, -1, &usbhostContext);
	/* FTDI:EDI */
	// init Android Accessory
	usbHostAndroidAccessory_init(VOS_DEV_ANDROID_ACCESSORY);

	/* FTDI:SCT Thread Creation */
	tcbusbFIRMWARE = vos_create_thread_ex(20, 2048, usb_host_processing,
	"usbApplication", 0);
	tcbgpioFIRMWARE = vos_create_thread_ex(20, 1024, firmware,
	"gpioApplication", 0);
	tcbpwmFIRMWARE = vos_create_thread_ex(20, 1024, pwm_processing,
	"pwmApplication", 0);
	/* FTDI:ECT */

	/*open the drivers*/
	 open_drivers();

	/*enable the PWM interrupts*/
	vos_enable_interrupts(VOS_PWM_TOP_INT_IEN);

	vos_start_scheduler();

main_loop:
	goto main_loop;

}

/* FTDI:SSP Support Functions */
```

```
unsigned char usbhost_connect_state(VOS_HANDLE hUSB)
{
	unsigned char connectstate = PORT_STATE_DISCONNECTED;
	usbhost_ioctl_cb_t hc_iocb;

	if (hUSB)
	{
		hc_iocb.ioctl_code = VOS_IOCTL_USBHOST_GET_CONNECT_STATE;
		hc_iocb.get        = &connectstate;
		vos_dev_ioctl(hUSB, &hc_iocb);
	}
	return connectstate;
}

VOS_HANDLE android_attach(VOS_HANDLE hUSB, unsigned char devANDACC,
	char *manufacturer, char *model, char *description,
	char *version, char *uri, char *serial)
{
	common_ioctl_cb_t androidAccessory_cb;
	usbHostAndroidAccessory_ioctl_cb_attach_t atInfo;
	VOS_HANDLE hANDACC;

	hANDACC = vos_dev_open(devANDACC);

	atInfo.hc_handle = hUSB;
	atInfo.manufacturer = manufacturer;
	atInfo.model = model;
	atInfo.description = description;
	atInfo.version = version;
	atInfo.uri = uri;
	atInfo.serial = serial;

	androidAccessory_cb.ioctl_code = VOS_IOCTL_USBHOSTANDROIDACCESSORY_ATTACH;
	androidAccessory_cb.set.data = &atInfo;
	if (vos_dev_ioctl(hANDACC, &androidAccessory_cb) !=
	USBHOSTANDROIDACCESSORY_OK)
	{
		vos_dev_close(hANDACC);
		hANDACC = NULL;
	}

	return hANDACC;
}

void android_detach(VOS_HANDLE hANDACC)
{
	common_ioctl_cb_t androidAccessory_cb;

	if (hANDACC)
	{
		androidAccessory_cb.ioctl_code =
VOS_IOCTL_USBHOSTANDROIDACCESSORY_DETACH;
		vos_dev_ioctl(hANDACC, &androidAccessory_cb);
		vos_dev_close(hANDACC);
	}
}

/* FTDI:ESP */

void open_drivers(void)
{
```

```
      /* Code for opening and closing drivers - move to required places in
Application Threads */
        /* FTDI:SDA Driver Open */
        hUSBHOST_1 = vos_dev_open(VOS_DEV_USBHOST_1);
        hGPIO_PORT_A = vos_dev_open(VOS_DEV_GPIO_PORT_A);
        hGPIO_PORT_B = vos_dev_open(VOS_DEV_GPIO_PORT_B);
        hPWM = vos_dev_open(VOS_DEV_PWM);
            //hANDROID_ACCESSORY = vos_dev_open(VOS_DEV_ANDROID_ACCESSORY);
}

void attach_drivers(void)
{
        /* FTDI:SUA Layered Driver Attach Function Calls */
        // Android Accessory Driver attach must specify various strings
        // Suggested values provided
        //char *manufacturer = "FTDI\0";
        //char *model = "VNC2\0";
        //char *description = "Vinculum Accessory Test\0";
        //char *version = "0.1.0\0";
        //char *uri = "http://www.ftdichip.com\0";
        //char *serial = "VinculumAccessory1\0";
        //hANDROID_ACCESSORY = android_attach(hUSBHOST_1,
VOS_DEV_ANDROID_ACCESSORY,
        // manufacturer, model, description, version, uri, serial);
        /* FTDI:EUA */
}

void close_drivers(void)
{
        /* FTDI:SDB Driver Close */
        vos_dev_close(hUSBHOST_1);
        vos_dev_close(hGPIO_PORT_A);
        vos_dev_close(hGPIO_PORT_B);
        vos_dev_close(hPWM);
            //vos_dev_close(hANDROID_ACCESSORY);
        /* FTDI:EDB */
}

/* Application Threads */

#define PUSH_BUTTON1_MAP      (0x02) //A1
#define PUSH_BUTTON2_MAP      (0x04) //A2
#define PUSH_BUTTON3_MAP      (0x08) //A3
#define PUSH_BUTTON4_MAP      (0x10)  //A4
#define PUSH_BUTTON5_MAP      (0x08)  //Not Connected

#define LED_BUTTON_MAP        (0x78)
#define PUSH_BUTTON_MAP       (0x1E)


/*FIXME, this fix si done to clear the issue of
                          wrong port read
*/
unsigned char u8PrevLedMap = 0x78;
/*to handle the long press*/
unsigned char u8PrevKeyMap = 0x1E;
/*temporary variable*/
unsigned char u8KeyMap = 0x1E;

/*since there is only one timer, I dont know how its going to
  workout
*/
void usb_host_processing()
```

