Future Technology Devices International Ltd.

Wireless Data Transfers with VNC2 Application Note AN_179

Document Reference No.: FT_000496

Version 1.0

Issue Date: 2011-08-17

This application note demonstrates how the FTDI VNC2 USB Host controller can connect to another device wirelessly with a USB Bluetooth dongle.

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

This application note will demonstrate how the VNC2 device can be used to host a Roving Networks USB Bluetooth dongle and configure it. Data will be passed to and from the dongle via the VNC2 to a PC connected to the VNC2 UART port. A second Roving Networks Bluetooth Dongle will be connected to another PC to allow for data transfer over the Bluetooth link. 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

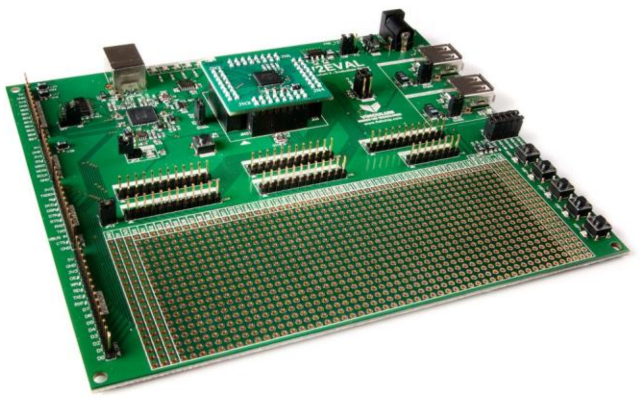


Figure 1.1 – V2-EVAL with daughter card

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 Roving Network USB Bluetooth Dongles

This application example uses two of the Roving Network USB Bluetooth dongles (RN-USB -X). These dongles were chosen as they use the FT232 device as the USB interface and the Vinculum IDE provides drivers for this chipset for free. Configuration of the dongle is done via this USB interface and the Bluetooth interface is handled entirely in the dongles.

See: http://www.rovingnetworks.com/Docs/Bluetooth-RN-USB-X-UM.pdf

For more information on the Bluetooth dongle.

