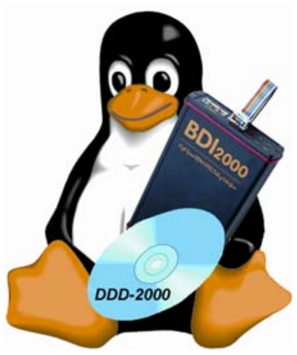
TMT
Tools Made Tough

DDD-2000 User's Manual
Version 1.0



DDD -2000
Table of Contents

Section 1. Installation

DDD-2000 has been extensively tested on the following commercial distributions:

- **RHL 7.x**
- **RHL 8**
- **RHL 9**

The following information has to be filtered when applied to different host distributions. If you did not install all packages doing a custom installation, you may need to install individual RPM files, as required.

DDD-2000 is based on DDD v3.3.8, which requires a 3.x version of GCC to build. If you are trying to install on a RHL 7.x system, you need to update the stock GCC compiler that came with the distribution. If you are installing on a RHL 8 or later system, the stock GCC compiler from the distribution should build DDD-2000.

R4I : 1 : Update your GCC version

R4I : 2 : Update your DDD version

TMT
Tools Made Tough

R4I : 1 : Update your GCC version

Stock installation of gcc-3.3.3 works out-of-box on above RHL platforms. If your host gcc compiler is stock from the distribution, gcc-3.3.3 is an update/upgrade to your current system. Binutils is included on downloads for those building a cross-development environment. For a strictly host development system, binutils should not be necessary.

http://www.recipes4linux.com/recipes/bdi2000/bdiddd/bdiddd.htm

The above URL allows you to download the gcc tarball referenced in the recipe steps below.

Recipe steps...

1. **cp <tarball> /usr/local/src/Archives**
2. **cd /usr/local/src**
3. **tar xvzf Archives/gcc-3.3.3.tar.gz**
4. **cd gcc-3.3.3**
5. **mkdir build**
6. **cd build**
7. **../configure**
8. **make**
9. **make install**

This installs gcc v3.3.3 into /usr/local/bin. If you want to use this compiler for host makes, you need to either modify your $PATH environment variable or explicitly reference the tool in Makefiles.

R4I : 2 : Update your DDD version

The tarball referenced in the recipe steps below is a patched ddd-3.3.8 tree. This tarball should be on the CD you received, and is named ddd-3.3.8-bdi-02.tgz. The recipe steps assume an appropriate GCC version is pathed in through $PATH.

Recipe steps...

1. **cp <tarball> /usr/local/src/Archives**
2. **cd /usr/local/src**
3. **tar xvzf Archives/ddd-3.3.8-bdi-02.tgz**
4. **cd ddd-3.3.8**
5. **mkdir build**
6. **cd build**
7. **../configure**
8. **make**
9. **make install**

This installs DDD-2000 into /usr/local/bin. You start DDD-2000 the same way you run DDD. Your $PATH environment variable may need to be adjusted, see the recipes in the next section for configuration information.

Section 2. Configuration

DDD-2000 has been extensively tested on the following commercial distributions:

- **RHL 7.x**
- **RHL 8**
- **RHL 9**

The following information has to be filtered when applied to different host distributions. If you did not install all packages doing a custom installation, you may need to install individual RPM files, as required.

R4C : 1 : Setting Environment Variables

This section describes how to set the environment variables that enable DDD-2000 to run on your host system. The following variables are described:

$PATH

$DDD_HOME, $XUSERSEARCHPATH

$LANG

$LD_LIBRARY_PATH

The typical way to set/change environment variables is in either the user's ~/.bashrc file or ~/.profile file. I usually set the variables in the ~/.bashrc file so I can kill the window I'm using and create a new one to see the changes take effect. When the ~/.profile file is used, the logged in user has to logout and login again before any changes are seen.

From the bash man page...

The following paragraphs describe how bash executes its startup files. If any of the files exist but cannot be read, bash reports an error. Tildes are expanded in file names as described below under Tilde Expansion in the EXPANSION section.

