



Application Note

AN_118

Migrating Vinculum Designs from VNC1L to VNC2-48L1C

Version 1.4

Issue Date: 29-03-2022

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

Use of FTDI devices in life support and/or safety applications is entirely at the user's risk, and the user agrees to defend, indemnify, and hold harmless FTDI from all damages, claims, suits, or expense resulting from such use.

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>
Copyright © Future Technology Devices International Limited

Table of Contents

| | | |
|------------|---|-----------|
| 1 | Introduction | 2 |
| 2 | VNC1L vs VNC2 | 3 |
| 3 | Reference Schematics | 4 |
| 3.1 | VNC1L Schematics | 4 |
| 3.2 | VNC2-48L1C Equivalent of VNC1L VDIP2 Reference Schematic. | 5 |
| 4 | BOM Change required to convert from VNC1L to VNC2-48L1C..... | 6 |
| 5 | Firmware..... | 7 |
| 6 | Loading Firmware | 8 |
| 7 | Using VPROG-1 to program the firmware | 12 |
| 8 | Summary..... | 13 |
| 9 | Contact Information..... | 14 |
| | Appendix A – SPI Clarification | 15 |
| | Appendix A – References | 16 |
| | Document References | 16 |
| | Acronyms and Abbreviations | 16 |
| | Appendix B – List of Tables & Figures | 17 |
| | List of Tables..... | 17 |
| | List of Figures | 17 |
| | Appendix C – Revision History | 18 |



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 six 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 focus of this document will be on how to migrate from a design using the VNC1L to a VNC2-48L1C device.

2 VNC1L vs VNC2

The key features of the two generations of devices are shown in Table 1. 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 |

Table 1 – VNC1L vs VNC2 - Key Features

3 Reference Schematics

3.1 VNC1L Schematics

This schematic is for the VDIP2 based on the VNC1L.

