Future Technology Devices International Ltd.

Vinco LCD Interface Example Application Note AN_153

Document Reference No.: FT_000327

Version 2.0

Issue Date: 2011-04-15

This application note describes how the Vinco module can be used to provide an interface to a LCD display.

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

Vinco is a development module inspired by the Arduino concept and uses the Vinculum II, VNC2 device. Vinco uses a VNC2-64Q package to facilitate 38 GPIO options on 0.1" pitch sockets. Vinco is designed as a prototyping platform for VNC2 based designs and applications.

This application note describes an example of how to use the Vinco module to create and display text messages on a 2 line, 16 character, monochrome LCD display. The application note also provides "C" source code examples to help the user get started with their own specific application. This source code can be downloaded from the FTDI website at:
http://www.ftdichip.com/Support/SoftwareExamples/VinculumIIProjects/Vinco_LCD.zip

Note: Any sample code provided in this note is for illustration purposes and is not guaranteed or supported.

Figure 1.1 - VINCO

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 Topway LCD Display

This application example uses the Topway LMB162ABC 16 character 2 line LCD display. The display is driven by a 5V power supply and 4 data lines control the characters displayed on the display. For more information on Topway displays see: http://www.topwaydisplay.com/Pub/Manual/LMB162ABC-Manual-Rev0.2.pdf

Figure 1.2 – Topway LMB162ABC LCD Module

Table of Contents

2 Block Diagram

This block diagram, Figure 2.1, shows the interconnect required for the Vinco to drive the LCD display.

The Vinco debug port is used to load the firmware onto the module. (Note that this requires a VNC2 DEBUG MODULE http://ftdichip.com/Support/Documents/DataSheets/Modules/DS_V2Debug_Module.pdf)

The interconnect between the two modules is used to transfer the text to be displayed on the LCD.