When bash is invoked as an interactive login shell, it first reads and executes commands from the file *~~/etc/profile~~*, if that file exists. After reading that file, it looks for *~/.bash_profile, ~/.bash_login, and ~/.profile*, in that order, and reads and executes commands from the first one that exists and is readable. The --noprofile option may be used when the shell is started to inhibit this behavior.

When a login shell exits, bash reads and executes commands from the file *~/.bash_logout*, if it exists.

When an interactive shell that is not a login shell is started, bash reads and executes commands from *~/.bashrc*, it that file exists. This may be inhibited by using the --norc option. The --rcfile *file* option will force bash to read and execute commands from *file* instead of *~/.bashrc*.

$PATH

The $PATH variable is used by the shell to look for executable programs. It is preset by the shell when the user first logs on and then it can be modified each time a shell is started. If DDD-2000 was installed to a directory not currently in the $PATH variable, the $PATH variable has to be adjusted. If DDD-2000 was not installed on your system (step 9 in recipe R4I : 2 : Update your DDD version) the $PATH variable should be adjusted to reflect the build directory for DDD-2000.

If you want your additions to the $PATH variable to be seen BEFORE the system settings, use the following line in your startup file (~/.bashrc)

```
EXPORT PATH=/usr/local/bin:$PATH
```

If you want your additions to the $PATH variable to be seen AFTER the system settings, use the following line in your startup file (~/.bashrc)

```
EXPORT PATH=$PATH:/usr/local/bin
```

$DDD_HOME & $XUSERFILESEARCHPATH

From the INSTALL file in the ddd-3.3.8 tarball...

12. If you want to use DDD on your own program, please install it first (see the Step 13, below). If you absolutely *must* run DDD without installing it, be sure that the following environment variables are properly set:

- DDD_HOME must point to the top-level source package directory (such that '$DDD_HOME/NEWS' is the DDD news)

- XUSERFILESEARCHPATH must contain the path of the 'Ddd' app-defaults file, where 'Ddd' is substituted by '%N'.

If you executed step 9 in recipe R4I : 2 : Update your DDD version, you can skip the rest of this section. However, if you did NOT execute step 9 in the above recipe, you should add the following lines to your shell startup file:

```
export DDD_HOME=/usr/local/src/ddd-3.3.8/

export
XUSERFILESEARCHPATH=$DDD_HOME/build/ddd:$XUSERFILESEARCHPATH
```

(Please note: the above export line for XUSERFILESEARCHPATH should all be on one line in your ~/.bashrc file)

$LANG

From Volume One, Xlib Programming Manual for Version 11, by Adrian Nye...

10.1.1 Internationalization with ANSI-C

...

The first step in any internationalized application is to establish the locale -- to cause the localization database to be read in. This is done with the C library function setlocale. It takes two arguments: a locale category and the locale name. The locale name specifies the database that should be used to localize the program, and the locale category specifies which behaviors (for example, the collation sequence of the alphabet or the formatting of times and dates) of the program should be changed. setlocale will most often be used as shown below:

setlocale(LC_ALL, "");

Passing the empty string as the locale name will cause setlocale to get the name of the locale from the operating system environment variable named LANG. This allows the application writer to leave the choice of locale to the end user of the application. There is no standard format for locale names, but they often have the form:

language[_territory[.codeset]]

From the setlocale man page...

...

If *locale* is "", each part of the locale that should be modified is set according to the environment variables. The details are implementation dependent. For glibc, first (regardless of *category*), the environment variable LC_ALL is inspected, next the environment variable with the same name as the category (LC_COLLATE, LC_CTYPE, LC_MESSAGES, LC_MONETARY, LC_NUMERIC, LC_TIME) and finally the environment variable LANG. The first existing environment variable is used. If its value is not a valid locale specification, the locale is unchanged, and setlocale returns NULL.

The locale "C" or "POSIX" is a portable locale; its LC_CTYPE part corresponds to the 7-bit ASCII character set.

A locale name is typically of the form *language*[_*territory*][.*codeset*][@*modifier*], where *language* is an ISO 639 language code, *territory* is an ISO 3166 country code, and *codeset* is a character set or encoding identifier like ISO-8859-1 or UTF-8. For a list of all supported locales, try "locale -a".

