Future Technology Devices International Ltd.

# Application Note AN_118

# Migrating Vinculum Designs From VNC1L to VNC2-48L1A

**Document Reference No.: FT_000159**

**Version 1.3**

**Issue Date: 2011-05-13**

The purpose of this document is to provide guidelines for migrating VNC1L designs to VNC2.

**Future Technology Devices International Limited (FTDI)**

Unit1, 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** Web: http://www.ftdichip.com

**TABLE OF CONTENTS**

# 1 Introduction

The Vinculum VNC2 is FTDI's 2nd generation USB host controller solution and it expands on the capabilities of the VNC1L. VNC2 is supplied in 6 different packages. These are 32 pin QFN and LQFP packages, 48 pin QFN and LQFP packages and 64 pin QFN and LQFP packages.

The smaller packages reduce the number of IO by 16 pins to allow for more compact designs where space limitations are a consideration.

The larger packages provide additional 16 I/O to allow for increased functionality. In addition to the increased number of VNC2 package options, there is also a new software development tool suite developed by FTDI to enable users to create their own customised firmware.

The main focus of this document will be on how to migrate from a design using the VNC1L to a VNC2-48L1A device.

## 2 VNC1L vs VNC2

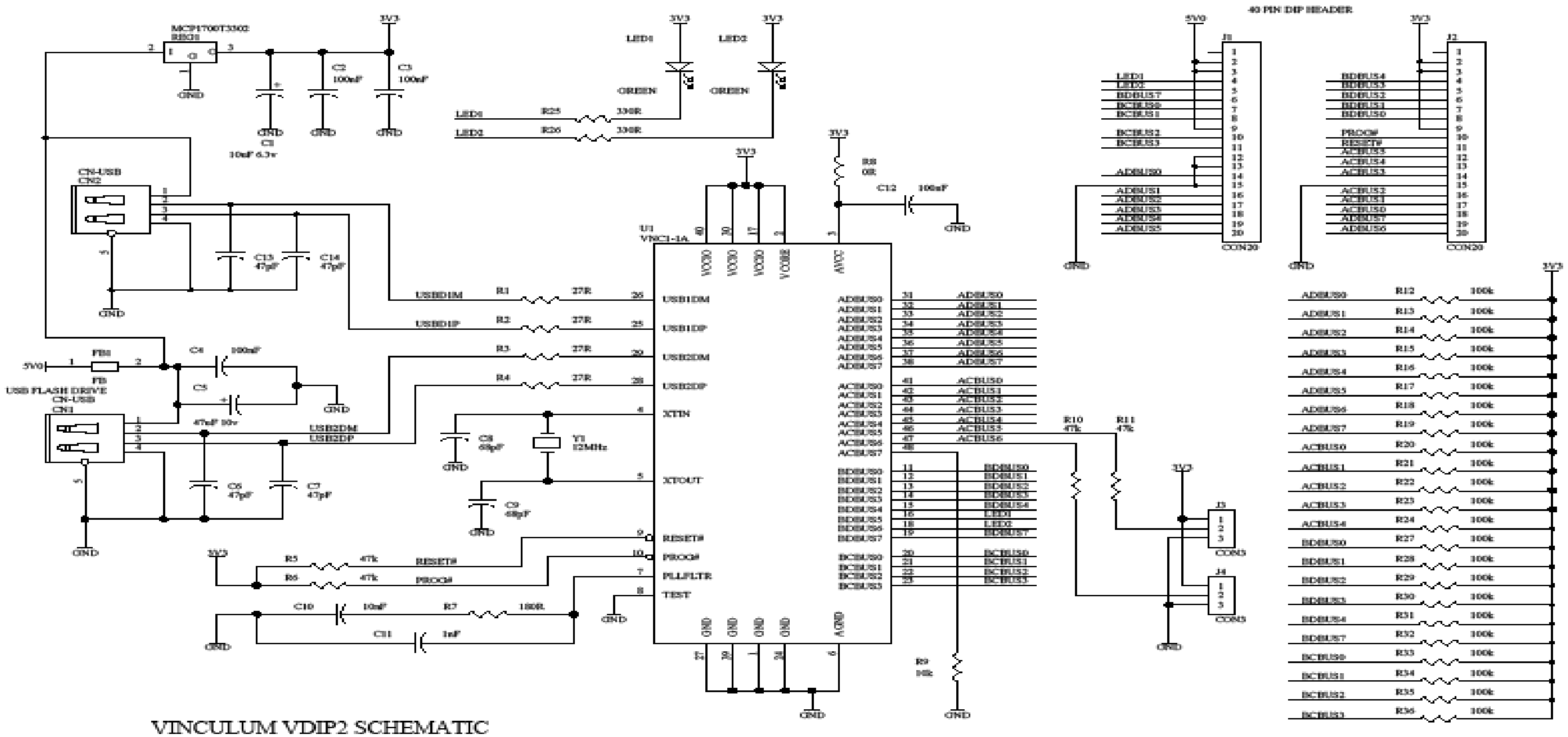
The main features of the two generations of devices are shown below. All features are available in the 64 pin packages. The 48 pin and 32 pin packages have the same memory and internal functions but less I/O. The 48 pin package matches the VNC1L device. For full details see the VNC2 Datasheet at www.ftdichip.com

