The NetBurner Tools User's Manual

Revision 1.2
March 5, 2009
Initial Release

Table of Contents

1. Introduction 4
2. Documentation 4
3. NBEclipse 6
4. NetBurner's Dev C++ IDE 8
5. MTTTY 9
6. IPSetup 14
7. NBFind 17
8. NBTFTP 19
9. CompCode 24
10. CompHtml 25
11. AutoUpdate 26
12. Application Identification 31
13. TaskScan 33
14. SmartTrap 38
15. WinAddr2Line 42
16. The NetBurner Monitor 45
    16.1. Download Commands 47
    16.2. Block Memory Operations 48
    16.3. Monitor Commands 49
        16.3.1. Boot the Application 49
        16.3.2. Display Help 50
        16.3.3. Display the Monitor Version 51
        16.3.4. Reset and Reinitialize the Monitor 52
        16.3.5. Set/Change Monitor Settings 53
        16.3.6. Start Execution 55
    16.4. Memory Operations 56
    16.5. Register Operations 57
17. GDB and NetBurner 58
18. Serial Debugging 60
    18.1. Serial Debugging Functions 68
        18.1.1. InitGDBStub 68
        18.1.2. InitGDBStubNoBreak 69
19. Network Debugging 70
    19.1. Network Debugging Functions 71
        19.1.1. InitializeNetworkGDB 71
        19.1.2. InitializeNetworkGDB_and_Wait 72
    19.2. Network Debugging Options 73

19.2.1. DebugIP 73
19.2.2. DebugRebootOnTrap 74
19.2.3. DebugRebootOnDisconnect 75
19.2.4. DebugNormalArp 76

1. Introduction

Thank you for purchasing your NetBurner Development Kit, and welcome to the NetBurner family.

Warning: The Networking contents of the NetBurner Network Development Kit User's Manuals do not apply to any of NetBurner's Non-Network Development Kits (e.g. Mod5213 Development Kit).

2. Documentation

- All **NetBurner Hardware** Documentation is in **C:\Nburn\docs\platform**
- All **Freescale Manuals** are in **C:\Nburn\docs\FreescaleManuals**
- A **Documentation Overview** (i.e. a **Getting Started Guide**) is in **C:\Nburn\docs**
- The **NetBurner Runtime Libraries** User's Manual and the **NetBurner uCOS Library** User's Manual are both in **C:\Nburn\docs\NetBurnerRuntimeLibrary**
- The **NNDK Programmer's Guide** is in **C:\Nburn\docs\NetworkProgrammersGuide**
- All **EFFS Documentation** (for the **PK70**, **Mod5234**, **Mod5270**, **Mod5272**, and **Mod5282** platforms only) is in **C:\Nburn\docs\EFFS**
- The **NBEclipse Getting Started** Guide is in **C:\Nburn\docs\Eclipse**
- All **NetBurner License Information** is in **C:\Nburn\docs\LicenseText**
- All **GNU Documentation** is in **C:\Nburn\docs\GNU**

The NetBurner User Manuals and Guides are intended as an introduction to developing Network/Internet (and Non-Networked) enabled products using NetBurner's Development Kits, but it is beyond the scope of any of these Manuals and Guides to tell you everything you need to know about embedded applications or about the C/C++ programming language. However, we do refer you to a variety of publications that explain the topics that we present in our manuals in more detail.

The software included in your kit is licensed to run only on NetBurner provided hardware. **If your application involves manufacturing your own hardware, please contact our Sales Department for details on a royalty free software license**.

NetBurner is your single source for hardware, software, development kits, tools, technical support, and custom design services. These elements are combined in a unique package that lets you concentrate on developing your product instead of reinventing network protocols and designing hardware. NetBurner solutions also allow you to reduce risk and improve functionality with a complete proven design, including hardware, TCP/IP Stack, RTOS, and all necessary tools. NetBurner is indeed the fastest way to network enable your product.

Whether you want to design your own hardware, or are looking for a standard off-the-shelf network solution - NetBurner provides the software, hardware, and tools to get your product to market in the shortest possible time. NetBurner offers a full line of services from board level designs and hourly consulting to complete turnkey systems.

Please ensure that your NetBurner Development Kit is registered by going to our **Support** site now to set up your account. **You must register your kit before you can receive Technical Support**. The registration data stored on NetBurner's Support Server will not be sold, exchanged, or knowingly released to third parties without prior written permission from the individuals affected.

NetBurner, Inc. makes no representations or warranties with respect to the contents of our manuals and specifically disclaims any implied warranties of merchantability or fitness for any particular purpose. Further, NetBurner, Inc. reserves the right to revise our manuals, make changes, and/or discontinue them without notice. Every effort has been made to ensure that all information is correct, but NetBurner, Inc. is not responsible for inadvertent errors.

NetBurner manuals contain links to third-party web sites that are not under the control of NetBurner, and NetBurner is not responsible for the content on any linked site. If you access any third-party sites listed in any of our manuals, then you do so at your own risk. NetBurner provides these links only as a convenience, and the inclusion of the link does not imply that NetBurner endorses or accepts any responsibility for the content on those third-party sites.

Life Support Disclaimer

NetBurner's products both hardware and software (including tools) are not authorized for use as critical components in life support devices or systems, without the express written approval of NetBurner, Inc. prior to use.

As used herein: (**1**) Life support devices or systems are devices or systems that (**a**) are intended for surgical implant into the body or (**b**) support or sustain life, and whose failure to perform, when properly used in accordance with instructions for use provided in the labeling, can be reasonably expected to result in a significant injury to the user. (**2**) A critical component is any component of a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.

NetBurner, Inc.
5405 Morehouse Drive, Suite 200
San Diego, CA 92121 USA
1-858-558-0293 (Telephone)
1-858-558-8549 (Fax)

http://support.netburner.com/ (NetBurner Technical Support)
http://www.netburner.com (NetBurner Website)
http://www.netburnerstore.com (NetBurner Store)
sales@netburner.com (NetBurner Sales Department)

3. NBEclipse

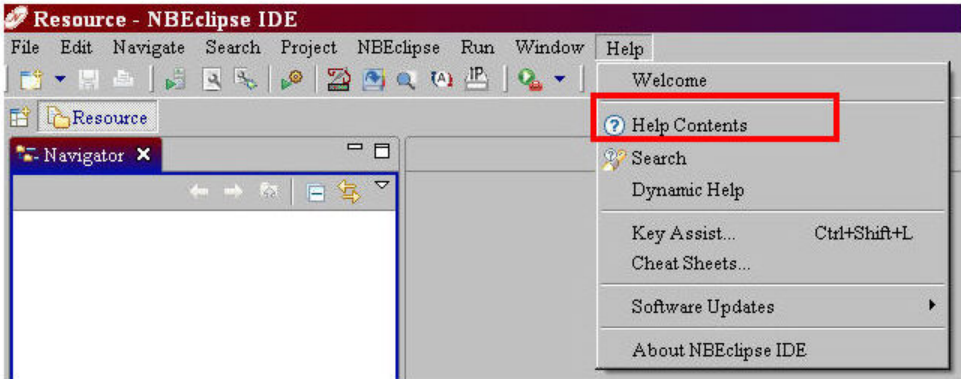
Important: NBEclipse requires Java Version 1.5 or higher installed on your host computer.

NBEclipse is built off the Eclipse platform IDE. Eclipse acts as a shell, and allows custom plugins to be created for it. Our plugin creates, manages, customizes, and builds NetBurner projects. However, you can easily add any other plugin on top of the NBEclipse IDE to give you the functionality that you want. For instance, plugins exist to mimic other editors (e.g. SlickEdit), GUI development, code development (across many different programming languages), debugging, and many more. **To use a custom plugin, just copy it into your C:\Nburn\NBEclipse\plugins directory before opening NBEclipse**.

To open up NBEclipse from Windows: Start → Programs → NetBurner NNDK → NBEclipse → NBEclipse. (**NBEclipse.exe** is in **C:\Nburn\NBEclipse**.)

NBEclipse offers a rich, highly customizable development environment. It allows you to work on multiple projects over multiple platforms simultaneously. NBEclipse also offers multiple compilation methods. In addition, you can choose to provide a custom makefile that you have complete control over, or you can choose to let NBEclipse control and update your makefile. However, you still have full access to all compiler flags/options to customize your project, as you like.

NBEclipse offers our full range of tools that are easily accessible within NBEclipse itself (as shown below). Simply click the NBEclipse pull-down menu and select your tool. A new tool (i.e. a view) to the NetBurner NBEclipse tool suite is NBFind. For additional information about NBFind, please refer to your NBEclipse (plug-in) User's Manual. From the **Help** pull-down menu in NBEclipse, select **Help Contents** (as shown below).



Debugging within NBEclipse is much easier then trying to use Insight/Dev-C++. Debugging is now much more **stable**, and is very easy to set up and use. **All** of the debugging options that you could use in Insight are here, and more are included on top of that.

Warning: If you are using Version 2.0 or greater of the NetBurner tools, you must use the NBEclipse integrated debugger. You cannot use Insight. However, you can however use GDB (the Command line debugger) with any NNDK software version.

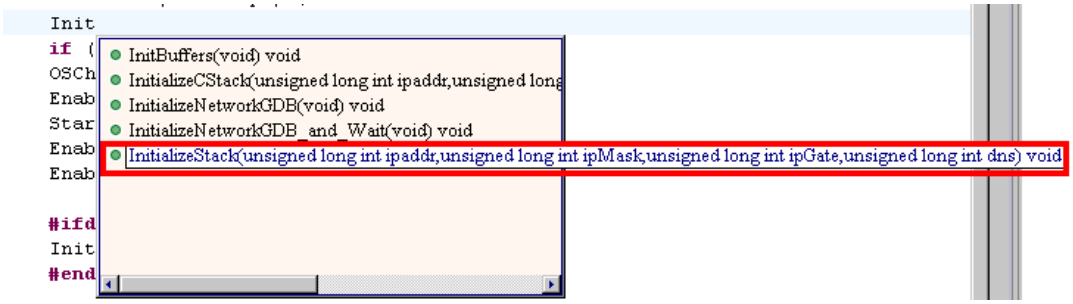
Note: To find out what **version** of the NetBurner tool set you are using, navigate to your Nburn (root) directory (**C:\Nburn** by default), and open up (e.g. in notepad) the **release_tag** file.

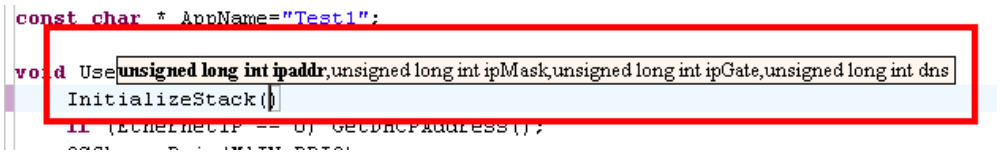
Please read your **NBEclipse Getting Started Guide** for step-by-step debugging instructions. From Windows: Start → Programs → Netburner NNDK → NBEclipse → NBEclipse Getting Started. By default, this PDF is located in **C:\Nburn\docs\Eclipse**.

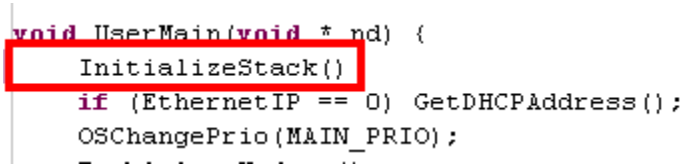
NBEclipse also offers great file management. **CVS support** is included out of the box and it is easy to integrate into your own CVS tree. Even without CVS, the 50 most recent saves per file are automatically saved, and can be reverted to, or diffed to the live version at any time.

When you are compiling your application, NBEclipse will find **code errors** and point them out to you through multiple means. NBEclipse can even locate errors on the fly, if you have **automatic building** turned **on**.

A **code complete "like" functionality** exists as well. This allows code completion and help when writing your code. For instance, you can type the start of a function (e.g. Init). Next, press the **Ctrl** key, and then the **space bar** on your keyboard (this is the **code complete shortcut**). A box will appear listing all of the functions/variables that can be completed with that start portion (i.e. Init). Select your function by double clicking it, and that function will be inserted in your code (as shown in the screen shots below).







Warning: You cannot compile an application that was created with NetBurner's Dev C++ using NBEclipse. You must create an NBEclipse project. Likewise, you cannot compile an application that was created with NBEclipse in NetBurner's Dev C++. You must create a NetBurner's Dev C++ project.

Please read your **NBEclipse Getting Started Guide** for step-by-step instructions. From Windows: Start → Programs → Netburner NNDK → NBEclipse → NBEclipse Getting Started. By default, this PDF is located in **C:\Nburn\docs\Eclipse**.

4. NetBurner's Dev C++ IDE

We recommend that you use the NBEclipse IDE to create, compile, and download applications to your NetBurner device.

Please read your **NBEclipse Getting Started Guide** for step-by-step instructions. From Windows: Start → Programs → Netburner NNDK → NBEclipse → NBEclipse Getting Started. By default, this PDF is located in your **C:\Nburn\docs\Eclipse** directory**.**

NetBurner's Dev C++ is a full-featured Integrated Development Environment (IDE) that enables you to create, edit, and download applications to your NetBurner device. Dev C++ is Free Software **distributed** under the terms of the **GNU General Public License** (GPL). A copy of the **GPL** is located in your **C:\Nburn\docs\GNU** directory.

A NetBurner Dev C++ Quick Start Guide (PDF) can be accessed directly from Windows: Start → Programs → Netburner NNDK → Dev C++ →Quickstart Guide. By default, this PDF is located in your C:\Nburn\devcpp\Help directory.

To open up NetBurner's Dev C++ from Windows: Start → Programs → Netburner NNDK → Dev C++ → Dev C++. For Dev C++ specific help, please refer to your Dev C++ User's Manual. From the Help pull-down menu in NetBurner's Dev C++ select Dev-C++ Help.

Warning: If you are using Version 2.0 or greater of the NetBurner tools, you must use the NBEclipse debugger (You cannot use Insight via Dev C++.) This means that you will have to import your Dev C++ project into NBEclipse.

Note: To find out what **version** of the tool set you are using, navigate to your Nburn (root) directory (**C:\Nburn** by default), and open up (e.g. in notepad) the **release_tag** file.

The features of NetBurner's Dev C++ include:

- An Application Wizard for creating new applications
- The ability to create new projects for existing applications
- Compile, link, and download your applications in one easy step
- Support for multiple languages (localization)
- Class browser
- Code completion
- Function listing
- Project manager
- Customizable syntax highlighting editor
- Support of templates for creating your own project types
- Makefile creation
- Tool manager
- Print support
- Find and Replace facilities

Warning: You cannot compile an application that was created with NetBurner's Dev C++ using NBEclipse. You must create an NBEclipse project. Likewise, you can not compile an application that was created with NBEclipse in NetBurner's Dev C++. You must create a NetBurner's Dev C++ project.

5. MTTTY

Synopsis:

`MTTTY.exe`

Description:

MTTTY (Multi-Threaded TTY) is an easy to use high performance Win 32 serial communications program. MTTTY can be executed in five ways.

- From **NBEclipse**: Click on the NBEclipse pull-down menu and then select Mttty
- From **NetBurner's Dev C++**: Click on the Tools pull-down menu and then select MTTTY
- From **Windows**: Start → Programs → Netburner NNDK → Mttty Serial Terminal.
- From the **DOS** command line: Navigate to the **C:\Nburn\pcbin** directory (default installation). **Type** the command **mttty** then **press** the **Enter** key.
- From Windows **Explorer**: Navigate to the **C:\Nburn\pcbin** directory (default installation) and **double click** the **mttty.exe** icon

Procedure:

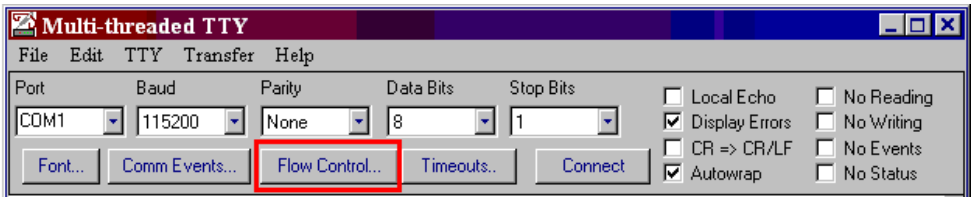
- **Attach** the supplied **Serial cable** (or the supplied **NULL Modem cable for the SB72EX and CB34EX**) from the serial port on your NetBurner device to the serial port on your host computer. For additional hardware information, please refer to your specific Hardware Manual. From Windows: Start → Programs → NetBurner NNDK → Platform Hardware.
  - If you are using the **Mod5272**, the **Mod5282**, the **Mod5234**, or the **Mod5270**, the serial port (**UART 0**) is the inner serial port (**J7**) on your **Mod-Dev-100** Carrier board. **Note:** If you are using **any** of the above NetBurner modules with the **Mod-Dev-50** (Promo) Carrier board, the serial port is the inner serial port (**J4**).
  - If you are using the **Mod5270LC** with the **Mod-Dev-70** Carrier board, the serial port (**UART 0**) is the inner serial port (**J4**).
  - If you are using the **SB72EX** (with the supplied **NULL Modem cable**), the serial port (default configuration) is the left serial port (**Port 0**). You **must** use a **NULL Modem cable** to connect your SB72EX to your host computer. **Warning: You cannot use a standard serial cable with your SB72EX device.**
  - If you are using the **CB34EX** (with the supplied **NULL Modem cable**), the serial port (default configuration) is the DB9 port (**Port 1**). You **must** use a **NULL Modem cable** to connect your CB34EX to your host computer. **Warning: You cannot use a standard serial cable with your CB34EX device.**

- If you are using the **Mod5213**, the serial port is the inner serial port on your **Mod-Dev-40** Carrier board (**UART0**).

- If you are using the **SB70** or **SB72**, the serial port is the inner serial port (**J2**) on the **SB72 Adapter/Evaluation board**.

- If you are using the **CFV2-66**, the serial port is labeled **J2**. If you are using the **CFV2-40**, the serial port is labeled **J4**.

- **Execute** MTTTY using any one of the five methods described above. See the screen shot below for the recommended (**factory default**) settings.



- **Port**: The communication port that you connected the serial cable to on your host computer (**usually COM1**)

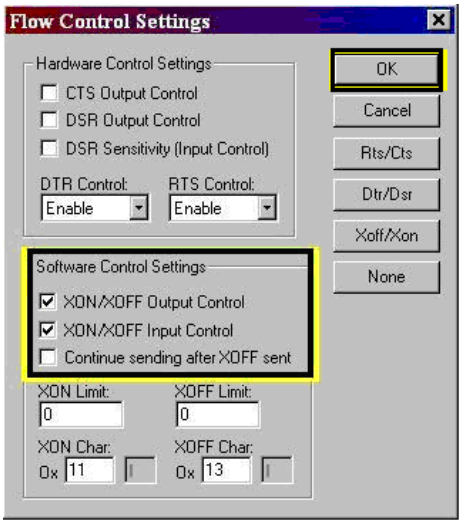
- **Baud**: The host computer and the attached serial (NetBurner) device must agree on a speed or baud rate to use for the serial connection (the **recommended** setting for both is **115200**).

