Future Technology Devices International Ltd.

Application Note AN_138

Vinculum-II Debug Interface Description

Document Reference No.: FT_000252

Version 1.0

Issue Date: 2010-03-25

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

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 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.

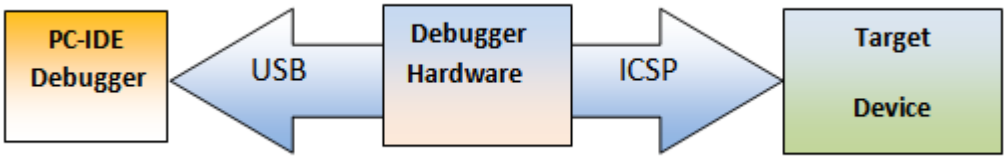


Figure 1.1 Typical Debugger Interface

Figure 1 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 "IN-CIRCUIT-SERIAL-PROGRAMMING" or ICSP.

VNC2 has only one Debugger Interface pin (Debug_IF) for carrying out debug operation. At power up, by default, this pin is available on IOBUS0. Although this pin can be relocated to a different GPIO pin using the IO Mux 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 an FT232R chip.

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

1. Use a debugger 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 the VNC2 debugger pin, power and crystal requirements followed by a description of the debugger interface hardware schematic (which is also available as the debugger module from FTDI). The document then explains the debug features available in the VNC2 IDE which allows debug operations to be carried out.

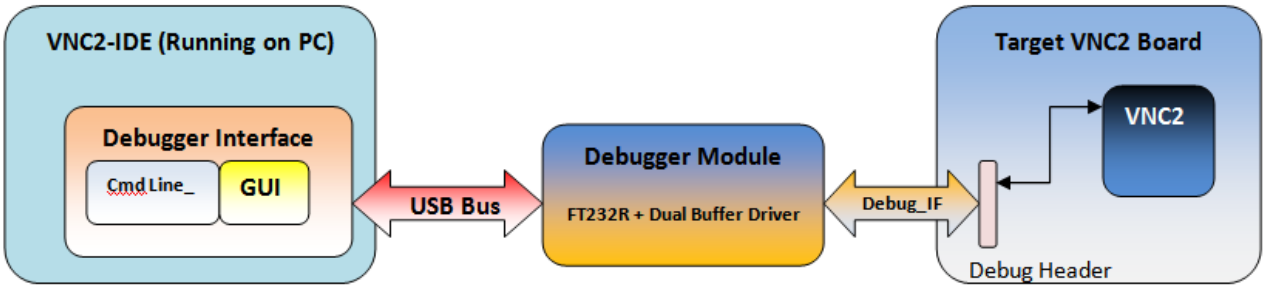
With a basic understating of the debug hardware, and using the VNC2 IDE debug features, this document then illustrates the debug operations using a V2-Eval board (which already has the debug hardware implemented on the board). This document describes a step by step process of using the debugger interface,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

2 Hardware Requirements

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

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

2.1 Interfaceing to a VNC2 Debug Module



When using the FTDI debugger module as an external accessory, the target customer application board requires a male debug header (which the debugger module plugs into) with following signals connected to VNC2.

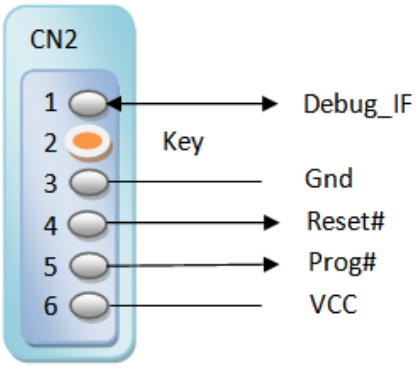


Figure 2.1 Debugger Interface Module Signal Details

Pin1 = Debug_IF (Serial Debug data interface pin).
Pin2 = [Key] (plastic Notch to prevent connecting connector other way round).
Pin3 = Ground.
Pin4 = Reset# (Optional: Needed for legacy programming designs).
Pin5 = Prog# (Optional: Legacy bootloader control pin).
Pin6 = VCC (target power supply).

The connector on the VNC2 Debugger Module is a 2mm pitch SIL socket and the target module should provide the male equivalent.

2.2 Designing On-Board Debugger Interface Hardware

The following section describes the second option of implementing 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 development environment. The conversion from the serial debug interface is carried out using a FTDI FT232R device.