At startup, DDD-2000 does, in fact, make the following function call

```
setlocale( LC_ALL, "" );
```

Which implies that the locale defaults to 7-bit ASCII, unless modified by an environment variable setting. 7-bit ASCII is problematic to the proper running of DDD-2000 because of the European origins of both ddd and the Abatron BDI-2000. Non-ASCII characters are displayed in the Abatron help menu and copyright notices for ddd.

Run the command 'locale -a' on your system to determine appropriate settings for the $LANG variable. At a minimum, the following should work on all systems:

```
export LANG=.UTF8
```

$LD_LIBRARY_PATH

This environment variable is used to find libraries when an executable file is loaded by the system. If you installed GCC v3.3.3 and used this compiler to build DDD-2000, you should add the following line to your ~/.bashrc file:

```
export LD_LIBRARY_PATH=/usr/local/lib:$LD_LIBRARY_PATH
```

TMT
Tools Made Tough

R4C : 2 : BDI-2000 Configuration File

The [HOST] portion of the BDI-2000 configuration file requires two settings to work correctly with DDD-2000.

Parameter Name	Setting
PROMPT	BDI>
TELNET	NOECHO

Section 3. DDD-2000

DDD-2000 has been extensively tested on the following commercial distributions:

- **RHL 7.x**
- **RHL 8**
- **RHL 9**

The following information has to be filtered when applied to different host distributions. If you did not install all packages doing a custom installation, you may need to install individual RPM files, as required.