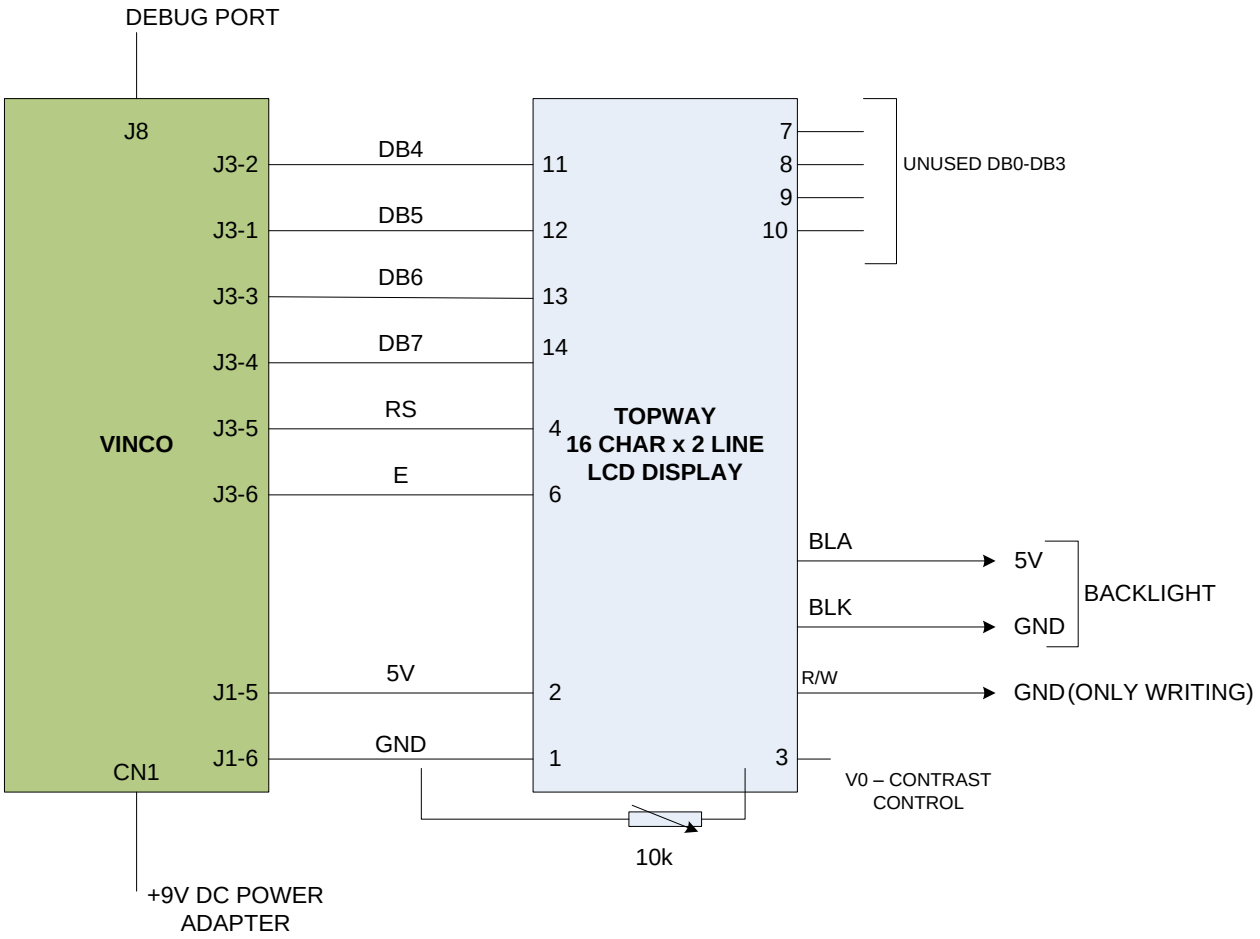

Figure 2.1 – Vinco Sign Writer Demo Block Diagram

3 Interconnect

3.1 Power

The Vinco module may be powered from the USB port on CN3 (5V) or via an external power converter (9V/1A DC) to CN1 (for example the FTDI VNCLO-PSU-UK)

As this application provides power to external circuitry (the LCD display), the Vinco is powered from an external 9V supply.

To ensure this power source is routed to the PCB, JP1 on the Vinco module must be set to the 2-3 position.

Power from the Vinco module is taken from J1 pin 5 to give a +5V supply for the Topway LCD display.

3.2 LCD Control

The Topway LCD display may be controlled in either an 4-bit data mode or an 8-bit data mode. This application uses the 4 bit data mode. These are the signals labelled DB4 – DB7 on the block diagram in section 2. These and the remaining signals required between Vinco and the LCD display are described in Table 3.1.

Signal	Function
DB4-DB7	**Data lines for sending information to the LCD**
RS	**Register Select is used to determine if the information being sent to the LCD panel is display data (RS=1) or an instruction (RS = 0).**
E	**The E bit is used to enable the device for access (E = 1).**
R/W	**Tied to GND as the LCD is only ever written in this demo.**
BLA/BLK	**Supply (5V) for the LCD backlight**
10k variable resistor between VO and GND	**allows for control of the LCD contrast**

Table 3.1 - Signal Name and Description – LCD Interface

3.3 Debugger Interface

The purpose of the debugger interface is to provide access to the VNC2 silicon/firmware debugger. The debug interface can be accessed by connecting a *VNC2_Debug_Module* (http://www.ftdichip.com/Support/Documents/DataSheets/ICs/DS_Vinculum-II.pdf) to the J8 connector. This debug module gives access to the debugger through a USB connection to a PC via the Integrated Development Environment (IDE). The IDE is a graphical interface to the VNC2 software development tool-chain and gives the following debug capabilities through the debugger interface:

- Flash Erase, Write and Program.
- Application debug - application code can have breakpoints, be single stepped and can be halted.
- Detailed internal debug - memory and register read/write access.

The IDE may be downloaded, free of charge, from
http://www.ftdichip.com/Firmware/V2TC/VNC2toolchain.htm

The Debugger Interface, and how to use it, is further described in the following applications Note
Vinculum-II Debug Interface Description

3.3.1 Signal Description - Debugger Interface

Table 3.2 shows the signals and pins description for the Debugger Interface pin header J8

Pin No.	*Name*	*Name On PCB*	*Type*	*Description*
J8-1	IO0	DBG	I/O	Debugger Interface
J8-2	-	[Key]	-	Not connected. Used to make sure that the debug module is connected correctly.
J8-3	GND	GND	PWR	Module ground supply pin
J8-4	RESET#	RST#	Input	Can be used by an external device to reset the VNCL2. This pin is also used in combination with PROG# and the UART interface to program firmware into the VNC2.
J8-5	PROG#	PRG#	Input	This pin is used in combination with the RESET# pin and the UART interface to program firmware into the VNC2.
J8-6	5V0	VCC	PWR Input	5.0V module supply pin. This pin can be used to provide the 5.0V input to the V2DIP2-32 from the debugger interface when the V2DIP2-32 is not powered from the USB connector (VBUs) or the DIL connector pins J1-1 and J3-6.

Table 3.2 - Signal Name and Description – Debugger Interface

4 Source code for the VNC2 writing to LCD Display

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, used to drive the LCD display via the Vinco module.