- **Parity**: Checks whether data has been lost or written over when transmitted between your host PC and your NetBurner device (the **recommended** setting is **None**)

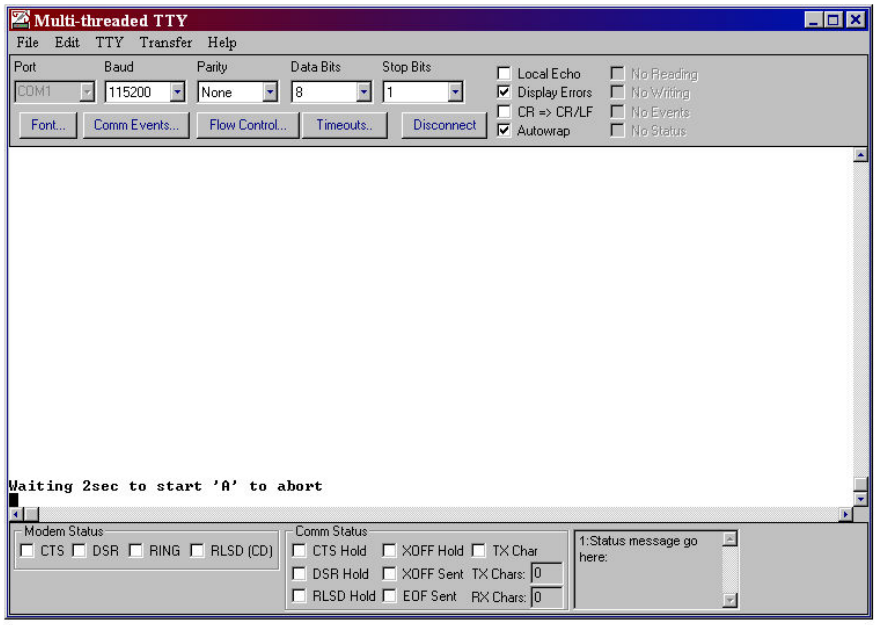
- **Data Bits:** The number of bits in a transmitted data package (the **recommended** setting is **8**)

- **Stop Bits**: The stop bit follows the data and parity bits in serial communication. It indicates the end of transmission (the **recommended** setting is **1**).

Warning: If you are using the Mod5213 kit, you must enable Flow Control. Just **click** the **Flow Control** (as shown above) button and **enable** flow control (as shown below).



- **Click** the **Connect** button and **cycle power** (a **hard reset**) to your NetBurner hardware. You **will** see a "Waiting Xsec to start 'A' to abort" message in the MTTTY window (as shown below).



- Next, **type** an **uppercase A** (i.e. **A**) in the MTTTY window **before** the time **expires** to get into the NetBurner monitor program. You will see the NetBurner prompt (i.e. **nb>**) in the MTTTY window (as shown in the screen shot on the next page).

MTTTY can be used to download your application into your NetBurner device's **SDRAM** (using the **DL** command) or **FLASH** Memory (using the **FLA** command) via the serial connection from your host computer to your NetBurner device.

Warning: If you are using the Mod5213 kit, you cannot use the DL command. You must download your application to FLASH Memory using the FLA command. Please read your (hard copy) Mod5213 Quick Start Guide for step-by-step instructions.

Warning: If you are **unable** to use MTTTY and get an **Error 5: CreateFile. Access is denied**. error box (as shown below), the currently selected communication port (i.e. COM1) is in use by another program or device. Common programs that also use the serial port would be other terminal programs such as HyperTerminal. In addition, if you have a second instance of MTTTY opened, you will get this error. The solution is to close all open programs, click the OK button (to dismiss this error box), and execute MTTTY again.

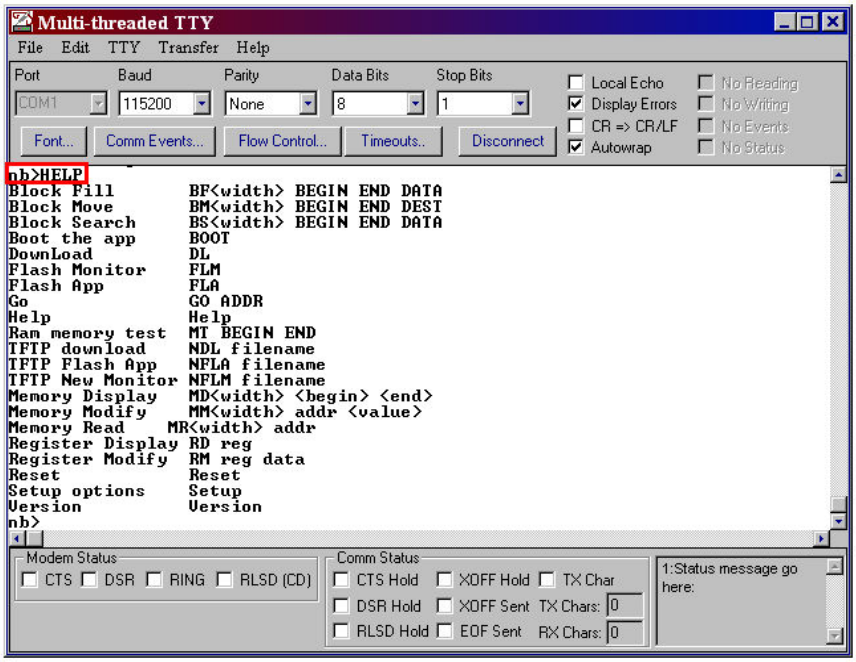
Warning: Programs downloaded to FLASH memory will overwrite the current program in your NetBurner device.

Note: If, after downloading your application into your NetBurner device's FLASH memory, it does **not** start automatically (e.g. you did not select "Boot to Application", item #8 in the MTTTY setup screen); you can start your application by **typing** the command **BOOT** at the nb> prompt and **pressing** the **Enter** key. See section **16.3.1** for additional information on the **boot** command.

Warning: Programs downloaded to SDRAM will be lost when the power to your NetBurner device is turned off.

Note: If you chose to download your application into your NetBurner device's SDRAM, just **type** the command **GO** at the nb> prompt and **press** the **Enter** key to execute your application. See section **16.3.6** for additional information on the **go** command.

Important: For help on running MTTTY (as well as seeing all the available monitor commands) **type** the command **HELP** at the nb> prompt and **press** the **Enter** key. (See the screen shot below for a list of all of the available monitor commands for the Mod5272.) See section **16.3.2** for additional information on the **help** command.

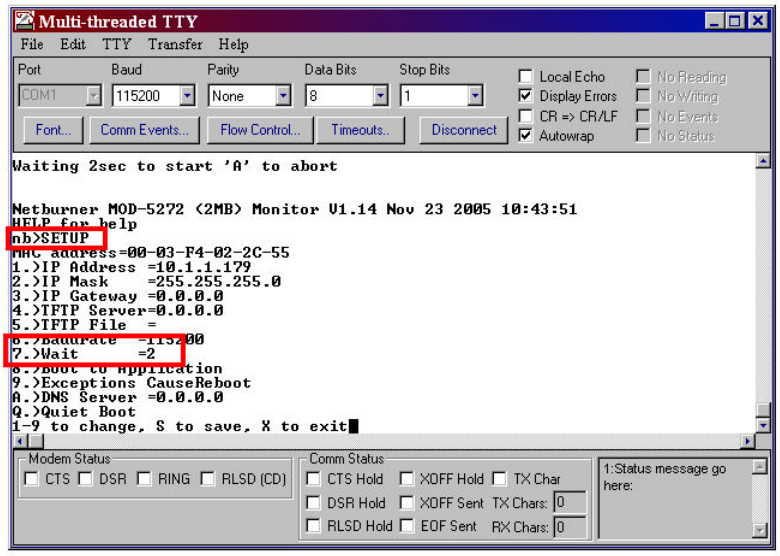


To **change** any the options in the monitor, **type** the command **SETUP** (at the nb> prompt) and **press** the **Enter** key. See the screen shot on the next page for a list of the parameters that can be changed. See section **16.3.5** for additional information on the **setup** command.

Note: The **Quiet Boot** option (item **Q**) will **disable** the boot message that is normally printed out when the system is booting. However, it is still possible to press an A (uppercase A) to access the nb> prompt during boot.

For additional information on the **NetBurner monitor**, please read to the NetBurner Monitor section (**Chapter 16**) in this Manual.

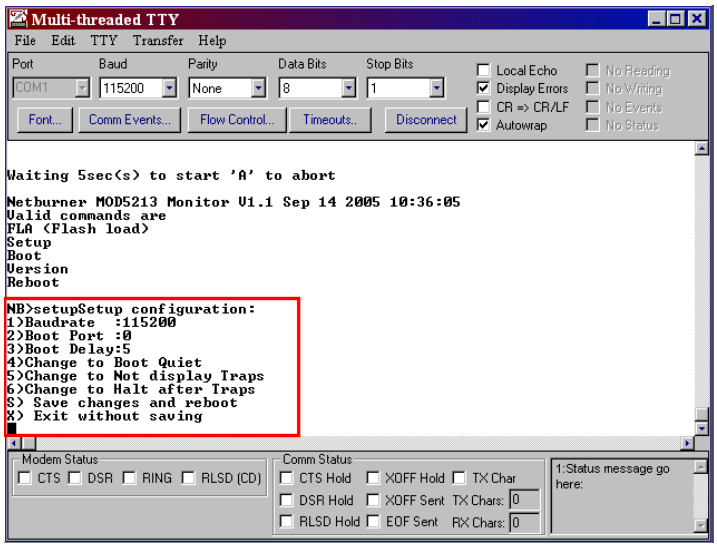
Warning: DO NOT CHANGE THE WAIT TIME (ITEM #7) TO 0 (ZERO)



The function of the boot monitor is for recovery. Therefore, if you change the Wait time (Item #7) to 0 (zero) and download an application in your NetBurner device that does not have Networking/AutoUpdate enabled, you will not be able to get into the NetBurner Monitor using MTTTY (i.e. you will be unable to access the nb> prompt). **Note:** This does **not** apply to any of the non-networking platforms**. Warning: Not all examples (in C:\Nburn\examples) in have Networking/AutoUpdate enabled.**

Since Networking/AutoUpdate is not enabled, you will not be able to see your NetBurner device in the IP Setup window to change the Boot Delay to a value other than zero. Therefore, you will be unable to download any programs in your NetBurner device, and you will have to send your Hardware to NetBurner as an RMA. The setup options for the Mod5213 are shown below.

Warning: IF YOU ARE USING THE MOD5213, DO NOT CHANGE THE BOOT DELAY (ITEM #3) TO 0 (ZERO)



6. IPSetup

Synopsis:

`IPSetup.exe`

Description:

IPSetup is a Win32 program used to configure NetBurner hardware through a network connection. **Warning: This section does not apply to any of NetBurner's Non-Network kits (e.g. Mod5213)**. Running IPSetup will **automatically** identify your NetBurner device (on your network) even if its IP Address is 0.0.0.0. IPSetup can be executed in five ways.

- From **NBEclipse**: Click on the NBEclipse pull-down menu and then select IPSetup
- From **NetBurner's Dev C++**: Click on the Tools pull-down menu and then on IPSETUP
- From **Windows**: Start → Programs → Netburner NNDK → IP Setup tool
- From the **DOS** command line: Navigate to the **C:\Nburn\pcbin** directory (default installation). **Type** the command **ipsetup** then **press** the **Enter** key.
- From Windows **Explorer**: Navigate to the **C:\Nburn\pcbin** directory (default installation) and **double click** the **IPSetup.exe** icon

Your NetBurner device **must** be **connected** to your Network via a standard ethernet cable (or connected to your host computer using a crossover cable) **and** be **recognized** in order to run IPSetup. For hardware setup instructions, please read your (hard copy) Quick Start Guide.

Warning: Not all of the NetBurner example programs (in C:\Nburn\examples) have networking enabled. If you download an application to your NetBurner device that does **not** have networking enabled, your NetBurner device will **not** be recognized by IPSetup, and you will be **unable** to use AutoUpdate to download applications to your NetBurner device.

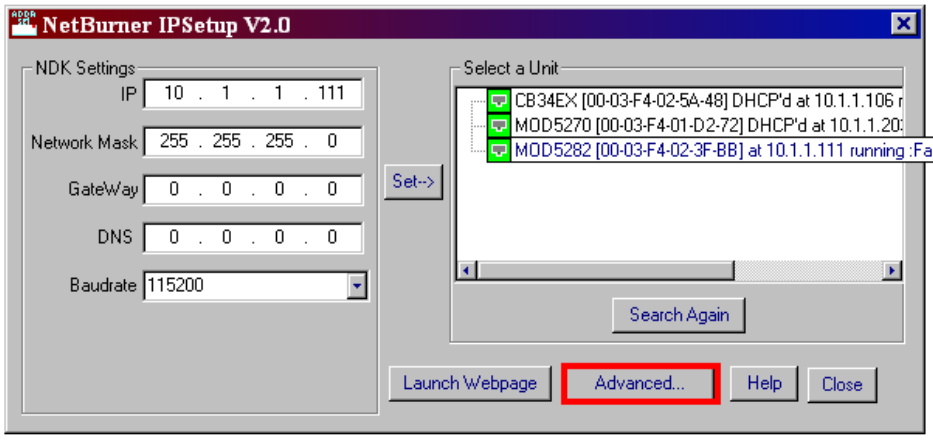
Note: If you **currently** have a program in your NetBurner device that does **not** have networking enabled, you **will** need to use MTTTY to download another application (with networking enabled) in your NetBurner device.

IPSetup will identify **all** NetBurner devices connected to your LAN. (See the screen shots on the next page.) **Note:** If your NetBurner device has **not** been initialized, and its IP Address is 0.0.0.0, it **can** be identified by its **unique** 48-bit Ethernet address.

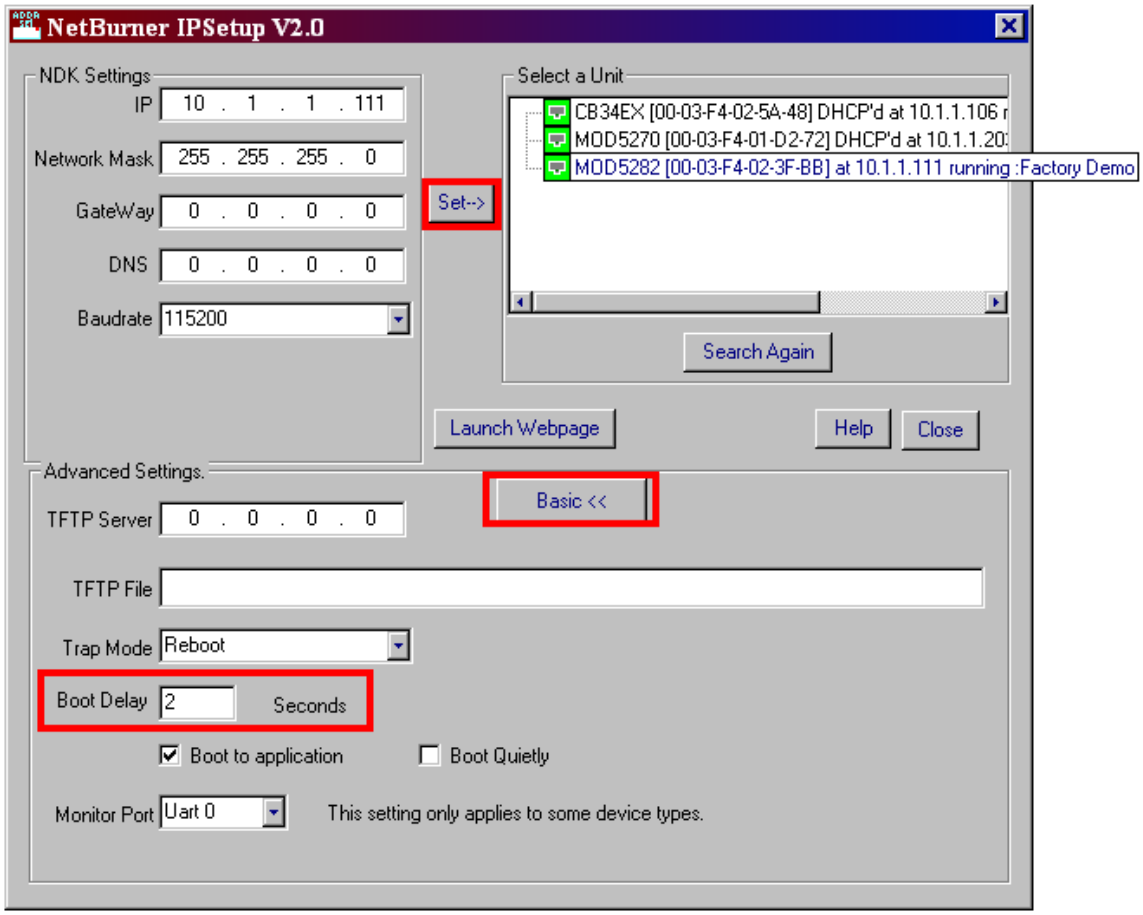
Procedure:

- **Open** up IPSetup using any one of the five methods described above. **Select** your NetBurner device from the device list (i.e. the Select a Unit pane). The IP Address, IP Mask, and IP Gateway **will** appear in their respective text boxes if you are using a Static IP Address as shown in the screen shot on the next page**. Note:** By **default**, all NetBurner devices are setup for **DHCP**. If only one NetBurner device has been identified, it is automatically selected.

- **Modify** the text box fields (if needed).



- **Click** the **Advanced** button to input **additional** values (**if needed**) for the selected NetBurner device. **Note:** The button name **will** change to **Basic** (and vice versa). The screen shot below shows IPSetup's Advanced Settings for a Mod5282 module with a Static IP Address.



If you are using a **TFTP Server**, enter the IP Address of your TFTP Server in the TFTP Server text box, **and** enter the file to use for TFTP downloads in the **TFTP File** text box (if needed).

The **Trap Mode** setting has three options: Reboot, Halt, and Quiet Reboot. **Note:** The factory default is **Reboot**.

The default **Boot Delay** for most NetBurner devices is 2 seconds. The function of the boot monitor is for recovery. Therefore, **it is very important that you set the Boot Delay to any number other than 0 (zero).**

We recommend a Boot Delay of 2 seconds.

Warning: Do not change the Boot Delay to 0 (zero).

By **default**, your NetBurner device boots to the application. If you uncheck **Boot to application**, your program will boot to the monitor.