```
{

	unsigned char i;
	unsigned char numRead = 0;
	usbhost_ioctl_cb_t                          usbhost_iocb;
	common_ioctl_cb_t                           androidAccessory_cb;
	usbHostAndroidAccessory_ioctl_cb_attach_t   atInfo;
	unsigned char                               status;

	// setup strings for the accessory
	// the manufacturer, model and version strings should match those in the
	// application's accessory_filter.xml file!
	// This will allow the application to auto-launch when the accessory is
	// connected
	char *manufacturer = "FTDI\0";
	char *model = "FTDIDemoKit\0";
	char *description = "Vinculum Accessory Test\0";
	char *version = "1.0\0";
	char *uri = "http://www.ftdichip.com\0";
	char *serial = "VinculumAccessory1\0";
	uint8 u8Data;
	unsigned char ledState = 0;
	//gpio_ioctl_cb_t gpio_iocb;
	unsigned char portData = 0xFF; // LEDs on when low, off when high
	unsigned short protocolRevision = 0;


	while (1)
	{
		if(usbhost_connect_state(hUSBHOST_1)== PORT_STATE_ENUMERATED)
		{

			hANDROID_ACCESSORY = android_attach(hUSBHOST_1,
VOS_DEV_ANDROID_ACCESSORY,

	manufacturer, model, description, version, uri, serial);

			if (hANDROID_ACCESSORY)
			{

				// successfully found and attached to an Android
Accessory device
				// Write data to the GPIO port - bit 3 to toggle
LED1 on V2EVAL board.
				vos_dev_write(hGPIO_PORT_B,&u8PrevLedMap, 1,
NULL);
				// get the protocol revision - should be 0x0100
for now
				androidAccessory_cb.ioctl_code =
VOS_IOCTL_USBHOSTANDROIDACCESSORY_GET_PROTOCOL_REVISION;
				androidAccessory_cb.get.data = &protocolRevision;
				status = vos_dev_ioctl(hANDROID_ACCESSORY,
&androidAccessory_cb);
				if (protocolRevision != 0x100)
				{
					vos_halt_cpu();
				}

				while (status == USBHOSTANDROIDACCESSORY_OK)
				{
					/*we are connected*/
					u8AccesoryConnected = 1;
					// read a message from the Android device
```

```
                                    // wrap this in the accessory driver read
function
                                    // NOTE: this call may return with less data
than was requested
                                    // In this case, we are requesting 64 bytes
to fill our buffer,
                                    // but our Android app will only send 1
byte!
                                    status =
vos_dev_read(hANDROID_ACCESSORY,(uint8 *)&gstAccPacketRead,
sizeof(gstAccPacketRead) , &numRead);

                                if(numRead == sizeof(gstAccPacketRead))
                                    {
                                    // process the message from the Android
device
                                    if(gstAccPacketRead.u8Type ==
DATA_TYPE_KEYPAD)
                                    {
                                          /*update the led bitmap*/
                                          u8Data = (gstAccPacketRead.u8Data <<
3);
                                          u8Data &= LED_BUTTON_MAP;
                                          u8PrevLedMap ^= u8Data;

      vos_dev_write(hGPIO_PORT_B,&u8PrevLedMap, 1, NULL);
                                    }
                                    else if(gstAccPacketRead.u8Type ==
DATA_TYPE_SLIDER)
                                    {
                                          u8Changed = 1;
                                          u8PwmDutyCycle =
gstAccPacketRead.u8Data;

                                    }
                                  }
                               }/*while(status == USBHOSTANDROIDACCESSORY_OK)*/
                               /*we are disconnected*/
                               u8AccesoryConnected = 0;
                               /*detach accessory*/
                               android_detach(hANDROID_ACCESSORY);

                               u8PrevLedMap = LED_BUTTON_MAP;
                               vos_dev_write(hGPIO_PORT_B, &u8PrevLedMap, 1,
NULL);

                               /*off the pwm led*/
                               u8Changed = 1;
                               u8PwmDutyCycle = 0x00;


                               /*wait unitl its uninstalled*/
                               while(usbhost_connect_state(hUSBHOST_1)==
PORT_STATE_ENUMERATED)
                               {
                                    vos_delay_msecs(10);
                               }
                          } /*if (hANDROID_ACCESSORY)*/
                    }/*if(usbhost_connect_state(VOS_HANDLE hUSB)==
                    PORT_STATE_ENUMERATED)*/
               } /*end of while 1*/
              /*wait for sometime*/
              vos_delay_msecs(10);
```

```
}



/* Application Threads */

void firmware()
{
      /* Thread code to be added here */
      unsigned char u8PortA = 0xF0;
      unsigned char u8PortB;
      unsigned char u8Temp;

      gpio_ioctl_cb_t gpio_ioca;

    // Set all pins to output using an ioctl.
      gpio_ioca.ioctl_code = VOS_IOCTL_GPIO_SET_MASK;
      gpio_ioca.value      = 0x78;
      // Send the ioctl to the device manager.
      vos_dev_ioctl(hGPIO_PORT_B, &gpio_ioca);
      /*clear the LEDs*/
    vos_dev_write(hGPIO_PORT_B,&u8PrevLedMap, 1, NULL);

      /*initialize the accessory, packet, for now
        it supports only leds*/
      gstAccPacketWrite.u8Type = DATA_TYPE_KEYPAD;
      while(1)
      {
            //svos_dev_write(hGPIO_PORT_A, &u8PortA, 1, NULL);
            /*check the key press*/
            vos_dev_read(hGPIO_PORT_A,&u8PortA, 1, NULL);
            /*take the one's cmplement*/
            u8PortA = ~u8PortA;
            /*the high ones are pressed*/
            u8PortA &= PUSH_BUTTON_MAP;

