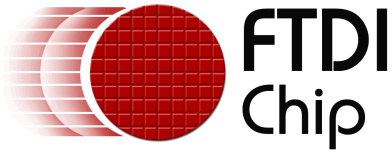




# Future Technology Devices International Ltd.

# ANVNC1L-01 Vinculum VNC1L Bootloader

# Table of Contents

# 1 Vinculum VNC1L Bootloader

This document describes the commands available for communicating with the VNC1L device's bootloader and how to use them.

A chapter is also included on how to interface to the Vinculum VNC1L device UART for reprogramming the Flash memory using the bootloader.

# 2 VNC1L Bootloader Commands

## 2.1 Set Data Variables

**Command Bytes**
*0x01,*
*TimerLow,*
*TimerHigh,*
*Baud1,*
*Baud2,*
*Baud3*

**Response Bytes**
*0x02*

**Remarks**
This command allows the Flash program time and Baud rate to be specified. A value of 6000 (0x1770) should be used for the program time (*TimerLow* = 0x70 and *TimerHigh* = 0x17 ). The Baud rate should be specified according to the table below. A value of 1MBaud gives fast and reliable Flash programming (*Baud1* = 0x03, *Baud2* = 0x00, *Baud3* = 0x00). The bootloader defaults to a Baud rate of 115200.

**Baud Rate Table**

| Baud Rate | Baud1 | Baud2 | Baud3 |
|---|---|---|---|
| 300 | 0x10 | 0x27 | 0x00 |
| 600 | 0x88 | 0x13 | 0x00 |
| 1200 | 0xC4 | 0x09 | 0x00 |
| 2400 | 0xE2 | 0x04 | 0x00 |
| 4800 | 0x71 | 0x02 | 0x00 |
| 9600 | 0x38 | 0x41 | 0x00 |
| 19200 | 0x9C | 0x80 | 0x00 |
| 38400 | 0x4E | 0xC0 | 0x00 |
| 57600 | 0x34 | 0xC0 | 0x00 |
| **115200** | **0x1A** | **0x00** | **0x00** |
| 230400 | 0x0D | 0x00 | 0x00 |
| 460800 | 0x06 | 0x40 | 0x00 |
| 921600 | 0x03 | 0x80 | 0x00 |
| 1000000 | 0x03 | 0x00 | 0x00 |
| 1500000 | 0x02 | 0x00 | 0x00 |
| 2000000 | 0x01 | 0x00 | 0x00 |
| 3000000 | 0x00 | 0x00 | 0x00 |

**Note**: Baud rates in gold cannot be used for Flash programming due to the time required to write to the Flash memory. 1000000 Baud (1MBaud) is the fastest Baud rate that should be used for Flash programming.

## 2.2 Read Flash Block

**Command Bytes**
*0x02,*
*FlashAddressLow,*
*FlashAddressHigh,*
*Count*

**Response Bytes**
*0x02*
*Data Block Read*

**Remarks**
This command reads *Count* bytes from the Flash memory starting at the address given by the Flash address word which can be calculated as follows:

Flash Address = ((*FlashAddressHigh* **shifted left** 8) **OR** 0xFF) **AND** *FlashAddressLow*

For example, to read a Flash address of 0xABCD the values to be sent in the command would be *FlashAddressHigh* = 0xAB and *FlashAddressLow* = 0xCD.

A *Count* value of 128 bytes should be used.

## 2.3 Write Flash Block

**Command Bytes**
*0x03,*
*FlashAddressLow,*
*FlashAddressHigh,*
*Count* (Followed by a response of 0x02)
*Data Block to Write*

**Response Bytes**
*0x02* after command (after *Count* sent)
*0x02* after data block sent

**Remarks**
This command writes *Count* bytes to the Flash memory starting at the address given by the Flash address word which can be calculated as follows:

Flash Address = ((*FlashAddressHigh* **shifted left** 8) **OR** 0xFF) **AND** *FlashAddressLow*

For example, to write a Flash address of 0xABCD the values to be sent in the command would be *FlashAddressHigh* = 0xAB and *FlashAddressLow* = 0xCD.

A *Count* value of 128 bytes should be used.

## 2.4 Echo

**Command Bytes**
*0xFF **OR** 0xFA*

**Response Bytes**
*0xFF **OR** 0xFA*

**Remarks**
This command is echoed by the VNC1L bootloader. If 0xFF is sent, it returns 0xFF. Similarly, if 0xFA is sent then 0xFA is returned. These commands can be used to synchronise to the bootloader before programming the Flash.