| FEATURE | VNC1L | VNC2 |
|---|---|---|
| | | |
| **MECHANICAL** | | |
| Package | 48 pin LQFP | 64, 48,32 pin QFN or LQFP |
| Temperature | -40 to +85C | -40 to +85C |
| | | |
| **ELECTRICAL** | | |
| VCC | 3V3 | 3V3 |
| VCCIO | 5V tolerant 3V3 | 5V tolerant 3V3 |
| CLK source | 12MHz (external) | 12MHz (external) |
| | | |
| **CPU** | 8-bit Harvard architecture | 16-bit Harvard architecture |
| | | |
| **INTERFACING** | | |
| USB ports | 2 | 2 |
| UART port | 1 | 1 |
| SPI slave port | 1 | 2 |
| SPI master port | 0 | 1 |
| FIFO monitor port option | 1 | 1 |
| Debug port | 0 | 1 |
| | | |
| **FIRMWARE** | | |
| Precompiled firmware | YES | YES |
| Tools for creating own firmware | NO | YES |
| | | |
| **MEMORY** | | |
| DATA RAM | 4k x 8 (4kbytes) | 4k x 32 (16kbytes) |
| E-FLASH | 64k x 8 (64kbytes) | 128k x 16 (256kByte) |
| | | |
| **USB MODES** | | |
| Speed | Full / low | Full / low |
| Transfer modes | Bulk / Interrupt | Bulk, interrupt, isochronous |
| | | |
| **CONFIGURATION PORTS** | | |
| UART | YES | YES |
| USB | YES (after initial programming) | YES |
| SPI | NO | YES |
| FIFO | NO | YES |
| DEBUG PORT | N/A | YES |
| | | |
| | | |

