bdiGDB

BDM interface for GNU Debugger

PowerPC MPC8xx/MPC5xx

User Manual

Manual Version 1.00 for BDI3000

abatron

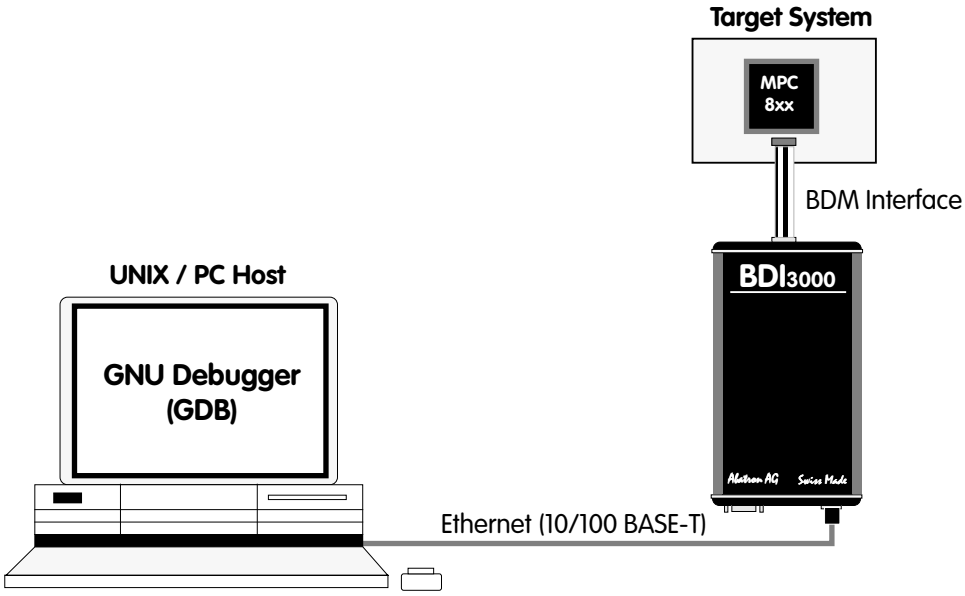
Appendices

1 Introduction

bdiGDB enhances the GNU debugger (GDB), with Background Debug Mode (BDM) debugging for MPC8xx/MPC5xx based targets. With the built-in Ethernet interface you get a very fast code download speed. No target communication channel (e.g. serial line) is wasted for debugging purposes. Even better, you can use fast Ethernet debugging with target systems without network capability. The host to BDI communication uses the standard GDB remote protocol.

An additional Telnet interface is available for special debug tasks (e.g. force a hardware reset, program flash memory).

The following figure shows how the BDI3000 interface is connected between the host and the target:



1.1 BDI3000

The BDI3000 is the main part of the bdiGDB system. This small box implements the interface between the JTAG pins of the target CPU and a 10/100Base-T Ethernet connector. The firmware of the BDI3000 can be updated by the user with a simple Linux/Windows configuration program or interactively via Telnet/TFTP. The BDI3000 supports 1.2 – 5.0 Volts target systems.

1.2 BDI Configuration

As an initial setup, the IP address of the BDI3000, the IP address of the host with the configuration file and the name of the configuration file is stored within the flash of the BDI3000.
Every time the BDI3000 is powered on, it reads the configuration file via TFTP.