Note the full project can be downloaded at:
http://www.ftdichip.com/Support/SoftwareExamples/VinculumIIProjects/Vinco_LCD.zip

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

Void main (void)

{

```
// GPIO context structure
    gpio_context_t gpioCtx;


// call VOS initialisation routines
    vos_init(10, VOS_TICK_INTERVAL, NUMBER_OF_DEVICES);
    vos_set_clock_frequency(VOS_48MHZ_CLOCK_FREQUENCY);

    if (vos_get_package_type() == VINCULUM_II_64_PIN)
    {
        // GPIO port A bit 0 to pin 51
        vos_iomux_define_output(51,IOMUX_OUT_GPIO_PORT_A_0); // DB4
        // GPIO port A bit 1 to pin 52
        vos_iomux_define_output(52,IOMUX_OUT_GPIO_PORT_A_1); // DB5
        // GPIO port A bit 2 to pin 55
        vos_iomux_define_output(55,IOMUX_OUT_GPIO_PORT_A_2); // DB6
        // GPIO port A bit 3 to pin 56
        vos_iomux_define_output(56,IOMUX_OUT_GPIO_PORT_A_3); // DB7
        // GPIO port A bit 4 to pin 57
        vos_iomux_define_output(57,IOMUX_OUT_GPIO_PORT_A_4); // RS
        // GPIO port A bit 5 to pin 58
        vos_iomux_define_output(58,IOMUX_OUT_GPIO_PORT_A_5); // E

        // UART to V2EVAL board pins
```

The *main* function will also initialise the device drivers used in this application (GPIO in this case), it then defines the threads that will be started (labelled firmware in this project) and finally it starts the VOS scheduler. This is shown as follows:

```
// initialise device drivers
// TODO: call initialisation routines for included device drivers

    gpioCtx.port_identifier = GPIO_PORT_A;
    gpio_init(VOS_DEV_GPIO,&gpioCtx);

    // create threads for firmware application (no parameters)
    tcbFirmware = vos_create_thread(29, SIZEOF_THREAD_MEMORY, firmware, 0);
    // start VOS scheduler

    vos_start_scheduler();
```

```
}
```

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

4.2 LCD Initialisation

The LCD must be put into an initial known state to be able to accept new data and this initialisation is done in the lcd_ini function.

```
void lcd_ini(VOS_HANDLE hLCD)
{
    vos_delay_msecs(100);
    // Send Reset command
    write_lcd_cmd(hLCD, 0x03);
    vos_delay_msecs(1);
    // Send Function Set
    write_lcd_cmd(hLCD, 0x28);
    vos_delay_msecs(1);
    write_lcd_cmd(hLCD, 0x28);
    vos_delay_msecs(1);
    // Send Display control command
    write_lcd_cmd(hLCD, 0x0C);
    vos_delay_msecs(1);
    // Send Display Clear command
    write_lcd_cmd(hLCD, 0x01);
    vos_delay_msecs(1);
    // Send Entry Mode Set command
    write_lcd_cmd(hLCD, 0x06);
    vos_delay_msecs(1);
}
```

The data values which can be sent to the LCD are defined in the LCD user manual. The write_lcd_cmd(hLCD, value) is an instruction to call the write_LCD_cmd function so that data may be is moved from the VNC2 GPIO lines to the LCD.

4.3 Writing Command Instructions to the LCD

The “write_lcd_cmd” command is used to send instructions to control the LCD panel.

The data is shifted 4 bits as only the upper 4 data lines of the LCD panel are used.

The example source code is detailed below

```
void write_lcd_cmd(VOS_HANDLE hLCD, unsigned char data)
{
    unsigned char cmd;
    // Write High nibble data to LCD
    cmd = (((data>>4)&0x0F) | lcd_e);
    cmd = (cmd &(~lcd_dat)); // Select Registers
    vos_dev_write(hLCD,&cmd,1,NULL);
    // Toggle 'E' pin
    cmd &= (~lcd_e);
    vos_dev_write(hLCD,&cmd,1,NULL);
    // Write Low nibble data to LCD
    cmd = ((data & 0x0F) | lcd_e);
    cmd = (cmd &(~lcd_dat)); // Select Registers
    vos_dev_write(hLCD,&cmd,1,NULL);
    // Toggle 'E' pin
```

```
    cmd &= (~lcd_e);
    vos_dev_write(hLCD,&cmd,1,NULL);
    vos_delay_msecs(1);

}
```

vos_dev_write is a VOS defined command for writing out data. The hLCD is the handle for the driver that the data is being sent to. In this example that means the GPIO lines.

4.4 Writing Data Bytes to the LCD Display

The "write_lcd_data" command is used to write data to be displayed by the LCD panel.

The data is shifted 4 bits as only the upper 4 data lines of the LCD panel are used.