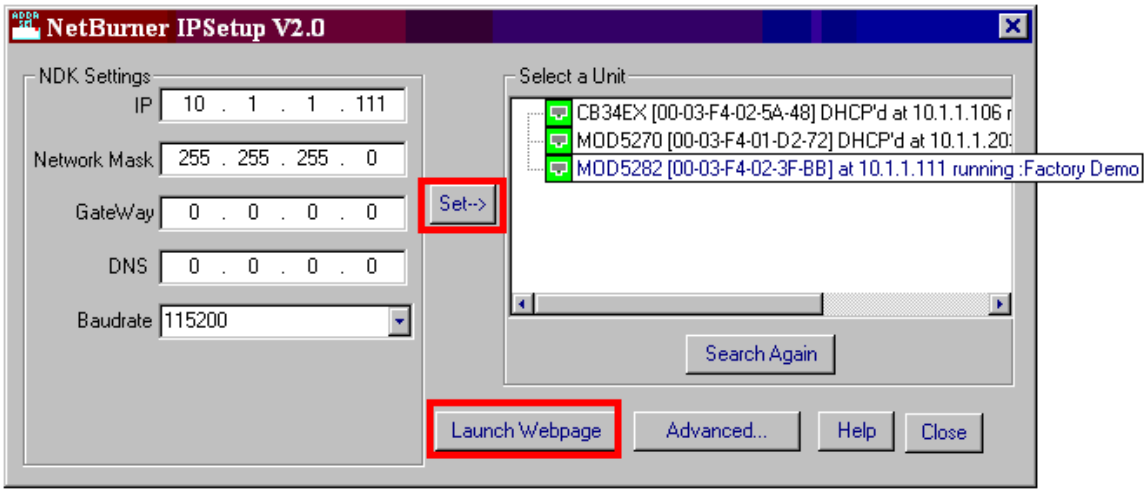
If you select **Boot Quietly**, the boot message that is normally printed out when the system is booting will be disabled. By default, this feature is **not** selected, and the boot message will be printed out when the system is booting.

The **Monitor Port** setting has two options: UART 0 and UART 1. The default is UART 0.

Click the **Set** button (in the center of the IPSetup window, as shown below), when you are finished programming the new values for the selected device**.**

Warning: If you do not click the Set button, your values will not be saved for the selected NetBurner device.

If you are using your NetBurner device as a **Web Server**, you can **view** your web page immediately by **clicking** the **Launch Webpage** button at the bottom of the IPSetup GUI as shown below.



7. NBFind

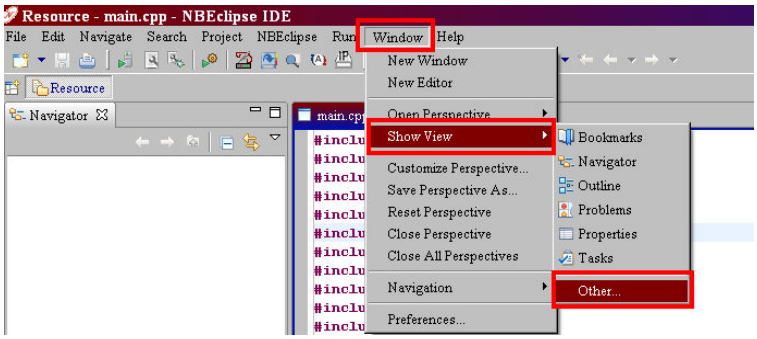
Description:

A new tool (a **view**) to the NetBurner **NBEclipse** tool suite is NBFind. **Warning: This section does not apply to any of NetBurner's Non-Network kits (e.g. Mod5213)**.

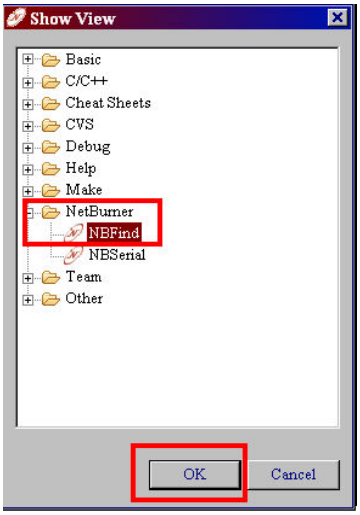
With a single click, NBFind allows you to build and load applications to multiple NetBurner hardware platforms across your network. **NBFind will only work when in the NBEclipse environment.**

NBFind requires both Java Version 1.5 (or greater) and Version 2.0 or greater of the NetBurner Tools installed on your host computer. To find out what **version** of the tool set you are using (using Window's Explorer), **navigate** to your Nburn (root) directory (**C:\Nburn** by default), and **open** up (e.g. in notepad) the **release_tag** file.

To access NBFind **within** NBEclipse, from the **Window** pull-down menu select **Show View** then select **Other...** as shown below.

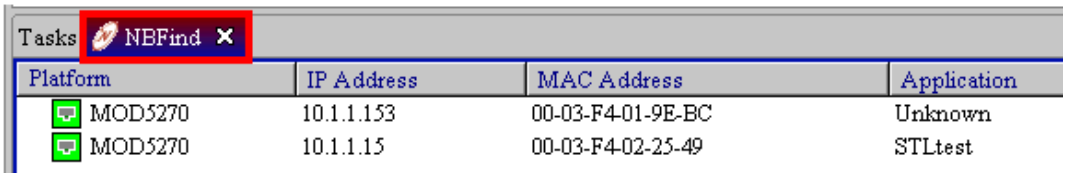


A **Show View** window will appear, **expand** the **NetBurner** folder (by **clicking** on the **+** sign), select **NBFind** (as shown below), and **click** the **OK** button.



The NBFind (view) pane will appear (by default) at the bottom of the NBEclipse IDE. For each NetBurner device found on your network, NBFind will display the NetBurner hardware (platform), its IP Address, its MAC Address, and the current Application's name (optional).

Note: If you did **not** give your application a **name** when you created it, you will see **Unknown** under the Application section (as shown below).



8. NBTFTP

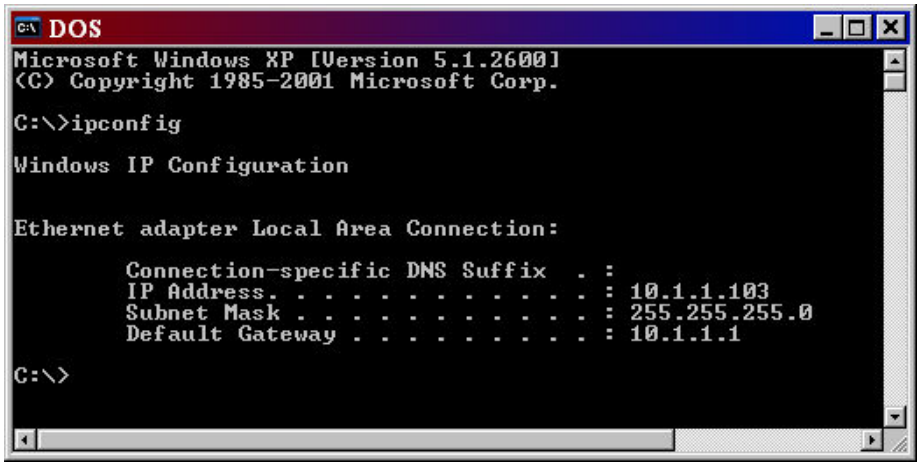
Synopsis

NBTFTP.exe

Description

NBTFTP is a simple TFTP Server for Win32 based computers**. You must have a Static IP Address and a Network Mask assigned to your NetBurner device in order to use NBTFTP**.
Warning: DHCP will not work.

You **will** also need to enter the IP Address of your host computer as the TFTP server (item # 4 in the MTTTY SETUP screen). **Note:** To find out what your host computer's IP Address is: **Open** up a **DOS** window, **type** the command **ipconfig**, and **press** the **Enter** key (an example is shown below).

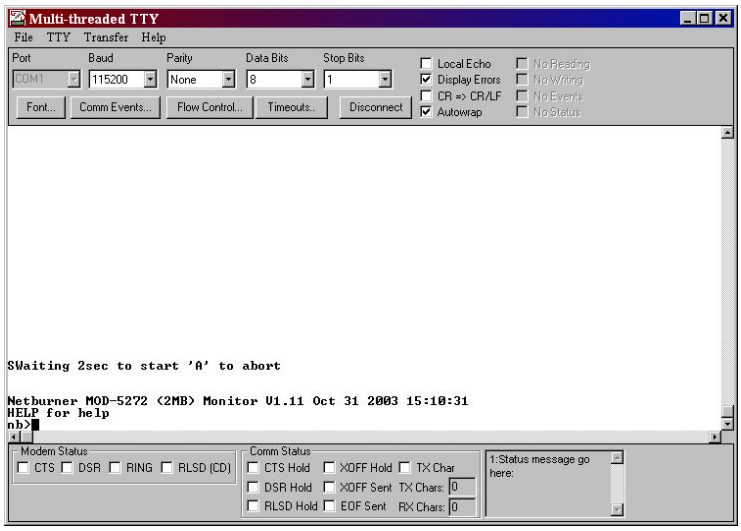


Warning: This section does not apply to any of NetBurner's Non-Network kits (e.g. Mod5213).

Procedure

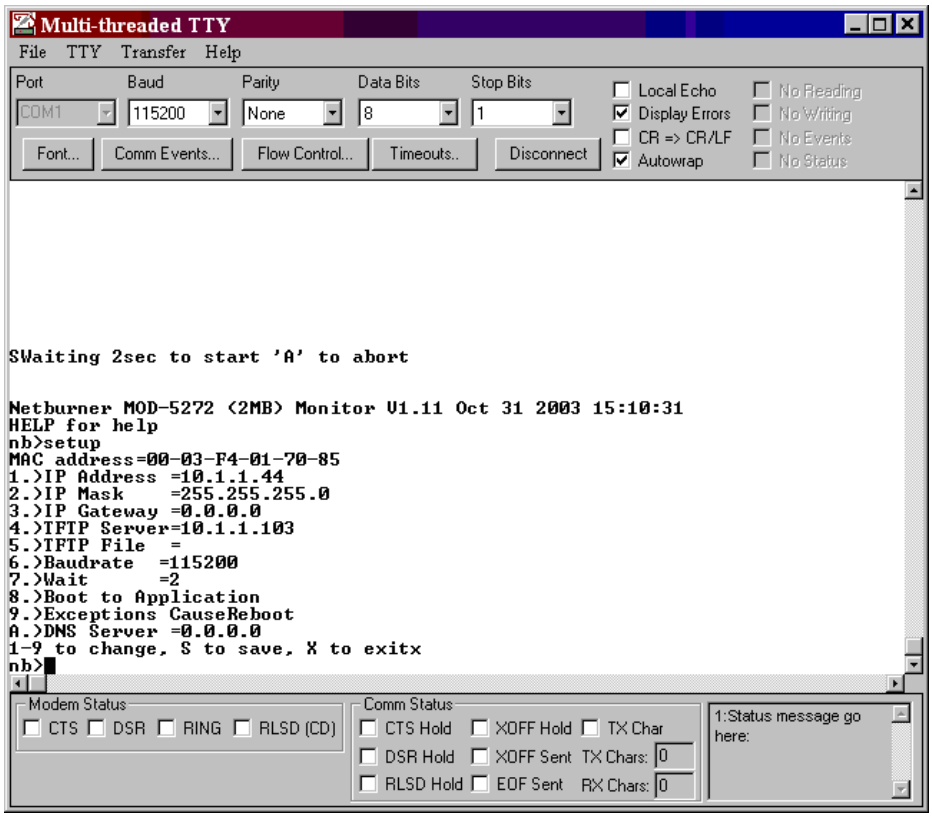
- **Connect** the supplied serial cable (or supplied NULL modem cable for the SB72EX and CB34EX platforms) from the serial port on your NetBurner device to your host computer's serial port.

- **Start MTTTY** on your host computer, **connect** to your NetBurner board, and **type** an **A** (i.e. an **uppercase A**) in the MTTTY window **before** the time expires to access the NetBurner monitor. You **will** see the **nb>** prompt (as shown in the screen shot on the next page).



The MTTTY screen shot **below** shows the **required** information you need in order to use NBTFTP (including an example of a Static IP Address (item 1) and an IP Mask (item 2) to use for your NetBurner device).

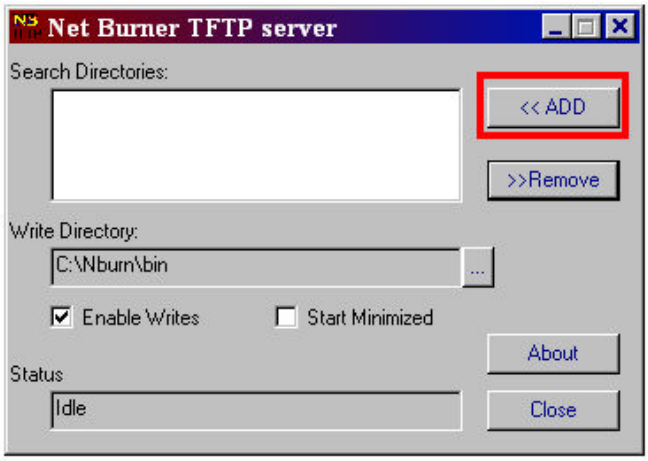
Remember to enter your **host computer's IP Address** as the **TFTP Server** (item 4) **before** running the NBTFTP application. After you are finished, **press** the **S** key on your keyboard to **save** your settings, and then **press** the **X** key on your keyboard to **exit** out of the MTTTY setup screen. You **will** be returned to the **nb>** prompt (as shown below). **Note:** Leave the MTTTY application **running** on your host computer.



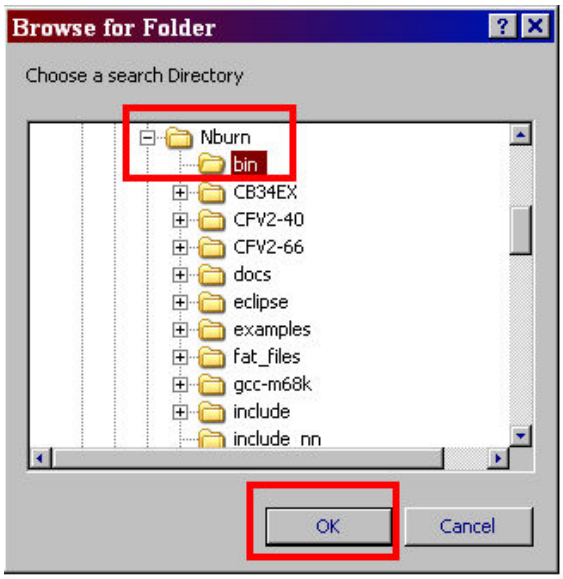
Execute the **NetBurner TFTP server** application (NBTFTP) by one of three ways:

- From **Windows**: Start → Programs → Netburner NNDK → TFTP Server.
- From **Windows Explorer**: First navigate to the **C:\Nburn\pcbin** directory (default installation). Next, double click the **NBTFTP.exe** icon.
- From a **DOS command line**: Open up a DOS window. Navigate to the **C:\Nburn\pcbin** directory (default installation). **Type** the command **nbtftp** then **press** the **Enter** key.

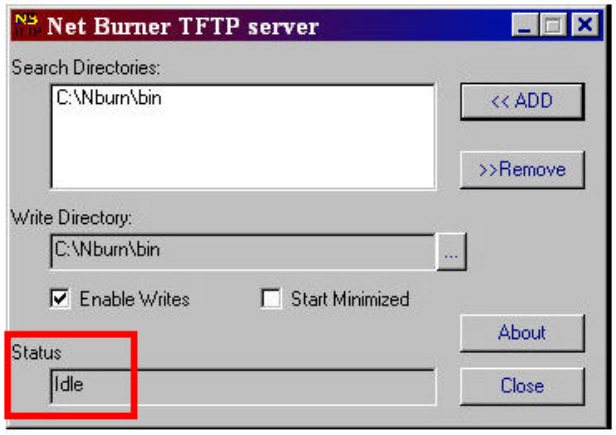
When the **Net Burner TFTP server** GUI appears (as shown below), **click** the **<<ADD** button.



A **Browse for Folder** window will appear (as shown below). **Navigate** the **C:\Nburn\bin** directory (the default location for applications compiled with NetBurner's Dev C++ or at the command line). **Click** the **OK** button **after** you have selected your directory. **Note:** If you are using **NBEclipse**, the **application** file will be in your **project's directory**. Therefore, navigate to that directory versus C:\Nburn\bin. **Click** the **OK** button **after** you have selected your project's directory.



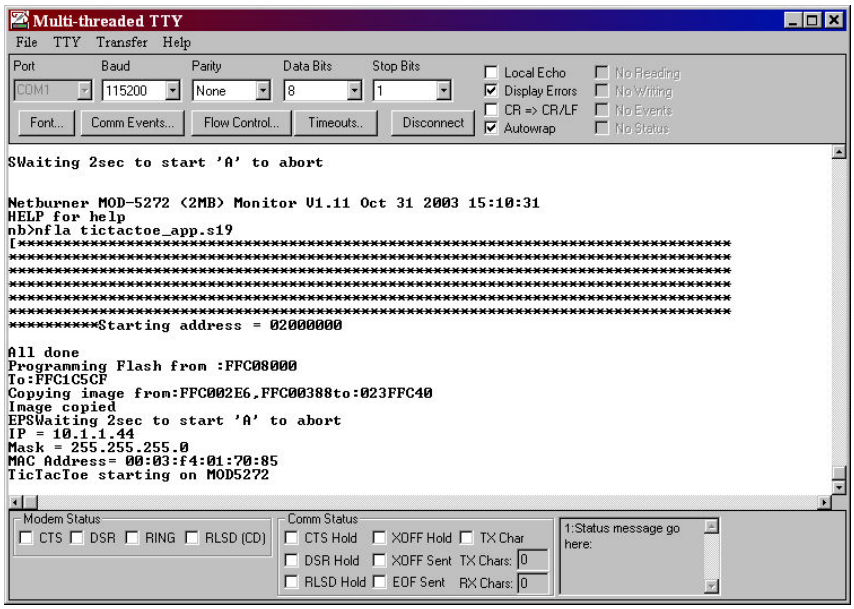
Your Net Burner TFTP server GUI should look like the screen shot below. **Note that the server status is Idle.**



Warning: Do not close the NetBurner TFTP server program.

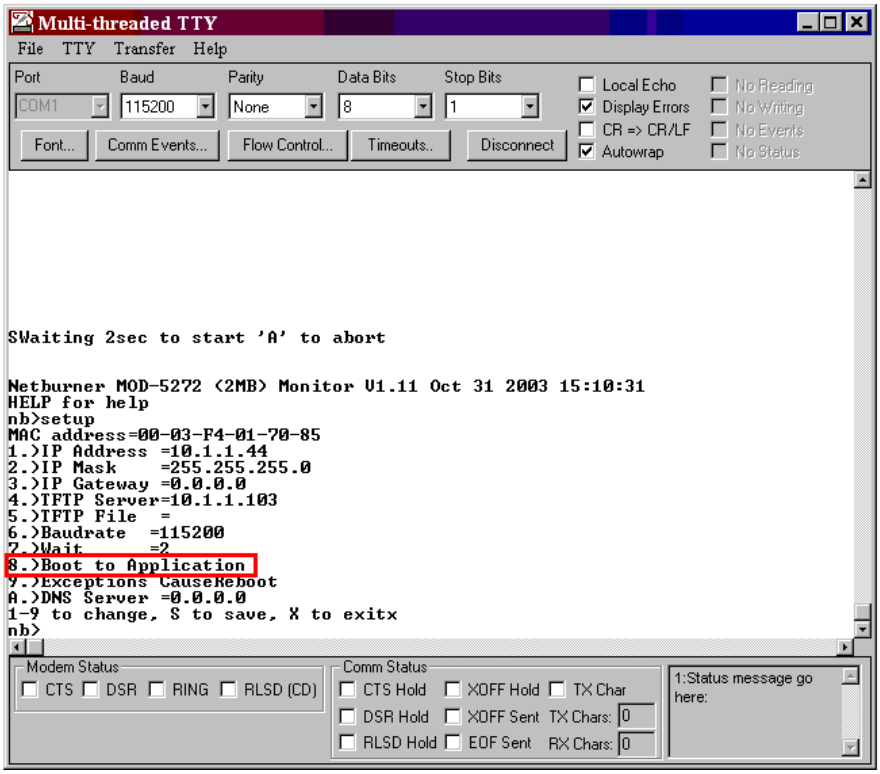
At the **nb>** prompt in the **MTTTY** window **type** the command **nfla <FileName_app.s19>** for a **FLASH** download, or **ndl <FileName.s19>** for a **SDRAM** download, (remember to replace FileName_app.s19 or FileName.s19 with your .s19 application file) then **press** the **Enter** key.

