Application Note

AN_138

Vinculum-II Debug Interface

Version 2.0

Issue Date: 2011-10-14

This document provides step by step guidelines on how to use the Debug Interface of the Vinculum-II (VNC2) device, how to carry out debug operations using the VNC2 IDE and details the debugger circuit.

Future Technology Devices International Limited (FTDI)
Unit 1, 2 Seaward Place, Glasgow G41 1HH, United Kingdom
Tel.: +44 (0) 141 429 2777 Fax: + 44 (0) 141 429 2758
Web Site: http://ftdichip.com

Table of Contents

1 Overview

This document provides step by step instructions for using the Vinculum-II VNC2 Integrated Development Environment (IDE) to debug applications using VNC2 family of USB embedded controllers. It also details the hardware needed to interface to the VNC2 debug port.

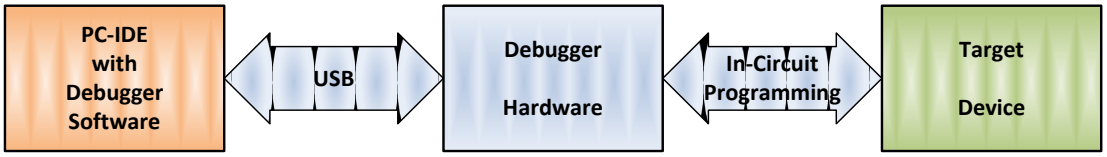

Figure 1.1 Typical Debugger Interface

Figure 1.1 shows a typical development/debugging environment that consists of IDE software running on a host PC with the debugger interface that communicates with debugger hardware on the USB bus. Debug hardware translates the data into serial format which is used by a target device to execute debug commands. This kind of debug implementation is commonly known as serial In-Circuit Programming.

VNC2 carries out debug operation through a single Debugger Interface pin (DEBUG_IF). At power up, by default, this pin is available on IOBUS0. Although this pin can be relocated to a different GPIO pin using the IOMux feature, it is recommended that the debugger interface pin is not relocated from IOBUS0.

Vinculum II IDE provides a debugger interface engine which provides all the serial interface debug commands and communicates with the target device using a USB to UART IC such as the FT232R.

There are two options for customers to implement a debugger interface from the host PC to a VNC2:

1. Use the VNC2 Debugger/Programmer Module from FTDI which can be connected to a compatible debug header on a customer target board.
2. Implement the debug module circuitry on the target board with a USB interface.

This document first describes a basic VNC2 circuit containing the minimum components to utilize the debugger. Next, a discussion of the debugger interface circuit and connector is provided to illustrate the available debug choices. Finally, the debug operations within the VNC2 are shown.

With a basic understanding of the debug hardware, and using the VNC2 IDE debug features, this document illustrates the debug operations using a V2-Eval board (which already has the debug hardware implemented on the board). This document describes a step by step process of using the debugger interface, adding watch variables, applying break points and stepping through demo code.

2 Hardware Requirements

There are two options in which debug interface can be implemented on target device

1. Using the FTDI VNC2 Debugger/Programmer Module
2. Designing on board Debugger Hardware Interface (as on the FTDI V2-EVAL Board)

2.1 Interfacing to a VNC2 Debugger/Programmer Module

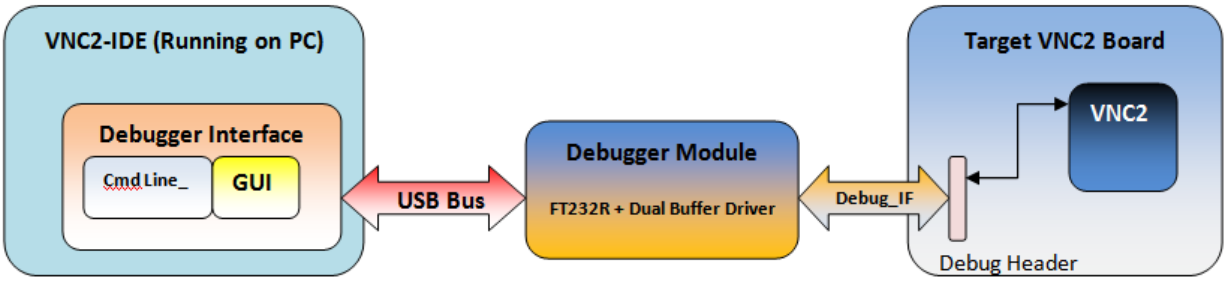

Figure 2.1 VNC2 Debugger Interface

When using the FTDI VNC2 Debugger/Programmer Module as an external accessory, the target circuit requires a male debug header which the debugger/programmer module plugs onto. The following signals must be connected to the VNC2 and made available on the debug header.