            /*check whether the key has been held*/
            if(u8PortA == u8PrevKeyMap)
                    continue;
            else
                  u8PrevKeyMap = u8PortA;

            /*if any button pressed??*/
            if(u8PortA)
            {
            /*look for new key*/
            //u8PrevKeyMap = u8PortA;
            //u8ValidKey = 1;



                  /*start the debounce of 200 msess, too much though*/
                  //wait_debounce(100);
                  vos_delay_msecs(100);
                  vos_dev_read(hGPIO_PORT_A, &u8Temp, 1, NULL);

                  u8Temp = ~u8Temp;
                  u8Temp &= PUSH_BUTTON_MAP;

                  u8PortA &= u8Temp;

                  /*check if any of the keys are presseed*/
```

```
                    if(u8PortA){
/*
                          u8Changed = 1;
                          if(u8PortA & PUSH_BUTTON1_MAP){
                                u8PwmDutyCycle++;
                           }else if(u8PortA & PUSH_BUTTON2_MAP){
                                      if(u8PwmDutyCycle !=0){
                                            u8PwmDutyCycle--;
                                      }
                                }else if(u8PortA & PUSH_BUTTON3_MAP){
                                      u8PwmDutyCycle = 50;
                          }
*/
                          /*led on positions*/
                          u8PortA <<= 2;
                          /*read the port A again to check which leds are high*/
                          //vos_dev_read(hGPIO_PORT_A, &u8Temp, 1, NULL);
                          //u8Temp1 &= 0xE0;
                          /*just take the map of LEDs only*/
                          /*FIXME, this fix si done to clear the issue of
                            wrong port read
                          */
                          u8PrevLedMap ^= u8PortA;
                          vos_dev_write(hGPIO_PORT_B,&u8PrevLedMap, 1, NULL);

                          /*prepare the packet*/
                          gstAccPacketWrite.u8Data = (u8PortA >> 3);

                          if(u8AccesoryConnected == 1){
                          /*send the usb data accross*/
                          vos_dev_write(hANDROID_ACCESSORY,(uint8
*)&gstAccPacketWrite, sizeof(gstAccPacketWrite), NULL);
                    }

                    }

             }


 } /*end of while */
}/*end of firmware*/


void pwm_processing()
{
      pwm_ioctl_cb_t pwm_iocb;
        uint16 u16OnTime = 500;

        // set counter prescaler value
      pwm_iocb.ioctl_code = VOS_IOCTL_PWM_SET_PRESCALER_VALUE;
        pwm_iocb.count.prescaler = 0x01;
      vos_dev_ioctl(hPWM, &pwm_iocb);

        /*FIXME, keep it like this, till I find a better way
          to enable/disable pwm
        */


      while(1) {

             /*max is 50*/
            if(u8PwmDutyCycle > PWM_MAX_DUTY_COUNT){
                        pwm_iocb.ioctl_code = VOS_IOCTL_PWM_SET_INITIAL_STATE;
```

```
                        pwm_iocb.output.init_state_mask = 0x00;
                        vos_dev_ioctl(hPWM, &pwm_iocb);
                        vos_delay_msecs(10);
                        continue;
                }

            /*take care of minimum duty cycle*/
            //if(u8PwmDutyCycle < PWM_MIN_DUTY_CYCLE){
            //          u8PwmDutyCycle = PWM_MIN_DUTY_CYCLE;

          //}

           /*calculate the ON time*/
          /*take care of the overflow*/
         if(u8PwmDutyCycle < PWM_MIN_DUTY_CYCLE){
                  pwm_iocb.ioctl_code = VOS_IOCTL_PWM_SET_INITIAL_STATE;
                  pwm_iocb.output.init_state_mask = 0xff;
                  vos_dev_ioctl(hPWM, &pwm_iocb);
                  vos_delay_msecs(10);
                  continue;
             }else{
                  u16OnTime = (1000*u8PwmDutyCycle)/PWM_MAX_DUTY_COUNT;
                  u16OnTime = (u16OnTime*48);
         }
         // *****************************************************************

          // Setting a count value of 0x00A0 with toggles at 0x0010 and 0x0060

          // will give a 50% duty cycle

          // *****************************************************************

          // set counter value - cycle complete when internal counter reaches
this value

          pwm_iocb.ioctl_code = VOS_IOCTL_PWM_SET_COUNTER_VALUE;
          pwm_iocb.count.value = 0xbb80;/*for 1 khz*/
          vos_dev_ioctl(hPWM, &pwm_iocb);



          // set comparator 0 value - toggle output at a value of 0x0010

          pwm_iocb.ioctl_code = VOS_IOCTL_PWM_SET_COMPARATOR_VALUE;
          pwm_iocb.identifier.comparator_number = COMPARATOR_0;
          pwm_iocb.comparator.value = u16OnTime;//0x0960; /*50 % duty cycle*/
          vos_dev_ioctl(hPWM, &pwm_iocb);



          // set comparator 1 value - toggle output at a value of 0x0060

          pwm_iocb.ioctl_code = VOS_IOCTL_PWM_SET_COMPARATOR_VALUE;
          pwm_iocb.identifier.comparator_number = COMPARATOR_1;
          pwm_iocb.comparator.value = 0xbb80;//
          vos_dev_ioctl(hPWM, &pwm_iocb);