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

1. VNC2 chip debug pin.
2. Dual buffer driver chip used for converting single debug pin into UART interface with separate input and outputs.
3. FT232R converter IC used for USB to serial UART conversion.
4. USB type B connector used to connect to PC debug environment.

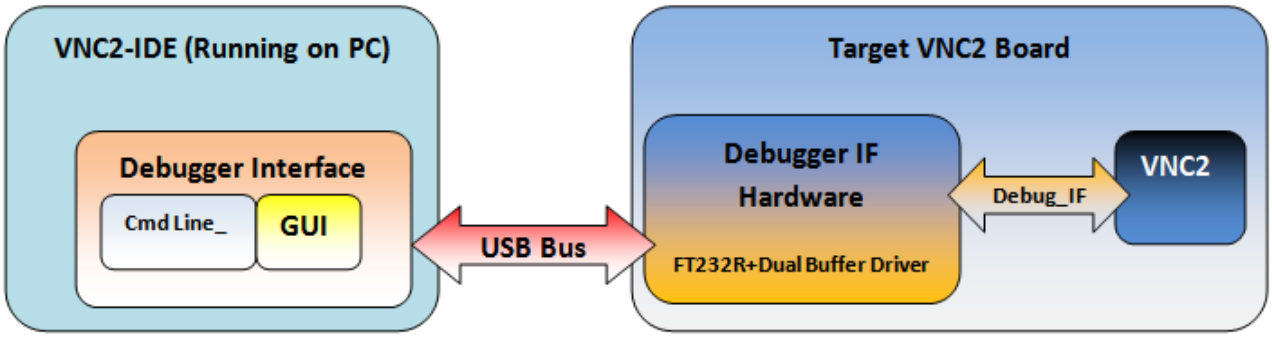


Figure 2.1 Debugger Interface Implementation

2.2.1 VNC2 Chip Hardware: Pins, Crystal, Power

The VNC2 chip uses the Debug_IF (IOBUS0 pin) signal to carry out debug operation on the actual chip. IOBUS0 pin is connected to an internal debug engine that decodes serial debug commands and executes all debug operations. Figure 2.3 indicates the connection of IOBUS0 for the VNC2 32 pin package.

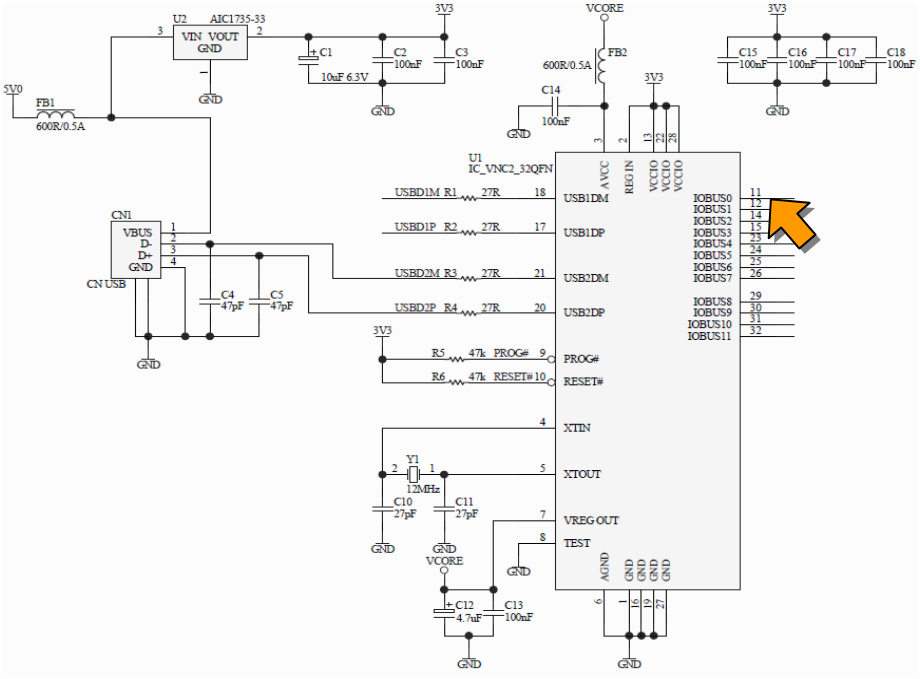


Figure 2.2 VNC2 Debugger Interface Schematic

2.2.2 Power-UP Debugger Defaults

A logic low on the Prog# pin will activate the default boot-loader on the target VNC2 device and allocate the Debug_IF pin to IOBUS0. At power up the IOBUS0 is input and will accept serial commands. The very first command sent to VNC2 is an IO_Mux command that sets the functionality of I/O pins. If the user does not set the Debug_IF in their application then by default this pin will remain at IOBUS0. It is recommended to leave the Debug_IF on this pin.