# 3 Reference Schematics

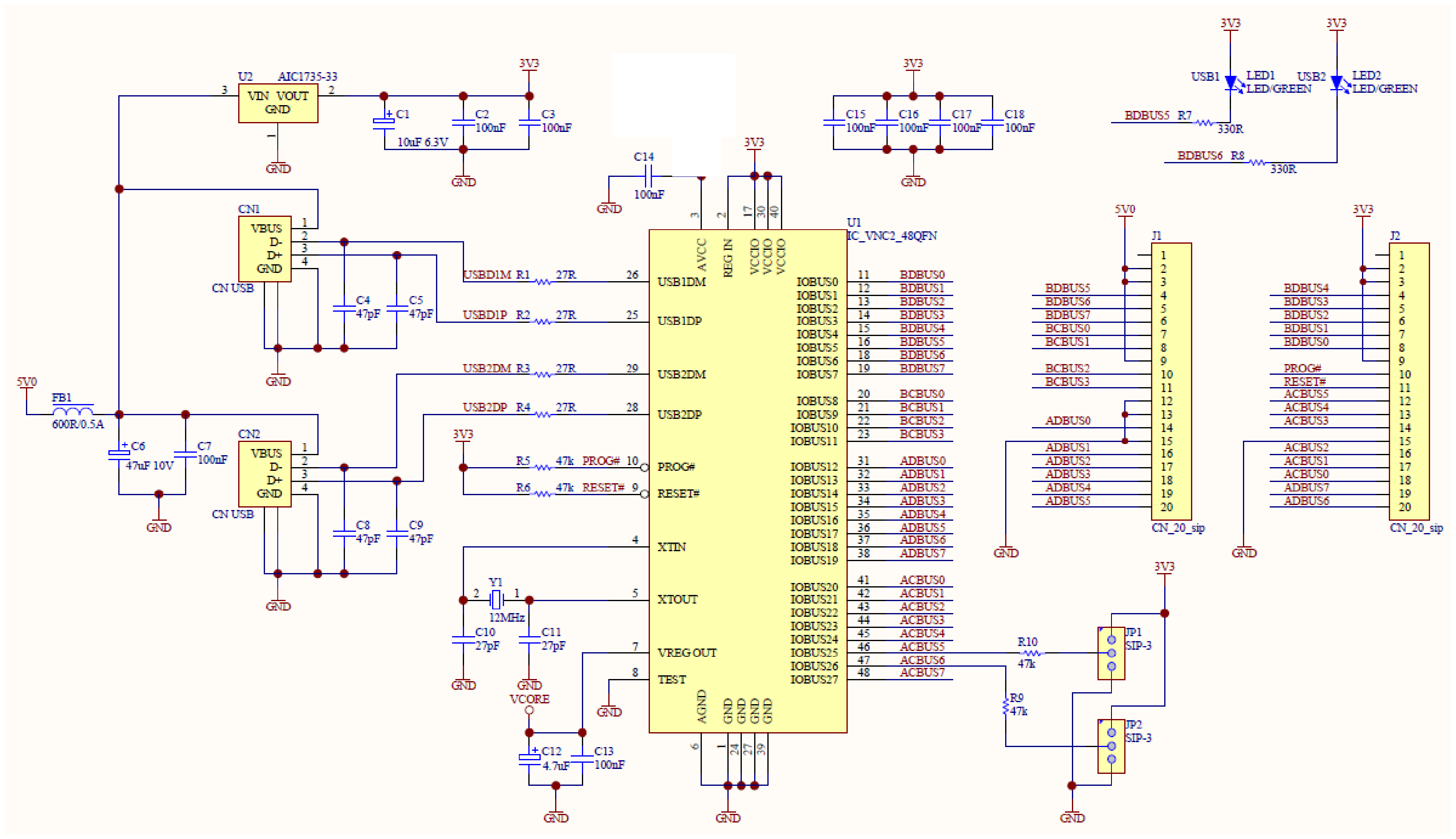
## 3.1 VNC1L schematic

This schematic is for the VDIP2 based on the VNC1L.



VINCULUM VDIP2 SCHEMATIC

## 3.2 VNC2-48L1A Equivalent of VNC1L VDIP2 Reference Schematic



## 4 BOM Change required to convert from VNC1L to VNC2-48L1A

| VNC1L Schematic Component | VNC2-48L1A Schematic Component | Description |
|---|---|---|
| U1 - VNC1L | U1 -VNC2-48L1A | Controller IC |
| C8, C9 – 68pF Capacitor | C10, C11 – 27pF Capacitor | Load capacitor on crystal |
| R9, 10k resistor | Do not fit pull down on U1 pin 48 | PLL enable |
| R7, 180R | Replace with 0R link or simply track over if redesigning PCB | VNC1L PLL Filter / VNC2 VREGOUT |
| C10, 10nF | C12, 4.7uF | VNC1L PLL Filter / VNC2 VREGOUT |
| C11, 1nF | C13, 100nF | VNC1L PLL Filter / VNC2 VREGOUT |
| R8, 0R to 3V3 | | Remove |
| | | |

The pull up resistors, shown on the VNC1L schematic R12-R36, are optional even on the VNC1L schematic.

Pin 3 AVCC of the VNC2-48L is internally bonded to the 1V8 internal regulator. This is the only package with this internal bond for ease of upgrade from a VNC1L.

## 5 Firmware

As the internals of the two devices is different, the firmware that runs on the devices is different. However to maintain backward compatibility there is equivalent firmware available for download.

| VNC1L Firmware | VNC2 Firmware |
|---|---|
| VDAP | V2DAP |
| VMSC | V2MSC |
| VDPS | V2DPS |
| VCDC | V2CDC |
| VDIF | V2DIF2 |

These builds may be downloaded from www.ftdichip.com (at the time of writing this application note, some of these builds were not available, but were scheduled to be put onto the website soon).

In addition to using these pre-compiled libraries, VNC2 is supplied with a software development tool chain to allow customers to customise the pre-compiled firmware or create their own firmware for greater flexibility in design.

Note: The precompiled builds are designed for the 48 pin pkg as that was all VNC1L was available in.

# 6 Loading Firmware

VNC1L was limited to using the UART to load firmware onto a blank device. The VNC2 may be programmed via the debug port or via the UART interface. Although FTDI will provide utilities for this, there may be situations where users prefer to develop their own programmer over the UART and there are some significant differences between VNC1L and VNC2.

Key differences between VNC1L and VNC2:

- Each block of the flash on the VNC2 is 128 bytes compared with 64 bytes on the VNC1L.

- The endianess of the VNC2 is different from that of the VNC1L meaning that data must be reversed before it can be sent to the device.