For this example, I previously compiled the tictactoe application (located in the C:\Nburn\examples directory). I used the tictactoe_app.s19 file (for a Flash download) for my TFTP download. At the nb> prompt, I typed the command nfla tictactoe_app.s19 then pressed the Enter key to start my TFTP download. As my download progressed, "********" characters appeared in the MTTTY window, and the progress bar on the lower left hand side of the window moved towards the right. When the FLASH download was finished, my NetBurner device automatically rebooted, and the TicTacToe application started, as shown in the screen shot below.

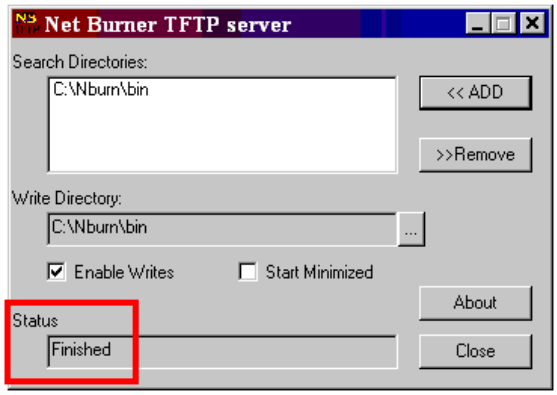


Important: If your new program does **not** automatically **start**, automatic loading may be turned off. If this is the case, just **type** the command **BOOT** at the nb> prompt and **press** the **Enter** key to start your new program. **Note:** To turn **automatic loading on**, just **change** item number **8** in the **MTTTY setup screen** from Boot to Monitor to **Boot to Application**. (See the screen shot below.) Remember, programs downloaded to FLASH memory will overwrite the existing program in your NetBurner device.

If downloading to **SDRAM**, just **type** the command **GO** at the **nb>** prompt and **press** the **Enter** key to start the program. Remember, programs downloaded to SDRAM will be lost when the power to your NetBurner device is turned off.



After the TFTP download is complete, the **Status** of the NBTFTP Server **will** change from Idle to **Finished** (as shown below).



9. CompCode

Synopsis

```
CompCode.exe <infile> <outfile> <outputaddress>
```

Description

CompCode converts an input file in S record format into binary. It then compresses this binary image, prepends a structure describing the compressed image, and outputs it as S records in a different file.

For additional **help** with compcode, (at the DOS command line) **type** the command **compcode -h** (in the **C:\Nburn\pcbin** directory) and **press** the **Enter** key.

Warning: The NetBurner Monitor will only store applications in FLASH that have been compressed with this utility.

Parameters

Name	Optional	Description
infile	No	The input S record file.
outfile	No	The output S record file.
outputaddress	Yes	The offset address for the output data. **Note:** This defaults to 0xFF00000.

Command Line Procedure

Open a DOS window and navigate to the **C:\Nburn\pcbin** directory (default installation). **Type** the command: **compcode myapp.s19 myapp_APP.s19 0xff00000** and **press** the **Enter** key.

Remember to replace myapp.s19 and myapp_APP.s19 with your files.

10. CompHtml

Synopsis

```
comphtml <directory> <options>
```

Description

The comphtml.exe command is designed to package web pages to be used in NNDK applications. **This section does not apply to any of NetBurner's non-network kits (e.g., MOD5213)**.

For additional help with comphtml.exe, type the following (in the \Nburn\pcbin directory unless you opted to install the NNDK environment variables at the time of installation) at the DOS command prompt and then press the <Enter> key:

```
comphtml -h
```

Comphtml.exe does three things:

- Encodes an entire directory structure into a *.cpp file
- Adds tables and encoding for dynamic HTML
- Determines the encoding type for each stored file

Parameters

Name	Optional	Description
<directory>	No	The path to the HTML directory to encode.
-o<outputfilename>	Yes	The file to put the encoded pages into. Note: This defaults to htmldata.cpp.
-d<defaultpage>	Yes	The default HTML page for the NetBurner web server to load when no page is specified. Note: This defaults to INDEX.HTM.

Command Line Procedure

1. Open a DOS command prompt window and navigate to the \Nburn\pcbin directory (you do not need to navigate to the path if you installed the NNDK environment variables at the time of installation).
2. Type the command comphtml –omydata.cpp –dfirstpage.html and then press the <Enter> key. Remember to replace "mydata.cpp" and "firstpage.html" with your files. All file names and web accesses are case sensitive.

11. AutoUpdate

Synopsis

`AutoUpdate.exe -I<ipaddr> -F<filename> -R -A`

Description

AutoUpdate will download a new code image to your NetBurner device. This update can be done from any point that has network access.

Warning: This section does not apply to any of NetBurner's Non-Network kits (e.g. Mod5213).

AutoUpdate typically starts as a WIN32 GUI Application, but it can be run without user intervention by using the -A option. The AutoUpdate application can be executed in five ways:

- From **NBEclipse**: Click on the NBEclipse pull-down menu and then select Auto Update. You can use NBEclipse to compile and load your application in one easy step.
- From **NetBurner's Dev C++**: Click on the Tools pull-down menu and select AUTOUPDATE. You can use NetBurner's Dev C++ IDE to compile and load your application in one easy step.
- From **Windows**: Start → Programs → Netburner NNDK → AutoUpdate tool.
- From the **DOS** command line: Navigate to the **C:\Nburn\pcbin** directory (default installation). **Type** the command **AutoUpdate** then **press** the **Enter** key.
- From Windows **Explorer**: Navigate to the **C:\Nburn\pcbin** directory (default installation). **Double click** the **AutoUpdate.exe** icon.

Note: You can create, compile, and download your program automatically (in one easy step) if you use **NBEclipse**.

In order to use AutoUpdate you must:

- Have either a Static IP Address, or DHCP IP Address and Network Mask assigned to your NetBurner device (the factory default is DHCP).
- Have a working network connection between your host computer and your NetBurner device. Your NetBurner device must be visible in IPSetup.
- Have included #include <AutoUpdate.h> in your application.
- Start network services and make the call to EnableAutoupdate(); in your application.

Parameters

Name	Optional	Description
-Iipaddr	Yes	The IP (either DHCP or Static) Address of the NetBurner device to update.
-FFilename	Yes	The input package file to use for updating your NetBurner device.
-R	Yes	This parameter tells the application to reboot after is has finished downloading. **Note:** If your NetBurner device is **not** set to start the application after reboot (item 8) - your application will **not** start.
-A	Yes	This parameter will run the download automatically **without** intervention.

Command Line Procedure

Open a **DOS** box, navigate to the **C:\Nburn\pcbin** directory (default installation), **type** the command **AutoUpdate -I10.1.1.99 -Fmyapp_APP.s19**, and **press** the **Enter** key.

Remember to replace myapp_APP.s19 with your file and replace 10.1.1.99 with the IP Address of your NetBurner device.

Warning: AutoUpdate functionality is included in most (but not all) NetBurner example programs (located by default in your C:\Nburn\examples directory).

An example of a (minimal) application using AutoUpdate

```
#include "predef.h"
#include <stdio.h>
#include <startnet.h>
#include <autoupdate.h>     // You must include <autoupdate.h>

void UserMain(void * pd)
 {
  InitializeStack();        // Initialize the TCP/IP Stack
  OSChangePrio(MAIN_PRIO); // Set user thread priority
  EnableAutoUpdate();       // Enable code update utility (i.e. AutoUpdate)
    printf("Application started\n");
  while (1)
   {
     // Your application code goes here
   }
 }
```

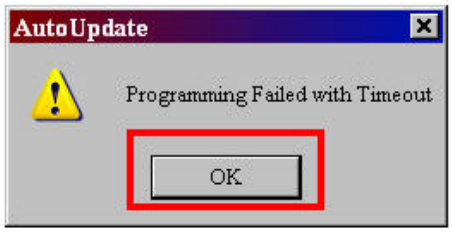
When it is time to compile **and** load your program (from the command line), **type** the command **make load**, then **press** the **Enter** key to execute AutoUpdate. **Important:** You **must** be in your program's directory when you execute this command.

AutoUpdate **will** automatically locate **all** NetBurner devices on your network and provide a list of hardware addresses. **Choose** your particular NetBurner device from the list, **select** your **_APP.s19** file, and **click** the **Update** button to download your application to your NetBurner device.

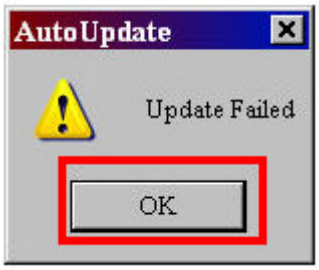
Important: Before an AutoUpdate, the system mallocs a block of RAM big enough to hold the entire image. If there is **not** enough **free RAM**, this step **fails**, and you will get an **insufficient memory** error message. More than likely, you have some process doing malloc or new, and **not** freeing the space.

Warning: The first time AutoUpdate is run, a (one time only) timeout error (shown below) will occur because a target NetBurner device has not been selected.

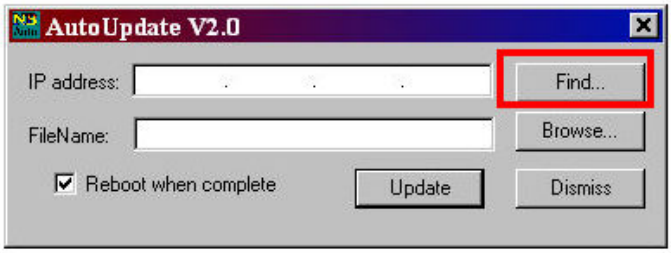
When the **Programming Failed with Timeout** box appears (as shown below) **click** the **OK** button.



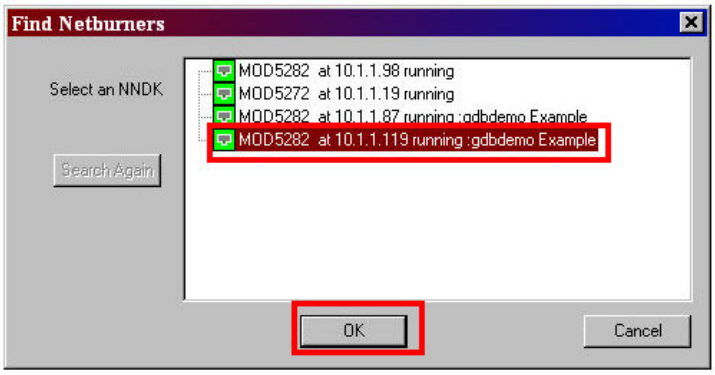
When the **Update Failed** box appears (as shown below) **click** the **OK** button.



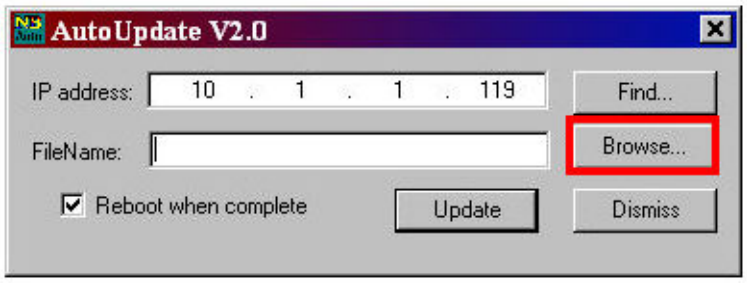
The AutoUpdate GUI will appear (as shown below). **Click** the **Find** button to locate your NetBurner device.



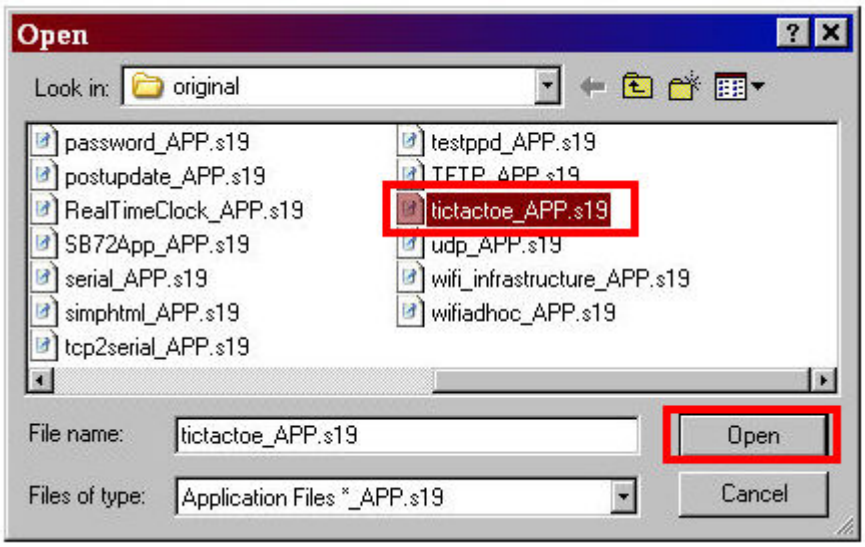
Select your NetBurner device (in the **Find Netburners** GUI), and **click** the **OK** button (as shown below).



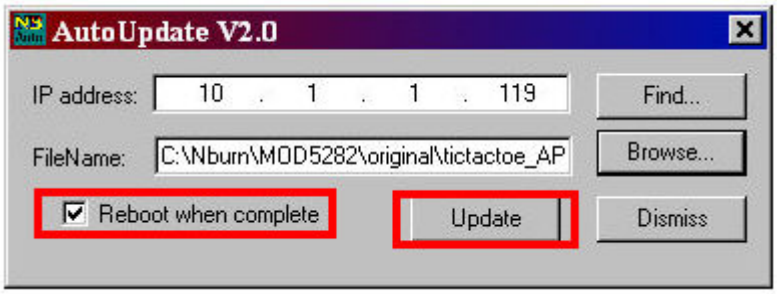
You will be returned to the AutoUpdate GUI. The **IP Address** of your NetBurner device **will** be **displayed.** Next, **click** the **Browse** button and an **Open** window will appear (as shown below).



Navigate to the directory where your **_App.s19** file is located. Next, **select your _App.s19 file and click** the **Open** button (as shown below).



Both the **IP Address** and **FileName** text boxes are **populated** with your information (as shown below). **Check** (enable**) Reboot when complete** to reboot your NetBurner device (as shown below) and **click** the **Update** button to download your application to your NetBurner device.



Important: If you are using either NetBurner's Dev C++ or the Command line to compile your program, your _App.s19 file is located (by default) in C:\Nburn\bin. If you are using NBEclipse, your _App.s19 file is located in your project's directory.

Remember you can create, compile, and download your program automatically (in one easy step) if you use NBEclipse.

12. Application Identification

Synopsis

```
const char * AppName="The name of your Application";
```

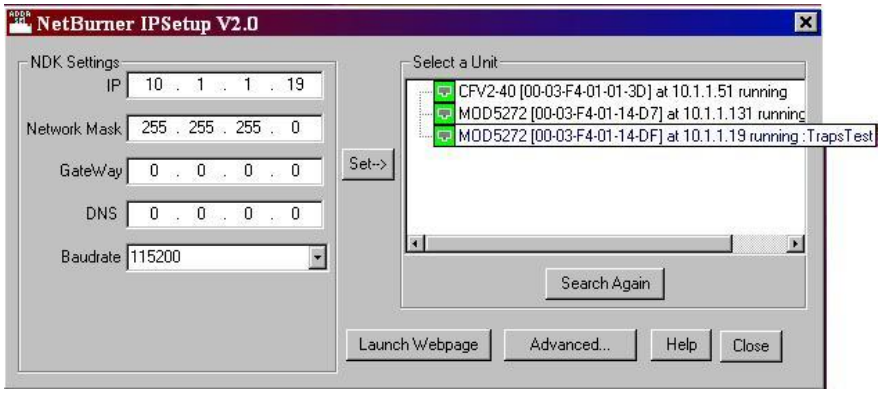
Description

The Application Identification (optional) feature allows for easy identification of your NetBurner device when you have multiple NetBurner devices on your network.

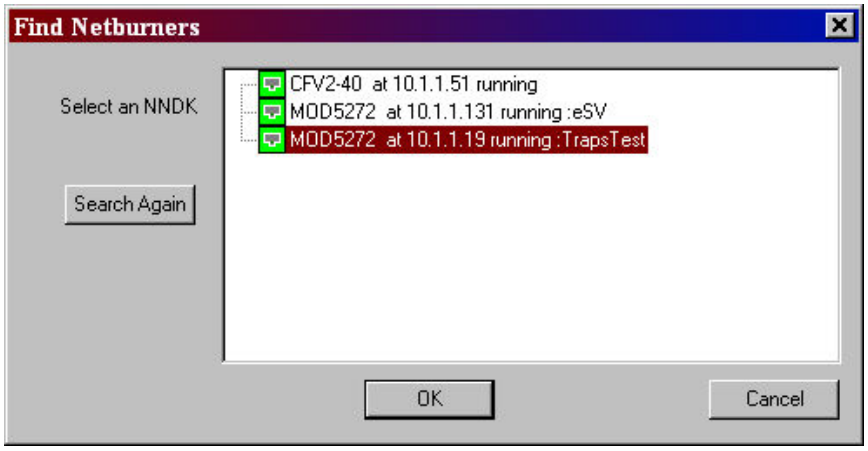
Warning: This section does not apply to any of NetBurner's Non-Network kits (e.g. Mod5213).