2.2.3 Debugger Interface Hardware

The debugger interface hardware uses an FT232R USB-Serial port device to converts USB packets containing VNC2 IDE debug information to TTL logic serial data format Debug_IF signal. The serial Debug_IF signal is decoded internally by VNC2 and internal debug operations are carried out appropriately.

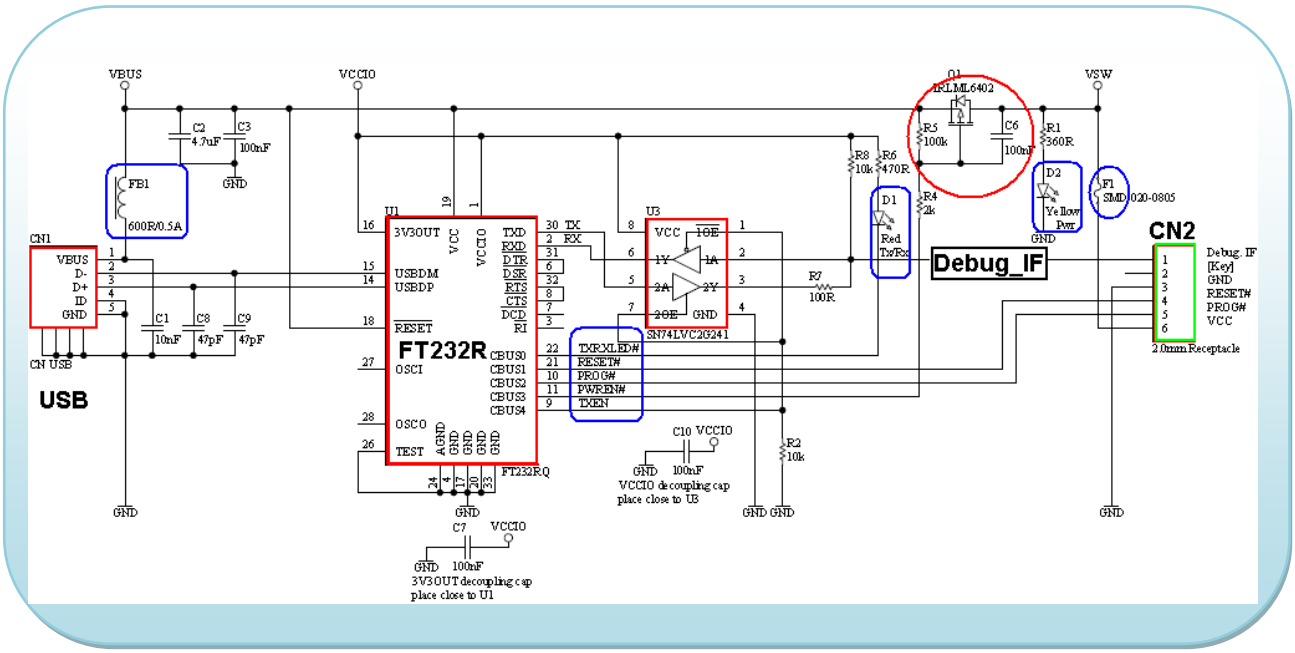


Figure 2.3 Debug IF Hardware Schematic

The important components on the USB side are the ferrite bead FB1 (600 Ohm, 0.5A) on USB power line VBUS, decoupling capacitors and filtering capacitors (C8, C9: 47pF) on USBDP and USBDM. Since VCCIO is connected to 3V3OUT, the logic level of the I/O pins is 3.3V DC. Serial port handshaking/flow control signals are connected in loopback mode i.e. Data Terminal Ready (DTR) is connected to Data Set Ready (DSR) and Request To Send (RTS) is connected to Clear To Send (CTS) pin. The TX and RX pins are converted to a half-duplex single wire serial Debug_IF signal using a dual buffer driver chip (U3: SN74LVC2G125).