          // enable comparators 0 and 1 for PWM 3

          // this will cause PWM output 1 to toggle on comparators 0 and 1
```

```
pwm_iocb.ioctl_code = VOS_IOCTL_PWM_SET_OUTPUT_TOGGLE_ENABLES;
pwm_iocb.identifier.pwm_number = PWM_3;
pwm_iocb.output.enable_mask = (MASK_COMPARATOR_0 | MASK_COMPARATOR_1);
vos_dev_ioctl(hPWM, &pwm_iocb);



// set initial state - all PWM outputs will be low (0) initially

pwm_iocb.ioctl_code = VOS_IOCTL_PWM_SET_INITIAL_STATE;
    if(u8PwmDutyCycle < PWM_MIN_DUTY_CYCLE){
   pwm_iocb.output.init_state_mask = 0xff;
   }else{
         pwm_iocb.output.init_state_mask = 0x00;
   }
vos_dev_ioctl(hPWM, &pwm_iocb);



// set restore state - PWM output 3 will return to low state (0)

// at end of cycle
pwm_iocb.ioctl_code = VOS_IOCTL_PWM_RESTORE_INITIAL_STATE;
pwm_iocb.output.restore_state_mask = (MASK_PWM_0);
vos_dev_ioctl(hPWM, &pwm_iocb);


// set mode to 25 cycles

pwm_iocb.ioctl_code = VOS_IOCTL_PWM_SET_NUMBER_OF_CYCLES;
pwm_iocb.output.mode = 0x50;
vos_dev_ioctl(hPWM, &pwm_iocb);
while(1)
         {

        // enable interrupt - this will fire when the specified number
of

        // cycles is complete

        pwm_iocb.ioctl_code = VOS_IOCTL_PWM_ENABLE_INTERRUPT;
        vos_dev_ioctl(hPWM, &pwm_iocb);


                // enable output
 //    if(u8PwmDutyCycle > PWM_MIN_DUTY_CYCLE){
        pwm_iocb.ioctl_code = VOS_IOCTL_PWM_ENABLE_OUTPUT;
        vos_dev_ioctl(hPWM, &pwm_iocb);
  // }

        // wait on interrupt
        pwm_iocb.ioctl_code = VOS_IOCTL_PWM_WAIT_ON_COMPLETE;
        vos_dev_ioctl(hPWM, &pwm_iocb);

         // disable output
        pwm_iocb.ioctl_code = VOS_IOCTL_PWM_DISABLE_OUTPUT;
         vos_dev_ioctl(hPWM, &pwm_iocb);

       if(u8Changed == 1){
                  u8Changed = 0;
       //         /*goto the external loop*/
                  break;
        }
```

```
                        }
                }
}
```

Android_Acc_IOMUX.c Contents

```
/*
** Filename: android_acc_iomux.c
**
** Automatically created by Application Wizard 1.4.2
**
** Part of solution android_acc in project android_acc
**
** Comments:
**
** Important: Sections between markers "FTDI:S*" and "FTDI:E*" will be
overwritten by
** the Application Wizard
*/
#include "vos.h"

void iomux_setup(void)
{
      /* FTDI:SIO IOMux Functions */
      unsigned char packageType;

      packageType = vos_get_package_type();
      if (packageType == VINCULUM_II_64_PIN)
      {
      // Debugger to pin 11 as Bi-Directional.
      vos_iomux_define_bidi(199, IOMUX_IN_DEBUGGER, IOMUX_OUT_DEBUGGER);

      /*input switched*/
      // GPIO_Port_A_1 to pin 12 as Input.
      vos_iomux_define_input(12, IOMUX_IN_GPIO_PORT_A_1);

      vos_iocell_set_config(12,VOS_IOCELL_DRIVE_CURRENT_4MA,VOS_IOCELL_
      TRIGGER_NORMAL,VOS_IOCELL_SLEW_RATE_SLOW, VOS_IOCELL_PULL_UP_75K);

      // GPIO_Port_A_2 to pin 13 as Input.
      vos_iomux_define_input(13, IOMUX_IN_GPIO_PORT_A_2);

      vos_iocell_set_config(13,VOS_IOCELL_DRIVE_CURRENT_4MA,VOS_IOCELL_TRIGGER_
      NORMAL,VOS_IOCELL_SLEW_RATE_SLOW, VOS_IOCELL_PULL_UP_75K);

      // GPIO_Port_A_3 to pin 14 as Input.
      vos_iomux_define_input(14, IOMUX_IN_GPIO_PORT_A_3);

      vos_iocell_set_config(14,VOS_IOCELL_DRIVE_CURRENT_4MA,VOS_IOCELL_TRIGGER_
      NORMAL,VOS_IOCELL_SLEW_RATE_SLOW, VOS_IOCELL_PULL_UP_75K);

      // GPIO_Port_B_3 to pin 32 as Input.
      //vos_iomux_define_input(32, IOMUX_IN_GPIO_PORT_B_3);
      //vos_iocell_set_config(32,VOS_IOCELL_DRIVE_CURRENT_4MA,VOS_IOCELL_TRIGGER
      _NORMAL,
      //VOS_IOCELL_SLEW_RATE_SLOW, VOS_IOCELL_PULL_UP_75K);

      // GPIO_Port_B_4 to pin 51 as Input.
      vos_iomux_define_input(51, IOMUX_IN_GPIO_PORT_A_4);

      vos_iocell_set_config(51,VOS_IOCELL_DRIVE_CURRENT_4MA,VOS_IOCELL_TRIGGER_
```

```
NORMAL,VOS_IOCELL_SLEW_RATE_SLOW, VOS_IOCELL_PULL_UP_75K);