# 3 VNC1L Bootloader Pseudo Code

A simplified VNC1L programming application outline is provided below, not taking account of error recovery.

A VNC1L programming application should perform the following steps:

- Initialise programmer UART with the following characteristics:
  Baud rate: 115200
  Data bits: 8
  Stop bits: 1
  Parity: None
  Handshaking: RTS/CTS

- Synchronise the programmer and VNC1L
  **LOOP**
  Send an Echo 6 command (0xFF)
  Wait for a response
  Read bytes available
  **UNTIL** last byte received is the Echo 6 response (0xFF)
  Send alternative Echo 6 command (0xFA)
  **LOOP**
  Check for data available
  If data available, read it. Otherwise wait for data
  **UNTIL** Last byte received is the Echo 6 response (0xFA)
  If last byte received was the alternative Echo 6 response, then synchronisation complete
  If last byte received was not the alternative Echo 6 response, repeat synchronisation

- Set desired VNC1L Baud rate and timer value of 6000 using the Set Data Variables 3 command

- Set programmer UART to the same Baud rate as the VNC1L

- Program flash from file
  **LOOP**
  Read a block of data from the firmware ROM file
  Send Write Flash Block 5 command for *Count* bytes (128 bytes required)
  Wait for acknowledgement (0x02)
  Write the block of data to the VNC1L
  **UNTIL** the complete firmware file has been written to the Flash memory

- Verify the Flash contents
  **LOOP**
  Read a block of data from the firmware ROM file
  Send the Read Flash Block 4 command requesting *Count* bytes (128 bytes required)
  Wait for acknowledgement (0x02)
  Wait for *Count* bytes to be returned
  Compare the data from the ROM file and the Flash memory
  If they do not match, flag an error
  **UNTIL** the entire firmware file has been compared to the entire Flash memory **OR** an error was found

- Flash programming complete

# 4 VNC1L Bootloader Hardware Interface

The Vinculum VNC1L bootloader uses the UART interface to load new firmware into the Vinculum Flash memory. The UART can be interfaced to in the following ways:

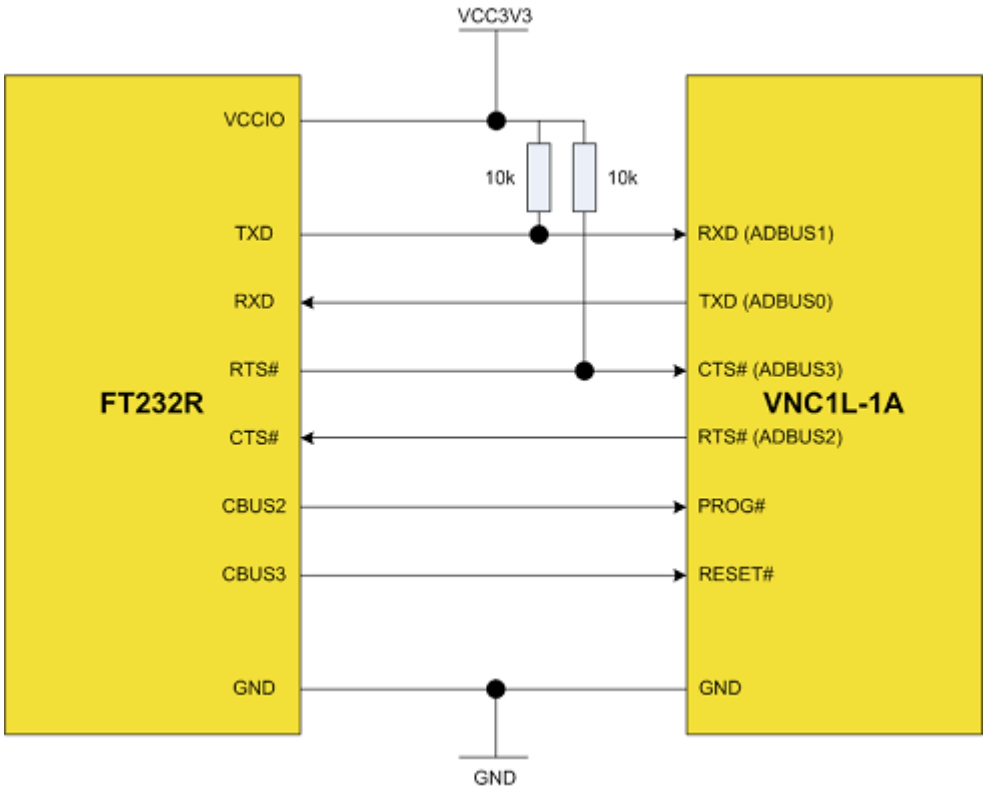
## 4.1 VNC1L Programming Over USB

The VNC1L device can be programmed from a PC USB port using an FTDI USB-serial converter device such as the FT232R, FT232BM or FT8U232AM. This is the fastest way to reprogram the VNC1L Flash memory as data can be transferred to the Vinculum device at up to 1MBaud.