When there is no data for transmission on TX, the Transmit Data Enable (TXDEN) pin is logic "LOW". This activates the 1st buffer in U3. The 1st buffer connects the Debug_IF signal to RX pin thus allowing flow of data from VNC2 to FT232R. This data is then sent to IDE via USB interface.

When IDE transmits debug commands, the TXDEN pin is driven logic "High". In this case the Debug_IF is connected to the TX pin using 2nd buffer and data is transmitted to VNC2.

This is summarised in Figure 2.4

TXDEN	Debuf_IF
0	TX
1	RX

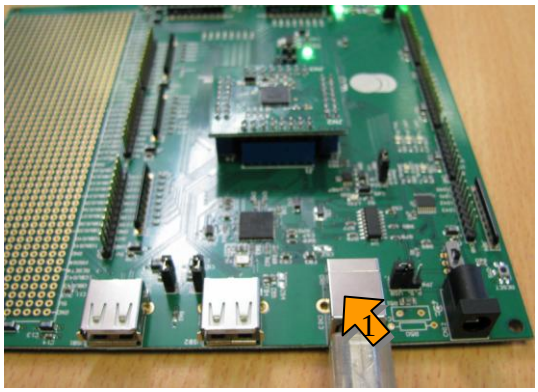
Figure 2.4 Debugger Interface Truth Table

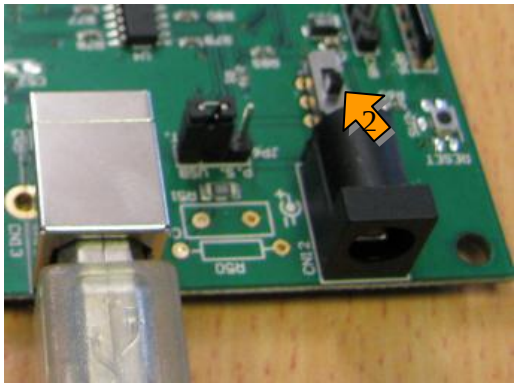
In this implementation the debugger interface hardware is powered by the USB bus. The PWREN# pin controls a P-FET to switch the USB power as VCC for the target device. The Prog# and Reset# pins provide backward compatibility with VNC1L devices, where they are used to activate boot loader mode for programming a target device.

The debugger interface hardware is the most critical portion of any embedded design. Designers must ensure that this section is free from noise sources such as electrical noise, EMI, high power switching, relays, motor controls or RF noise in the designed system. Discussion of implementing isolation between USB and serial interface is beyond the scope of this application note.

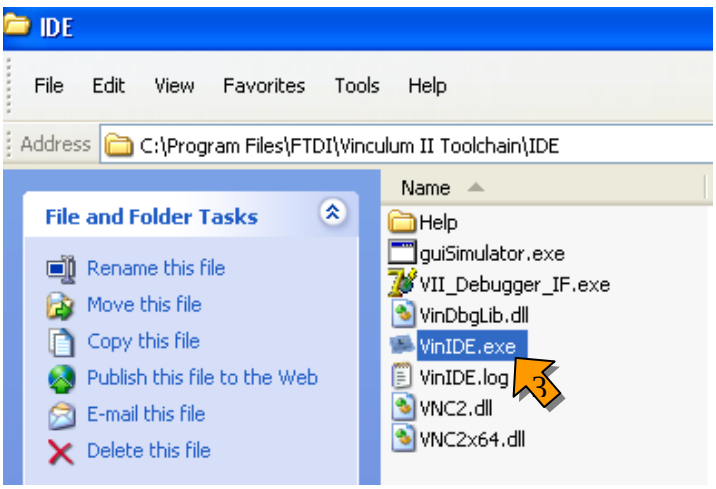
2.2.4 Connecting to the V2-EVAL Board Debugger Interface

Connect the debugger interface hardware to a PC via a USB cable. Apply the power to the V2-EVAL board and turn "ON" the power switch.





Run "Vinculum II IDE" from the following path
"C:\Program Files\FTDI\Vinculum II Toolchain\IDE\VinIDE.exe"
(Installation of the toolchain will also have provided the option for an icon on the desktop)



3 Programming the Target Device

3.