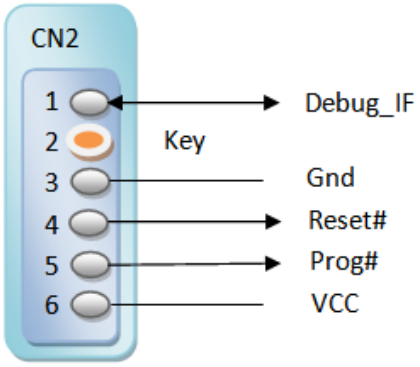

Figure 2.2 VNC2 Debugger/Programmer Module Connector

The VNC2 Debugger/Programmer Module uses a 2mm right-angle single-row female header. Pin 2 is blocked as a polarity keyway. The mating connector on the target circuit must have pin 2 removed. A suggested mating connector is the Sullins NRPN061PARN-RC. This is a standard part number that does not have the pin 2 keyway. Pin 2 can be removed or cut from the connector. Customized connectors are available through many manufacturers.

The length of the pins should be approximately 4mm for a solid connection.

Pin	Signal Name	Type	Description
1	DEBUG_IF	Bi-Directional	Serial debug data interface pin – half-duplex
2	N/A	N/A	Keyway for proper attachment of VNC2 Debugger Module
3	GROUND	Power	Signal ground
4	RESET#	Input	VNC2 Reset – Active LOW
5	PROG#	Input	VNC2 Program – Active LOW For legacy programming designs and recovery of the debug interface
6	VCC	Power	+5V DC from VNC2 Debugger/Programmer Module (250mA max) See notes below

Table 2.1 VNC2 Debugger/Programmer Module interface on Target Hardware

NOTES:

1. The +5V DC available on VCC through the VNC2 Debugger/Programmer Module is limited to approximately 250mA. If the target circuit is powered through the VNC2 Debugger Module, care must be taken to ensure the target circuit does not draw more than this limit. USB Host applications commonly require up to 500mA per port; an external power supply is recommended in these situations.
2. Care should also be taken to prevent any current from the target circuit from being applied to the VNC2 Debugger/Programmer Module. A Schottky diode in series with the Anode connected to the VNC2 Debugger/Programmer Module VCC will prevent this situation.
3. RESET# and PROG# are outputs from the VNC2 Debugger/Programmer Module.

2.2 Designing An On-Board Debugger Interface

A second debug option is to incorporate the debug module equivalent circuitry directly onto the customer application board. The debug circuitry converts the single pin debug interface into a USB type B interface which enables a user to connect the debug hardware to a PC running the VNC2 Toolchain IDE. The conversion from the serial debug interface is accomplished through a USB to UART IC such as the FT232R.

The following components are used to implement the debug circuitry. (A further detailed description of each component is outlined in the following sections).

1. Vinculum II (VNC2)
2. Dual buffer driver chip to convert the single debug signal into UART interface with separate transmit and receive signals
3. FT232R converter IC to convert USB to serial UART
4. USB type B socket connector for connection with the host PC running the VNC2 Toolchain IDE

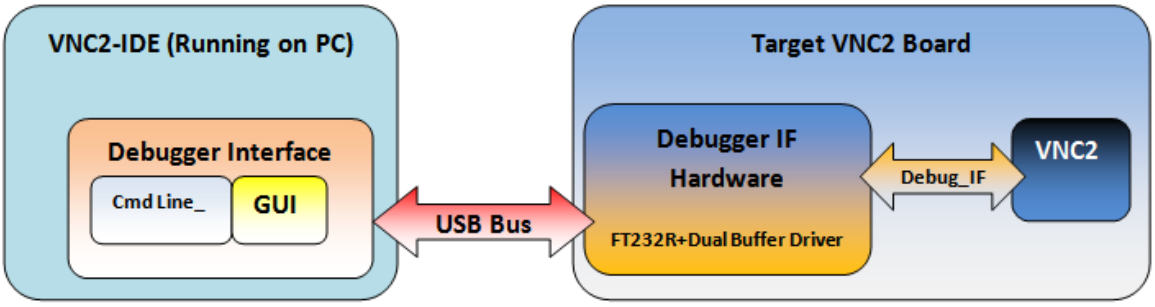

Figure 2.3 Debugger Interface on Target VNC2 Board

2.2.1 VNC2 DEBUG_IF Assignment

The VNC2 chip uses the DEBUG_IF signal to carry out debug operation on the chip. The DEBUG_IF signal is connected to an internal debug engine that decodes serial debug commands and executes all debug operations.

The default IOMux assignment for DEBUG_IF is IOBUS0. This assignment can be changed through the IOMux; however, FTDI recommends leaving the default assignment in place.

In the event that the debug interface is moved to a different IO pin and the user program does not properly initialize the debug interface, the VNC2 may appear to be non-responsive. It is possible to recover from this state; however the target circuit must provide a connection to IOMUX0. In addition to IOBUS0, the recovery process requires both PROG# and RESET#:

1. Drive PROG# and RESET# low – VNC2 is in reset and ready for recovery
2. Drive RESET# high – VNC2 comes out of reset and activates IOBUS0 as the DEBUG_IF signal
3. Proceed with flashing the VNC2 with a new user application on IOBUS0
4. Drive RESET# low – VNC2 is in reset
5. Drive PROG# high – VNC2 is in reset and ready for normal operation
6. Drive RESET# high – VNC2 is out of reset and under normal operation
   The DEBUG_IF signal is assigned through the program that was flashed in step 3

2.2.2 Debugger Interface Hardware

An FT232R USB-Serial chip and dual buffer is used to create the half-duplex asynchronous serial signals required by the DEBUG_IF signal. Commands are sent through the VNC2 Toolchain IDE through this interface and decoded and executed through the debug engine on the VNC2.