Following an example of a typical configuration file:

```
; bdiGDB configuration file for MPC860ADS board
; ---------------------------------------------
;
[INIT]
; init core register
WSPR    638             0x02200000      ;IMMR : internal memory at 0x02200000
WSPR    158             0x00000007      ;ICTRL:
; init SIU register
WM32    0x02200000      0x01632440      ;SIUMCR
WM32    0x02200004      0xFFFFFF88      ;SYPCR
WM16    0x02200200      0x0002          ;TBSCR
WM16    0x02200220      0x0102          ;RTCSC
WM16    0x02200240      0x0002          ;PTSCR
; init UPM
SUPM    0x02200168      0x0220017c      ;set address for MCR and MDR
WUPM    0x00000000      0x8FFFEC24      ;UPMA single read
WUPM    0x00000001      0x0FFFEC04
WUPM    0x00000002      0x0CFFEC04
WUPM    0x00000003      0x00FFEC04
              ..........

WUPM    0x0000003C      0x33FFCC07      ;UPMA exception
WUPM    0x0000003D      0xFFFFFFFF
WUPM    0x0000003E      0xFFFFFFFF
WUPM    0x0000003F      0xFFFFFFFF
; init memory controller
WM32    0x02200104      0xFFE00D34      ;OR0 : 2MB, all accesses, 6ws, time relax
WM32    0x0220010C      0xFFFF8110      ;OR1
WM32    0x02200114      0xFFC00800      ;OR2
WM32    0x02200100      0x02800001      ;BR0
WM32    0x02200108      0x02100001      ;BR1
WM32    0x02200110      0x00000081      ;BR2
WM16    0x0220017A      0x0400          ;MPTPR : divide by 16
WM32    0x02200170      0x13A01114      ;MAMR

[TARGET]
CPUCLOCK    25000000  ;the CPU clock rate after processing the init list
BDIMODE     AGENT     ;the BDI working mode (LOADONLY | AGENT | GATEWAY)
BREAKMODE   SOFT      ;<AGENT> SOFT or HARD, HARD uses PPC hardware breakpoints

[HOST]
IP          151.120.25.114
FILE        C:\cygnus\b19\demo\mpc860\vxworks
FORMAT      ELF
LOAD        MANUAL      ;load code MANUAL or AUTO after reset
DEBUGPORT   2001
START       0x10000
```

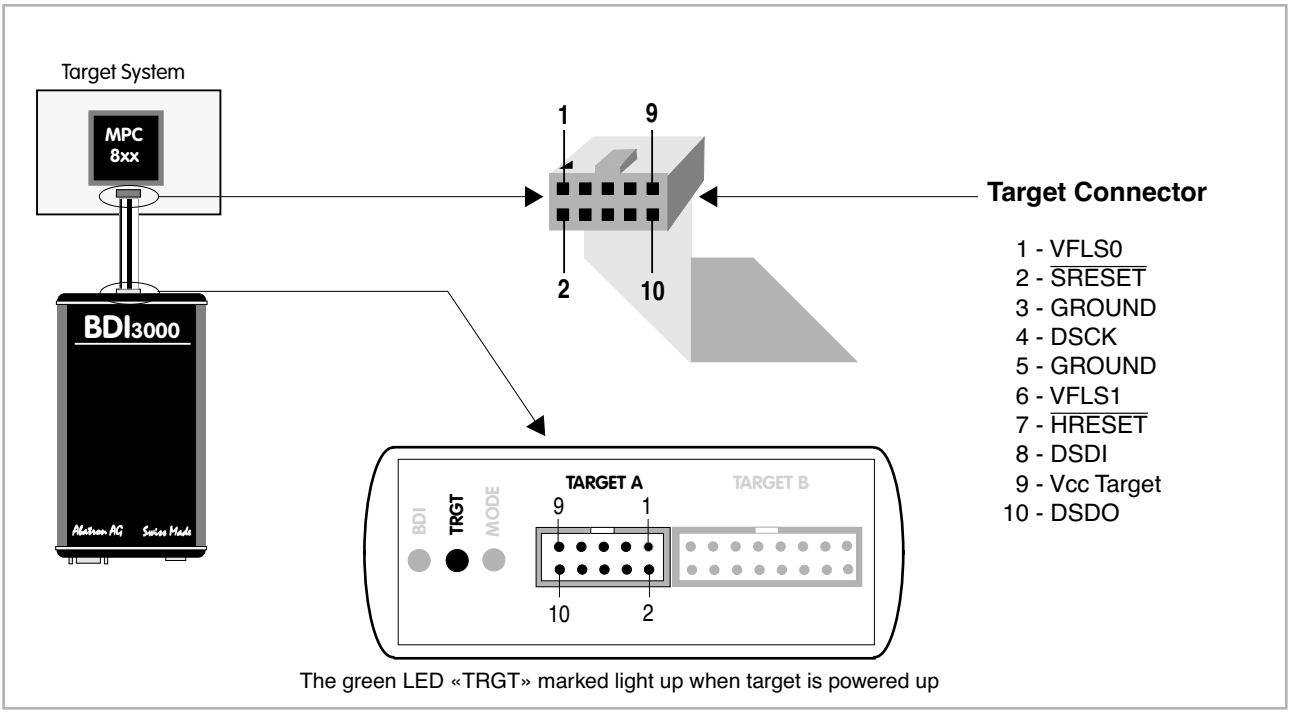
Based on the information in the configuration file, the target is automatically initialized after every reset.