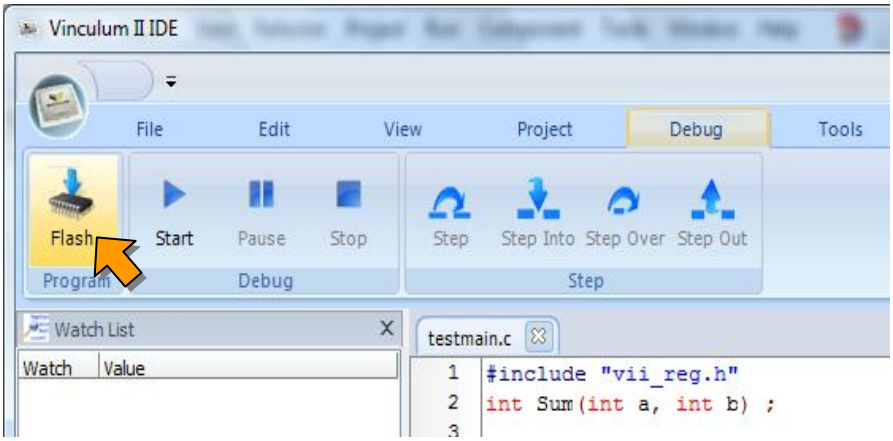
4.

3.1 Using VNC2 IDE

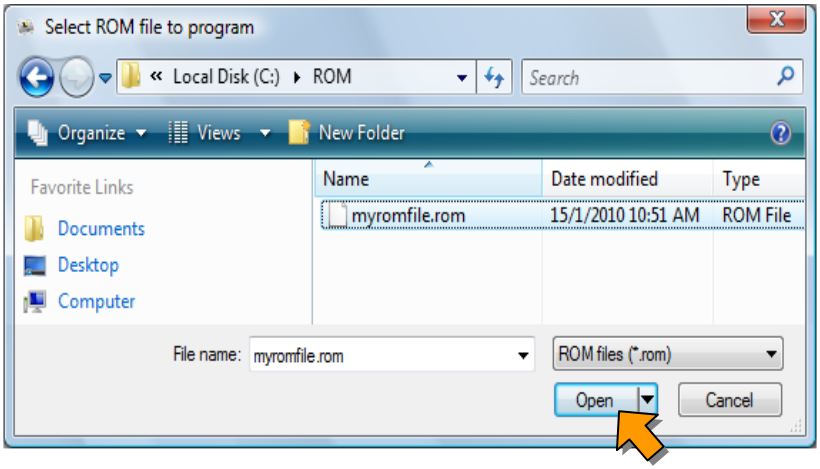
This section describes how to use the Debugger functions from the Debug menu of the IDE. It describes loading, running and debugging code in the VNC2 device.

After writing the source code for a project the output that should be programmed into a device is labelled a .ROM file. This is produced as the result of a successful build (not within the scope of this document). The .ROM file which can be "Flashed" to the target device from the Debug menu as shown below.

From VNC2 IDE "Debug" menu, click on "Flash" icon.



Next select the ROM file from your project directory and click on "Open" button. This initiates the programming of ROM file into flash memory of target device.

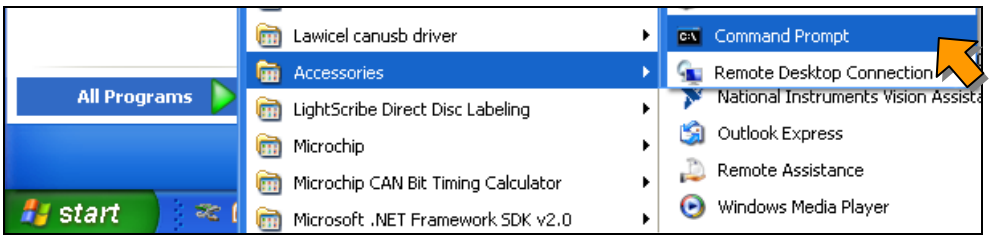


Once the programming is successful, a "ROM file programming Successful" message appears, click on "OK" of message box to continue debug operations.