- The size of the ROM files on VNC1L were all 64k, the ROM files on the VNC2 can be anything up to the size of the flash.

To program the VNC2 firmware over the UART the device must be reset in prog mode; this is achieved by driving the **prog#** pin low and then driving the **reset#** pin low then high.  When the device is in prog mode it is ready be programmed over the UART using the following command / response standard.

| **Command Name** | **Command(s):** | **Response** | **Comment** |
|---|---|---|---|
| Echo | 0xFF | 0xFF | |
| Echo | 0xFB | 0xFB | VNC1L used the 0xFA command to echo, VNC2 uses 0xFB to distinguish between the devices. |
| Set Baud Rate | 0x01<br>BaudRate(1, 2 or 3) | 0x02 | (115200 Baud == 1, 1 MBaud == 2, 3 MBaud == 3) |
| Read Address | 0x02<br>FlashAddrLow<br>FlashAddrHigh | 0x02<br>Data to Read | |
| Write Address | 0x03<br>FlashAddrLow<br>FlashAddrHigh<br>Data to Write | 0x02 | The 0x02 response will be sent after FlashAddrHigh has been received by the device. |

Below is a snippet of the test code that has been used to write and verify the flash of the VNC2....

```
// Sample code for writing flash of VNC2 written in C/C++.

// Function:   sendData - writes the data read from the ROM file to the VNC2 and verifies it.
// Parameters: romfile - a handle to the ROM file obtained from the open(....) function.
//             filename - a char array containing the name of the rom file that we have opened.
//             ftHandle - handle to the TTL cable that we are communicating over.
// Return:     returns 1 on successful completion, !1 otherwise.


char sendData(int romfile, char *filename, FT_HANDLE ftHandle)
{
        unsigned long filesize;
        unsigned long pageSegCount;          // The amount of pages needed to store the ROM file.
        unsigned long pageRem = 0;           // Portion of a ROM file left over that doesn't fill a full
                                             // page.
        unsigned long numwrit;               // Not required to be used....
        unsigned long padByteCnt = 0;
        unsigned long padSegRem = 0;
        unsigned long padSegSize = (1 * 128);
        unsigned char databuf[PAGESIZE];     // Data read from the ROM file.
unsigned char endianBuf[PAGESIZE];   // Data read from the ROM file reversed to accommodate
                                     // VNC2 endianess
unsigned char receiveBuf[PAGESIZE];  // Data received from the VNC2
unsigned char j = 0;
unsigned char cmd;
unsigned char dataread = 0x00;
unsigned char baudbuf[6];
FT_STATUS ftStatus;
unsigned long i;                     // Index used as the location within the device flash.

struct stat stbuf;
stat(filename, &stbuf);

// FILE SIZE CALCULATIONS
filesize = stbuf.st_size;
pageSegCount = filesize / PAGESIZE;
pageRem = filesize % PAGESIZE;

//echo from the device...
cmd = 0xFF;
ftStatus = FT_Write(ftHandle, &cmd, 1, &numwrit);
numwrit = 0;
ftStatus = FT_Read(ftHandle, &dataread, 1, &numwrit);

cmd = 0xFB;
ftStatus = FT_Write(ftHandle, &cmd, 1, &numwrit);
numwrit = 0;
ftStatus = FT_Read(ftHandle, &dataread, 1, &numwrit);

// Set the baud rate of the chip...
cmd = 0x01;
baudbuf[0] = cmd;
baudbuf[1] = 0x03;            // 1, 2 or 3

ftStatus = FT_Write(ftHandle, baudbuf, 2, &numwrit);

// Allow some time for the baud rate to settle...
Sleep(100);

// Set the baud rate of the TTL cable.
FT_SetBaudRate(ftHandle, 3000000);

numwrit = 0;
ftStatus = FT_Read(ftHandle, &dataread, 1, &numwrit);
if(dataread != 0x02)
{
        printf("The baud rates aren't synced, exiting.\n");
        return -1;
}

//echo from the device...
cmd = 0xFB;
FT_Write(ftHandle, &cmd, 1, &numwrit);
numwrit = 0;
FT_Read(ftHandle, &dataread, 1, &numwrit);

cmd = 0xFF;
FT_Write(ftHandle, &cmd, 1, &numwrit);
numwrit = 0;
FT_Read(ftHandle, &dataread, 1, &numwrit);

printf("Writing File");

// Increment by one to accommodate the fractional page, the fractional page will be memset with
// 0xFF...
```

```
pageSegCount++;

for(i = 0; i < pageSegCount; i++)
{
        unsigned char transdata[2];
        int numread = 0;

        // Memset the data buffer...
        memset(databuf, 0xFF, PAGESIZE);

        numread = read(romfile, databuf, PAGESIZE);

        // Change the data to little? endian for VNC2.
        for (j=0; j < PAGESIZE; j += 2)
        {
                endianBuf[j] = databuf[j+1];
                endianBuf[j+1] = databuf[j];
        }
/*
        WRITE FORMAT
        0x03
        FlashAddressLow
        FlashAddressHigh
        Data block to write.
*/
        cmd = 0x03;
        FT_Write(ftHandle, &cmd, 1, &numwrit);

        transdata[0] = (i & 0xFF);
        transdata[1] = ((i >> 8) & 0xFF);

        // Send the data address....
        FT_Write(ftHandle, transdata, 2, &numwrit);

        // Check for a response from this....
        FT_Read(ftHandle, &dataread, 1 , &numwrit);
        if(dataread != 0x02)
        {
                printf("!");
                return -1;
        }

        // Send the data.
        // Send the data address....
        FT_Write(ftHandle, endianBuf, PAGESIZE, &numwrit);

        // Check for a response again....
        FT_Read(ftHandle, &dataread, 1 , &numwrit);
        if(dataread == 0x02)
        {
                printf(".");
        }
        else
        {
                printf("!");
                return -1;
        }

/*
        READ FORMAT
        0x02
        FlashAddressLow
        FlashAddressHigh
*/
        // Read the data back to make sure that it has been programmed correctly.
```

```
                cmd = 0x02;
                transdata[0] = (i & 0xFF);
                transdata[1] = ((i >> 8) & 0xFF);

                // Send the command
                FT_Write(ftHandle, &cmd, 1, &numwrit);

                // Send the data address....
                FT_Write(ftHandle, transdata, 2, &numwrit);

                // Check for a response from this....
                FT_Read(ftHandle, &dataread, 1 , &numwrit);
                Sleep(10);

                // Make sure that we have 128 bytes to read from the chip!!!
                FT_GetQueueStatus(ftHandle, &numwrit);
                if(numwrit < PAGESIZE)
                {
                        printf("Not enough data to read");
                        return -1;
                }

                // Read 128 bytes back from the chip...
                FT_Read(ftHandle, receiveBuf, PAGESIZE, &numwrit);

// Compare the data read back from the flash with the data sent....
// NOTE: The data that is returned is reversed due to the endianess of the VNC2,
// compare the returned data with the data that we reversed to send to the chip.
                for(j = 0 ; j < PAGESIZE; j++)
                {
                        if(endianBuf[j] != receiveBuf[j])
                        {
                                printf("The data returned is not the same!");
                                return -1;
                        }
                }

        }
        return 1;
}
```