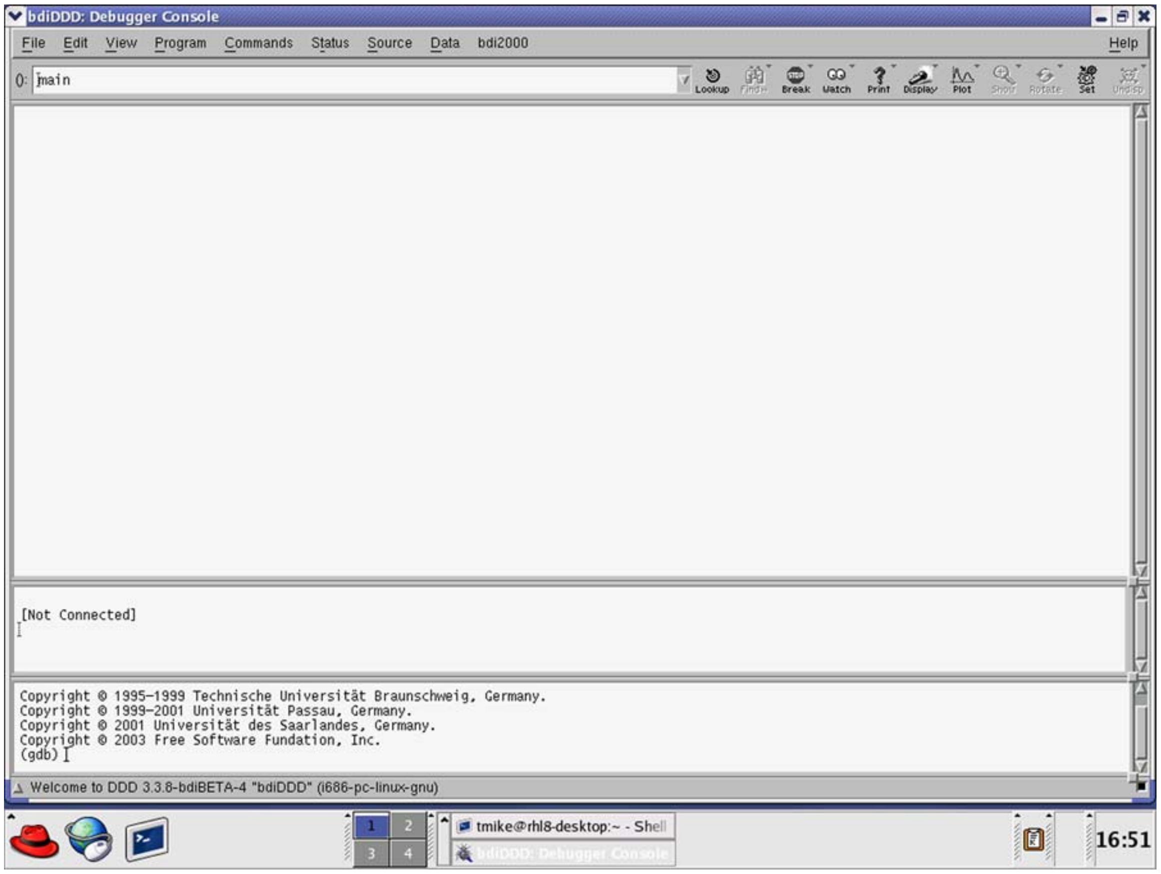
R4DDD-2000 : 1: Telnet console window

R4DDD-2000 : 2 : Edit | Preferences | bdi2000 window

R4DDD-2000 : 3 : bdi2000 menu

R4DDD-2000 : 1 : Telnet console window

Just above the gdb console window, a new console window is displayed, with the prompt [Not Connected]. This is essentially a telnet interface window with simple command-line editing capabilities.


bdiDDD: Debugger Console
File Edit View Program Commands Status Source Data bdi2000 Help
0: main
Lookup Find Break Watch Print Display Plot Show Rotate Set Undisp
tmike@rhl8-desktop:~ - Shell
bdiDDD: Debugger Console
16:51

```
[Not Connected]
```

```
Copyright © 1995-1999 Technische Universität Braunschweig, Germany.
Copyright © 1999-2001 Universität Passau, Germany.
Copyright © 2001 Universität des Saarlandes, Germany.
Copyright © 2003 Free Software Fundation, Inc.
(gdb)
```

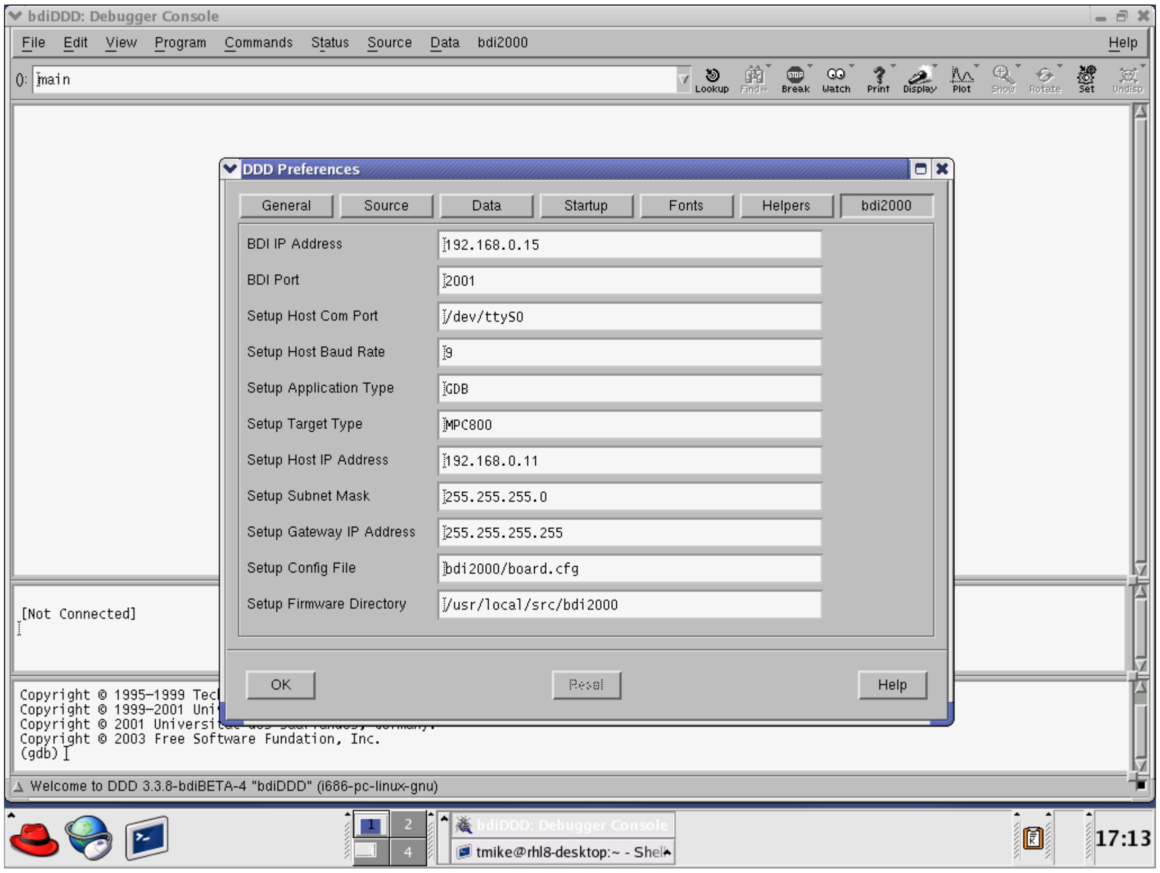
Welcome to DDD 3.3.8-bdiBETA-4 "bdiDDD" (i686-pc-linux-gnu)