Figure 1.2 – Roving network Bluetooth Dongle

Table of Contents

2 Block Diagram

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

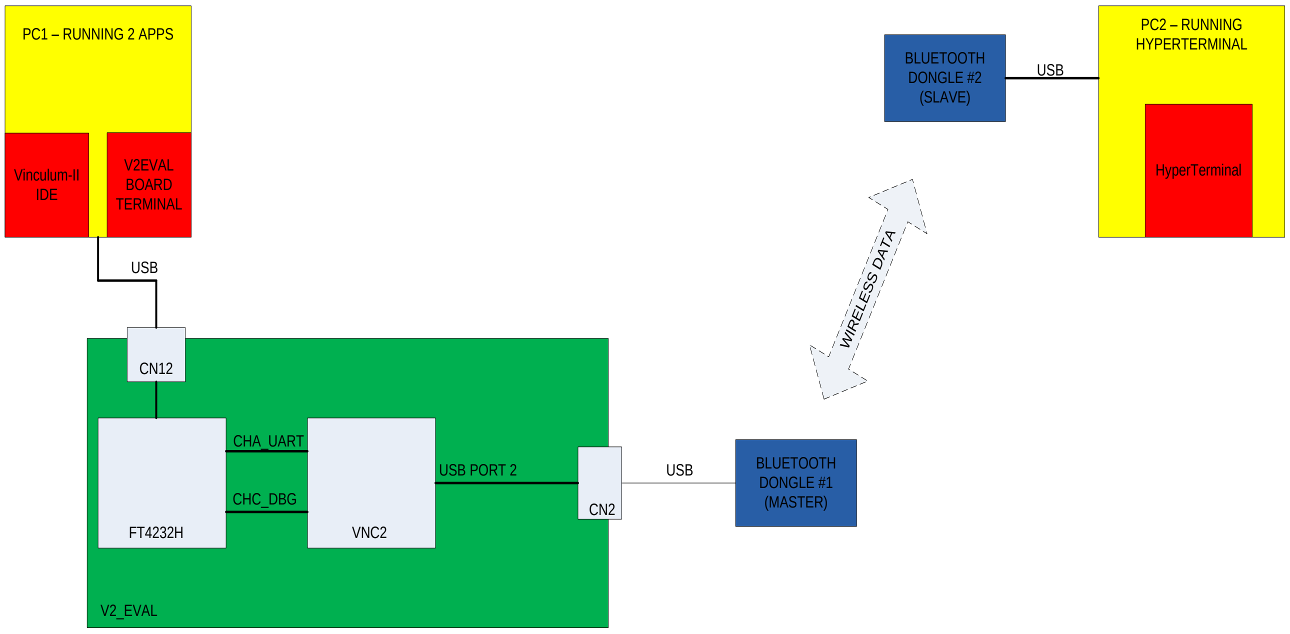


Figure 2.1 – VNC2 Bluetooth Demo Block Diagram

3 Interconnect

3.1 PC 1 – V2EVAL

PC 1 is the development PC. This PC is used to run the FTDI VNC2 toolchain IDE to develop the VNC2 application code and load it onto the V2EVAL board via the debugger interface. The FT4232H channel C on the V2EVAL board is the debugger interface which the IDE will automatically find.

When the application code is compiled and loaded into the VNC2, the V2EVAL Board Terminal application can be used to configure the Bluetooth dongle #1 and is also used for sending and receiving data between the dongles.

The FT4232H channel A on the V2_EVAL board provides the data channel to connect the terminal application on the PC to the UART port of the VNC2.

3.2 PC 2

PC 2 is simply another Bluetooth terminal to allow for data to be transferred in both directions. The PC connects to Bluetooth Dongle #2 via a terminal emulator such as HyperTerminal, TeraTerm or TTY. The default settings of the dongles were used - 115200 -8 –N -1, and no flow control.

4 Source code for the VNC2 Application

The Vinculum II IDE is used to create application code to run on VNC2. This section gives some example source code, and explains its operation.

In essence the code breaks down into simple steps:

Initialise the Vinculum Operating System (VOS)

Initialise the UART

Configure the UART

Initialise the USB Host

Initialise the USBFT232host

Attach the USBFT232Host to the USBHost and configure the interface.

  Then in a loop:

    Check for data on the UART

    Read data from UART

    Write data to USB

    Check for data on the USB

    Read data from USB

    Write data to UART

All the Bluetooth protocol is handled in the dongle and is not something the VNC2 firmware needs to handle.

Note the full project can be downloaded at:

http://www.ftdichip.com/Support/SoftwareExamples/VinculumIIProjects/Bluetooth/bluetooth.zip

The source code is provided on an "as is" basis and is neither guaranteed nor supported.

4.1 VNC2 Initialisation

When generating firmware for VNC2, the first steps are to enable the Vinculum Operating System (VOS), which controls the VNC2 services and device manager, defines the clock speed the core will use, and defines the VNC2 pins that will be used. This is done in the function labelled *main*. The "*main*" function for this application is shown as follows

```
/************************************************************************
/* Main code - entry point to firmware */
/***********************************************************************/

void main(void)
{
	/* 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(); //This is where the pinout is defined. It is generated
	               //automatically by the application wizard

	init_devices();

	/* FTDI:SCT Thread Creation */
	tcbFIRMWARE = vos_create_thread_ex(20, 0x2000, firmware, "Application",
0);
	/* FTDI:ECT */

	vos_start_scheduler();

main_loop:
	goto main_loop;
}
```

Note: Starting the VOS scheduler is always the last thing to be done as all configuration must be complete before this starts.

4.2 Init_devices()

This section initialises the drivers for the USB host port, the UART port, and the usbHostFT232 class driver.

```
void init_devices(void)
{
	// UART Driver configuration context
	uart_context_t uartContext;

	// USB Host configuration context
	usbhost_context_t usbhostContext;

	// Initialise USB Host
	usbhostContext.if_count = 8;
	usbhostContext.ep_count = 16;
	usbhostContext.xfer_count = 2;
	usbhostContext.iso_xfer_count = 2;
	usbhost_init(-1, VOS_DEV_USBHOST_2, &usbhostContext);

	// Initialise USB Host FT232 Driver
	usbHostFt232_init(VOS_DEV_USBHOST_FT232);

	// Initialise UART
	uartContext.buffer_size = VOS_BUFFER_SIZE_128_BYTES;
	uart_init(VOS_DEV_UART,&uartContext);

}
```

4.3 The Main Thread - firmware()

This is the section where the USB host port is checking for the Bluetooth dongle, then enumerating and attaching it, if it is present.

If this succeeds, the code runs in an inner loop that checks and transfers data between the VNC2 UART and VNC2 USB port 2.