// FIFO_Data_0 to pin 15 as Bi-Directional.
vos_iomux_define_bidi(15, IOMUX_IN_FIFO_DATA_0, IOMUX_OUT_FIFO_DATA_0);
// FIFO_Data_1 to pin 16 as Bi-Directional.
vos_iomux_define_bidi(16, IOMUX_IN_FIFO_DATA_1, IOMUX_OUT_FIFO_DATA_1);
// FIFO_Data_2 to pin 17 as Bi-Directional.
vos_iomux_define_bidi(17, IOMUX_IN_FIFO_DATA_2, IOMUX_OUT_FIFO_DATA_2);
// FIFO_Data_3 to pin 18 as Bi-Directional.
vos_iomux_define_bidi(18, IOMUX_IN_FIFO_DATA_3, IOMUX_OUT_FIFO_DATA_3);
// FIFO_Data_4 to pin 19 as Bi-Directional.
vos_iomux_define_bidi(19, IOMUX_IN_FIFO_DATA_4, IOMUX_OUT_FIFO_DATA_4);
// FIFO_Data_5 to pin 20 as Bi-Directional.
vos_iomux_define_bidi(20, IOMUX_IN_FIFO_DATA_5, IOMUX_OUT_FIFO_DATA_5);
// FIFO_Data_6 to pin 22 as Bi-Directional.
vos_iomux_define_bidi(22, IOMUX_IN_FIFO_DATA_6, IOMUX_OUT_FIFO_DATA_6);
// FIFO_Data_7 to pin 23 as Bi-Directional.
vos_iomux_define_bidi(23, IOMUX_IN_FIFO_DATA_7, IOMUX_OUT_FIFO_DATA_7);
// GPIO_Port_B_0 to pin 24 as Output.
vos_iomux_define_output(24, IOMUX_OUT_GPIO_PORT_B_0);
// FIFO_TXE_N to pin 25 as Output.
vos_iomux_define_output(25, IOMUX_OUT_FIFO_TXE_N);
// FIFO_RD_N to pin 26 as Input.
vos_iomux_define_input(26, IOMUX_IN_FIFO_RD_N);
// FIFO_WR_N to pin 27 as Input.
vos_iomux_define_input(27, IOMUX_IN_FIFO_WR_N);
// FIFO_OE_N to pin 28 as Input.
vos_iomux_define_input(28, IOMUX_IN_FIFO_OE_N);
// UART_DSR_N to pin 29 as Input.
vos_iomux_define_input(29, IOMUX_IN_UART_DSR_N);
// UART_DCD to pin 31 as Input.
vos_iomux_define_input(31, IOMUX_IN_UART_DCD);
// UART_TXD to pin 39 as Output.
vos_iomux_define_output(39, IOMUX_OUT_UART_TXD);
// UART_RXD to pin 40 as Input.
vos_iomux_define_input(40, IOMUX_IN_UART_RXD);
// UART_RTS_N to pin 41 as Output.
vos_iomux_define_output(41, IOMUX_OUT_UART_RTS_N);
// UART_CTS_N to pin 42 as Input.
vos_iomux_define_input(42, IOMUX_IN_UART_CTS_N);
// UART_DTR_N to pin 43 as Output.
vos_iomux_define_output(43, IOMUX_OUT_UART_DTR_N);
// UART_DSR_N to pin 44 as Input.
vos_iomux_define_input(44, IOMUX_IN_UART_DSR_N);
// UART_DCD to pin 45 as Input.
vos_iomux_define_input(45, IOMUX_IN_UART_DCD);
// UART_RI to pin 46 as Input.
vos_iomux_define_input(46, IOMUX_IN_UART_RI);
// UART_TX_Active to pin 47 as Output.
vos_iomux_define_output(47, IOMUX_OUT_UART_TX_ACTIVE);

// SPI_Slave_0_MOSI to pin 52 as Input.
vos_iomux_define_input(52, IOMUX_IN_SPI_SLAVE_0_MOSI);
// SPI_Slave_0_MISO to pin 55 as Output.
vos_iomux_define_output(55, IOMUX_OUT_SPI_SLAVE_0_MISO);

// SPI_Slave_1_CLK to pin 57 as Input.
vos_iomux_define_input(57, IOMUX_IN_SPI_SLAVE_1_CLK);
// SPI_Slave_1_MOSI to pin 58 as Input.
vos_iomux_define_input(58, IOMUX_IN_SPI_SLAVE_1_MOSI);
// SPI_Slave_1_MISO to pin 59 as Output.
vos_iomux_define_output(59, IOMUX_OUT_SPI_SLAVE_1_MISO);
// SPI_Slave_1_CS to pin 60 as Input.
```

```
        vos_iomux_define_input(60, IOMUX_IN_SPI_SLAVE_1_CS);

        // PWM_3 to pin 56 as Output.
        vos_iomux_define_output(64, IOMUX_OUT_PWM_3);

        // PWM_0 to pin 61 as Output.
        vos_iomux_define_output(56, IOMUX_OUT_GPIO_PORT_B_3);

        vos_iocell_set_config(56,VOS_IOCELL_DRIVE_CURRENT_4MA,VOS_IOCELL_TRIGGER_
        NORMAL,VOS_IOCELL_SLEW_RATE_FAST, VOS_IOCELL_PULL_NONE);

        // GPIO_Port_A_5 to pin 62 as Output.
        vos_iomux_define_output(61, IOMUX_OUT_GPIO_PORT_B_4);

        vos_iocell_set_config(61,VOS_IOCELL_DRIVE_CURRENT_4MA,VOS_IOCELL_TRIGGER_
        NORMAL,VOS_IOCELL_SLEW_RATE_FAST, VOS_IOCELL_PULL_NONE);