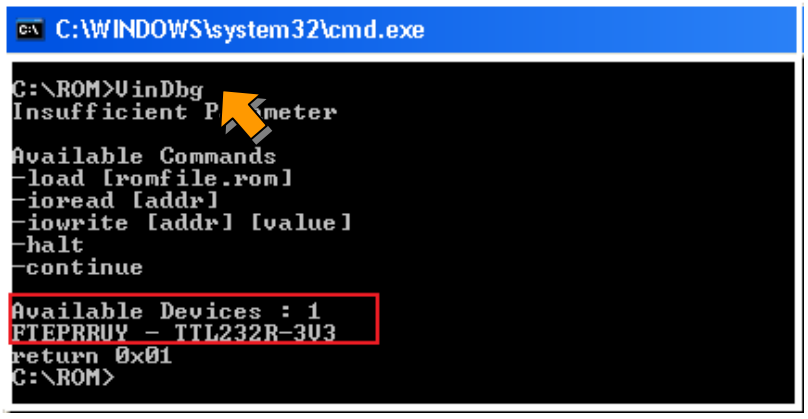
3.2 Using VinDbg Command Line Interface

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

1. Run "Command Prompt" using following shortcut
   Start→All Programs →Accessories → Command Prompt



2. Run "VinDbg" under command prompt window to check if debugger is active and available. This command will also enlist all debugger interfaces currently available on the system with their serial number and manufacturing string details.



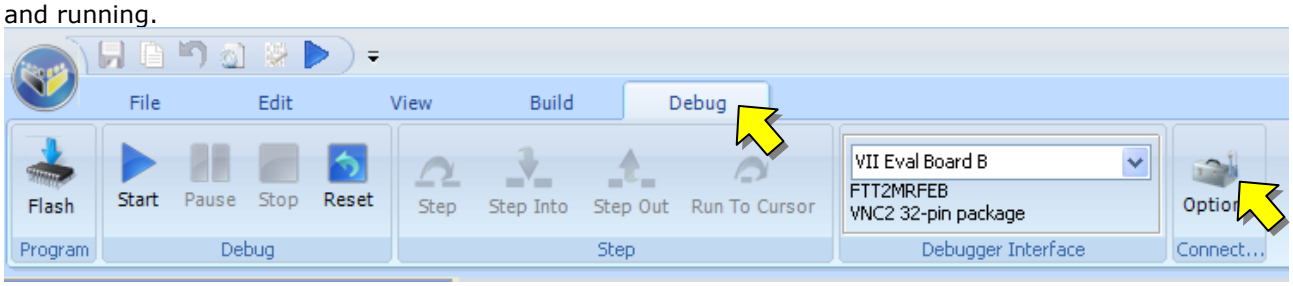
3. To program the target device through command prompt use the "load" switch parameter with VinDbg command specifying the file name and debug interface name, as shown below. On successful programming of the target device it will return 0x00.

4 Software: VNC2 IDE Debug Features

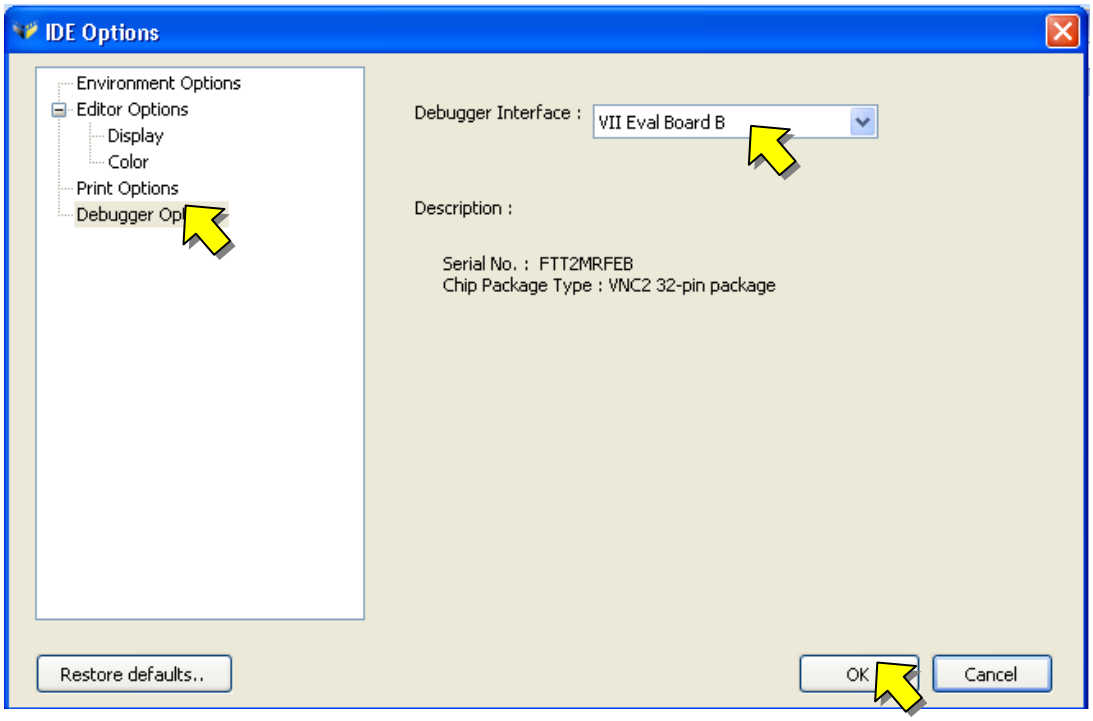
The following section provides step-by-step instructions for using various debugging capabilities available in VNC2 IDE.