NOTE: The UART port settings can be changed by the user but must match the setting used in the V2EVAL Board terminal application on PC 1.

NOTE 2: The USBHOST_FT232 settings are selected to match the default of the Bluetooth dongle.

```
void firmware()
{
      //declare types used

      unsigned char i;
      enum USBHOST_STATUS usbStatus;
      unsigned char status;
      unsigned char buf_USB_to_UART[64];
      unsigned char buf_UART_to_USB[64];
      unsigned short counter;
      unsigned short num_read;
      unsigned short num_written;

      //open UART port
      hUART = vos_dev_open(VOS_DEV_UART);

      //configure UART port

      //set uart baud rate to 115200
    CommonIoctl.ioctl_code = VOS_IOCTL_UART_SET_BAUD_RATE;
    CommonIoctl.set.uart_baud_rate = UART_BAUD_115200;
    //send setup command to uart(determined by handle)
    vos_dev_ioctl(hUART, &CommonIoctl);

    //set uart data bits to 8
    CommonIoctl.ioctl_code = VOS_IOCTL_UART_SET_DATA_BITS;
    CommonIoctl.set.param = UART_DATA_BITS_8;
    //send setup command to uart(determined by handle)
    vos_dev_ioctl(hUART, &CommonIoctl);

    //set uart stop bits to 1
    CommonIoctl.ioctl_code = VOS_IOCTL_UART_SET_STOP_BITS;
    CommonIoctl.set.param = UART_STOP_BITS_1;
    //send setup command to uart(determined by handle)
    vos_dev_ioctl(hUART, &CommonIoctl);

    //set uart parity to none
    CommonIoctl.ioctl_code = VOS_IOCTL_UART_SET_PARITY;
    CommonIoctl.set.param = UART_PARITY_NONE;
    //send setup command to uart(determined by handle)
    vos_dev_ioctl(hUART, &CommonIoctl);

    //set uart flow control to RTS/CTS
    CommonIoctl.ioctl_code = VOS_IOCTL_UART_SET_FLOW_CONTROL;
    CommonIoctl.set.param = UART_FLOW_RTS_CTS;
    //send setup command to uart(determined by handle)
    vos_dev_ioctl(hUART, &CommonIoctl);

    //enable UART DMA
    CommonIoctl.ioctl_code = VOS_IOCTL_COMMON_ENABLE_DMA;
    CommonIoctl.set.param = DMA_ACQUIRE_AND_RETAIN;
    vos_dev_ioctl(hUART,&CommonIoctl);

    //open host driver and FT232 host class driver
    hUSBHOST_2 = vos_dev_open(VOS_DEV_USBHOST_2);

/*******************************************************************
//Loop to check device is enumerated and send data back and forward
/*******************************************************************/
```

```
        do
        {
                //wait for enumeration to complete
                vos_delay_msecs(250);
                // user ioctl to see if bus available
                hc_iocb.ioctl_code = VOS_IOCTL_USBHOST_GET_CONNECT_STATE;
                hc_iocb.get = &i;
                vos_dev_ioctl(hUSBHOST_2, &hc_iocb); //checking USB port2 - the host

                if (i == PORT_STATE_ENUMERATED)
                {
                        status = 0;

                        // find FTDI vendor device
                        hc_iocb_vendor.vid = USB_VID_FTDI;
                        hc_iocb_vendor.pid = USB_PID_ANY;

                        // user ioctl to find first hub device
                        hc_iocb.ioctl_code =
                       VOS_IOCTL_USBHOST_DEVICE_FIND_HANDLE_BY_VID_PID;
                        hc_iocb.handle.dif = NULL;
                        hc_iocb.set = &hc_iocb_vendor;
                        hc_iocb.get = &ifDev2;

                        usbStatus = vos_dev_ioctl(hUSBHOST_2, &hc_iocb);
                        if (usbStatus != USBHOST_OK)
                        {
                                status = 1;
                        }

                        if (status == 0)
                        {
                                // now we have a device, intialise a driver for it
                                hUSBHOST_FT232 = vos_dev_open(VOS_DEV_USBHOST_FT232);

                                // ft232_attach
                                ft232HostAttach.hc_handle = hUSBHOST_2;
                                ft232HostAttach.ifDev = ifDev2;
                                ft232HostAttach.ftPort = USBHOSTFT232_PORTA;

                                CommonIoctl.ioctl_code = VOS_IOCTL_USBHOSTFT232_ATTACH;
                                CommonIoctl.set = &ft232HostAttach;

                                 if (vos_dev_ioctl(hUSBHOST_FT232, &CommonIoctl) !=
                                        USBHOSTFT232_OK)
                                {
                                                status = 1;
                                }
                                else
                                {
/*******************************************************************
//Now that device is connected it must be configured
//NOTE BAUD RATE uses set.uart_baud_rate, all other parameters are
//set with set.param.