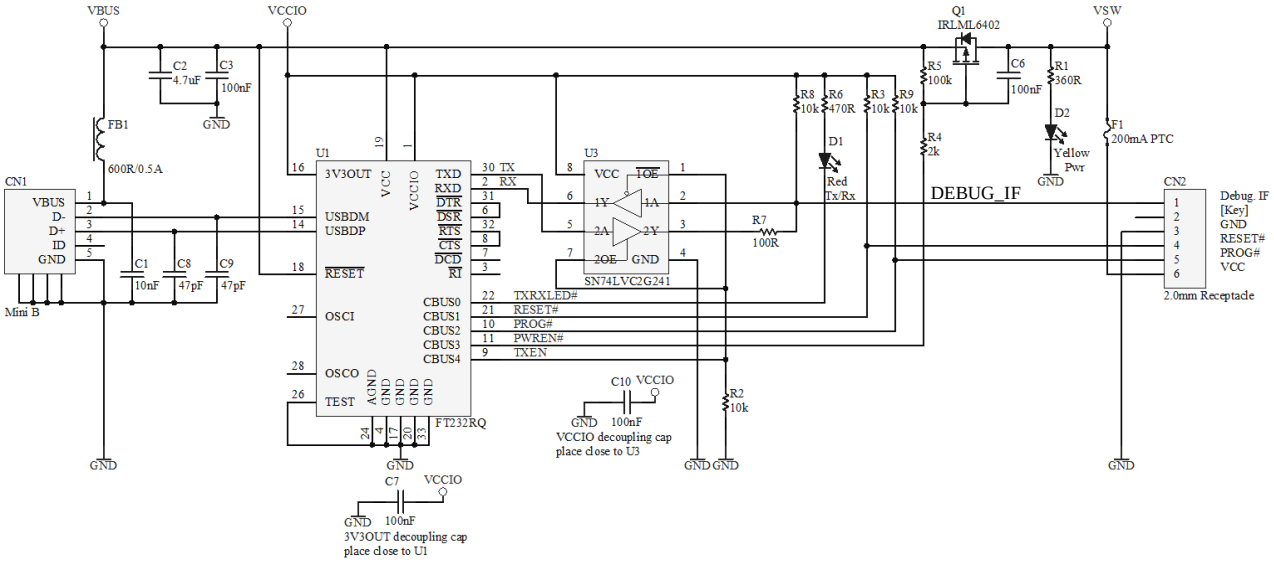

Figure 2.4 VNC2 Debugger Module Schematic

This circuit starts on the left with a standard USB mini-B connector. VBUS is filtered through the C1 (10nF) and FB1 (~600ohms @ 100MHz, 0.5A rating). C8 and C9 (47pF) provide filtering for the USB signals. 3V3OUT is decoupled through C7, and provides the +3.3V DC power input to VCCIO.

On the peripheral side of the FT232R, the UART flow control signals are in a loop-back configuration (RTS# to CTS# and DTR# to DSR#). TXD and RXD are connected to U3 to provide the half-duplex single-wire DEBUG_IF signal which operates at 1Mbps. TXEN controls the direction of the transceiver. Other logic-level buffers may be used in place of the dual buffer shown for U3. Timing of the TXEN direction control is not critical with most logic families. R7 provides protection in the event both the VNC2 and debugger circuit are both transmitting at the same time.

PWREN# controls a P-channel FET. After the debug circuit is recognized by the PC USB host and operating system (this is called "enumeration"), PWREN# will be driven low and power is applied to the VCC pin of the debugger interface. VCC is at +5.0V DC; however, the data and control signals are referenced to +3.3V DC.

PROG# and RESET# are connected to the pins of the same name on the VNC2. These signals provide a means of controlling the VNC2 when flashing firmware and a means to recover the debug interface in the event it is reassigned or disabled by the application.

The FT232R CBUS signals must be configured through the internal EEPROM:

CBUS Pin	EEPROM Setting	Signal Name	Description
0	TX & RXLED#	TXRXLED#	Flickers LED when data is being transmitted or received
1	I/O MODE	RESET#	Connects to VNC2 RESET#
2	I/O MODE	PROG#	Connects to VNC2 PROG#
3	PWRON#	PWREN#	Switches VCC power to the debug connector after enumeration
4	TXDEN	TXEN	Controls direction of the half-duplex buffer 1 = Transmit, 0 = Receive
N/A	Power Type	N/A	"Bus Powered" if configured similar to Figure 2.4
N/A	Max Bus Power	N/A	300mA – the FT232R will request 300mA from the PC USB host port.
N/A	All Others	N/A	Default settings are acceptable

Table 2.2 FT232R EEPROM & CBUS Settings

The FTDI FT_Prog utility is used to perform the FT232R EEPROM programming.