The MM232R and UM232R provide a convenient way of interfacing the FT232R to a Vinculum VNC1L design or the VDIP1 evaluation module and will allow for programmatically controlling the state of the PROG# and RESET# pins.

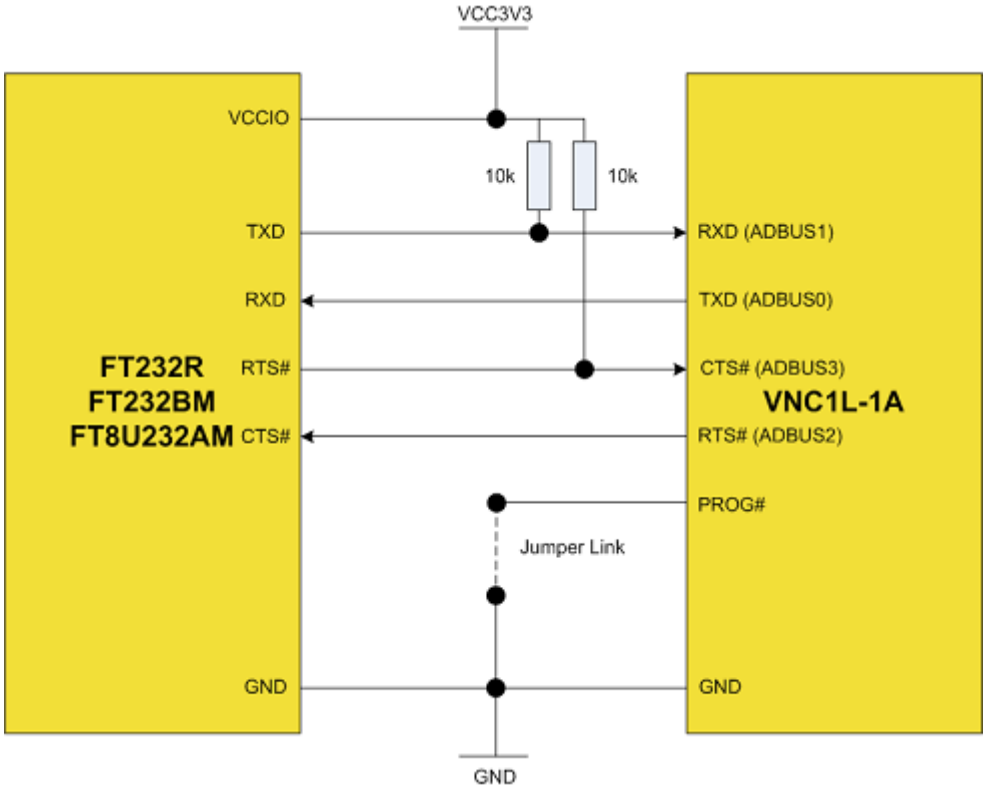
FTDI's standard Vinculum evaluation kits (VDRIVE1, VMUSIC1, VF2F) all have a 6-pin header which can be interfaced to the FT232R-based TTL-232R-3V3 cable to allow firmware reflashing and require a jumper to be set to enable the bootloader at power up.

The required connections between the VNC1L and an FT232R device for programmatically controlling the PROG# and RESET# pins is shown below:




Note that CBUS Bit Bang mode must be enabled in the FT232R EEPROM for CBUS2 and CBUS3 for this to function (see AN232R-01 Bit Bang Modes for the FT232R and FT245R for details of CBUS Bit Bang mode). To enable the bootloader, the PROG# pin must be driven low and the VNC1L must then be reset by driving the RESET# pin low then high. Run mode can be enabled by driving the PROG# pin high and then resetting the VNC1L by driving the RESET# pin low then high.