******************************************************************/

                                        // Set baud rate 115200baud
                                        CommonIoctl.ioctl_code =
                                                VOS_IOCTL_USBHOSTFT232_SET_BAUD_RATE;
                                        CommonIoctl.set.uart_baud_rate =
                                                USBHOSTFT232_BAUD_115200;
                                        vos_dev_ioctl(hUSBHOST_FT232,&CommonIoctl);
```

```
                            // Set number of data bits to 8
                            CommonIoctl.ioctl_code =
                                        VOS_IOCTL_USBHOSTFT232_SET_DATA_BITS;
                            CommonIoctl.set.param = USBHOSTFT232_DATA_BITS_8;
                            vos_dev_ioctl(hUSBHOST_FT232,&CommonIoctl);

                            // Set number of stop bits to 1
                            CommonIoctl.ioctl_code =
                                  VOS_IOCTL_USBHOSTFT232_SET_STOP_BITS;
                            CommonIoctl.set.param = USBHOSTFT232_STOP_BITS_1;
                            vos_dev_ioctl(hUSBHOST_FT232,&CommonIoctl);

                            // Set parity to none
                            CommonIoctl.ioctl_code =
                                  VOS_IOCTL_USBHOSTFT232_SET_PARITY;
                            CommonIoctl.set.param = USBHOSTFT232_PARITY_NONE;
                            vos_dev_ioctl(hUSBHOST_FT232,&CommonIoctl);

                            // Set flow control to none
                            CommonIoctl.ioctl_code =
                                  VOS_IOCTL_USBHOSTFT232_SET_FLOW_CONTROL;
                            CommonIoctl.set.param = USBHOSTFT232_FLOW_NONE;
                            vos_dev_ioctl(hUSBHOST_FT232,&CommonIoctl);

                            // Set DTR - needed to start the Bluetooth dongle
                            CommonIoctl.ioctl_code =
                                  VOS_IOCTL_USBHOSTFT232_SET_DTR;
                            vos_dev_ioctl(hUSBHOST_FT232,&CommonIoctl);

                            // Set RTS - needed to start the Bluetooth dongle
                            CommonIoctl.ioctl_code =
                                  VOS_IOCTL_USBHOSTFT232_SET_RTS;
                            vos_dev_ioctl(hUSBHOST_FT232,&CommonIoctl);

                            //start polling the connected device
                            CommonIoctl.ioctl_code =
                                  VOS_IOCTL_USBHOSTFT232_START_POLL;
                            vos_dev_ioctl(hUSBHOST_FT232,&CommonIoctl);


/*******************************************************************
//Loop data between UART and USB port
*******************************************************************/

//UART checking queue status
do
{
                  memset(buf_UART_to_USB,0x0,64);

                  num_written = 0;
                  CommonIoctl.ioctl_code = VOS_IOCTL_COMMON_GET_RX_QUEUE_STATUS;

                  vos_dev_ioctl(hUART, &CommonIoctl);

                  num_written = CommonIoctl.get.queue_stat;

                  if (num_written != 0)
                  {

                  //UART reading
                  num_read = 0;
                  if (vos_dev_read(hUART, buf_UART_to_USB, num_written,
                        &num_read) == UART_OK)
                        {
                // write it out the USB port.
                        counter = 0;
                        vos_dev_write(hUSBHOST_FT232, buf_UART_to_USB, num_read,
                                    &counter);
                        }
                  }

//USB host checking for data
                  num_written = 0;
                  CommonIoctl.ioctl_code = VOS_IOCTL_COMMON_GET_RX_QUEUE_STATUS;

                  vos_dev_ioctl(hUSBHOST_FT232, &CommonIoctl);

                  num_written = CommonIoctl.get.queue_stat;

                  if (num_written != 0)
                  {
                  //USB HOST READING
                  memset(buf_USB_to_UART,0x0,64);

                  num_read = 0;

      if (vos_dev_read(hUSBHOST_FT232, buf_USB_to_UART, num_written, &num_read)
            == USBHOSTFT232_OK)

                      {
                       // write it out the UART port
                        counter = 0;
                        vos_dev_write(hUART, buf_USB_to_UART, num_read,
                        &counter);
                      }
                  }

            } while (1);
                  }
      }
vos_dev_close(hUSBHOST_FT232);
            }
      } while (1);

}
```

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.