The debugger interface hardware is a critical portion of any embedded design. Designers must ensure that this circuit is free from noise sources such as electrical noise, EMI, high power switching, relays, motor controls or RF noise in the designed system. In addition, it is common practice to provide ESD protection on all circuits that may be connected to the "outside world", including the debug interface.

A discussion of implementing isolation and protection between USB and serial interface is beyond the scope of this application note.

2.2.3 V2-EVAL Board Debugger Interface

The V2-EVAL board is an "all-in-one" development platform for the VNC2. The V2-EVAL board provides:

- Headers to connect a VNC2 daughterboard
  Daughterboards are available for 32-, 48- and 64-pin VNC2 devices
- External power supply connector
- USB type A sockets for each VNC2 USB port
- Generous prototyping area
- All I/O pins made available on header pins and grouped by port name and function
- User programmable LEDs and pushbutton switches
- FTDI FT4232H to provide the debug interface, UART connection, Synchronous Serial connection (MPSSE) and "spy" port to monitor certain signal functions

Full details can be found through the V2-EVAL Datasheet.

The debug interface is made available through Port C of the FT4232H. No additional modules or circuitry is necessary.

2.2.4 V2DIP and Vinco Debugger Interfaces

Aside from the V2-EVAL and customer-created circuitry, FTDI also sells a series of V2DIP modules. These modules are on a Dual-Inline Packet (DIP), intended for direct insertion to a breadboard or PCB with similar mounting options. Each module is provided with a debugger connector as shown in Figure 2.2 to accept the VNC2 Debugger/Programmer Module.

Another popular development platform is the FTDI Vinco board. This has an Arduino™-like form-factor and can accept existing Arduino shields. As with the V2DIP modules, a debugger connector is included for connection to a VNC2 Debugger/Programmer Module.

3 Programming the Target Device

3.1 Using the VNC2 IDE

This section describes how to use the Debugger functions from the Debug menu of the IDE. It describes loading, running and debugging code in the VNC2 device.

Once a user application is written and successfully compiled, a programming file is created with a ".ROM" extension. It is this ROM file that is programmed into the flash memory of the VNC2. The debug section of the IDE is used to perform the flash operation.

The .ROM file which can be "Flashed" to the target device from the Debug menu as shown below. First, be sure a Debugger Interface is active. See section 4.1 to select the active debugger interface.

Next, from VNC2 IDE "Debug" menu, click on "Flash" icon.

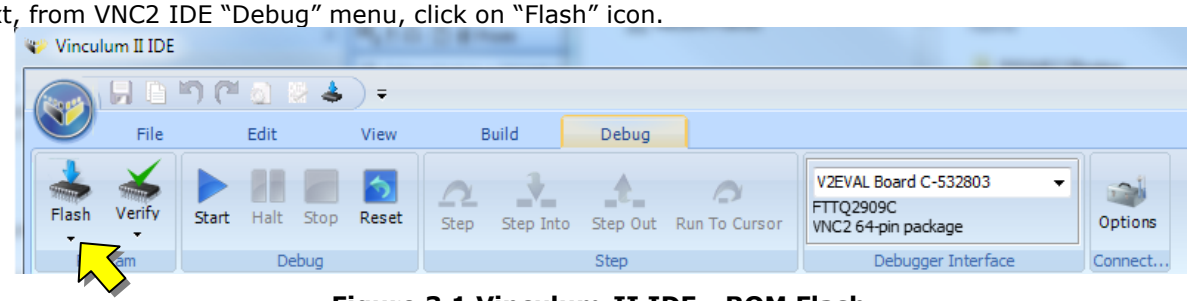

Figure 3.1 Vinculum-II IDE - ROM Flash

If a project is loaded, the last build will be flashed into the VNC2. While a project is loaded and the programmer wants to try a different ROM file, click on the small down arrow at the bottom of the Flash button. This will bring up a standard Windows Open File dialog box to allow a different selection.

If a project is not currently open, the standard Windows file open dialog will appear allowing you to select a ROM file. This is useful for the pre-compiled ROM files available on the FTDI Website.

3.2 Using VinPrg Command Line Interface

The following section describes how to program a target device using the command line utility VinPrg.exe

1. Run the Windows "Command Prompt" through the standard start menu location:

   Start → All Programs → Accessories → Command Prompt

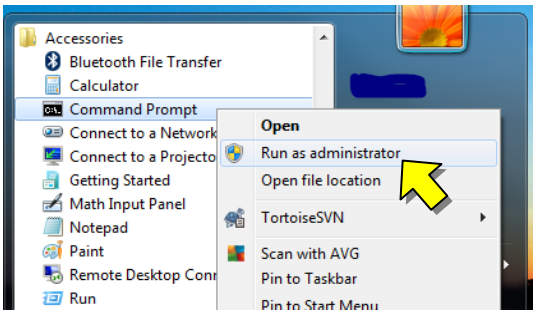

Figure 3.2 Command Line VinPrg – Getting Started

NOTE: Windows 7 users will need to right-click on the Command Prompt menu item and select "Run as Administrator".