If using an FT232BM, FT8U232AM or an FT232R without using CBUS bit bang on CBUS2 and CBUS3 to programmatically control the PROG# and RESET# pins, a jumpered configuration is required as follows:




Note that for the bootloader to be active the jumper must be fitted before powering the VNC1L. If the jumper is not fitted, the VNC1L will power up in run mode with the bootloader inactive.

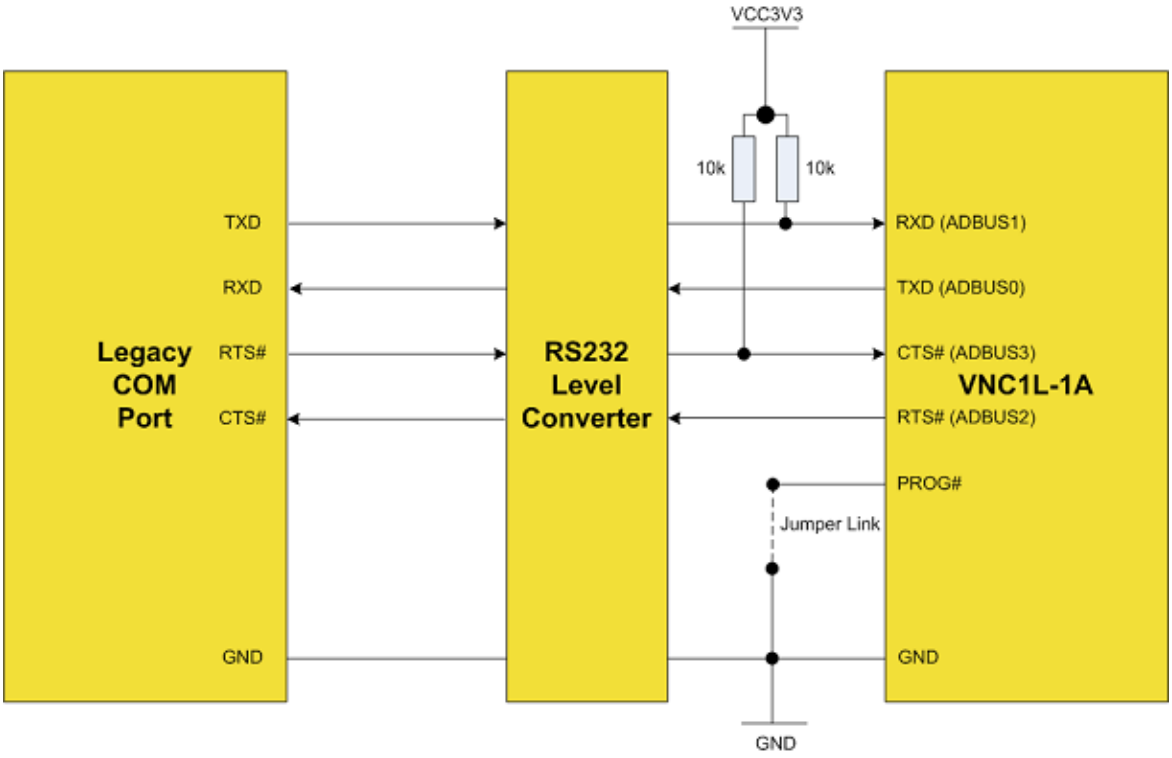
Both of the configurations above are supported by the VPROG application which can be downloaded from the Vinculum web site.

## 4.2 VNC1L Programming From a COM Port

It is possible to program firmware into the VNC1L using a legacy PC COM port, provided that the COM port to be used is capable of at least 115200 Baud. This is required because the VNC1L bootloader defaults to 115200 Baud. The Baud rate may be changed by issuing a Set Data Variables 3 command to the VNC1L after establishing communication at 115200 Baud.

Programming the VNC1L device using a legacy COM port requires the use of an RS232 level converter on the UART of the VNC1L to convert between the RS232 signal level and the VNC1L 3.3V level (5V safe). In addition, a jumper has to be fitted to the PROG# pin of the VNC1L to allow for switching between program mode and run mode. Note that for the bootloader to be active the jumper must be fitted before powering the VNC1L. If the jumper is not fitted, the VNC1L will power up in run mode with the bootloader inactive.