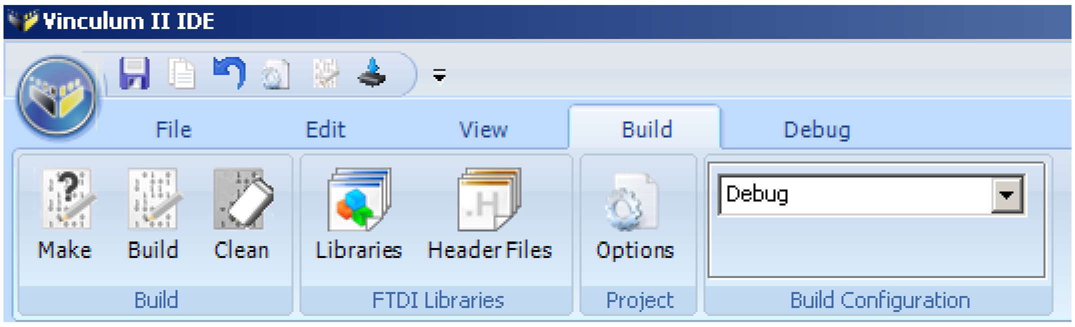
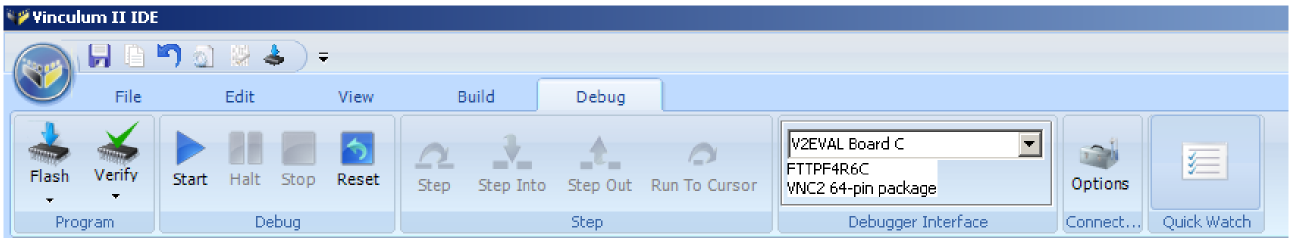


Figure 5.1 – Vinculum II IDE Build Button

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



V2EVAL Board C
FTTPF4R6C
VNC2 64-pin package

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.

6 Configuring the Bluetooth Dongles (and running the app)

The Bluetooth dongles are configured over the USB port.