After a **successful** download, your application's name (in this case TrapsTest) will automatically appear in:

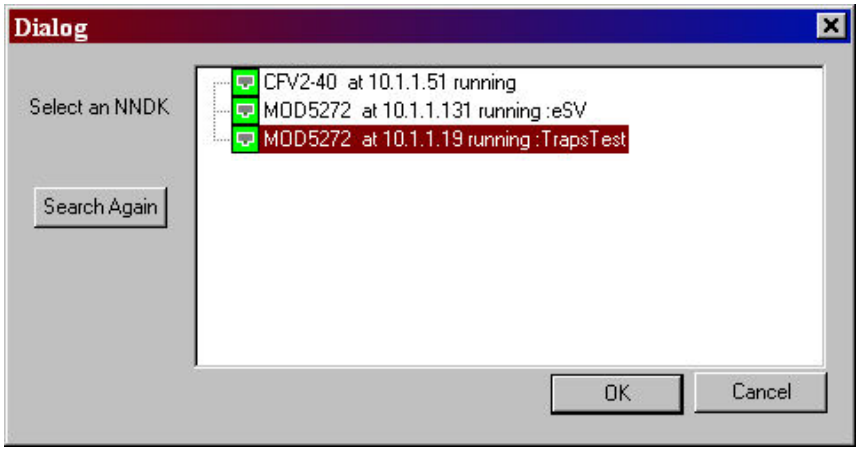
1. The **IPSetup** Select a Unit pane:



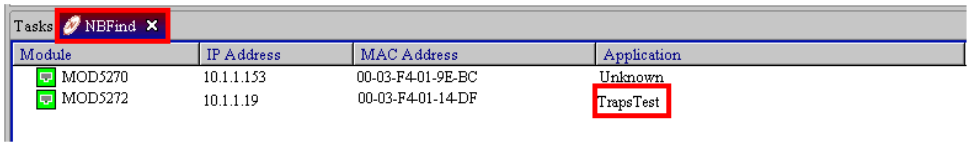
2. The **AutoUpdate** Find Netburners window:



3. The **TaskScan** Dialog window:



4. The **NBFind** (in **NBEclipse**) view:



Example Application with the Application Identification Feature Included

```
#include "predef.h"
#include <stdio.h>
#include <ctype.h>
#include <startnet.h>
#include <autoupdate.h>
#include <dhcpclient.h>
#include <taskmon.h>
#include <smarttrap.h>

// Instruct the C++ compiler not to mangle the function name
extern "C"
{
void UserMain( void * pd);
}

// Name for development tools to identify this application
const char * AppName="My Application";

// Main task
void UserMain( void * pd)
{
    InitializeStack();
    if (EthernetIP==0)GetDHCPAddress();
    OSChangePrio( MAIN_PRIO );
    EnableAutoUpdate();
    StartHTTP();
    EnableTaskMonitor();
    EnableSmartTraps();

    iprintf( "Application started\r\n" );
    while ( 1 )
    {
        OSTimeDly( TICKS_PER_SECOND );
    }
}
```

13. TaskScan

Synopsis

`taskscan.exe`

Description

TaskScan is an **optional** network-debugging tool that is used to view all of the tasks (and their status) in your application. **Note:** To use TaskScan you **must** add **#include <taskmon.h>** in your application's main.cpp file and **EnableTaskMonitor();** in UserMain.

For additional help, please look at the **Serial example** application with TaskScan included. This application is located by default in your **C:\Nburn\examples** directory.

Warning: This section does not apply to any of NetBurner's Non-Network kits (e.g. Mod5213).

Important: Your NetBurner device **must** be recognized by IPSetup (if you wish to use the TaskScan tool). **Note:** There are **no** performance hits if you include TaskScan in your application **unless** you call it. TaskScan can be executed in the five ways:

- From **NBEclipse**: Click on the NBEclipse pull-down menu and then select Task Scan.
- From **NetBurner's Dev C++**: Click on the Tools pull-down menu and then select TaskScan.

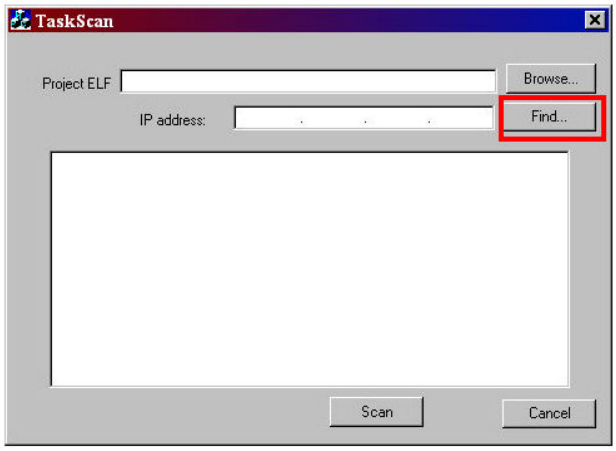
- From **Windows**: Start → Programs → Netburner NNDK → TaskScan.
- From the **DOS** command line: Navigate to the **C:\Nburn\pcbin** directory (default installation). **Type** the command **taskscan** then **press** the **Enter** key.
- From Windows **Explorer**: Navigate to the **C:\Nburn\pcbin** directory (default installation) and **double click** the **taskscan.exe** icon.

Procedure

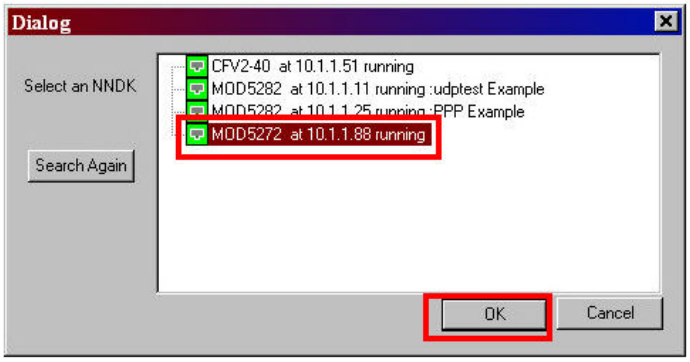
When the TaskScan application is executed (by any of the ways listed above), the TaskScan GUI will appear.

Note: In order to use TaskScan, you **must** know the IP Address of your NetBurner device, and you **must** select your Project's **.elf** file. (This file is always located in the **same** directory as your application.)

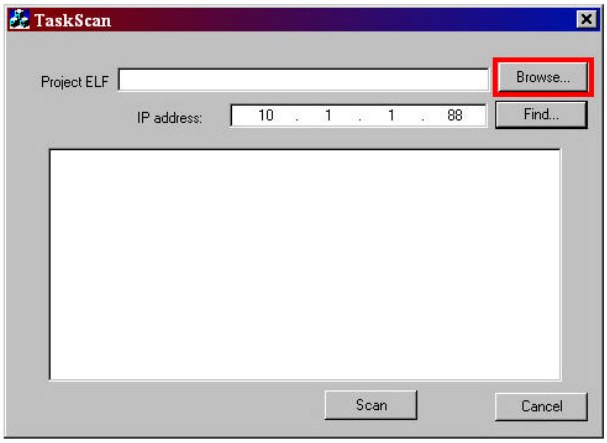
You can **locate** your NetBurner **device** by **clicking** the **Find** button in the TaskScan GUI as shown below.



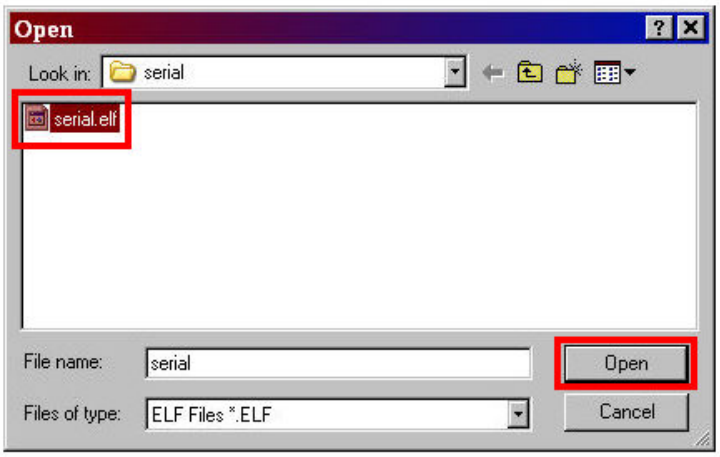
Select your NetBurner device from the NetBurner device(s) listed in the **Dialog** window (as shown below), and **click** the **OK** button when you are finished. **Note:** If only one NetBurner device has been identified, it is automatically selected.



Next, **select** your project's **.elf** file by **clicking** on the **Browse...** button in the TaskScan GUI (as shown below).

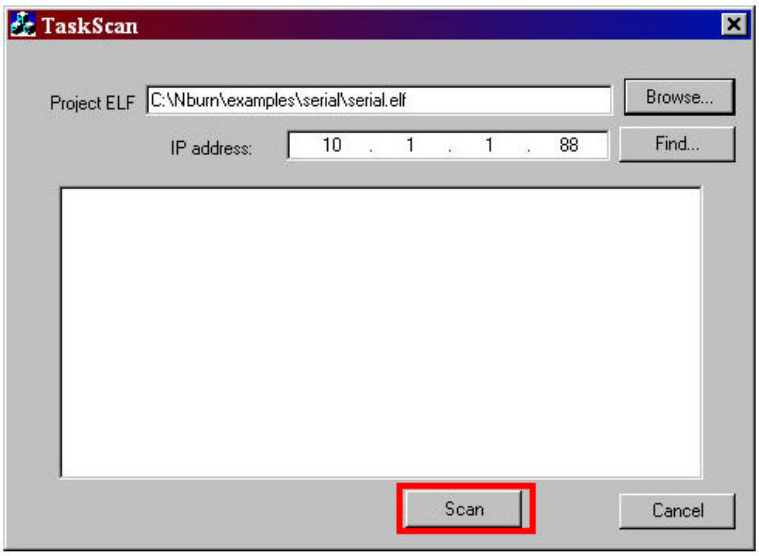


An **Open** window will appear. **Select** your project's **.elf file** and **click** the **Open** button (as shown below). **The .elf file is always located in the same directory as your application.**

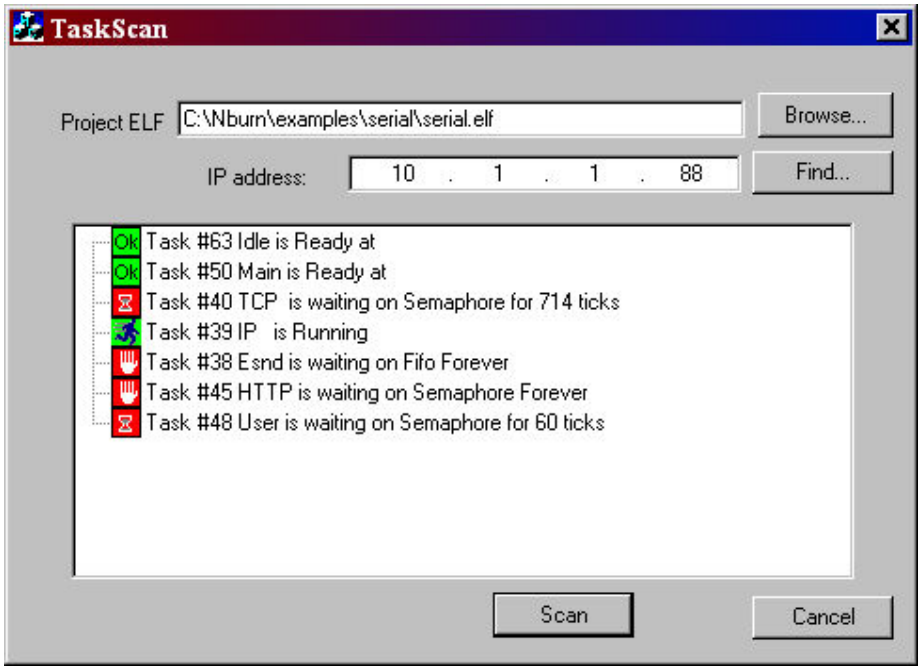


When you are **finished**, both the **Project ELF** and **IP address** text boxes **will** be populated with **your information** (i.e. you have identified your NetBurner device via its IP Address and have

opened up your application's **.**elf file as shown below). Finally, **click** the **Scan** button in the TaskScan GUI.



The TaskScan application with your tasks **will** appear (as shown in the example below).



Legend

Icon	Description
Green "Ok"	The task is ready to run
Green "running man"	The task is currently running
Red "hour glass"	The task is waiting (for a period of time) to start
Red "hand"	The task is waiting and has not started

There are **64 uC/OS tasks**. Task 1 has the highest priority and task 63 has the lowest. **Seven** task priorities are **predefined** in (located in **C:\Nburn\include\constants.h**). They are:

```
#define MAIN_PRIO (50)
#define HTTP_PRIO (45)
#define PPP_PRIO (44)
#define TCP_PRIO (40)
#define IP_PRIO (39)
#define ETHER_SEND_PRIO (38)
#define WIFI_TASK_PRIO (37)
```

For more information on uC/OS tasks, please refer to The NetBurner uC/OS RTOS Library User's Manual.

Example Application with the TaskScan Feature Included

```
1  #include "predef.h"
2  #include <stdio.h>
3  #include <ctype.h>
4  #include <startnet.h>
5  #include <autoupdate.h>
6  #include <dhcpclient.h>
7  #include <taskmon.h>
8  #include <smarttrap.h>
9
10 // Instruct the C++ compiler not to mangle the function name
11 extern "C"
12 {
13 void UserMain( void * pd);
14 }
15
16 // Name for development tools to identify this application
17 const char * AppName="My Application";
18
19 // Main task
20 void UserMain( void * pd)
21 {
22     InitializeStack();
23     if (EthernetIP==0)GetDHCPAddress();
24     OSChangePrio( MAIN_PRIO );
25     EnableAutoUpdate();
26     StartHTTP();
27     EnableTaskMonitor();
28     EnableSmartTraps();
29
30     iprintf( "Application started\r\n" );
31     while ( 1 )
32     {
33         OSTimeDly( TICKS_PER_SECOND );
34     }
35 }
```

14. SmartTrap

Description

SmartTrap is an **optional** debugging tool to help you find out (troubleshoot) where your application trapped. In order to use this tool you **must** include **#include <smarttrap.h>** and **EnableSmartTraps();** in your application.

By default, **smarttrap.h** is located in **C:\Nburn\include** and **smarttrap.cpp** is located in **C:\Nburn\system**.

For the **Non-Network kits (e.g. Mod5213), smarttrap.h** is located in **C:\Nburn\include_nn** and **smarttrap.cpp** is located in **C:\Nburn\system_nn.**

Warning: If you are using the Network debugger, you must put the SmartTrap function before the debugging function. If you put the debugging function first (i.e. before the SmartTrap function), the debugger will not work. This does **not** apply to any of NetBurner's **Non-Network** kits (e.g. **Mod5213**).

Example Program:

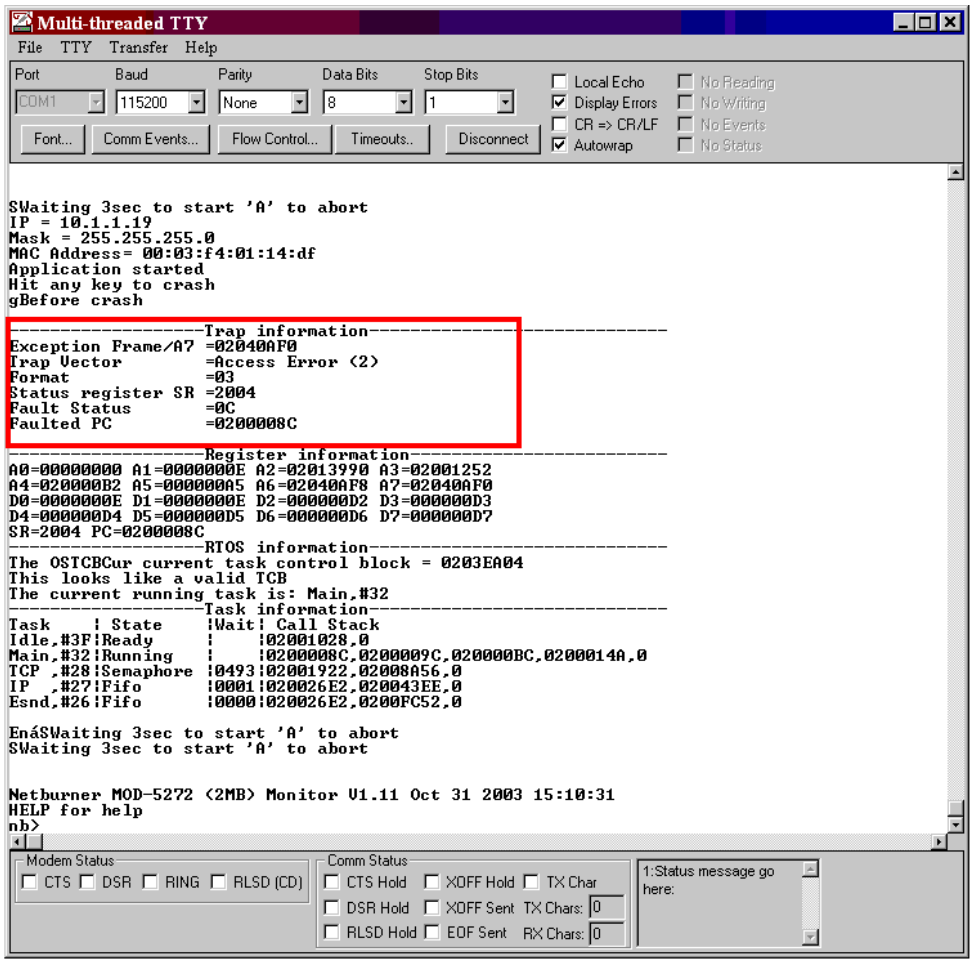
In order to demonstrate the SmartTrap tool, I have created a simple application that will trap when any key on the keyboard is pressed.

```
#include "predef.h"
#include <stdio.h>
#include <ctype.h>
#include <startnet.h>
#include <autoupdate.h>
#include <dhcpclient.h>
#include <smarttrap.h>  /* You must include <smarttrap.h> */

extern "C" {
   void UserMain(void * pd);
 }
  /* The function that will eventually crash. */
  int func4(int i)
 {
  int rv=*(int *)i; /* Line 17 */
  return rv;
 }
  /* func3 calls func4 */
  int func3(int i)
 {
  int rv=func4(i);
  return rv;
 }
  /* func2 calls func3 */
  int func2(int i)
{
  int rv=func3(i);
  return rv;
 }
/* func2 calls func1 */
  int func1(int i)
 {
```

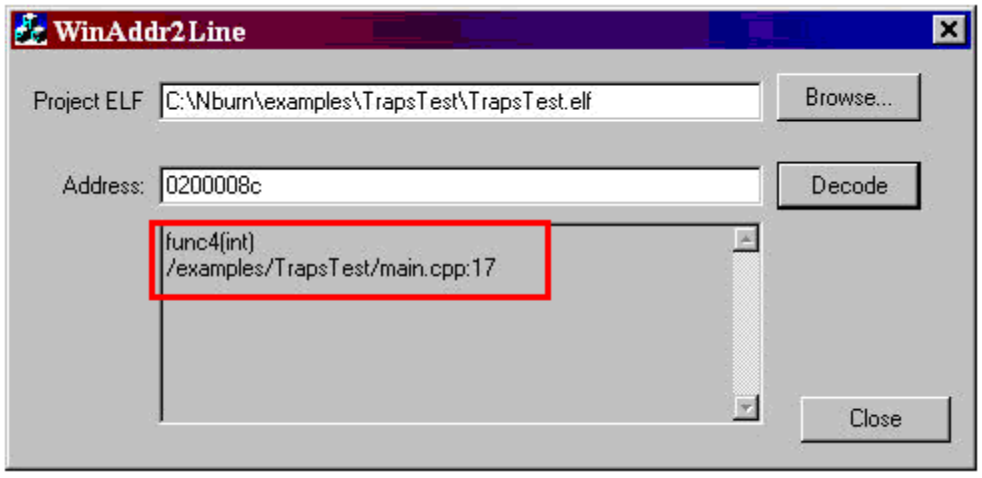
```
 int rv=func3(i);
 return rv;
}
 const char * AppName="TrapsTest using SmartTraps";
 void UserMain(void * pd)
{
 InitializeStack();
 if (EthernetIP==0)GetDHCPAddress();
 OSChangePrio(MAIN_PRIO);
 EnableAutoUpdate();

/* You must call the SmartTraps function in UserMain. */
 EnableSmartTraps();

 iprintf("Application started\n");
 iprintf("Hit any key to crash\r\n");
 getchar(); /* Wait here before we crash */
 while (1)
  {
   iprintf("Before crash\r\n");
   int i=0;
   int q=func1(i);
   iprintf("Func 1= %d\r\n",q);
   OSTimeDly(20);
  }
}
```

I used MTTTY to run my application (after compiling and downloading it to my Mod5272 module). On the next page is a screen shot showing the trap information I received (after pressing the "g" key on my keyboard - causing my program to crash).