A sample block diagram of the hardware requirements for programming from a legacy COM port is shown below:



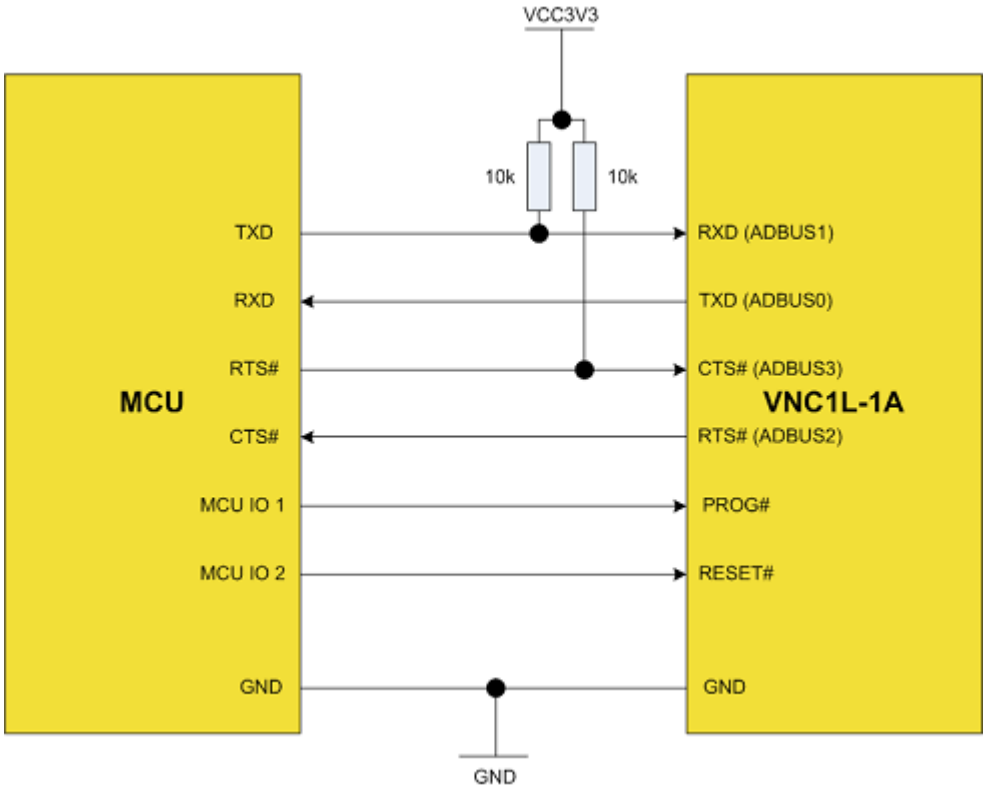

At present, programming from a legacy COM port will require software development to create a Flash programming application.

## 4.3 VNC1L Programming From an MCU

VNC1L firmware can also be updated via a microcontroller with a UART. The microcontroller must be capable of at least 115200 Baud. This is required because the VNC1L bootloader defaults to 115200 Baud. The Baud rate may be changed by issuing a Set Data Variables 3 command to the VNC1L after establishing communication at 115200 Baud.

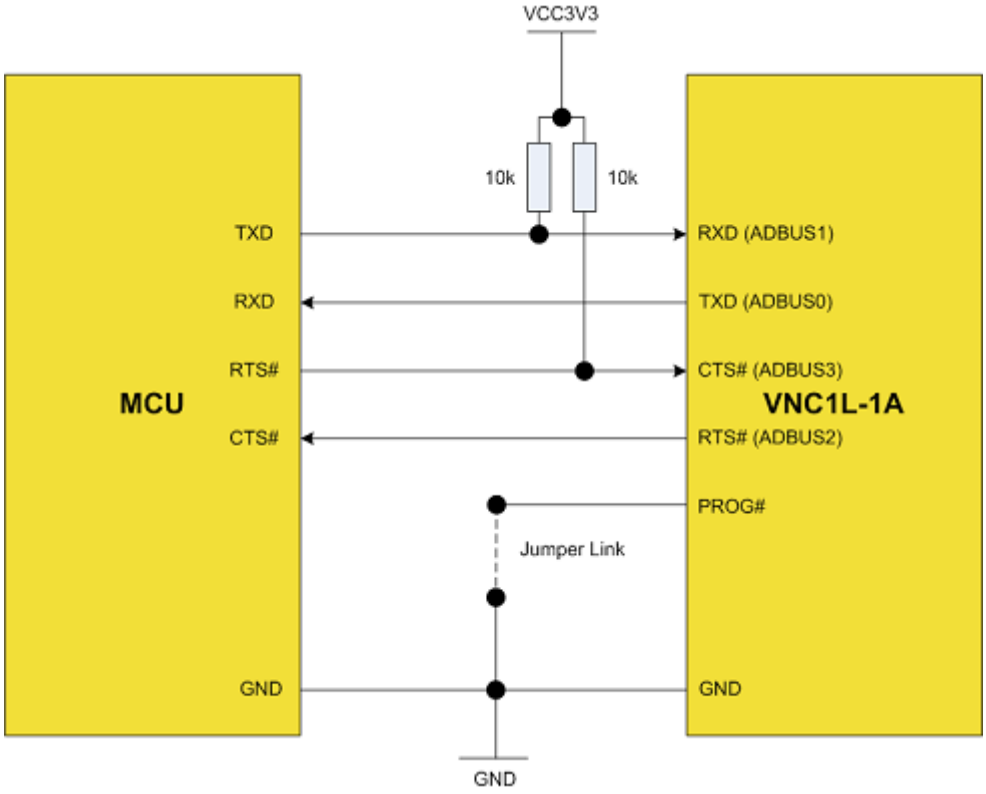
Also, the VNC1L UART requires the use of RTS/CTS flow control. If the MCU UART does not automatically handle hardware handshaking, this must be implemented in the MCU firmware.