2. Run "VinPrg" with no options from the command line to list the available options and usage guide:

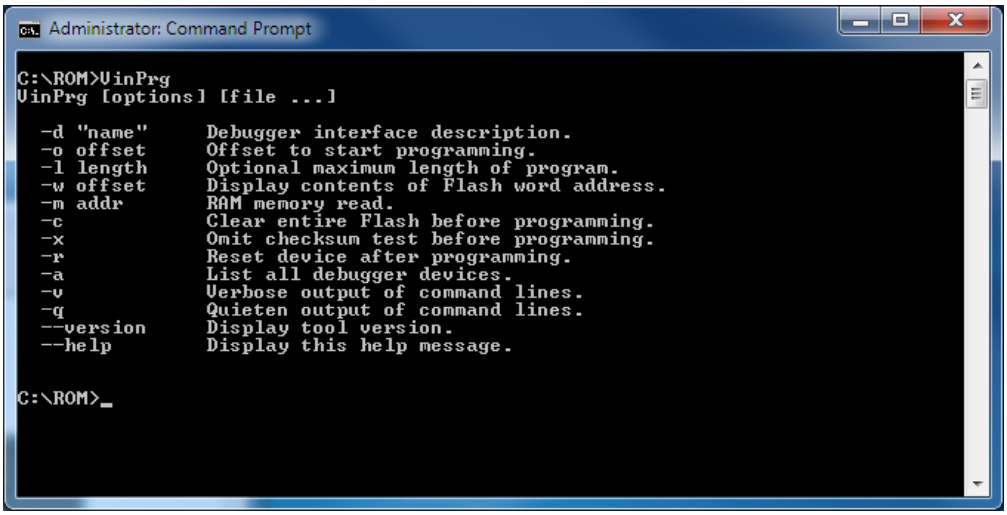

Figure 3.3 Command Line VinPrg - Show Available Options

3. Run "VinPrg –a" to list the available debugger interfaces:

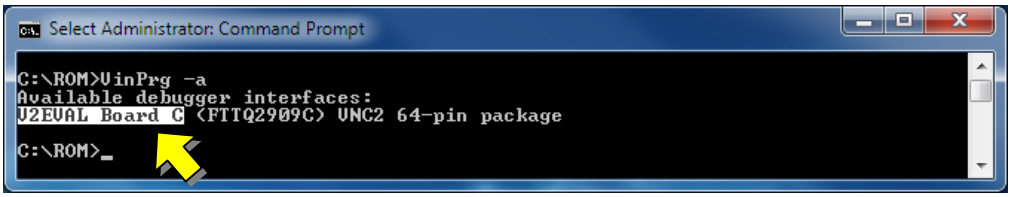

Figure 3.4 Command Line VinPrg - List Available Debugger Interfaces

4. Now that the interface name is known, a ROM file can be programmed. Two parameters are needed for VinProg, the file name and debugger interface name. Notice the quotes around the debugger interface name. In this example, a pre-compiled ROM file, FT232Uart.ROM, was downloaded from the FTDI website:

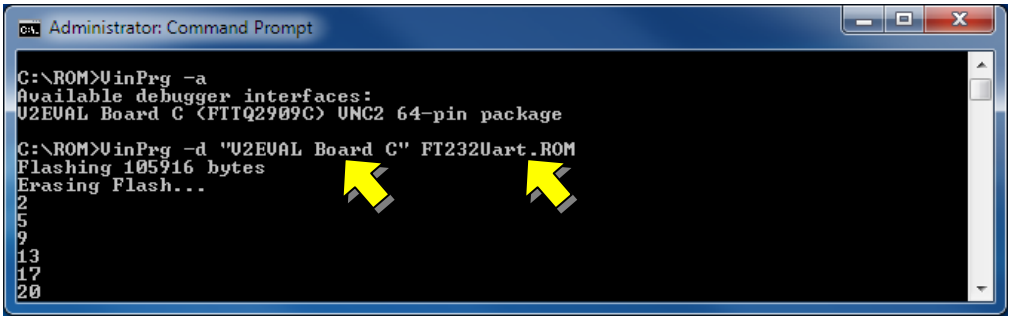

Figure 3.5 Command Line VinPrg - Flash a ROM File

5. At this point, VinPrg will erase any existing application and then program the new application contained in the ROM file. Status is shown within the Command Prompt window as these steps proceed. When complete, "Flash write done." will be displayed:

Figure 3.6 Command Line VinPrg - Flash Write Done

4 Software: VNC2 IDE Debug Features

The following section provides step-by-step instructions for using various debugging capabilities available in VNC2 IDE.

4.1 Selecting Debug Interface

This section describes how to select the debugger from the VNC2 IDE Tools Options menu and explains various debugger button options under the debugger ribbon tab. <selecting debugger, debugger toolbar>

Run VNC2 IDE (Vinculum II IDE) and ensure the debugger interface hardware or VNC2 demo board is up and running.