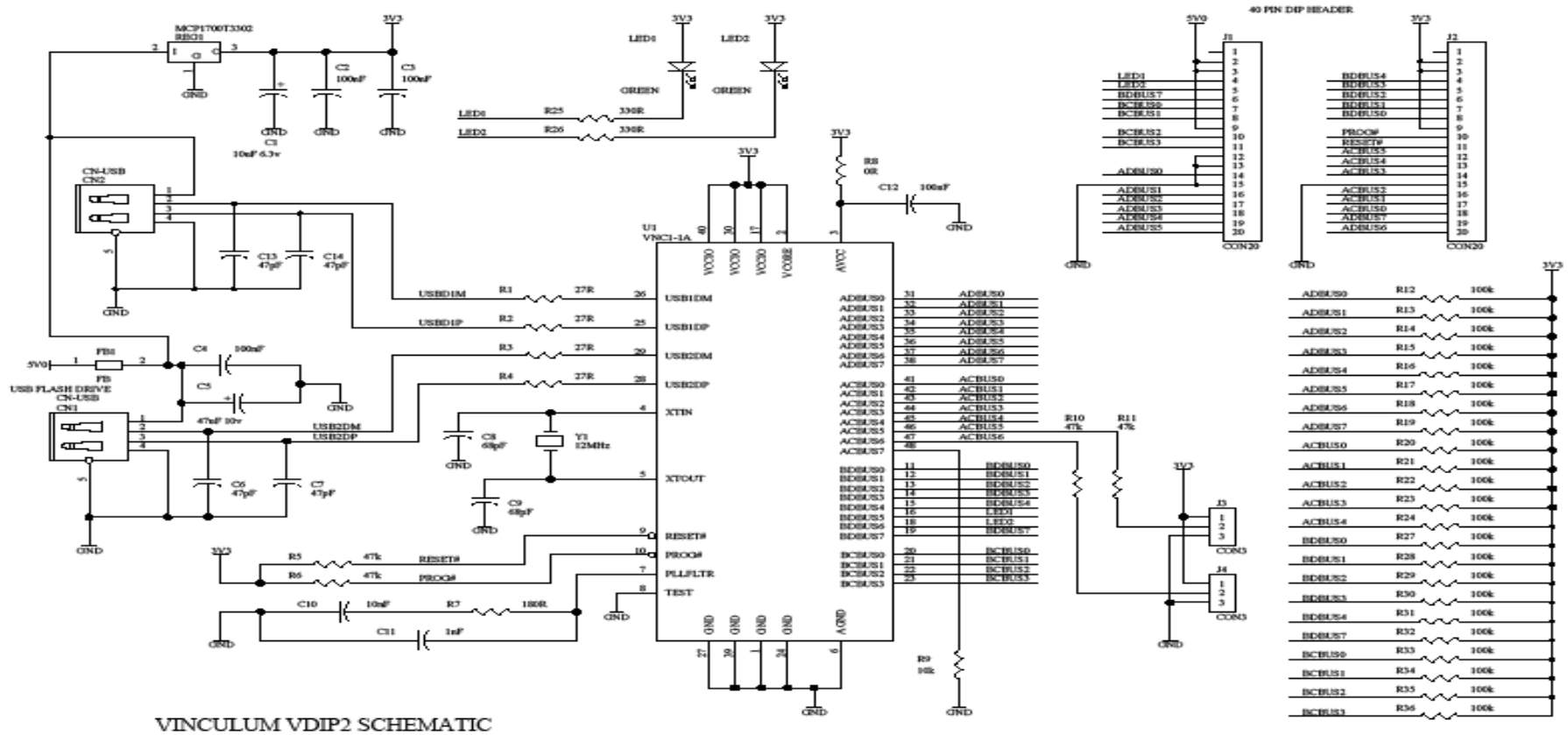


Figure 1 - VNC1L Schematics

3.2 VNC2-48L1C Equivalent of VNC1L VDIP2 Reference Schematic

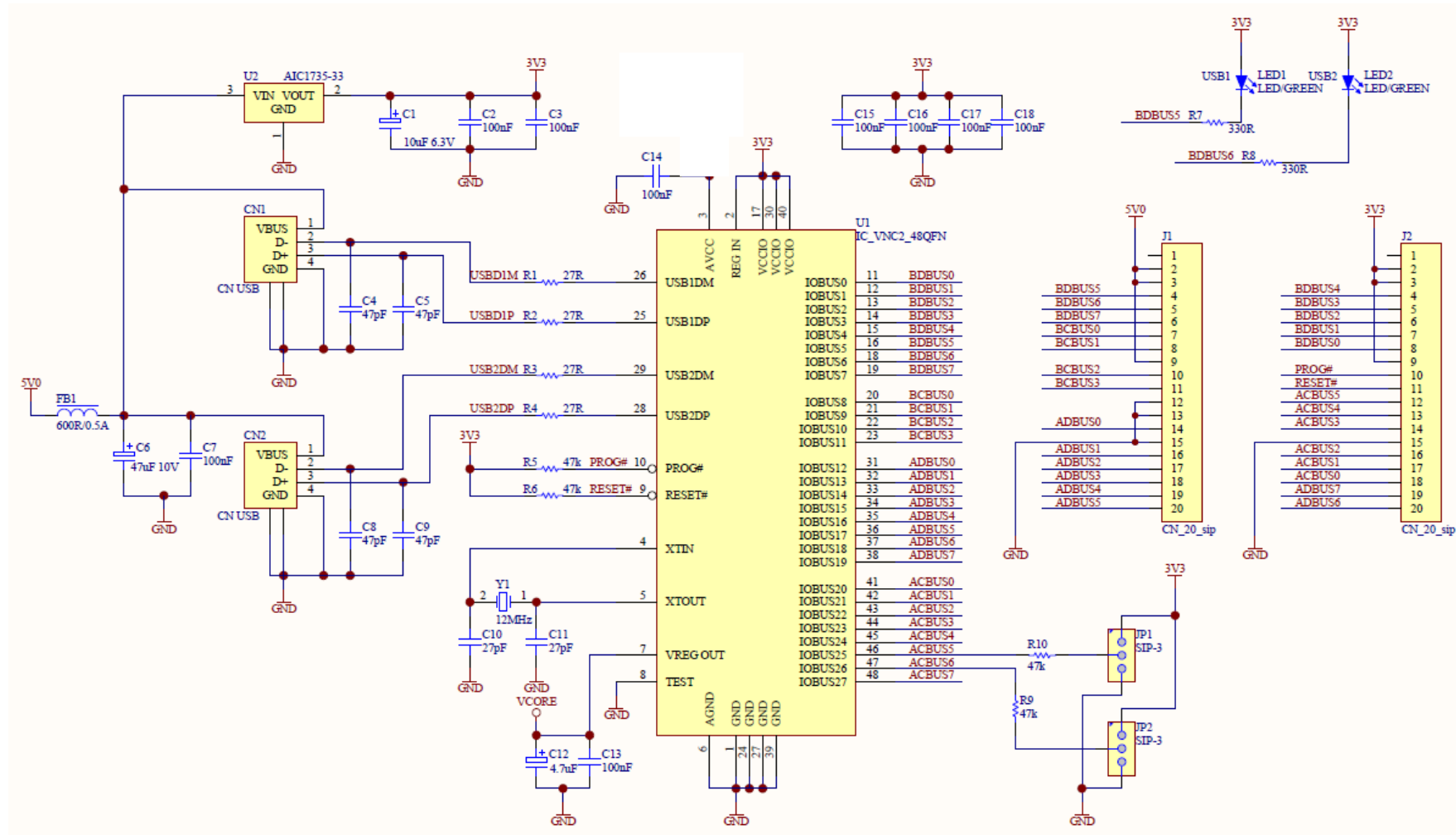


Figure 2 – VNC2-48L1C Equivalent of VNC1L VDIP2 Reference Schematic

4 BOM Change required to convert from VNC1L to VNC2-48L1C

| VNC1L Schematic Component | VNC2-48L1C Schematic Component | Description |
|---------------------------|--|---------------------------------|
| U1 - VNC1L | U1 -VNC2-48L1C | 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 endianness 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 BaudRate (1, 2 or 3) | 0x02 | (115200 Baud == 1, 1 Mbaud == 2, 3 Mbaud == 3) |
| Read Address | 0x02 FlashAddrLow FlashAddrHigh | 0x02 Data to Read | FlashAddrLow/FlashAddrHigh are 128-byte block aligned addresses, so 0x0000=block 0 of 128 bytes, 0x0001=block 1 of 128 bytes, etc., up to 2047 for a total of 256K bytes |
| Write Address | 0x03 FlashAddrLow FlashAddrHigh Data to Write | 0x02 | The 0x02 response will be sent after FlashAddrHigh has been received by the device. FlashAddrLow/FlashAddrHigh are 128-byte block aligned addresses, so 0x0000=block 0 of 128 bytes, 0x0001=block 1 of 128 bytes, etc., up to 2047 for a total of 256K bytes |

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 number of pages needed to store the ROM file.
    unsigned long pageRem = 0;           // Portion of a ROM file left over that does not fill a full
    PAGESIZE = 128;                       // page.
    unsigned long numwrit;                // Not required to be used....
    unsigned long padByteCnt = 0;
    unsigned long padSegRem = 0;
    unsigned long padSegSize = (1 * PAGESIZE);
    unsigned char databuf[PAGESIZE];     // Data read from the ROM file.
    unsigned char endianBuf[PAGESIZE];   // Data read from the ROM file reversed to accommodate
                                         // VNC2 endianness
}
```

```
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 endianness 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 Using VPROG-1 to program the firmware

There is a difference when using VPROG-1 to program the firmware. The original socket VPROG-1-S-LQFP48 was designed for VNC1-48L chip where AVCC – pin 3 was powered from 3.3V. The new VNC2-48L has this pin powered from the output of internal 1.8V voltage regulator. When VNC2-48L chip was used with the original socket, 3.3V was applied to the AVCC pin causing strain on regulator output and PLL supply – however the device was still fully operational. The latest version of the socket adds a 2.0mm jumper added next to the crystal to select VNC1 or VNC2. Please refer to Figure 3 for the jumper setting.