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.



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.



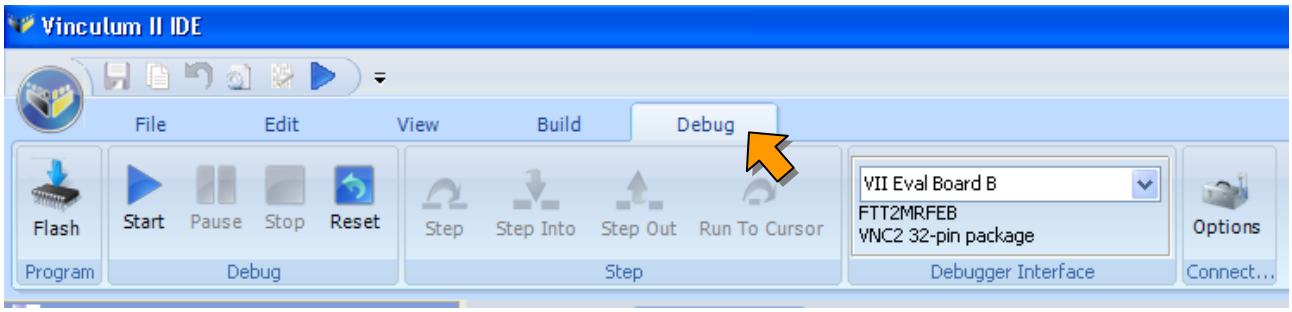
From the "Options" pop up window

1. Select "Debugger Options" from left side menu.
2. From the Debugger Interface drop down menu select "V2EVAL Demo B".
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 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.



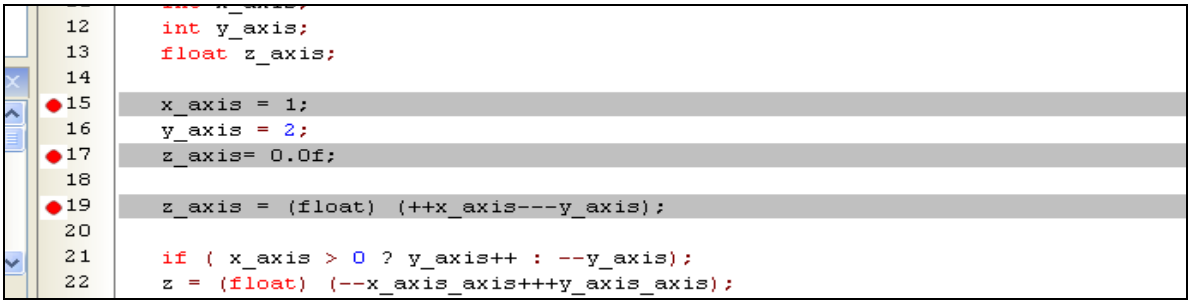
1. **Program:** This section has one icon called "Flash". After successful compilation of code, clicking on this icon will initiate the programming of flash memory on target device with the contents of ROM file.

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.

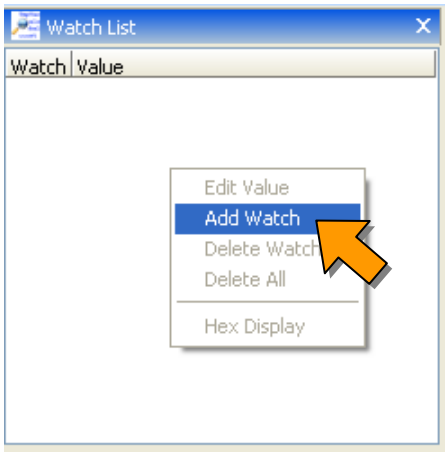
4.3 Breakpoints, Start/Stop, Watch and Step