        // GPIO_Port_A_6 to pin 63 as Output.
        vos_iomux_define_output(62, IOMUX_OUT_GPIO_PORT_B_5);

        vos_iocell_set_config(62,VOS_IOCELL_DRIVE_CURRENT_4MA,VOS_IOCELL_TRIGGER_
        NORMAL,VOS_IOCELL_SLEW_RATE_FAST, VOS_IOCELL_PULL_NONE);

        // GPIO_Port_A_7 to pin 64 as Output.
              vos_iomux_define_output(63, IOMUX_OUT_GPIO_PORT_B_6);

        vos_iocell_set_config(63,VOS_IOCELL_DRIVE_CURRENT_4MA,VOS_IOCELL_TRIGGER_
        NORMAL,VOS_IOCELL_SLEW_RATE_FAST, VOS_IOCELL_PULL_NONE);

        }

        /* FTDI:EIO */
}
```

Appendix B

Android JAVA Application

Also available from http://www.ftdichip.com/Support/SoftwareExamples/Android/android_acc_appl.zip

```
package FTDI.LED;

import java.io.FileDescriptor;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;

import FTDI.LED.R.drawable;
import android.app.Activity;
import android.app.AlertDialog;
import android.app.PendingIntent;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.IntentFilter;
import android.hardware.usb.UsbAccessory;
import android.hardware.usb.UsbManager;
import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
import android.os.ParcelFileDescriptor;
import android.util.Log;
import android.view.View;
import android.widget.ImageButton;
import android.widget.ImageView;
import android.widget.ProgressBar;
import android.widget.SeekBar;

public class LEDActivity extends Activity{

      private static final String ACTION_USB_PERMISSION =
"FTDI.LED.USB_PERMISSION";
      public UsbManager usbmanager;
      public UsbAccessory usbaccessory;
      public PendingIntent mPermissionIntent;
      public ParcelFileDescriptor filedescriptor;
      public FileInputStream inputstream;
      public FileOutputStream outputstream;
      public boolean mPermissionRequestPending = true;

      //public Handler usbhandler;
      public byte[] usbdata;
      public byte[] writeusbdata;
      public byte  ledPrevMap = 0x00;
      //public byte[] usbdataIN;

      public SeekBar volumecontrol;
    public ProgressBar slider;

    public ImageButton button1; //Button led1;
    public ImageButton button2; //Button led2;
    public ImageButton button3; //Button led3;
    public ImageButton button4; //Button led4;
```

```
    public ImageView led1;
    public ImageView led2;
    public ImageView led3;
    public ImageView led4;
    public ImageView ledvolume;

    public int readcount;
    /*thread to listen USB data*/
    public handler_thread handlerThread;


      /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
      super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        usbdata = new byte[4];
        writeusbdata = new byte[4];

        usbmanager = (UsbManager) getSystemService(Context.USB_SERVICE);
        Log.d("LED", "usbmanager" +usbmanager);
        mPermissionIntent = PendingIntent.getBroadcast(this, 0, new
Intent(ACTION_USB_PERMISSION), 0);
        IntentFilter filter = new IntentFilter(ACTION_USB_PERMISSION);
        filter.addAction(UsbManager.ACTION_USB_ACCESSORY_DETACHED);
        Log.d("LED", "filter" +filter);
        registerReceiver(mUsbReceiver, filter);

        led1 = (ImageView) findViewById(R.id.LED1);
            led2 = (ImageView) findViewById(R.id.LED2);
            led3 = (ImageView) findViewById(R.id.LED3);
            led4 = (ImageView) findViewById(R.id.LED4);

        button1 = (ImageButton) findViewById(R.id.Button1);
        button1.setOnClickListener(new View.OnClickListener()
        {
            public void onClick(View v)
            {
                  byte ibutton = 0x01;
                  Log.d("LED", "Button 1 pressed");

                  ledPrevMap ^= 0x01;

                  if((ledPrevMap & 0x01) == 0x01){
                              led1.setImageResource(drawable.image100);
                        }
                        else{
                              led1.setImageResource(drawable.image0);
                  }



                  //v.bringToFront();
                  WriteUsbData(ibutton);
            }
            });

        button2 = (ImageButton) findViewById(R.id.Button2);
        button2.setOnClickListener(new View.OnClickListener()
        {
```

```
            public void onClick(View v)
            {
            byte ibutton = 0x02;
            //v.bringToFront();

            ledPrevMap ^= 0x02;

            if((ledPrevMap & 0x02) == 0x02){
                        led2.setImageResource(drawable.image100);
                  }
                  else{
                        led2.setImageResource(drawable.image0);
            }

            WriteUsbData(ibutton);
            }
      });

button3 = (ImageButton) findViewById(R.id.Button3);
button3.setOnClickListener(new View.OnClickListener()
{
            public void onClick(View v)
            {
            byte ibutton = 0x04;
            //v.bringToFront();

            ledPrevMap ^= 0x04;

            if((ledPrevMap & 0x04) == 0x04){
                        led3.setImageResource(drawable.image100);
                  }
                  else{
                        led3.setImageResource(drawable.image0);
            }
            WriteUsbData(ibutton);
            }
      });

button4 = (ImageButton) findViewById(R.id.Button4);
button4.setOnClickListener(new View.OnClickListener()
{
            public void onClick(View v)
            {
            byte ibutton = 0x08;
            //v.bringToFront();
            ledPrevMap ^= 0x08;

            if((ledPrevMap & 0x08) == 0x08){
                        led4.setImageResource(drawable.image100);
                  }
                  else{
                        led4.setImageResource(drawable.image0);
            }
            WriteUsbData(ibutton);
            }
      });

volumecontrol = (SeekBar)findViewById(R.id.seekBar1);