At this point, you **need** to notice the **Faulted PC address of 0200008C** (in the Trap information section in MTTTY, shown on the next page). This is where my program crashed.



Using the **WinAddr2Line** application (**see the instructions on the next page**), we find that my program crashed in func4(int) at line 17 in main.cpp.



15. WinAddr2Line

Description:

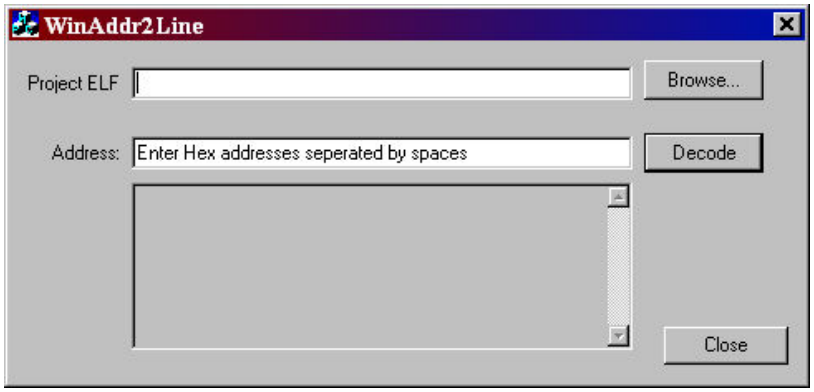
In order to find out where your program crashed using a GUI (versus the command line), use the NetBurner WinAddr2Line application The WinAddr2Line application can be executed in five ways:

- From **NBEclipse**: Click on the NBEclipse pull-down menu and then select WinAddr2Line
- From **NetBurner's Dev C++**: Click on the Tools pull-down menu and then select WinAddr2Line
- From **Windows**: Start → Programs → Netburner NNDK → WinAddr2Line
- From the **DOS** command line: Navigate to the **C:\Nburn\pcbin** directory (default installation). **Type** the command **WinAddr2Line** then **press** the **Enter** key.
- From Windows **Explorer**: Navigate to the **C:\Nburn\pcbin** directory (default installation) and **double click** the **windddr2line.exe** icon

Important: In order to use the WinAddr2Line application, you **must** enter both the **.elf** file and the **Faulted PC address**.

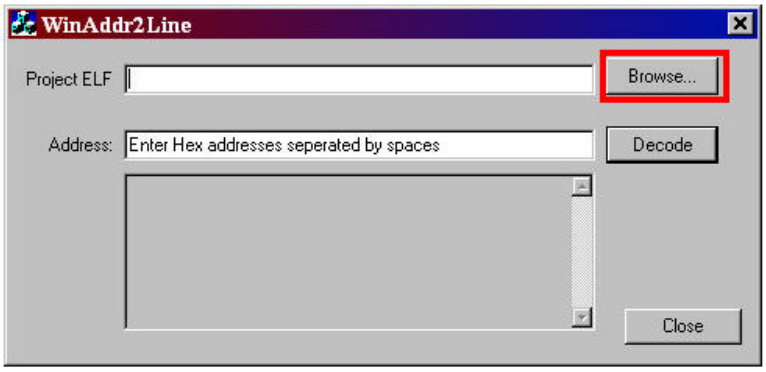
Procedure:

Execute the WinAddr2Line application by any of the five ways listed above. The WinAddr2Line GUI will appear as shown below.

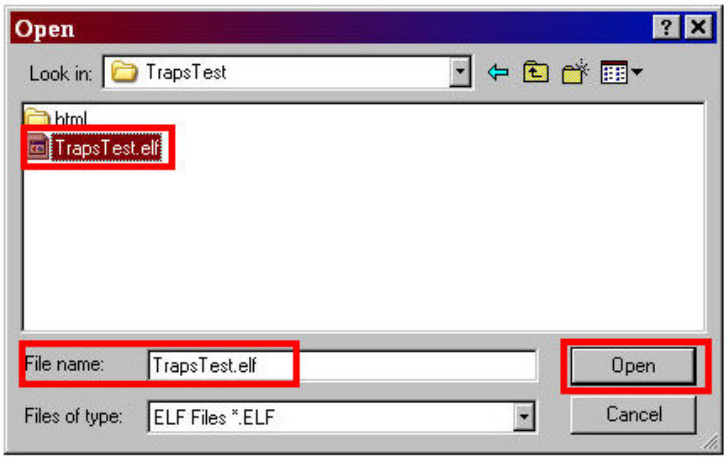


Click the **Browse** button to locate your **Project ELF** file as shown on the next page.

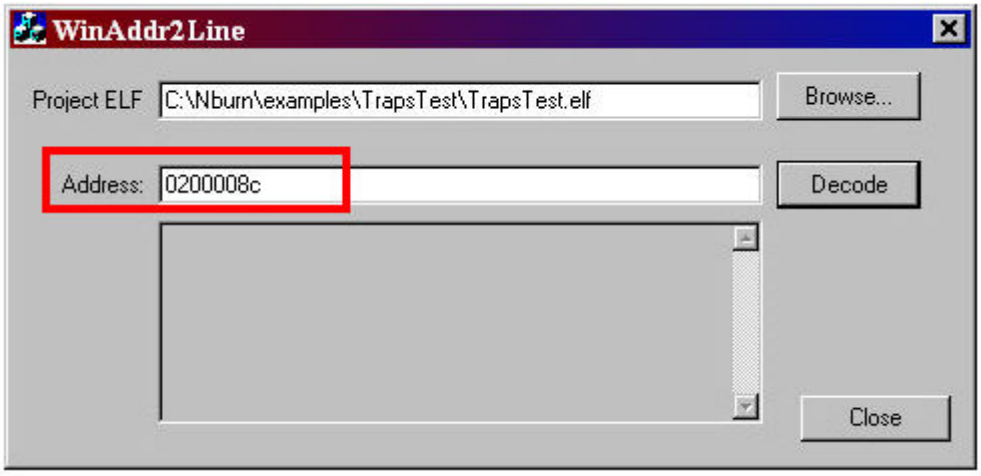
Note: The **.elf file** is always located in the **same** directory as your application.

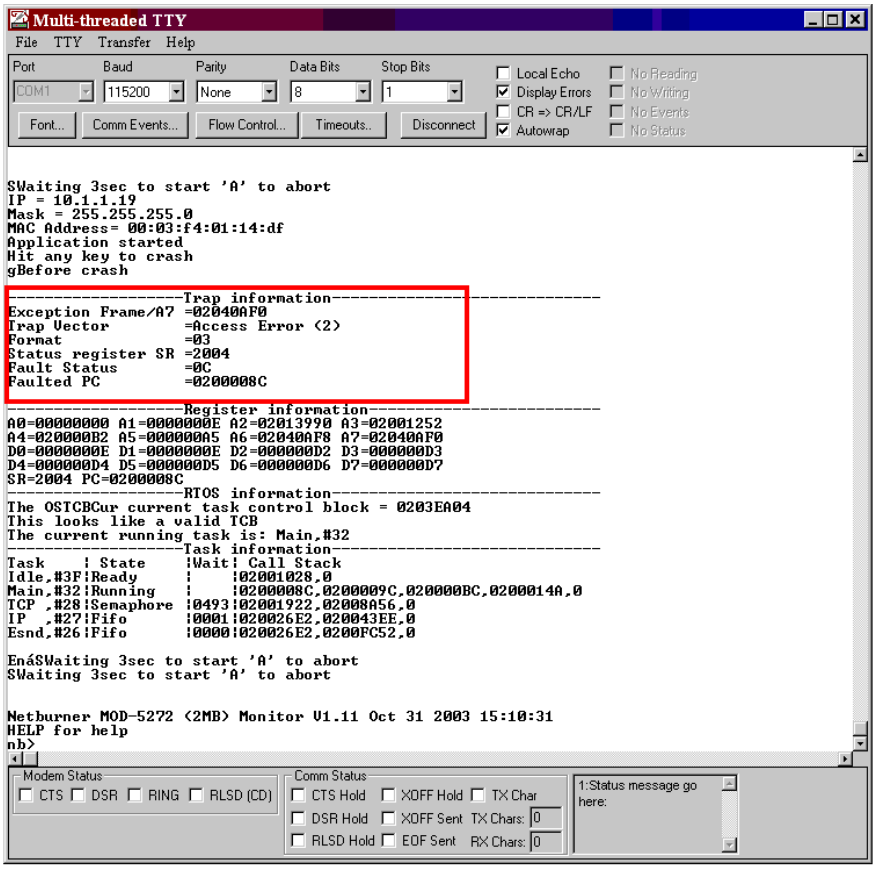


An **Open** window will appear. **Navigate** to your **project's directory**, select the **.elf** file, and **click** the **Open** button (as shown below).

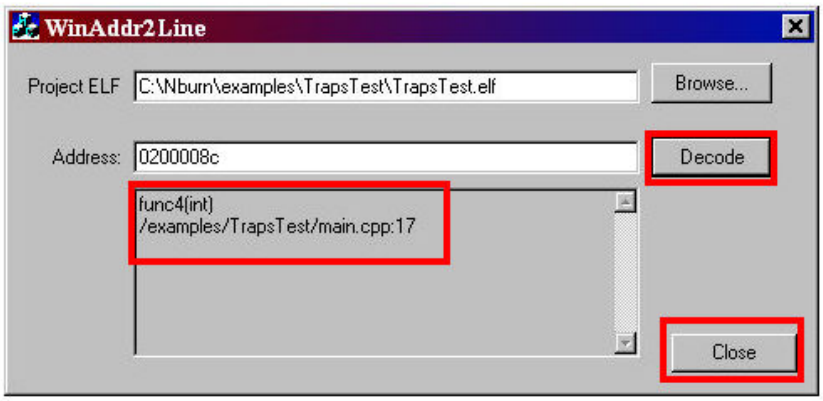


The Project ELF text box is now populated with your **.elf** file path. Next, **type** the **Faulted PC address** (located in the **Trap information** section in the **MTTTY** window, as shown on the next page) in the **Address** text box.





Both the **Project ELF** and **Address** text boxes are **populated** with your information (as shown in the screen shot below). Just **click** the **Decode** button to find out where your application crashed. **Click** the **Close** button to **exit** the WinAddr2Line application.



Important: If the **current** application in your NetBurner device traps, and you have used SmartTrap to (successfully) troubleshoot your problem, download the **original factory application** to your NetBurner device (using MTTTY) to restore it to its factory state. This application is located (by default) in your C:\Nburn\<Your Hardware Platform>\original directory.

16. The NetBurner Monitor

Description:

The NetBurner monitor is embedded in Flash. It initializes your NetBurner device and starts the loaded applications. The monitor provides a user interface, and loads/decompresses the application stored in Flash. It also allows the user to modify settings that control how the device behaves.

Warning: The Mod5213 does not support all of the monitor commands listed on the next page.

Boot Process:

1. The board resets to setup code
2. The board setup code sets up the board's hardware environment
3. The monitor reads the flash parameter area for baud rates etc. and sets them
4. The monitor code looks in Flash to see if an application is to be loaded
5. If no application is to be loaded, it stops and runs the diagnostic and flash loading monitor functions
6. The monitor then pauses for a programmable number of seconds to allow the user to abort the loading process
7. If an application is to be loaded, it uncompress it from Flash to DRAM
8. Checksum the uncompressed image. If this fails, it start the diagnostic and flash loading monitor functions
9. Jump to the _START entry point in the application and run the stock unmodified CRT0

Diagnostic and FLASH Loading Monitor:

To communicate with the diagnostic monitor, attach a serial cable from the serial port on your NetBurner device to the serial port on your host computer, and use a serial communication program such as MTTTY.

- If you are using the **Mod5234, Mod5270, Mod5272, or the Mod5282**, the serial port is the inner serial port (**J7**) on your **Mod-Dev-100** Carrier board. **Note:** If you are using any of the above NetBurner modules with the **Mod-Dev-50** (Promo) Carrier board, the serial port is the inner serial port (**J4**).

- If you are using the **Mod5270LC** with the **Mod-Dev-70** Carrier board, the serial port (**UART 0**) is the inner serial port (**J4**).

- If you are using the **Mod5213** with the **Mod-Dev-40** Carrier board, the serial port (**UART 0**) is the inner serial port (**J4**).

- If you are using the **SB72-EX** (with the supplied NULL Modem cable), the serial port is the left serial port (**Port 0**). You **must use a NULL Modem cable** to connect your SB72EX to your host computer. **Warning: You cannot use a standard serial cable with your SB72EX device.**

- If you are using the **CB34EX** (with the supplied NULL Modem cable), the serial port (default configuration) is the DB9 port (**Port 1**). You **must use a NULL Modem cable** to connect your CB34EX to your host computer**. Warning: You cannot use a standard serial cable with your CB34EX device.**

- If you are using the **SB70** or **SB72** board, the serial port is the inner serial port (**J2**) on the **Adapter/Evaluation board**.

- If you are using the **CFV2-66** board, the serial port is labeled **J2.**

- If you are using the **CFV2-40** board, the serial port is labeled **J4.**

Monitor Commands:

The NetBurner monitor supports the following (simple) commands:

Command	Description
BF	Memory Block Fill
BM	Memory Block Move
BS	Memory Block Search
BOOT	Boots the Application
DL	Downloads S records to DRAM using the Serial port
FLA	Downloads a new application image to FLASH using the Serial port
GO	Starts the execution at the specified address
HELP	Displays a list of the available commands
NDL	Downloads S records to DRAM using TFTP
NFLA	Downloads a new application to FLASH using TFTP
MD	Memory Display
MM	Memory Modify
RD	Register Display
RM	Register Modify
RESET	Resets the NetBurner Hardware
SETUP	Displays the Setup Options
VERSION	Displays the Monitor Version

Warning: The Mod5213 does not support all of the monitor commands listed in this table.

16.1. Download Commands

Synopsis:

Command	Description
DL	Serial download an application to DRAM
NDL <filename>	TFTP download an application to DRAM
FLA	Serial download an application to FLASH
NFLA <filename>	TFTP download an application to FLASH
FLM	Serial download a new monitor image to FLASH
NFLM <filename>	TFTP download a new monitor image to FLASH

Description:

The NetBurner monitor **only** knows how to download Motorola format S records. These S-record files can be sent to your NetBurner device using the serial port, or over the network using the TFTP protocol. **Warning: The Mod5213 does not support all of the monitor commands listed in this section.**

When you **download** an application using the **FLA** or **NFLA** command, the application **must** be in **compressed** form. This compression is done using the **CompCode** utility.

Important: Attempts to **download** any formats **other than S records** will generate an **error**. In addition, if you attempt to **download** an S-record that extends **beyond** the address limits of the **memory map** for the download type specified, the download **will** generate an error. Please refer to your hardware PDF (in **C:\Nburn\docs\platform**) the address limits of your NetBurner device.

Warning: If you download a new monitor image that is bad, there is no way to recover. Your NetBurner device would then need to be returned to NetBurner as an **RMA**, or reprogrammed using a BDM Flash programmer.

- **Serial Downloading:** After executing the serial download command, send the S record over the same serial link. (If you are using MTTTY, press the F5 key on your keyboard to do this.) If the S-record has a terminating S7 record, the download will complete automatically. **Note:** If there is **no** terminating S7 record, type a "**.**" (i.e. a period) to **terminate** the download.

- **TFTP Network Downloading:** The TFTP download commands allow you to use Ethernet to download code from your computer to your NetBurner device. In order for TFTP to work all of the following steps must be done:

    - Set a **valid Static IP Address** for your **NetBurner** device. (**Warning: DHCP will not work**)

    - A TFTP Server program **must** be running on your host computer. **Note:** NetBurner provides **NBTFTP** for this purpose. Please read the NBTFTP section in this manual for additional information and step-by-step instructions.

    - Set the **valid TFTP Server IP Address** in your NetBurner device.

    - The TFTP Server **must** be configured to provide access to the desired files.

16.2. Block Memory Operations

Synopsis:

Command	Description
BF<.w> <start> <end> <value>	Block Fill
BM<.w> <start> <end> <destination>	Block Move
BS<.w> <start> <end> <value>	Block Search

Parameters:

Parameter	Optional	Description
<.w>	Yes	This sets the width of the memory transfer. Legal values are: .b BYTE, .w WORD (default), and .l LONG.
<start>	No	The starting address for the block operation.
<end>	No	The ending address for the block operation.
<value>	No	The value to use in the search or fill.
<Destination>	No	The destination for the copy operation.

All numerical parameters are in hexadecimal

Examples:

BM.b 10000 100ff 20000 --- Will move ff bytes from 10000 to 20000

BM 10000 100fe 20000 --- Will move words from 10000 to 20000

BF.l 10000 11000 DeadBeef --- Will fill 10000 to 11000 with DEADBEEF

BS.l 10000 30000 42414420 --- Will find 42414420 ("BAD_")

16.3. Monitor Commands

16.3.1. Boot the Application

Synopsis:

```
Boot
```

Description:

This command causes the monitor to decompress and execute the application stored in Flash.
Warning: If there is no valid application stored in flash, an error message is generated.

Procedure:

Type the command **boot** at the nb> prompt and **press** the **Enter** key on your keyboard.

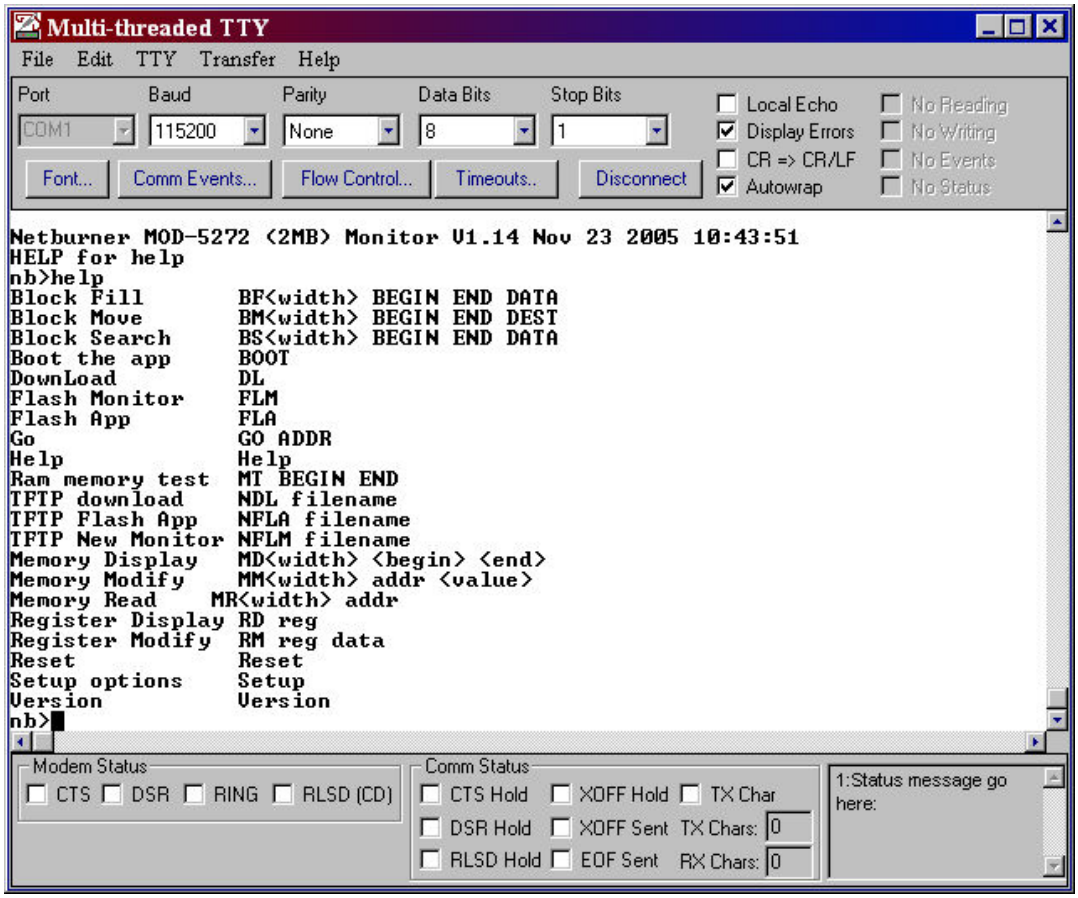
16.3.2. Display Help

Synopsis:

HELP

Description:

This command causes the monitor to display all legal commands, and brief help for each one. See the (for reference only) screen shot for a Mod5272 module below. **Warning: The Mod5213 does not support all of the monitor commands listed in this section.**



Procedure:

Type the command **help** at the nb> prompt and **press** the **Enter** key on your keyboard.