The telnet window has a simple command-line editor. Input is accepted at the prompt at the bottom of the window. Regardless of where the cursor is positioned, when keyboard input is displayed, the window is scrolled to the last prompt line.

A command line is buffered until a newline is parsed, then the command string is sent to the telnet agent. Backspace is supported to delete characters, but the user is not allowed to backspace over the prompt.

R4DDD-2000 : 2 : Edit | Preferences | bdi2000 window

The system specific parameters required to access the BDI2000 are collected in this menu. All parameter values are text strings and can be modified to reflect your system settings. The settings include parameters that allow the user to telnet to, and connect gdb to, the BDI2000. Also included are all of the bdisetup parameters required to configure the BDI2000 the first time you use it. Please see the Abatron BDI2000 User's Guide for more information on these parameters. The Help button on this dialog box displays information about these parameters, as well. By default, your new settings will be saved when DDD-2000 exits, allowing your settings to survive across multiple-invocations of DDD-2000.


bdiDDD: Debugger Console
File Edit View Program Commands Status Source Data bdi2000 Help
Lookup Find Break Watch Print Display Plot Show Rotate Set Undisp
DDD Preferences
General Source Data Startup Fonts Helpers bdi2000
OK Reset Help
17:13

BDI IP Address	192.168.0.15
BDI Port	2001
Setup Host Com Port	/dev/ttyS0
Setup Host Baud Rate	9
Setup Application Type	GDB
Setup Target Type	MPC800
Setup Host IP Address	192.168.0.11
Setup Subnet Mask	255.255.255.0
Setup Gateway IP Address	255.255.255.255
Setup Config File	bdi2000/board.cfg
Setup Firmware Directory	/usr/local/src/bdi2000

```
(): main
[Not Connected]
Copyright © 1995-1999 Tec
Copyright © 1999-2001 Uni
Copyright © 2001 Universit
Copyright © 2003 Free Software Fundation, Inc.
(gdb)
```

Welcome to DDD 3.3.8-bdiBETA-4 "bdiDDD" (i686-pc-linux-gnu)

R4DDD-2000 : 3 : bdi2000 menu

This menu supports the following commands:

COMMAND	**DESCRIPTION**
connectBDI	**open telnet session to BDI2000**
connectGDB	**issue gdb target remote call to BDI2000**
reset	**issue reset command to BDI2000**
bdiSetup->Version	**issue bdisetup -v command (requires serial connection)**
bdiSetup->Erase	**issue bdisetup -e command (requires serial connection)**
bdiSetup->Update	**issue bdisetup -u command (requires serial connection)**
bdiSetup->Configure	**issue bdisetup -c command (requires serial connection)**
registers	**issue rdump command to BDI2000**
breakHard	**set breakpoint mode to Hard**
breakSoft	**set breakpoint mode to Soft**

When the bdiSetup commands are running, the status bar at the bottom of the main window is updated to reflect this state. The GUI is unresponsive during these times. One command in particular, the update operation, can take a number of minutes to complete. When any of the bdiSetup commands complete, a dialog box is displayed with information about the command, including the exact command-line that was issued and the response from the BDI2000.