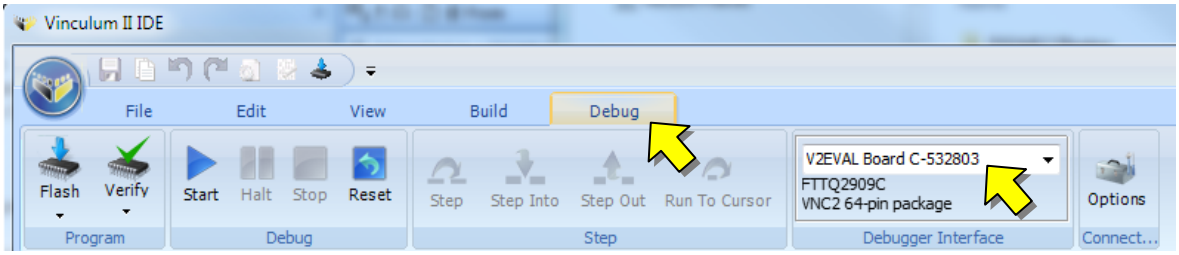

Figure 4.1 Vinculum-II IDE Debug Interface Selection

1. Click on the "Debug" tab from the top menu bar of VNC2 IDE.
2. Select the "Options" button from the Options tool ribbon bar. This will show "Options" pop-up window as shown below.

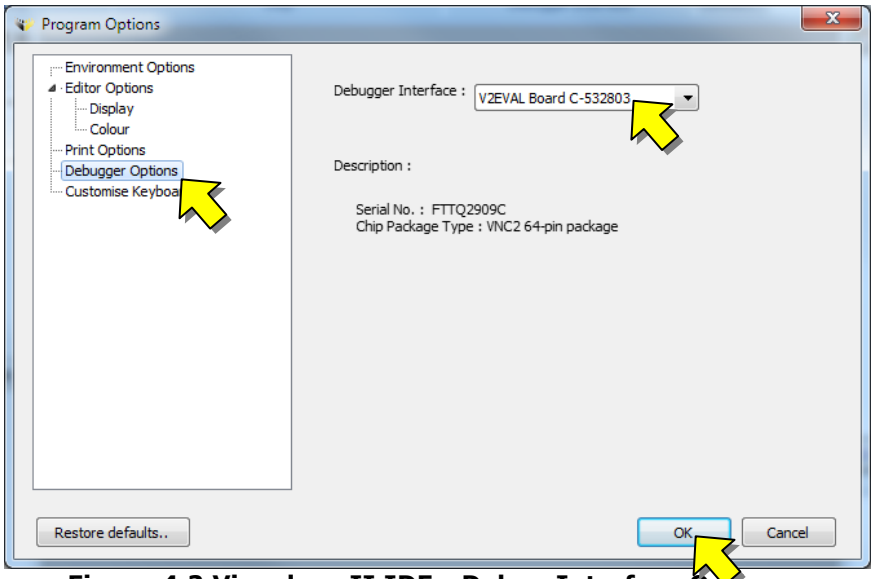

Figure 4.2 Vinculum-II IDE - Debug Interface Options

From the "Options" pop up window

1. Select "Debugger Options" from left side menu
2. From the Debugger Interface drop down menu select "V2EVAL Board C-nnnnn". The name may vary depending on which debug interface type is in use.
3. Finally click on "OK" Button

That's it; we have successfully selected a "Debugger IF" from the VNC2 IDE. Now let's have a look at various debugger buttons available under debugger menu and their functionality.

4.2 VNC2 IDE Debug Features

All of the GUI debugger options are available under the "Debug" menu tab. Clicking on the Debug tab will show all the sections in the ribbon menu namely: "Program", "Debug" and "Step". Each section in the menu has icons for carrying out a specific debug task.

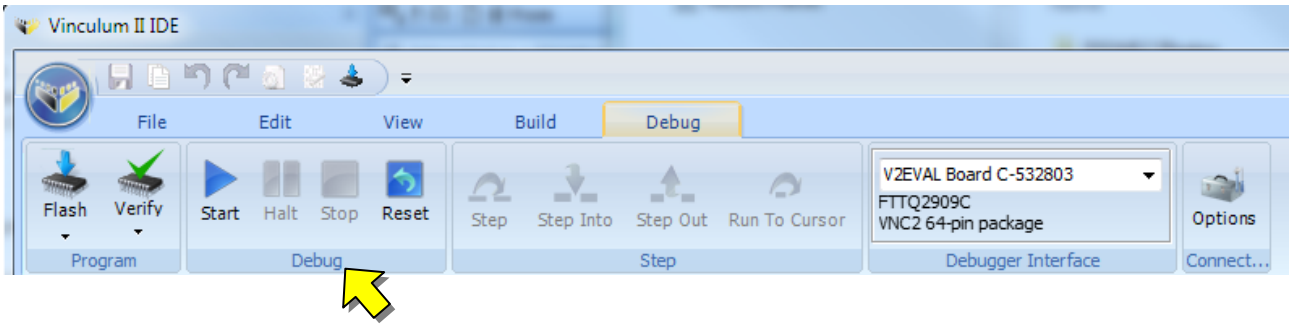

Figure 4.3 Vinculum-II IDE Interactive debugging

1. **Program:** This section has two icons: "Flash" and "Verify". After successful compilation of code, clicking on the Flash button will initiate the programming of flash memory on target device with the contents of ROM file, as shown in section 3.1. Clicking Verify will confirm the VNC2 memory contents against a ROM file.