16.3.3. Display the Monitor Version

Synopsis:

VERSION

Description:

This command causes the monitor to display the version and build date.

Procedure:

Type the command **version** at the nb> prompt and **press** the **Enter** key on your keyboard.

16.3.4. Reset and Reinitialize the Monitor

Synopsis:

```
Reset
```

Description:

This command causes the monitor to restart **and** reinitialize all hardware. It will then follow the normal booting procedure.

Procedure:

Type the command **reset** at the nb> prompt and **press** the **Enter** key on your keyboard.

16.3.5. Set/Change Monitor Settings

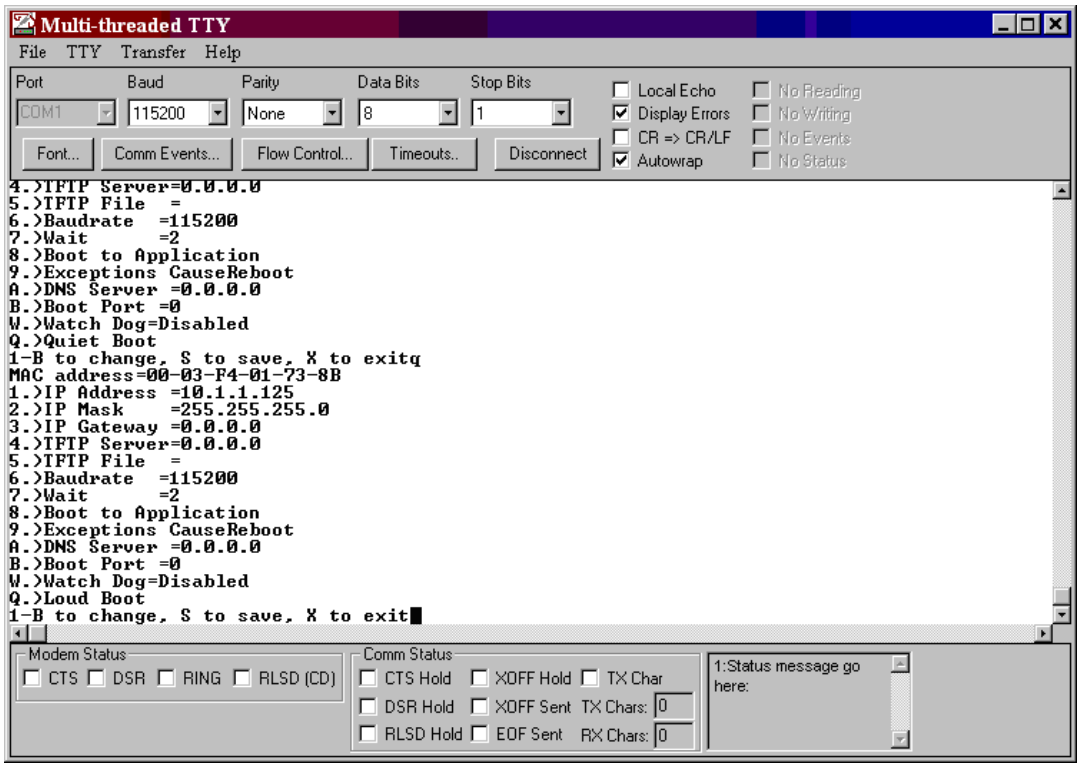
Synopsis

SETUP

Description

This command will cause the monitor to start an **interactive** routine to set and display the (monitor) configuration variables. When you issue the setup command (by **typing** the command **setup** at the **nb>** prompt and **pressing** the **Enter** key) you will see a display similar to the one below. **Note:** Items 1, 2, 3, and 6 can be set automatically by running the IPSetup program.

Warning: The Mod5213 does not support the commands listed in the screen shot below.



To **change** one or more of the variables choose the desired variable via its number. When you have changed all of the variables that you want to change, press the **S** key on your keyboard to **save** your settings. Remember, if you do **not** press the S (for Save) key on your keyboard, the settings will **not** take effect. **After** you have **saved** your settings (by **pressing** the **S** key on your keyboard), **press** the **X** key to **exit t**he setup program. **Note:** For **all** IP Addresses you are **expected** to enter them **in dotted decimal format** (e.g. 10.1.1.11).

Procedure:

Type the command **setup** at the nb> prompt and **press** the **Enter** key on your keyboard.

Parameters:

Name	Description	Possible Values
IP Address	The NetBurner device's IP Address.	Any valid IP address in dotted decimal form.
IP Mask	The NetBurner device's IP Network Mask value.	A valid IP Mask in dotted decimal form.
IP Gateway	The routing Gateway for the NetBurner devices. Any packets not destined for the local segment are sent here.	A valid IP Gateway in dotted decimal form.
TFTP Server	The IP Address of the TFTP Server used when TFTP download commands are used.	A valid IP address of the TFTP server in dotted decimal form.
TFTP File	The file name to use for TFTP downloads when no name is specified.	Any legal ASCII Value.
BaudRate	The default system baudrate.	300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, or 115200 **Note:** Any other values **will** default to **115200**.
Wait	The number of seconds to wait from startup until the Application in flash is loaded and executed. **WARNING: If you do not have networking enabled in your application and you set the wait time to 0, you will be unable to change the wait time.**	0 to 32000. For the **CFV2-40** and **CFV2-66** platforms **only**: A value of Zero can be **overridden** by setting the **dipswitch closest to the board center** in the **ON** position. For all **other platforms**, you can modify this value **only** by using IP Setup.
Boot Mode	The action to take on boot up. There are two choices: **1.** Boot to Application **2.** Boot to Monitor.	Selecting this parameter for change will cycle through the **two** possibilities (Boot to Application or Boot to Monitor).
Watch Dog	A timer that will count down and reset your NetBurner device unless it is reset by the user. The suggested setting for this option, if you are using any of the NetBurner example programs is: **Watch Dog=Disabled**.	Selecting this parameter for change will cycle through the two possibilities (Watch Dog=Disabled or Watch Dog=Enabled). **Note:** By **default,** this parameter is **disabled**.
Quiet Boot	Disables the boot message that is normally printed out when the system is booting.	Selecting this parameter for change will cycle through the two possibilities (Quiet Boot or Loud Boot).
Exception Mode	The Action to take when an exception is encountered: **1.** Cause Reboot **2.** Cause Halt **3.** Cause Quiet Reboot	Selecting this parameter for change will cycle through the **three** possibilities (Cause Reboot, Cause Halt, or Cause Quiet Reboot)

16.3.6. Start Execution

Synopsis:

GO <address>

Description:

This command causes the monitor to begin execution at the **specified** address. **Note:** You will regain control of the system until your NetBurner device is rebooted, or the running code causes an exception.

Parameter:

Parameter	Optional	Description
<address>	Yes	The address at which to start execution. **Note:** If **no** address is **specified**, then the value of the **stored PC register** will be used.

Procedure:

Type the command **go** at the nb> prompt and **press** the **Enter** key on your keyboard.

16.4. Memory Operations

Synopsis:

Command	Description
MD<.w> <start> <end>	Memory Display
MM<.w> <location> <value>	Memory Modify

Parameters:

Parameter	Optional	Description
<.w>	Yes	This sets the width of the memory transfer. Legal values are: .b BYTE, .w WORD (default), and .l LONG.
<start>	No	The starting address for the block operation.
<end>	Yes	The ending address for the display operation.
<value>	Yes	The value to set in the modified location.

Note: Type a "**.**" (i.e. a period) to terminate an interactive modification session.

All numerical parameters are in hexadecimal

Examples:

`MD.b 100000 ---` Will display one screen of bytes starting at 100000

`MM 100000 1234 ---` Will write the word 0x1234 to 100000

`MM.l 100000 ---` Will interactively modify longs starting at 100000

16.5. Register Operations

Synopsis:

Command	Description
RD	Register - Display All
RD <reg>	Register - Display Specific
RM <reg> <value>	Register – Modify

Parameters:

Parameter	Optional	Description
<reg>	Optional for display only	The register to display or modify. [D0..D7 A0..A7 PC SR]
<value>	No	The value to set in the modified register.

All numerical parameters are in hexadecimal

Examples:

RD A0 --- Will display the stored value of A0

RM.SR 2700 --- Will modify SR to 2700

17. GDB and NetBurner

Overview

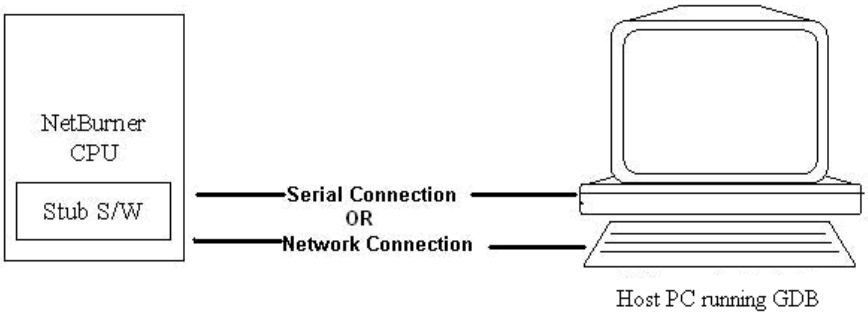
GDB (i.e. the **GNU Project Debugger**) is free software, protected by the **GNU General Public License** (GPL). A copy of the GPL can be found in **C:\Nburn\docs\GNU** directory (default location). You **can** use GDB (the **Command line GNU debugger**) with **any** version of the NetBurner tool set. **Note:** To find out what **version** of the tool set you are using, navigate to your Nburn (root) directory (**C:\Nburn** by default), and open up (e.g. in notepad) the **release_tag** file.

Warning: If you are using NNDK Version 2.0 or greater, you must use the NBEclipse Debugger if you want a GUI. You cannot use Insight.

If you want to use **NBEclipse** with its **fully integrated debugger**, please read your **NBEclipse Getting Started Guide** for step-by-step instructions. From Windows: Start → Programs → Netburner NNDK → NBEclipse → NBEclipse Getting Started. By default, this PDF is located in **C:\Nburn\docs\NetworkProgrammersGuide**.

The NetBurner debugging solution uses a small piece of debugging software, referred to as the **debugging stub**, to connect the **running** NetBurner environment with the GDB debugger using one of the two serial ports (all NetBurner platforms) or your network connection (for all NetBurner network platforms **except** the **CFV2-40** and all of the **Non-Network Development Platforms** (e.g. **Mod5213**).

The NetBurner solution allows you to start, stop, and set breakpoints to control an executing program. When the program execution is paused, you can examine **all** of the local and global variables.



To get the GDB debugger up and running you need **all** of the following:

- A **running program** on your NetBurner device. This program **must** include the **GDB stub** (either serial or network) in the source code, and it **must** have been compiled with the debug option. **Important:** The debug application in your NetBurner device **must** have networking **enabled** in order to use the Network Debugger.

- A **working** serial connection between your host computer and the GDB stub on your NetBurner Device (for Serial Debugging) or a **working** Network connection between your host computer and your NetBurner device (for Network Debugging).

- The **NetBurner** provided version of **GDB** running on your host computer.

- Access to **both** the **source** and the **.elf** files for the debug program **currently** running on your NetBurner device. **Note:** This is most easily accomplished if you build your debug program on the **same** computer that is running the debugger.

Compiling Debug-Enabled Applications Using NBEclipse

This is the **preferred** method for **creating**, **compiling**, **downloading**, and **debugging** your application in one easy step. Please read your **NBEclipse Getting Started Guide** for step-by-step instructions. From Windows: Start → Programs → Netburner NNDK → NBEclipse → NBEclipse Getting Started. By default, this PDF is located in your **C:\Nburn\docs\Eclipse** directory.

Compiling Debug-Enabled Applications Using NetBurner's Dev C++

Once the header file (serial or network) and GDB Stub function have been added to the application source code, your application **must** be built with the debug option. When building an application for debugging purposes, use either the Make Debug Version option or the Debug & Load option from the Build menu in NetBurner's Dev C++.

Using the **Make Debug** option will only **build** your debug application. Using the **Debug & Load** option will **build and download** your debug application to your NetBurner device (if the current application in your NetBurner device supports network downloads).

Warning: If you are using NNDK Version 2.0 or greater, you must use the NBEclipse Debugger if you want a GUI. You cannot use Insight. You can use GDB (command line).

Compiling Debug-Enabled Applications Using the DOS Command Line

Executing the **make debug** command at the Command line will only **build** your debug application; just as the **make** command does for non-debug builds. **Important:** You **must** be in your **project's directory** when issuing this command.

Executing the **make loaddebug** command at the Command line will **build and download** your debug application; just as the "make load" command does for non-debug builds. **Important:** You **must** be in your **project's directory** when issuing this command.

If you want to **build and download** your debug application, the current application in your NetBurner device **must** support network downloads. **Note:** If you are using any of NetBurner's **Non-Network Development Kits** (e.g. **Mod5213**), the current application in your hardware **must** support SerialLoad downloads

When a NetBurner **debug application** is built (using **any** method), the **output image** file will be **prefixed** with **DB** (DB stands for debug). For example, if your application name were "MyProject", the output file would be called "**DB**MyProject_APP.s19".

Remember, **NBEclipse** is the **preferred** method for **creating**, **compiling**, **downloading**, and **debugging** your application in one easy step.

Warning: You cannot use the NBEclipse debugger to debug an application created with NetBurner's Dev C++ or with a third party editor. You must create an NBEclipse project.

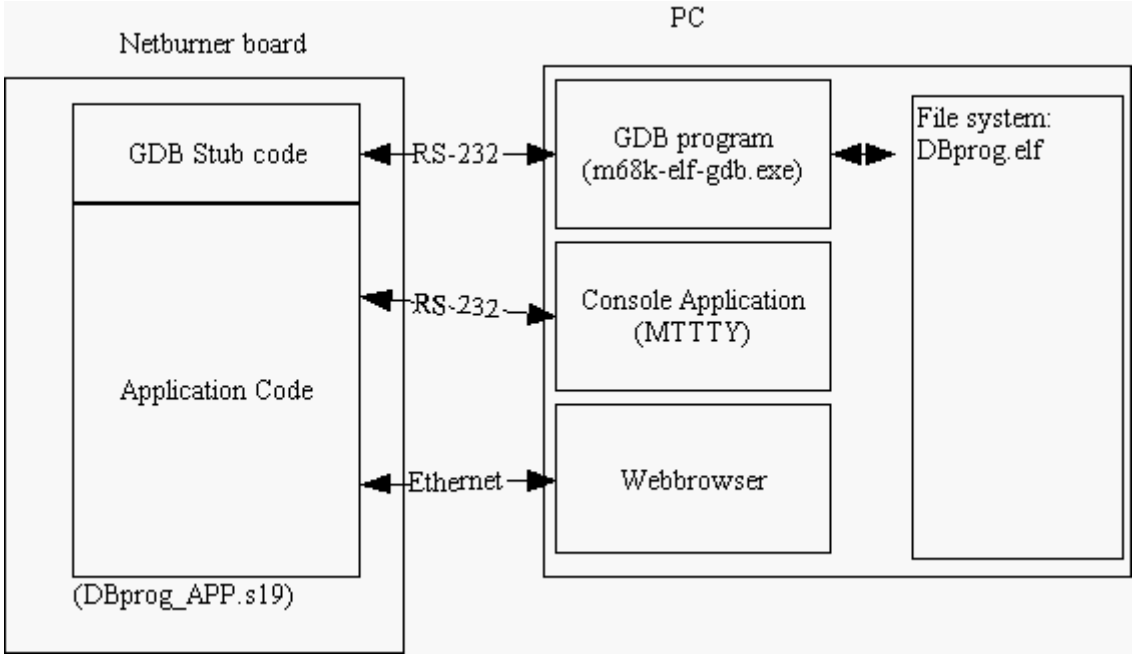
18. Serial Debugging

Introduction

The serial debugging stub is configured to use the second serial port with a **level 7 nonmaskable interrupt**. This level of control **allows** the debugger to debug **all** application code, even the operating system interrupt routines. **Warning: The only code you cannot debug is code within a level 7 interrupt.**

The serial stub uses the GDB remote serial protocol to communicate with the host computer. This allows the host computer to stop and start execution, set breakpoints, and examine memory. For the curious, the details of this protocol are all covered in the GDB remote serial protocol documentation on Red Hat's web site.

The GDB program runs on your host computer and communicates with your NetBurner device over one of the serial ports. A typical serial debugging setup looks like the diagram below.



In this example, there are four programs running and three communications links.

The four programs running are:

- The NetBurner application program including the GDB stub running on your NetBurner device. This is your application program doing whatever you wrote it to do.

- The GDB program running on the host computer. The GDB debugger provides the user interface to the debugging. It also controls the NetBurner device over the debugging serial (or network) connection with the code in the GDB stub.

- The Console (MTTTY application) running on the host computer. This is a normal serial console, so that you can interact with the program, and use the normal stdio to/from the application.

- The Web Browser running on the host computer. This allows you to see whatever web pages your application generates. (This is **optional**.)

The three communications links are:

- The RS232 connection from the stub to GDB. This connection is used to (serially) communicate debugging commands from the host computer to the NetBurner device, and to communicate state information from the NetBurner device to the host computer.

- The RS-232 connection from the Application to the Console.

- The Ethernet/IP connection from the Application to the Web Browser. (This is **optional** and does **not** apply to **any** of NetBurner's **Non-Network** Development Platforms (e.g. **Mod5213**) and/or the CFV2-40 Platform.)

In order to **effectively** use this debugging environment, you **must** master **two** things. First, setting up your NetBurner application to be debugged, and secondly, using the GDB Debugger.

Setting up your Debug Application

- **Before** attempting these steps you **should** become familiar with making and downloading a file to your NetBurner device. The following example assumes that you are building a project named "prog" using either NetBurner's Dev C++ IDE or the DOS Command Line.