As with the FT232R programming the VNC1L over USB 9, it is possible to use two of the available MCU IOs to control the PROG# and RESET# pins from the MCU firmware. The following diagram illustrates this arrangement:




To enable the bootloader, the PROG# pin must be driven low and the VNC1L must then be reset by driving the RESET# pin low then high. Run mode can be enabled by driving the PROG# pin high and then resetting the VNC1L by driving the RESET# pin low then high.

In a situation where IOs from the MCU are not being used to control the PROG# and RESET# pins, a jumper must be used to manually enable program or run mode. A sample block diagram of the hardware requirements for programming from an MCU with a jumper controlling PROG# is shown below:




Note that for the bootloader to be active the jumper must be fitted before powering the VNC1L. If the jumper is not fitted, the VNC1L will power up in run mode with the bootloader inactive.

Programming the VNC1L from an MCU will require firmware development to allow Flash programming.

# 5 Revision History

| Version | Release Date | Comments |
|---|---|---|
| 1.0 | September 2006 | Initial release. |
| 1.1 | September 2006 | Typo correction and synchronisation reorder. |
| 1.2 | October 2006 | Added statement for pseudo code not including error recovery.<br>Added reference to AN232R-01 for CBUS Bit Bang mode.<br>Added #s to RTS and CTS signals in diagrams. |
| 1.3 | November 2006 | Changed 128 byte reocmmendation for Write Flash Block and Read Flash Block to a requirement. |

# 6 Contact

**Head Office - Glasgow, UK**
Future Technology Devices International Limited
373 Scotland Street
Glasgow
G5 8QB
United Kingdom
Tel: +44 (0) 141 429 2777
Fax: +44 (0) 141 429 2758

E-Mail (Sales): sales1@ftdichip.com
E-Mail (Support): support2@ftdichip.com
E-Mail (General Enquiries): admin1@ftdichip.com
Web Site URL: http://www.ftdichip.com
Web Shop URL: http://apple.clickandbuild.com/cnb/shop/ftdichip

**Branch Office - Taiwan**
Future Technology Devices International Limited (Taiwan)
4F, No 16-1, Sec. 6 Mincyuan East Road
Neihu District
Taipei 114
Taiwan
ROC
Tel: +886 2 8791 3570
Fax: +886 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)
5285 NE Elam Young Parkway
Suite B800
Hillsboro, OR 97124-6499
USA
Tel: +1 (503) 547-0988
Fax: +1 (503) 547-0987

E-Mail (Sales): us.sales@ftdichip.com
E-Mail (Support): support2@ftdichip.com
E-Mail (General Enquiries): admin1@ftdichip.com
Web Site URL: http://www.ftdichip.com

**Agents and Sales Representatives**
Please visit the Sales Network page of the FTDI web site for the contact details of our distributor(s) in your country.

# 7 Disclaimer



This product and its documentation are supplied on an as-is basis and no warranty as to their suitability for any particular purpose is either made or implied.
Future Technology Devices International Ltd. will not accept any claim for damages howsoever arising as a result of use or failure of this product. Your statutory rights are not affected.

This product or any variant of it is not intended for use in any medical appliance, device or system in which the failure of the product might reasonably be expected to result in personal injury.

This document provides preliminary information that may be subject to change without notice.