You can also refer to [PCN_FT_013](#) VPROG 48 pin socket revised from rev 1 to rev 2.

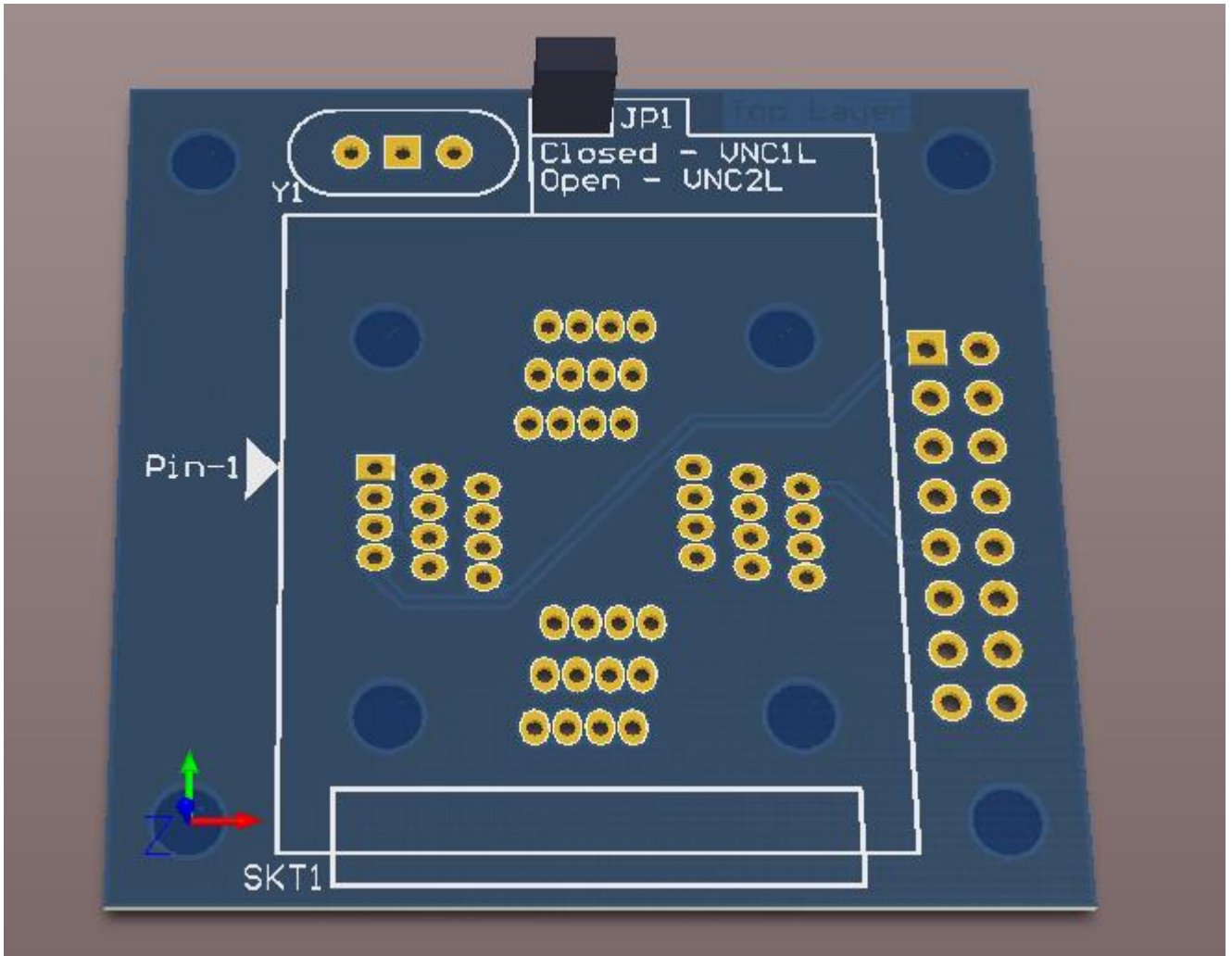


Figure 3 - Jumper Settings

8 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.