2. **Debug:** This section consists of four icons, namely "Start", "Pause", "Stop" and "Reset". Clicking the "Start" icon initiates the sequential execution of instructions at the target device as per the contents of ROM file loaded in program memory or flash memory. Clicking the "Pause" will result in halt of command execution. Clicking on "Start" after "Pause" will resume the execution of program from where it was paused. Clicking Reset will reset the code to the first line for execution. Clicking "Stop" will result in termination of program execution and next time when "Start" is clicked the program execution will begin from the start of program memory.

3. **Step:** This section consists of four icons "Step", "Step Into", "Step Out" and "Run to Cursor". Clicking on "Step" will result in execution of a single instruction from program memory. Every time "Step" is clicked, the program counter gets incremented by one and one instruction is executed. "Step Into" will result in stepping into the function calls. "Step Out" will exit the function call it was executing without executing rest of the instructions in a function call. "Run to Cursor" will allow the user to place the cursor on a line of code in the editor window and execute all instructions until the cursor is reached. So basically step features are used to do a line by line execution of the program, ensure user must halt execution first (using "Pause") before doing a step.

4.3 Breakpoints, Start/Stop, Watch and Step

Breakpoints are used to interrupt and halt execution of the program for debugging purposes. Clicking on the line number in the gutter part of the source editor corresponding to the instruction will add a breakpoint in VNC2 IDE, where the program execution will halt. Click on the line number again to remove the breakpoint. Breakpoints can also be set, activated and cleared through the breakpoint panel. Multiple breakpoints maybe set, though a maximum of 3 can be active at one time.

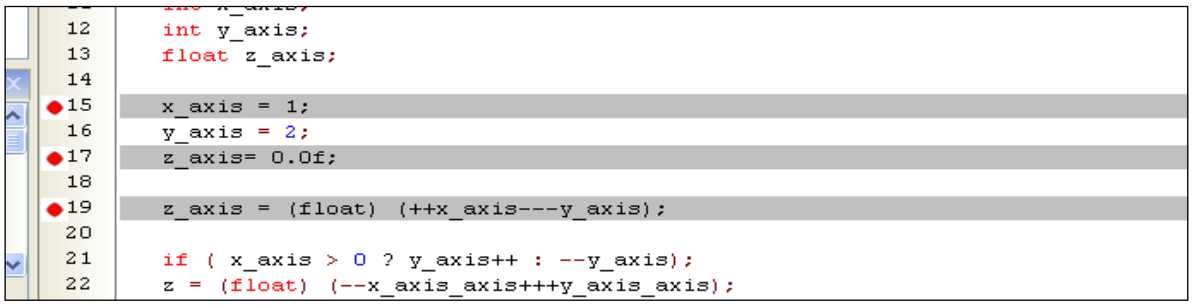

Figure 4.4 Vinculum-II IDE Breakpoints

The watch list window lists the variables that are being evaluated during the debugging process. These variables are updated after the program has paused execution. Right-clicking on the Watch window will give options to add a watch variable, edit its value, delete it, delete all watch variables or display their value in Hex format.

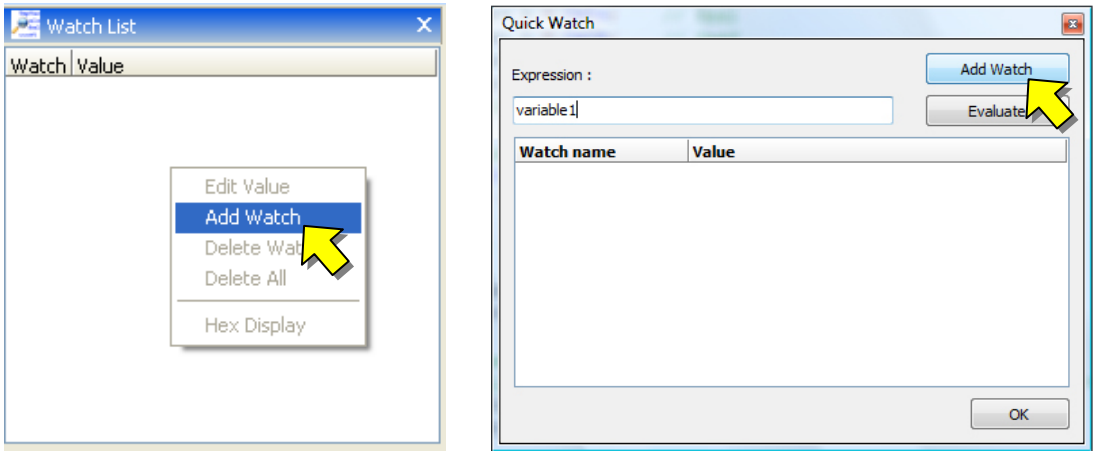

Figure 4.5 Vinculum-II IDE Add Watch

Clicking on "Add Watch" option will bring up a "Quick Watch" pop-up window, in which one can specify the watch variable name under Expression text box. Clicking on "Add Watch" button will add it to the Watch List. This way a new variable will be added in the Watch window.