Section 4. Usage Scenarios

DDD-2000 has been extensively tested on the following commercial distributions:

- **RHL 7.x**
- **RHL 8**
- **RHL 9**

The following information has to be filtered when applied to different host distributions. If you did not install all packages doing a custom installation, you may need to install individual RPM files, as required.

R4US : 1 : Configuring the Abatron BDI2000

Built-in to DDD-2000 is an interface to the bdisetup utility that must be run to configure the BDI2000 the first time you use it. Outside of the DDD-2000 installation, the user must build the bdisetup utility that is part of the firmware update files provided by Ultimate Solutions. The bdisetup utility must be in the $PATH setting (typically /usr/local/bin) in order for DDD-2000 to access the utility. Please note that a serial connection is required between host and the BDI2000 unit when the bdisetup utility is used.

See R4DDD-2000 : 2 : Edit | Preferences | bdi2000 window for a description of the setup variables that should be updated for your system. Please use the Help button on this display for additional information about each Preference variable/setting. Alternatively, from the command line, after bdisetup is built and installed, you can issue this command to receive a help display directly from the utility:

```
bdisetup > bdi.help
```

This command creates a text file in the current working directory that can be viewed for additional information about the bdisetup commands and parameters.

Whenever firmware updates are issued by Ultimate Solutions, or some network configuration option has changed, you should run the bdisetup utility. Please follow these steps.

Recipe steps...

1. **Remove power from target board and BDI2000**
2. **Disconnect JTAG cable between target and BDI2000**
3. **Establish RS-232 connection between BDI2000 and host**
4. **Install firmware update files on host**
5. **Modify Edit | Preferences | bdi2000 settings, as required**
6. **Run appropriate bdisetup utility function**
7. **Cycle power on the BDI2000 unit (leave power off for 10 seconds)**

bdisetup utility function	Description
bdi2000 | bdiSetup | bdiVersion	runs bdisetup -v to determine current version information
bdi2000 | bdiSetup | bdiErase	runs bdisetup -e to erase current firmware and logic
bdi2000 | bdiSetup | bdiUpdate	runs bdisetup -u to update firmware and logic
bdi2000 | bdiSetup | bdiConfigure	runs bdisetup -c to configure network parameters

Please note that when any of the bdisetup functions are executed, the X display "freezes" until the command completes. The status line at the bottom of the display is updated to reflect the utility function running, but no X control is possible until the utility completes. When the function is complete an X dialog window appears with the output of the command.

TMT
Tools Made Tough

R4US : 2 : Loading and debugging a Linux Kernel

This section describes the same sequence of steps as detailed in the App-Note available from Ultimate Solutions on debugging a Linux Kernel. The App-Note is titled "Using the Abatron BDI2000 to Debug a Linux Kernel" and is Number 02-001. A PDF version of this App-Note is available here at this URL:

http://www.recipes4linux.com/recipes/bdi2000/kerneldebug/Tool Talk 02-001a-Debug-Linux Kernel.pdf

Ultimate Solutions can provide a hard-copy of the App-Note, if this is desirable.

The sequence of steps, as presented here, are in a slightly different order, because DDD-2000 now controls the BDI2000 unit. The hardware used (RPX-Lite PPC/823 board from Embedded Planet) is the same, but the kernel version is now slightly different. The steps described here are based on using the Journeyman edition of HHL 2.0. Only the actual steps are presented here, for a description of the steps, please see the App-Note link above.

Preparing the Kernel to be Debugged

1. **cd [root of kernel source tree]**
2. **vi +/'CFLAGS :=' Makefile**
3. **add "-g -ggdb" to the end of the CFLAGS macro definition**
4. **make clean dep**
5. **make zImage**
6. **make modules**
7. **make modules_install**
8. **cp arch/ppc/boot/images/zImage.mbx /tftpboot/vmlinuz.bdi**

Some of the steps above, like 2-3, you will only do once, but most of them you will do whenever you modify the kernel, or more likely, a driver that is linked with the kernel. Step 8 gives the BDI2000 visibility to the kernel image that will be downloaded. The filename specified in step 8 should be the same as that specified in the configuration file used by the BDI2000.