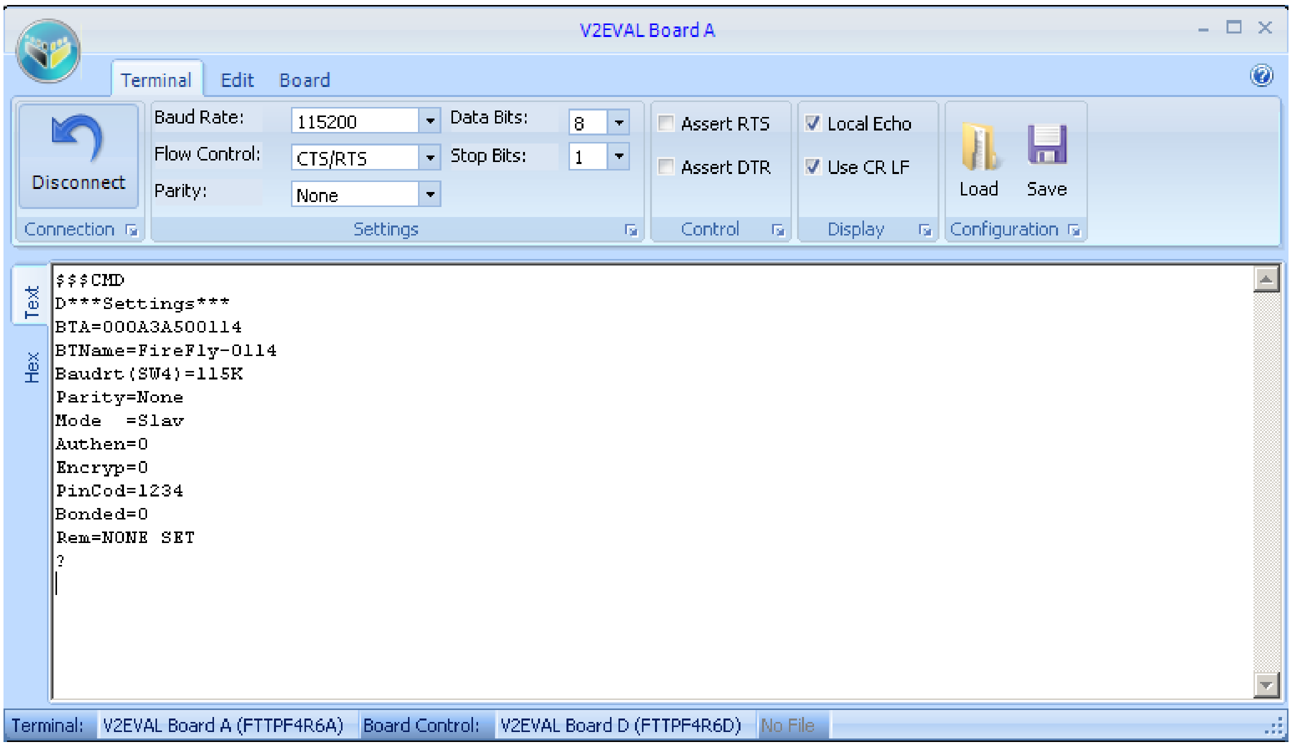
One device is set to be the master and the other the slave. Either the master or slave dongle can be connected to the Vinculum host.

You can do this configuration via the VNC2 or by simply connecting them to a PC and setting them up before use with your terminal application.

$$$ will enter the dongle into command mode.

D <rtn> will give the basic configuration of the device.

The most interesting configuration parameter at this stage will be the address (BTA):



Baud Rate: 115200
Flow Control: CTS/RTS
Parity: None
Data Bits: 8
Stop Bits: 1
Assert RTS [ ]
Assert DTR [ ]
Local Echo [✓]
Use CR LF [✓]

```
$$$CMD
D***Settings***
BTA=000A3A500114
BTName=FireFly-0114
Baudrt(SW4)=115K
Parity=None
Mode  =Slav
Authen=0
Encryp=0
PinCod=1234
Bonded=0
Rem=NONE SET
?
```

Terminal: V2EVAL Board A (FTTPF4R6A) Board Control: V2EVAL Board D (FTTPF4R6D) No File

Figure 6.1 – Bluetooth Dongle descriptors read by the Vinculum

In this example the dongle connected to the Vinculum device is shown to be in slave mode with address 000A3A500114.

To connect to this device with the master dongle, use the terminal connected to the master dongle to send the command " C, 000A3A500114 <rtn>". This instructs the master Bluetooth Dongle to wirelessly connect with Bluetooth device address 000A3A500114 whcih is connected to the Vinculum.

When connected any data typed in the V2EVAL Board terminal on PC 1 will appear in the terminal application on PC 2 and vice versa.

For additional commands for the Bluetooth dongle see the user manual:

http://www.rovingnetworks.com/Docs/Bluetooth-RN-USB-X-UM.pdf

7 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 – References

Application and Technical Notes available at

http://www.ftdichip.com/Support/Documents/AppNotes.htm

V2-EVAL datasheet

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

Vinculum-II IO Cell Description

http://www.ftdichip.com/Support/Documents/AppNotes/AN_137_Vinculum-II%20IO_Cell_Description.pdf

Vinculum-II Debug Interface Description

http://www.ftdichip.com/Support/Documents/AppNotes/AN_138_Vinculum-II_Debug_Interface_Description.pdf

Vinculum-II IO Mux Explained

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

Vinculum-II Errata Technical Note

http://www.ftdichip.com/Support/Documents/TechnicalNotes/TN_118_VNC2%20Errata%20Technical%20Note.pdf

Roving Networks USB Bluetooth Dongle User Manual

http://www.rovingnetworks.com/Docs/Bluetooth-RN-USB-X-UM.pdf

Appendix B – List of Figures and Tables

List of Figures

Appendix C – Revision History

Version 1.0	First release	17th August, 2011