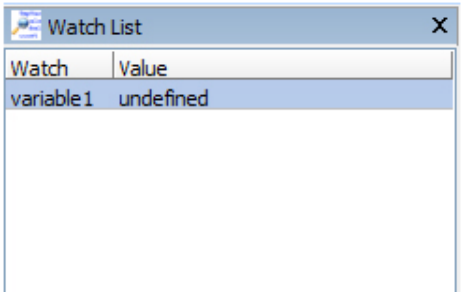

Figure 4.6 Vinculum-II IDE Watch List

5 Troubleshooting

This section discusses some common issues with VNC2 debug operation

1. **"Unable to LOAD the ROM file" message at command prompt**

   This is caused when the debugger has failed to properly load the ROM file into the VNC2 target device (VII). This could be due to noisy environment, unstable power supply, crystal oscillator unstable or out of spec, long debug wires with reflection issues, corrupted ROM file, ROM file not present in specified path, debugger not selected properly. Be sure to check all possible scenarios.

2. **"ROM file not found" message**

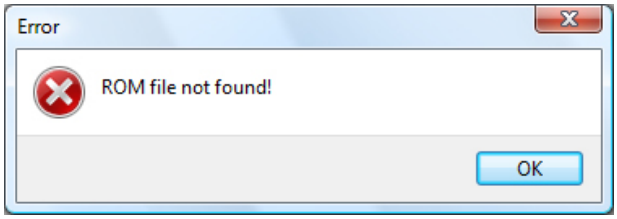

Figure 5.1 Troubleshooting - ROM File Not Found

   This is caused when the VNC2 IDE has failed to find the specified ROM file to load. Ensure the ROM file exists and double check its location or path.

3. **"Error on Loading the ROM file"**

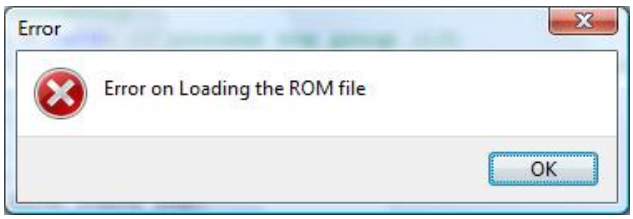

Figure 5.2 Troubleshooting – Error Loading ROM File

   This error is caused when the Debugger has failed to load ROM file. Make sure that the VNC2 is properly connected and operational.

4. **No power (VCC) at target device**

   Ensure there is no short circuit on board. For the V2-EVAL, check the Fuse F1, PWREN# pin and MOSFET Q1. Finally make sure that the debugger circuit is properly connected to target device. For other circuits, check these similar functions with the design.

5. **What is required to be able to use debug in the VNC2 Toolchain IDE?**
   a. The Vinculum-II Toolchain – Ensure the toolchain is fully installed. Check the FTDI Website for new releases.
   b. The Debugger Interface – This can be one of many forms: VNC2 Debugger/Programmer Module, V2-EVAL inbuilt debugger circuit or a circuit added to the target hardware.
   c. The FTDI device drivers required to interact with the debugger interface. The drivers can be found on the FTDI Website.

6 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 – 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

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

Branch Office – Shanghai, China

Future Technology Devices International Limited (China)
Room 408, 317 Xianxia Road,
Shanghai, 200051
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.

Appendix A – References

Document and Website References

Vinculum-II Datasheet

Vinculum-II Product Page

Vinculum-II IDE Product Page

V2-EVAL Product Page

Vinco Product Page

V2DIP modules Product Page

FT_Prog FT-series EEPROM Utility

Sullins Connectors

Acronyms and Abbreviations

Term	Description
CMD	Command Prompt
EEPROM	Electrically Erasable Read Only Memory
FTDI	Future Technology Devices International
GUI	Graphical User Interface
ICSP	In Circuit Serial Programming
I/F	Interface
IDE	Integrated Development Environment
I/O	Input/Output, usually logic level
ROM	Read Only Memory
RX or RXD	Receive
TX or TXD	Transmit
TXDEN	Transmit Data Enable
USB	Universal Serial Bus
VNC2	Vinculum-II chip

Appendix B – List of Tables & Figures

List of Tables

List of Figures

Appendix C – Revision History

Document Title: AN_138 Vinculum-II Debug Interface
Document Reference No.: FT000252
Clearance No.: FTDI# 147
Document Folder: http://www.ftdichip.com/Support/FTDocuments.htm
Document Feedback: Send Feedback

Revision	Changes	Date
1.0	Initial Release	2010-03-25
2.0	Formatted with new document style, Updated images for newer IDE version, edited copy throughout	2011-10-14