//set the max value to 50
volumecontrol.setMax(50);
```

```
        volumecontrol.setOnSeekBarChangeListener(new
SeekBar.OnSeekBarChangeListener()
        {
                public void onStopTrackingTouch(SeekBar seekBar)
                {
                      // TODO Auto-generated method stub

                }

                public void onStartTrackingTouch(SeekBar seekBar)
                {
                      // TODO Auto-generated method stub

                }

                public void onProgressChanged(SeekBar seekBar, int progress,
boolean fromUser)
                {
                      writeusbdata[0] = 1;
                      writeusbdata[1] = 1;
                      writeusbdata[2] = 2;
                      writeusbdata[3] = (byte) progress;
                      try
                      {
                            if(outputstream != null)
                            {
                                  outputstream.write(writeusbdata,0,4);
                            }
                      }
                      catch (IOException e)
                      {
                      }

                      ledvolume = (ImageView) findViewById(R.id.LEDvolume);
                      if(progress == 0)
                      {
                            ledvolume.setImageResource(drawable.image0);
                      }
                      else if(progress > 0 && (int)progress < 11)
                      {
                            ledvolume.setImageResource(drawable.image10);
                      }
                      else if (progress > 10 && progress < 21)
                      {
                            ledvolume.setImageResource(drawable.image20);
                      }
                      else if (progress > 20 && progress < 36)
                      {
                            ledvolume.setImageResource(drawable.image35);
                      }
                      else if (progress > 35 && progress < 51)
                      {
                            ledvolume.setImageResource(drawable.image50);
                      }
                      else if (progress > 50 && progress < 66)
                      {
                            ledvolume.setImageResource(drawable.image65);
                      }
                      else if (progress > 65 && progress < 76)
                      {
                            ledvolume.setImageResource(drawable.image75);
                      }
                      else if (progress > 75 && progress < 91)
```

```
                        {
                                ledvolume.setImageResource(drawable.image90);
                        }
                        else
                        {
                                ledvolume.setImageResource(drawable.image100);
                        }
                }
        });
    }


    @Override
    public void onResume()
    {
            super.onResume();
            Intent intent = getIntent();
            if (inputstream != null && outputstream != null) {
                  return;
            }

            UsbAccessory[] accessories = usbmanager.getAccessoryList();
            UsbAccessory accessory = (accessories == null ? null :
accessories[0]);
            if (accessory != null) {
                  if (usbmanager.hasPermission(accessory)) {
                        OpenAccessory(accessory);
                  }
                  else
                  {
                        synchronized (mUsbReceiver) {
                              if (!mPermissionRequestPending) {
                                    usbmanager.requestPermission(accessory,
                                                mPermissionIntent);
                                    mPermissionRequestPending = true;
                              }
                  }
            }
            } else {}
    }

      @Override
      public void onDestroy()
      {
            unregisterReceiver(mUsbReceiver);
            //CloseAccessory();
            super.onDestroy();
      }


      /*open the accessory*/
      private void OpenAccessory(UsbAccessory accessory)
      {
            filedescriptor = usbmanager.openAccessory(accessory);
            if(filedescriptor != null){
                  usbaccessory = accessory;
                  FileDescriptor fd = filedescriptor.getFileDescriptor();
                  inputstream = new FileInputStream(fd);
                  outputstream = new FileOutputStream(fd);
                  /*check if any of them are null*/
                  if(inputstream == null || outputstream==null){
                        return;
                  }
```

```
        }

        handlerThread = new handler_thread(handler, inputstream);
        handlerThread.start();

    } /*end OpenAccessory*/

    public void ReadUsbData()
    {
        if(usbdata[0] == 0)
        {
            /*
            led1 = (ImageView) findViewById(R.id.LED1);
            led2 = (ImageView) findViewById(R.id.LED2);
            led3 = (ImageView) findViewById(R.id.LED3);
            led4 = (ImageView) findViewById(R.id.LED4);
            */
            ledPrevMap ^= usbdata[3];
            usbdata[3] = ledPrevMap;

            if((usbdata[3]& 0x01) == 0x01)
            {
                led1.setImageResource(drawable.image100);
            }
            else{
                led1.setImageResource(drawable.image0);
            }

            if((usbdata[3]& 0x02) == 0x02){
                led2.setImageResource(drawable.image100);
            }else{
                led2.setImageResource(drawable.image0);
            }

            if((usbdata[3]& 0x04) == 0x04){
                led3.setImageResource(drawable.image100);
            }else{
                led3.setImageResource(drawable.image0);
            }

            if((usbdata[3]& 0x08) == 0x08){
                led4.setImageResource(drawable.image100);
            }else{
                led4.setImageResource(drawable.image0);
            }
        }
        else if (usbdata[0] == 1)
        {
            ledvolume = (ImageView) findViewById(R.id.LEDvolume);
            if((int)usbdata[3] == 0)
            {
                ledvolume.setImageResource(drawable.image0);
            }
            else if((int)usbdata[3] > 0 && (int)usbdata[3] < 11)
            {
                ledvolume.setImageResource(drawable.image10);
            }
            else if ((int)usbdata[3] > 10 && (int)usbdata[3] < 21)
            {
                ledvolume.setImageResource(drawable.image20);
            }
            else if ((int)usbdata[3] > 20 && (int)usbdata[3] < 36)
            {
```