- **Edit** your source code to include and start the (serial) debugger

  - Somewhere in you application startup code, (usually in UserMain) you **must** include the following code. **Important:** You **must** build your program **without** optimization, so that the debugger can figure out what is going on.

```
#include <gdbstub.h>    // Required for Serial Debugging
*
 {
    /* Somewhere in your start up code */
    /* This starts GDB on port 1 at 57600 baud */
     InitGDBStub( 1, 57600 );  }
```

- If you are using **NBEclipse** (the **preferred** method), please read your **NBEclipse Getting Started Guide** for step-by-step instructions using the NBEclipse Debugger. From Windows: Start → Programs → Netburner NNDK → NBEclipse →NBEclipse Getting Started. By default, this PDF is located in **C:\Nburn\docs\Eclipse**.

- From the **Build** menu in **NetBurner's Dev C++**, select **Make Debug Version**.

- From the **DOS** command line, **navigate** to your **project's** directory. Next, **type** the command **make debug** and **press** the **Enter** key.

This command (executed using **any** method, including NBEclipse) **will** build the following two files:

1. The **DBprog.elf** file with symbol information, located in your project's directory

2. The **DBprog _APP.s19** file (the compressed version of the program to be loaded into FLASH)

- Finally, **load** the **DBprog_APP.s19** file (DB stands for Debug) into your NetBurner device's **FLASH** memory using MTTTY. **Note:** You can compile **and** download your debug application to your NetBurner device in one easy step (**if** the **current** application in your NetBurner device **supports** direct downloads) by selecting the **Debug & Load** option from the Build menu in NetBurner's Dev C++. If building your program at the **Command line**, issue the command **make loaddebug** versus make debug.

Remember, NBEclipse is the preferred method for creating, compiling, downloading, and debugging your application in one easy step.

Using the GDB (Command Line) Debugger

Before you start the debugger, you **must** prepare a **symbol .elf** file (i.e. DBprog.elf).The following example will take you through a simple session with GDB. GDB is a complicated piece of software. **It is recommended that you spend some time with the full GDB documentation**. The example below debugs (at the command line) a trivial program that is located at the end of this section.

Open a **DOS** window. **Navigate** to your **project's directory**. Make sure that a DBprog.elf file **exists** in that directory. At the DOS prompt **type** the command **m68k-elf-gdb** and **press** the **Enter** key. This command **will** start GDB, and you **will** see a prompt similar to the one below:

```
GNU gdb 19991101
Copyright 1998 Free Software Foundation, Inc.
GDB is free software, covered by the GNU General Public
License, and you are
welcome to change it and/or distribute copies of it under
certain conditions.
Type "show copying" to see the conditions.
There is absolutely no warranty for GDB.  Type "show
warranty" for details.
This GDB was configured as "--host=i686-pc-cygwin --
target=m68k-elf".
(gdb)
```

All commands from here on **will** assume that you are typing them at the gdb prompt. **Before** you can **use GDB** you **must** do **two** things. First, **load the symbol file** and second, **establish communications**.

1. Load the symbol file:

```
sym DBprog.elf
```

2. Establish communications:

```
set remotebaud 57600
target remote /dev/com2
```

GDB should respond with:

```
 InitGDBStub ( port=1, baudrate=57600 ) at   gdbstub.c:824
824 }
```

At this point, GDB has **control** of your NetBurner device, and your program is **not** running. Now, let's set a **break point** in a specific function -"TestCase". In order to do this, we type the command **list TestCase**. GDB responds with:

```
22
23
24  void TestCase( int i, char * c )
25  {
26    printf("Buffer[%d]= [%s] %p\r\n",i,c,c);
27
28  }
29
```

We will now set a **break point at line 26**, so we **type** the command **break 26**. GDB responds with:

```
Breakpoint 1 at 0x20000a0: file main.cpp, line 26.
(gdb)
```

We will now **continue execution**, so we **type** the letter **c**. GDB responds with:

```
Continuing.
```

GDB will **not** do anything more until the break point is hit. Therefore, we type something (anything) on the keyboard, and then press the Enter key. GDB responds with:

```
Breakpoint 1, TestCase (i=1, c=0x204871c "something") at
main.cpp:26
26     printf("Buffer[%d]= [%s] %p\r\n",i,c,c);
Current language:  auto; currently c++
(gdb)
```

Now we are stopped again. We can **show the variables i or c** by **typing** either **print i or print c**. We can also **step**. To step into the printf call **typ**e the letter **s.** GDB Responds with:

```
printf (fmt=0x201a6f4 "Buffer[%d]= [%s] %p\r\n")
    at /source/gcc-2.95.2/newlib/libc/stdio/printf.c:56
56        _stdout_r (_REENT)->_data = _REENT;
Current language:  auto; currently c
(gdb)
```

Common GDB Commands

Command	Description
c	Continue
s	Step into
n	Step over
break x	Put a break point at line x
clear x	Clear the break point at line x
list x	List the function x
print x	Show the value of the expression x
set x=n	Set the variable x to n
quit	Quit debugging
info locals	Print out all of the local variables

The Program used in this Example

```
#include "predef.h"
#include <stdio.h>
#include <ctype.h>
#include <includes.h>
#include <serial.h>
#include <iosys.h>
#include <constants.h>
#include <utils.h>
#include <constants.h>
#include <ip.h>
#include <autoupdate.h>
#include <gdbstub.h>

extern "C" {
  void UserMain( void * pd );
 }
void TestCase( int i, char * c )
 {
 printf( "Buffer[%d]= [%s] %p\r\n", i, c, c );
 }
void UserMain( void * pd )
  {
   register int n;
   n=0;
  // Close the serial ports in case they are already open.
  printf( "Starting\n" );
  InitializeStack();
  OSChangePrio( MAIN_PRIO );
  EnableAutoUpdate();
  printf( "Before Init\n" );
  InitGDBStub( 1, 57600 );  /* This starts GDB on port 1 at 57600 baud */
  printf( "In While\r\n" );
  while(1)
  {
   char buffer[120];
   gets(buffer);
   n++;
   TestCase( n, buffer );
  }
}
```

Linking the Serial GDB Stub with your Executable

To **link** the **serial GDB** stub with your **executable** (for **all** NetBurner hardware platforms), you **will** need to **add** the following to your source code: the **GDB Stub header file** and a call to the **GDB stub function**. In the source code file that includes the UserMain( ) function, include the following header file:

```
#include <gdbstub.h>
```

Then, somewhere in your start up code call **one** of the following two functions:

```
void InitGDBStub( int port, int baudrate );
```

or

```
void InitGDBStubNoBreak( int port, int baudrate );
```

The **InitGDBStub** function **will** cause the program to **stop** execution at the point the stub is called, and **wait** for the debugger to connect and provide instructions.

The **InitGDBStubNoBreak** function **will** initialize the debugger, but it **will** allow program execution to continue normally.

Note: For debugging **during** your development process, we **recommend** that you use the **InitGDBStubNoBreak** function. The **recommended** debugger serial port **baud rate is 57600**. Using the InitGDBStubNoBreak function is good for attaching the debugger to a program that is **already** running to hunt down **intermittent** bugs.

Serial Port Wiring and Configuration

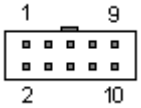
99% of all serial GDB communication problems occur because of **improper wiring** between the serial ports on the NetBurner board and host computer.

- If you are using either the **PK70** or **CB34EX**, we **recommend** that you **use** the **Network Debugger**.
- If you are using the **Mod5234, Mod52720, the Mod5272, or the Mod5282**, the debug port is the outer serial port (**J8)** on your **Mod-Dev-100** Carrier board. If you are using the any of these modules with the **Mod-Dev-50** (Promo) Carrier board, the debug port is the outer serial port (**J3**). We **recommend** that you **use** the **Network Debugger**.
- If you are using the **Mod5270LC** kit with the **Mod-Dev-70** Carrier board, the serial port (**UART 1**) is the inner serial port (**J3**). We **recommend** that you **use** the **Network Debugger**.
- If you are using the **SB72EX** (with a **NULL Modem cable**), the debug port is the right serial port (**Port 1**). We **recommend** that you **use** the **Network Debugger**. **Warning: You must use a NULL Modem cable. You cannot use a standard serial cable.**
- If you are using either the **SB70 or SB72** board with the SB72 Adapter/Evaluation board, the debug port is the outer serial port (**J3**) on the **Adapter/Evaluation board**. **Note:** We **recommend** that you **use** the **Network Debugger**.
- For the **CFV2-40 and CFV2-66**, the default GDB port is the **second serial port**, which is brought out on a **10-pin header**. We **recommend** that you **use** the **Network Debugger if you are using the CFV2-66 platform**.

1. The typical connection scheme is to use an **AT style IDC-10 pin to DB9 cable** (which is included in the **CFV2-40** and **CFV2-66** NetBurner Network Development Kits) and then connect the DB9 of the AT cable to a DB9 connector on the back of the computer that will run GDB.

2. **Note:** The pin-out of both DB9 connectors are identical, so you will need either a null-modem adapter or a cross-wired cable (swap rx and tx) to achieve proper communication between the computer and your NetBurner device. The description on the next page provides detailed wiring information.

3. There are **two** standard types of IDC-10 pin header to DB-9 ribbon cables: AT (also called Everex or Everest), and DTK (essentially a parallel ribbon crimp style). The 10-pin header for the CFV2-40 and CFV2-66 is configured for the AT style.



4. Above is a picture of a DB9 connector to indicate pin-outs. The table below provides the signal definitions for each connector. The IDC-10 pin-out is included for reference, but the AT style is the pin-out you need to implement. **Note:** An AT style IDC-10 to DB9 cable is included in your (CFV2-40 and CFV2-66) Development Kit.

Signal Definitions (For CFV2-40 or CFV2-66 kits only)

DB-9	IDC-10 (AT, Everex, & Everest)	IDC-10 (DTK)	Standard Male DB-9 Serial Port (PC) RS-232 Signals
1	**1**	**1**	DCD (input)
2	**2**	**3**	RX (input)
3	**3**	**5**	TX (output)
4	**4**	**7**	DTR (output)
5	**5**	**9**	GND
6	**6**	**2**	DSR (input)
7	**7**	**4**	RTS (output)
8	**8**	**6**	CTS (input)
9	**9**	**8**	RI (input)

Depending on how you wire from the 10-pin header to a DB-9 (CFV2-40 and CFV2-66), you may or may not need a NULL modem adapter. If wired per the above chart, you **will** need a null modem adapter. This is how the included serial cable is wired as provided in your CFV2-40 and CFV2-66 Network Development Kits.

If you want to build a **dedicated debugging cable** for the second serial port that does **not** require a **NULL modem cable**, then **replace** the male DB9 with a female DB9 **and** wire as follows:

DB-9 Female Pin	IDC-10 (AT) Ribbon Cable Wire Number
2	3
3	2
5	5

Testing the Serial Connection (All NetBurner platforms)

To proceed with this test you **must** have completed the all of the following steps:

- **Installed** the **GDB** software (this should have been done automatically when you installed the NetBurner Software) on your host computer
- **Built** your **application** code with the **GDB stub and debugging information**
- **Downloaded** your **debug application** code into **FLASH** memory
- **Connected** your NetBurner device to your host computer via their **Serial** ports

Warning: If the following serial test does not work as described, correct the problems before moving to the next step.

- **Load** your **debug** program with the **GDB stub** in your NetBurner device FLASH memory (if you have not done so already)
- **Start** the **MTTTY** serial terminal program and **connect** (make sure to click the Connect button) at the **correct** baud rate. **Note:** The default **baud** rate is **57600**.
- **Reset** your NetBurner device by **cycling power** (i.e. a hard reset).
    - If your stub was started with the **InitGDBStub()** function, then you **will** see the GDB stub report: **$S05#b8** in the MTTTY window.
    - If your stub was started with the **InitGDBStubNoBreak()** function, **type** the command **Ctrl-c** on the keyboard. You **will** see the GDB stub report: **$S02#b5** in the MTTTY window.
- **Type** a **-** (**hyphen**), and the stub **will** repeat the **last** report (e.g. $S02#b5 for no break).
- **Type** a **+** (**plus**), and the stub will **not** echo **any** characters.
- **Type** the command **$D#44** to **start** the application program, and you **will** see the stub report a **+** (**plus**).

You have now **passed** the serial connection test!

18.1. Serial Debugging Functions

18.1.1. InitGDBStub

Required Header File:

```
#include <gdbstub.h>       // Found in C:\Nburn\include
```

Note: If you are using any of the **Non-Network platforms** (e.g. **Mod5213**), this header file is found in **C:\Nburn\include_nn**.

Synopsis:

```
void InitGDBStub( int port, int baudrate );
```

Description:

This function will cause the program to **stop** execution at the point the stub is called, and **wait** for the debugger to connect and provide instructions. **The recommended debugger serial port baudrate is 57600**.

Parameters:

Name	Type	Description
port	int	Determines which Serial port will be used.
baudrate	int	Specifies the baudrate of the opened Serial port, **Note:** This value **must** be: 300, 600, 1200, 2400, 4800, 9600, 119200, 38400, 57600, or 115200.

Usage Example:

```
/* This starts GDB on Serial port 1 at 57600 baud */
InitGDBStub( 1, 57600 );
```

Returns:

Nothing --- This is a void function

18.1.2. InitGDBStubNoBreak

Required Header File:

```
#include <gdbstub.h>       // Found in C:\Nburn\include
```

Note: If you are using any of the **Non-Network platforms** (e.g. **Mod5213**), this header file is found in **C:\Nburn\include_nn**.

Synopsis:

```
void InitGDBStubNoBreak( int port, int baudrate );
```

Description

This function will initialize the debugger, but it will allow program execution to continue normally.

For debugging during your development process, we recommend using this function.

Using this function is a good way to attach the debugger to a program that is already running, to hunt down intermittent bugs. **The recommended debugger serial port baudrate is 57600**.

Parameters:

Name	Type	Description
port	int	Determines which Serial port will be used.
baudrate	int	Specifies the baudrate of the opened Serial port, **Note:** This value **must** be: 300, 600, 1200, 2400, 4800, 9600, 119200, 38400, 57600, or 115200.

Usage Example:

```
/* This starts GDB on Serial port 1 at 57600 baud */
InitGDBStubNoBreak( 1, 57600 );
```

Returns:

Nothing -- This is a void function

19. Network Debugging

Introduction

The NetBurner Network Debugger allows full functionality debugging over your network connection. You will be able to debug everything (including interrupt routines). The only limitation, **you cannot debug non-maskable interrupts**.

Warning: The NetBurner Non-Networking Platforms (e.g. Mod5213 kit) and the CFV2-40 Network Development platform are not supported.

If you are using NNDK Version 2.0 or greater, you must use the NBEclipse integrated debugger. You cannot use Insight (**GUI**). However, you can use **GDB** (the **Command line** Debugging) with **any** NNDK software version.

Note: To find out what **version** of the tool set you are using, navigate to your Nburn (root) directory (**C:\Nburn** by default), and open up (e.g. in notepad) the **release_tag** file.

Please read your **NBEclipse Getting Started Guide** for step-by-step instructions using the NBEclipse Debugger. From Windows: Start → Programs → Netburner NNDK → NBEclipse → NBEclipse Getting Started. By default, this PDF is located in **C:\Nburn\docs\Eclipse**.

Setting up the Application to be debugged

Before attempting these steps you should become familiar with making and downloading a file to your NetBurner device. The following example assumes that you are building a project named "prog".

Change your source code to include and start the network debugger. Somewhere in you application startup code, (usually in UserMain) you **must** include the following code:

```
#include <NetworkDebug.h>   // Required for Network Debugging
  {
     // Somewhere in your start up code
     InitializeNetworkGDB();
     // Or you can use - InitializeNetworkGDB_and_Wait();
  }
```

Note: You **must** build your program **without** optimization, so that the debugger can figure out what is going on.

19.1. Network Debugging Functions

19.1.1. InitializeNetworkGDB

Required Header File:

```
#include <NetworkDebug.h>   // Found in C:\Nburn\include
```

Synopsis:

```
void InitializeNetworkGDB();
```

Description:

This function **initializes** the Network debugger. **Note:** Add this function **after** your other initialization code.

Warning: The NetBurner Non-Networking Platforms (e.g. Mod5213 kit) and the CFV2-40 Network Development platform are not supported.

Parameters:

None

Returns:

Nothing --- This is a void function

19.1.2. InitializeNetworkGDB_and_Wait

Required Header File:

```
#include <NetworkDebug.h>   // Found in C:\Nburn\include
```

Synopsis:

```
void InitializeNetworkGDB_and_Wait();
```

Description:

This function **initializes** the Network debugger and **waits** for you to connect **before** you start debugging. **Note:** Add this function **after** your other initialization code.

Warning: The NetBurner Non-Networking Platforms (e.g. Mod5213 kit) and the CFV2-40 Network Development platform are not supported.

Parameters:

None

Returns:

Nothing --- This is a void function

19.2. Network Debugging Options

The network debugger has four options that **change** its default behavior. To set one of these options just include the appropriate Macro right **before** UserMain.

Warning: The NetBurner Non-Networking Platforms (e.g. Mod5213 kit) and the CFV2-40 Network Development platform are not supported.

19.2.1. DebugIP

Introduction

There are times when you want the debugger to connect via a different IP Address (versus its "normal" IP Address). **Note:** It is **not** possible for a **single** network interface to have **multiple** DHCP addresses, so you **cannot** have a DHCP Address for the debugger and a separate DHCP Address for your NetBurner device.

The DebugIP option allows you to set a Static IP Address for your debugging. The DebugIP option also automatically tells the debugger to use ARP for its IP Address.

Warning: This option cannot be disabled with the DebugNormalArp option.

Required Header File:

```
#include <NetworkDebug.h>   // Found in C:\Nburn\include
```

Usage:

```
DebugIP( "IP Address" );
```

Example:

```
DebugIP( "10.1.1.125" );
```

19.2.2. DebugRebootOnTrap

Introduction

This option is used when the Network debugger is running, but **no** debugging session is connected. The normal action when your program faults or traps is to sit and wait for the debugger to connect so that you can debug the faulty code.

This option **forces** the system to **reboot** when it hits a fault, breakpoint, or trap while it is **not** connected to a debugging session.

Required Header File:

```
#include <NetworkDebug.h>   // Found in C:\Nburn\include
```

Usage:

```
DebugRebootOnTrap( );
```

19.2.3. DebugRebootOnDisconnect

Introduction

When the debugging session disconnects from the device being debugged, the normal operation is to remove all breakpoints and continue from where it stopped. This option **forces** the device to reboot and restart when the debugging session is disconnected or terminates.

Required Header File:

```
#include <NetworkDebug.h>   // Found in C:\Nburn\include
```

Usage

```
DebugRebootOnDisconnect( );
```

19.2.4. DebugNormalArp

Introduction

There are times when you might want to debug the NetBurner ARP response code. Since the debugger takes over this function, you **cannot** debug this code. Turning this option **on** prevents the debugger from doing this.

Warning: If you sit for a long time in a breakpoint, any communicating device may loose its ARP entry for that device, and be unable to resolve the ethernet address.

Required Header File:

```
#include <NetworkDebug.h>   // Found in C:\Nburn\include
```

Usage:

```
DebugNormalArp( );
```