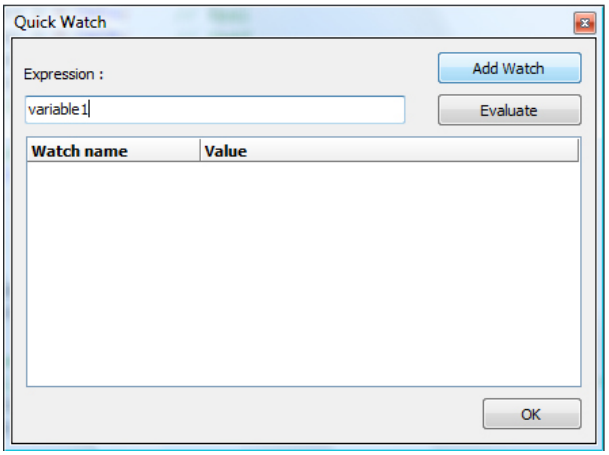
Breakpoints are used to interrupt and halt execution of the program for debugging purposes. Clicking on the line number in the gutter part of the source editor corresponding to the instruction will add a breakpoint in VNC2 IDE, where the program execution will halt. Click on the line number again will remove the breakpoint. There are a maximum of 3 active breakpoints that may be set at any time.



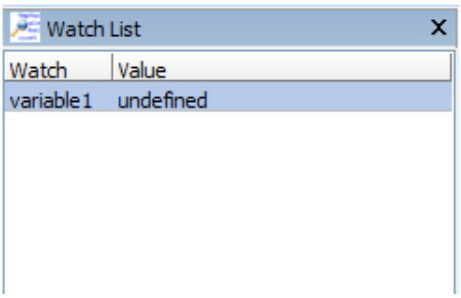
The watch list window lists the variables that are being evaluated during the debugging process. These variables are updated after the program has paused execution. Right-clicking on the Watch window will

give options to add a watch variable, edit its value, delete it, delete all watch variables or display their value in Hex format.





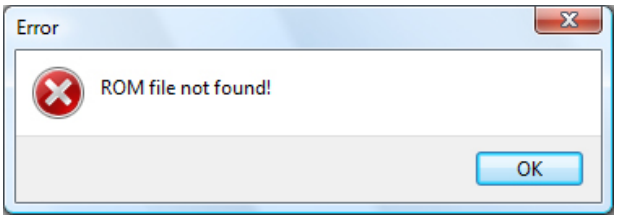
Clicking on "Add Watch" option will bring up a "Quick Watch" pop-up window, in which one can specify the watch variable name under Expression text box. Clicking on "Add Watch" button will add it to the Watch List. This way a new variable will be added in the Watch window.



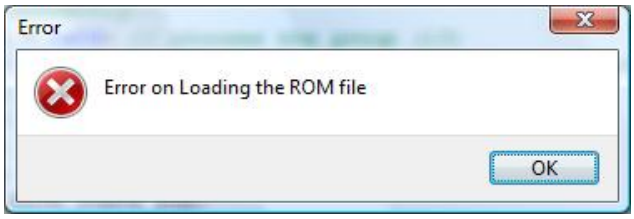
5 FAQ

This section discusses some frequently asked questions regarding 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. Ensure to check out all the possible scenarios.
2. **Getting "ROM file not found" message**



   This is caused when the VNC2 IDE has failed to find the specified ROM file to load. Please ensure the ROM file exists and double check its location or path.
3. **Getting "Error on Loading the 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. Check the Fuse F1, PWREN# pin and MOSFET Q1. Finally make sure that the connecter C2 is properly connected to target device.
5. **What is required to be able to use debug in VNC2 IDE?**
   a. Make sure you have these files to be able to use the debugging functionalities:
      The debugger (VinDbg.exe)
      The VinDbgLib.dll1 library
   b. Make sure the paths of the above executables are declared either by adding them to the path environment variable or by explicitly declaring them in the IDE's options module
   c. Ensure that the evaluation board is connected and turned on.

6 Abbreviations and Terms

CMD	Command
FTDI	Future Technology Devices International
GUI	Graphical User Interface
ICSP	In Circuit Serial Programming
I/F	Interface
IDE	Integrated Development Environment
ROM	Read Only Memory
RX	Receive
TX	Transmit
TXDEN	Transmit Data Enable
USB	Universal Serial Bus
VNC2	Vinculum II chip

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.admin@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 – Revision History

Version draft	First draft	21/01/2010
Version 1.0	First release	25/03/2010