The example source code is detailed below

```
void write_lcd_data(VOS_HANDLE hLCD, unsigned char data)
{
    unsigned char cmd;

    // Write High nibble data to LCD
    cmd = (((data>>4)&0x0F) | lcd_dat);
    cmd = (cmd | lcd_e); // Select DDRAM
    vos_dev_write(hLCD,&cmd,1,NULL);
    // Toggle 'E' pin
    cmd &= (~lcd_e);
    vos_dev_write(hLCD,&cmd,1,NULL);
    // Write Low nibble data to LCD
    cmd = ((data & 0x0F) | lcd_dat);
    cmd = (cmd | lcd_e); // Select DDRAM
    vos_dev_write(hLCD,&cmd,1,NULL);
    // Toggle 'E' pin
    cmd &= (~lcd_e);
    vos_dev_write(hLCD,&cmd,1,NULL);
    vos_delay_msecs(1);

}
```

4.5 Writing Data Strings to the LCD Display

The "write_lcd_str" command is used to write strings, as opposed to individual bytes, to the LCD.

The example source code is detailed below

```
void write_lcd_str(VOS_HANDLE hLCD, unsigned char *str)
{
    while(*str != '\0')
    {
        write_lcd_data(hLCD, *str);
        ++str;
    }

}
```

4.6 The Firmware function

The "firmware" function in this example defines the variables used, the text to be sent to the display and calling the LCD control functions.

```
void firmware(void)
{
// VOS handles for opened devices
    VOS_HANDLE hGpio;

// GPIO IOCTL request block
    gpio_ioctl_cb_t gpio_iocb;

// general purpose variables
    unsigned char data = 1;
    unsigned char i;
// string to display (include space for terminating NULL)
    unsigned char *lcd_str = "Vinco";
    unsigned char data_buf[16];

// find and open GPIO device port A
    hGpio = vos_dev_open(VOS_DEV_GPIO);

    gpio_iocb.ioctl_code = VOS_IOCTL_GPIO_SET_MASK;
    gpio_iocb.value = 0xFF;    // set all as output

    vos_dev_ioctl(hGpio, &gpio_iocb);

    lcd_ini(hGpio);
    lcd_str = "Vinco";

// Set 1-st line address
// Send Display Clear command
    write_lcd_cmd(hGpio, 0x01);
    vos_delay_msecs(2);
    write_lcd_cmd(hGpio, (0x05 | 0x80));
    write_lcd_str(hGpio, lcd_str);

// Set 2-nd line address
    lcd_str = "www.ftdichip.com";
    write_lcd_cmd(hGpio, (0x40 | 0x80));
    write_lcd_str(hGpio, lcd_str);

    vos_delay_msecs(2000);
    lcd_str = "The Best Board";

// Set 1-st line address
    write_lcd_cmd(hGpio, 0x01);
    vos_delay_msecs(2);
    write_lcd_cmd(hGpio, (0x01 | 0x80));
    write_lcd_str(hGpio, lcd_str);

// Set 2-nd line address
    lcd_str = "Ever!!!";
    write_lcd_cmd(hGpio, (0x45 | 0x80));
    write_lcd_str(hGpio, lcd_str);

    vos_delay_msecs(2000);
    } while (1);
}
```

5 Programming Vinco

When Vinco has been connected to the LCD panel and the firmware has been built in the IDE, the next step is to transfer the .ROM file generated by the IDE to the Vinco module. The IDE generates the .ROM file with a single button click of the "Build" button.

Connect the USB port of the VNC2 Debug Module to a PC and load the free FTDI drivers for the FT232R device on the debug module. This will happen automatically via Windows Update if you are connected to their internet. Otherwise refer to the installation guide for your OS:

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

The IDE should now automatically detect the VNC2 debug module.

Connect the other end of the VNC2 Debug Module to the J8 connector of the Vinco.

Use the IDE FLASH button to load the .ROM file into the Vinco. A getting started guide for using the Vinculum IDE may be downloaded from:

http://www.ftdichip.com/Support/Documents/AppNotes/AN_142_Vinculum-II_Tool_Chain_Getting_Started_Guide.pdf

The IDE will report back a successful programming. At this point the VNC2 Debug module may be removed from the Vinco J8 connector.

The .rom file can also be downloaded from the following location:

http://www.ftdichip.com/Support/SoftwareExamples/VinculumIIProjects/Vinco_LCD.zip

6 Running the firmware

The Vinco may be reset by power cycling the unit and then the firmware will run...

The user will observe text on the LCD being updated as per the firmware code:

Vinco

www.ftdichip.com

The best board

Ever !!!

It is left to the user to experiment with changing the displayed text by modifying the sample project code.

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

Vinco datasheet

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

VNC2 Debug Module

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

Topway LMBABC Display

(http://www.topwaydisplay.com/Pub/Manual/LMB162ABC-Manual-Rev0.2.pdf)

Appendix B – List of Figures and Tables

List of Figures

List of Tables

Appendix C – Revision History

Version draft	First draft	August 2010
Version 1.0	First Release	25th October 2010
Version 2.0	Change brand name from Vinculo to Vinco	14th April 2011