9 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

Branch Office – Tigard, Oregon, USA

Future Technology Devices International Limited (USA)
7130 SW Fir Loop
Tigard, OR 97223-8160
USA
Tel: +1 (503) 547 0988
Fax: +1 (503) 547 0987

E-mail (Sales) us.sales@ftdichip.com
E-mail (Support) us.support@ftdichip.com
E-mail (General Enquiries) us.admin@ftdichip.com

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 8791 3576

E-mail (Sales) tw.sales1@ftdichip.com
E-mail (Support) tw.support1@ftdichip.com
E-mail (General Enquiries) tw.admin1@ftdichip.com

Branch Office – Shanghai, China

Future Technology Devices International Limited (China)
Room 1103, No. 666 West Huaihai Road,
Shanghai, 200052
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.admin@ftdichip.com

Web Site

<http://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.

System and equipment manufacturers and designers are responsible to ensure that their systems, and any Future Technology Devices International Ltd (FTDI) devices incorporated in their systems, meet all applicable safety, regulatory and system-level performance requirements. All application-related information in this document (including application descriptions, suggested FTDI devices and other materials) is provided for reference only. While FTDI has taken care to assure it is accurate, this information is subject to customer confirmation, and FTDI disclaims all liability for system designs and for any applications assistance provided by FTDI. Use of FTDI devices in life support and/or safety applications is entirely at the user's risk, and the user agrees to defend, indemnify, and hold harmless FTDI from any and all damages, claims, suits, or expense resulting from such use. This document is subject to change without notice. No freedom to use patents or other intellectual property rights is implied by the publication of this document. Neither the whole nor any part of the information contained in, or the product described in this document, may be adapted, or reproduced in any material or electronic form without the prior written consent of the copyright holder. Future Technology Devices International Ltd, Unit 1, 2 Seaward Place, Centurion Business Park, Glasgow G41 1HH, United Kingdom. Scotland Registered Company Number: SC136640

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.

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 do not 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 A – References

Document References

[PCN_FT_013](#)

Acronyms and Abbreviations

| Terms | Description |
|-------|---|
| FIFO | First In First Out |
| LQFP | Low Profile Quad Flat Package |
| QFN | Quad Flat No Lead Package |
| ROM | Read Only Memory |
| SPI | Serial Peripheral Interface |
| USB | Universal Serial Bus |
| UART | Universal Asynchronous Receiver Transmitter |



Appendix B – List of Tables & Figures

List of Tables

Table 1 – VNC1L vs VNC2 - Key Features..... 3

List of Figures

Figure 1 - VNC1L Schematics4
Figure 2 – VNC2-48L1C Equivalent of VNC1L VDIP2 Reference Schematic.....5
Figure 3 - Jumper Settings..... 12

Appendix C – Revision History

Document Title: AN_118 Migrating Vinculum Designs from VNC1L to VNC2-48L1C
Document Reference No.: FT_000159
Clearance No.: FTDI# 139
Product Page: <http://www.ftdichip.com/FTProducts.htm>
Document Feedback: [Send Feedback](#)

| Revision | Changes | Date |
|----------|--|------------|
| 1.0 | Initial Release | 19-02-2010 |
| 1.1 | Added new chapter 6 for loading firmware over UART | 22-03-2010 |
| 1.2 | Updated Section 6 | 22-04-2010 |
| 1.3 | Correction to write opcode in section 6; Added Appendix A | 13-05-2011 |
| 1.4 | Updated references to VNC2-48L1C (rev C of device) updated VPROG-1 setting for VNC1L and VNC2 | 29-03-2022 |