## 7 Summary

In summary VNC2 will allow for existing PCBs to be upgraded to the more powerful device without needing to change the PCB, but will also allow for greater customisation of the firmware and hence a better product and user experience.

New designs, requiring new PCBs may also opt for different device packages.
The 32 pin option will allow for smaller designs where space is a consideration.
The 64 pin package will allow for additional functionality to be added via the extra pin count.

## 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 8797 1330
Fax: +886 (0) 2 8751 9737

|  |  |
|---|---|
| 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 – SPI Clarification

There is a subtle difference in the SPI mode of operation between VNC1L and VNC2 in VNC1L SPI backward compatible mode.

In essence VNC1L waits for the chip select line to go active and then the start bit must be a ‘1’ for the transaction to begin.

In VNC2 backward compatibility mode the chip waits for the chip select to go active and then counts the next 12 clocks as the full transmission. The start bit is effectively a don’t care state.

If the SPI interface is accessed by the external controller (master)  by bit-banging the SPI bits of the interface there will be no new problems.

If the SPI interface is accessed by the external controller using its in-built 8-bit wide SPI interface and simply padding with 0’s either at the front or the end of the message it will not work.

As such the interface must be bit-banged to ensure it works with VNC1L and VNC2 in this mode. The first bit after the chip select is active must be the start bit. No padding is allowed.

## Appendix B – Revision History

| | | |
|---|---|---|
| **Version Preliminary** | First Preliminary version available | 1st Feb 2010 |
| **Version 1.0** | First release Rev 1.0 available | 19th Feb 2010 |
| **Version 1.1** | Added new chapter 6 for loading firmware over UART | 22nd March 2010 |
| **Version 1.2** | Modified chapter 6 | 22nd April 2010 |
| **Version 1.3** | Correction to write opcode in section 6<br>Added appendix A | 13th May 2011 |