```
                        ledvolume.setImageResource(drawable.image35);
                }
                else if ((int)usbdata[3] > 35 && (int)usbdata[3] < 51)
                {
                        ledvolume.setImageResource(drawable.image50);
                }
                else if ((int)usbdata[3] > 50 && (int)usbdata[3] < 66)
                {
                        ledvolume.setImageResource(drawable.image65);
                }
                else if ((int)usbdata[3] > 65 && (int)usbdata[3] < 76)
                {
                        ledvolume.setImageResource(drawable.image75);
                }
                else if ((int)usbdata[3] > 75 && (int)usbdata[3] < 91)
                {
                        ledvolume.setImageResource(drawable.image90);
                }
                else
                {
                        ledvolume.setImageResource(drawable.image100);
                }
        }
    }

    private void CloseAccessory()
    {
        try{
                filedescriptor.close();
        }catch (IOException e){}

        try {
                inputstream.close();
        } catch(IOException e){}

        try {
                outputstream.close();

        }catch(IOException e){}
        /*FIXME, add the notfication also to close the application*/
        //unregisterReceiver(mUsbReceiver);
        //CloseAccessory();
        //super.onDestroy();
        filedescriptor = null;
        inputstream = null;
        outputstream = null;

        System.exit(0);

    }


    final Handler handler =  new Handler()
{
    @Override
    public void handleMessage(Message msg)
    {
        ReadUsbData();
    }

};
```

```
	private class handler_thread  extends Thread {
		Handler mHandler;
		FileInputStream instream;

		handler_thread(Handler h,FileInputStream stream ){
			mHandler = h;
			instream = stream;
		}

		public void run()
		{

			while(true)
			{
				Message msg = mHandler.obtainMessage();
				try{
					if(instream != null)
					{
					readcount = instream.read(usbdata,0,4);
					if(readcount > 0)
					{
						msg.arg1 = usbdata[0];
						msg.arg2 = usbdata[3];
					}
					mHandler.sendMessage(msg);
					}
					}catch (IOException e){}
			}
		}
	}

	public void WriteUsbData(byte iButton){
		writeusbdata[0] = 0;
		writeusbdata[1] = 1;
		writeusbdata[2] = 2;
		writeusbdata[3] = iButton;

		Log.d("LED", "pressed " +iButton);

		try{
			if(outputstream != null){
				outputstream.write(writeusbdata,0,4);
			}
		}
		catch (IOException e) {}
	}

    private final BroadcastReceiver mUsbReceiver = new BroadcastReceiver()
	{
		@Override
		public void onReceive(Context context, Intent intent)
		{
			String action = intent.getAction();
			if (ACTION_USB_PERMISSION.equals(action))
			{
				synchronized (this)
				{
					UsbAccessory accessory = (UsbAccessory)
intent.getParcelableExtra(UsbManager.EXTRA_ACCESSORY);
					if
(intent.getBooleanExtra(UsbManager.EXTRA_PERMISSION_GRANTED, false))
					{
						OpenAccessory(accessory);
```

```
                                    }
                                    else
                                    {
                                          Log.d("LED", "permission denied for
accessory "+ accessory);

                                    }
                                    mPermissionRequestPending = false;
                              }
                        }
                        else if
(UsbManager.ACTION_USB_ACCESSORY_DETACHED.equals(action))
                        {
                              UsbAccessory accessory =
(UsbAccessory)intent.getParcelableExtra(UsbManager.EXTRA_ACCESSORY);
                              if (accessory != null )//&&
accessory.equals(usbaccessory))
                              {
                                    CloseAccessory();
                              }
                        }else
                        {
                              Log.d("LED", "....");
                        }
                  }
            };
};
```

Appendix C – References

Application and Technical Notes available at
http://www.ftdichip.com/Support/Documents/AppNotes.htm

VNC2 Datasheet

http://www.ftdichip.com/Support/Documents/DataSheets/ICs/DS_Vinculum-II.pdf

V2-EVAL datasheet

http://www.ftdichip.com/Support/Documents/DataSheets/Modules/DS_V2EVAL_Rev2.pdf

Vinculum II Toolchain

http://www.ftdichip.com/Firmware/vnc2toolchain/Vinculum%20II%20Installer%20V1.4.2.exe

AN_139 IO_Mux explained

http://www.ftdichip.com/Support/Documents/AppNotes/AN_139_Vinculum-II%20IO_Mux%20Explained.pdf

AN_140 PWM Example

http://www.ftdichip.com/Support/Documents/AppNotes/AN_140_Vinculum-II_PWM_Example.pdf

AN_151 Vinculum II User Guide

http://www.ftdichip.com/Support/Documents/AppNotes/AN_151%20Vinculum%20II%20User%20Guide.pdf

TN_133_ Vimculum II toolchain Release Notes

http://www.ftdichip.com/Support/Documents/TechnicalNotes/TN_133_Vinculum_II_Toolchain_Release_Notes.pdf

Eclipse JAVA Development tools

http://www.eclipse.org/jdt/

Managing Android Apps from Eclipse

http://developer.android.com/guide/developing/projects/projects-eclipse.html

XOOM Tablets

http://www.motorola.com/Consumers/GB-EN/Consumer-Products-and-Services/ANDROID-TABLETS/MOTOROLA-XOOM-with-Wi-Fi-GB-EN?WT.srch=1&WT.mc_id=EMEA_GB-EN_XOOM_Aug-2011&WT.mc_ev=click

Appendix D – List of Figures and Tables

List of Figures

Appendix E – Revision History

Version 1.0	First release	22nd Aug 2011