Starting minicom (assumes serial connection between host and target)

1. **minicom**

Starting DDD-2000 and downloading Linux kernel

Steps in this recipe involve interacting with 3 components of DDD-2000: the gdb console window [gdb], the bdi console window [bdi] and the menu system.

1. **cd [root of kernel source tree]**
2. **ddd --debugger ppc_8xx-gdb --gdb vmlinux**
3. **bdi2000 | connectBDI**
4. **bdi2000 | reset**
5. **[bdi] go**
6. **in minicom window, wait for firmware prompt**
7. **[bdi] halt**
8. **[bdi] load**
9. **[bdi] bi 0xc0002180 (address of start_here)**
10. **[bdi] go**
11. **in minicom window, wait for target to hit breakpoint and return control**
12. **[bdi] ci**
13. **bdi2000 | connectGDB**
14. **[gdb] b panic**
15. **[gdb] b sys_sync**
16. **[gdb] cont**

At this point, in the minicom window, the kernel should complete booting and a login prompt should appear. The kernel has booted, with two breakpoints set and you can begin your debug session. The address above (0xc0002180) is the address for 'start_here' in my kernel image. This address will differ in most kernel images, and the user may wish to place the initial breakpoint at another place in the kernel image. Unless you need to debug the very initial start-up code, it is recommended that the initial breakpoint be placed after the kernel has enabled the MMU.

R4US : 3 : Cycling power on the target board

Sometimes during a debug session it becomes necessary or advantageous to reset the target and/or cycle power on the target. Please follow the instructions in this recipe when you need to do this with your system.

Once the target is reset or power-cycled, issue the BDI reset command:

bdi2000 | reset

Whatever was running on the target is lost and has to be reloaded. If you were debugging a Linux kernel, follow the steps in recipe R4US : 2 : Loading and debugging a Linux Kernel.

From the perspective of DDD-2000, this is a benign event.

All breakpoint information in gdb is still accurate and breakpoint(s)/watchpoint(s) do not have to be redone.

R4US : 4 : Cycling power on the BDI2000

Unlike R4US : 3 : Cycling power on the target board, this event must be managed carefully. Basically, the power sequence documented in the bdiGDB User Manual, must be followed.

The following reference is on the bottom of page 8 of my manual (v1.23 for PowerPC MPC8xx/MPC5xx)...

Please switch on the system in the following sequence:

1. **external power supply (for BDI2000)**
2. **target system**

If power is cycled on the BDI2000 unit, and the target is not turned off, the telnet interface to the BDI2000 will appear to hang. If a bdi2000 menu command is attempted at this time, more than likely DDD-2000 will crash with a core file. If the system hasn't crashed, cycle power on the target and the expected behavior should occur.

Expected behavior is that when the BDI2000 unit is power-cycled, the telnet process is killed and restarted. When the system is operating normally, the [Not Connected] prompt appears in the bdi console window after a BDI2000 power-cycle. At this point, issue the bdi2000 | connectBDI command and a new telnet session is established with the BDI2000.

The user now has to restore the debug session to the state it was in prior to the power-cycle.

From the perspective of DDD-2000, the expected behavior from this event is benign. The fact that DDD-2000 cannot, at this time, protect against the unexpected behavior, is known and the user is warned.

All breakpoint information in gdb is still accurate and breakpoint(s)/watchpoint(s) do not have to be redone.

You will probably have to issue the GDB connect command:

bdi2000 | connectGDB

twice. The first time an error message is displayed in the dialog window, related to a Broken pipe and the putpkt function. The second time the command is issued, no error message occurs and the connection between GDB and the BDI-2000 is re-established.

PLEASE NOTE: If you try to exit DDD-2000 with an inconsistent state between GDB and the BDI-2000 unit, the program does not exit normally. The DDD-2000 process needs to be killed at the command-line. A core file is likely to be generated. In this context, inconsistent means that GDB believes a connection is established with a gdb-agent process (BDI-2000), and the connection no longer exists because the BDI-2000 has been reset.

Section 5. Troubleshooting