2 Installation

2.1 Connecting the BDI3000 to Target

The cable to the target system is a ten pin flat ribbon cable. In case where the target system has an appropriate connector, the cable can be directly connected. The pin assignment is in accordance with the Motorola specification.

In order to ensure reliable operation of the BDI (EMC, runtimes, etc.) the target cable length must not exceed 20 cm (8").



The green LED «TRGT» marked light up when target is powered up

For TARGET A connector signals see table on next page.

Warning:
Before you can use the BDI3000 with an other target processor type (e.g. PPC <--> ARM), a new setup has to be done (see chapter 2.5). During this process the target cable must be disconnected from the target system.

To avoid data line conflicts, the BDI3000 must be disconnected from the target system while programming a new firmware for an other target CPU.

TARGET A Connector Signals

Pin	Name	Description
1	VFLS0	These pin and pin 6 (VFLS1) indicate to the debug port controller whether or not the MPC is in debug mode. When both VFLS0 and VFLS1 are at "1", the MPC is in debug mode.
2	$\overline{\text{SRESET}}$	This is the Soft-Reset bidirectional signal of the MPC8xx. On the MPC5xx it is an output. The debug port configuration is sampled and determined on the rising-edge of $\overline{\text{SRESET}}$ (for both processor families). On the MPC8xx it is a bidirectional signal which may be driven externally to generate soft reset sequence.
3+5	GND	**System Ground**
4	DSCK	**Debug-port Serial Clock** During asynchronous clock mode, the serial data is clocked into the MPC according to the DSCK clock. The DSCK serves also a role during soft-reset configuration.
6	VFLS1	These pin and pin 1 (VFLS0) indicate to the debug port controller whether or not the MPC is in debug mode. When both VFLS0 and VFLS1 are at "1", the MPC is in debug mode.
7	$\overline{\text{HRESET}}$	This is the Hard-Reset bidirectional signal of the MPC. When this signal is asserted (low) the MPC enters hard reset sequence which include hard reset configuration.
8	DSDI	**Debug-port Serial Data In** Via the DSDI signal, the debug port controller sends its data to the MPC. The DSDI serves also a role during soft-reset configuration.
9	Vcc Target	**1.2 – 5.0V:** This is the target reference voltage. It indicates that the target has power and it is also used to create the logic-level reference for the input comparators. It also controls the output logic levels to the target. It is normally fed from Vdd I/O on the target board.
10	DSDO	**Debug-port Serial Data Out** DSDO is clocked out by the MPC according to the debug port clock, in parallel with the DSDI being clocked in. The DSDO serves also as "READY" signal for the debug port controller to indicate that the debug port is ready to receive controller's command (or data).

Mention of sources used: MPC860ADS User's Manual, Revision A

Enhanced Debug Mode Detection:

For MPC8xx and MPC555 targets, debug mode (Freeze) detection also works when the BDM connector pins VFLS0 and VFLS1 are not connected to the target. If not connected to VFLSx, this BDM connector pins should be left open or tied to Vcc. The BDI uses the following algorithm to check if the target is in debug mode (freezed):

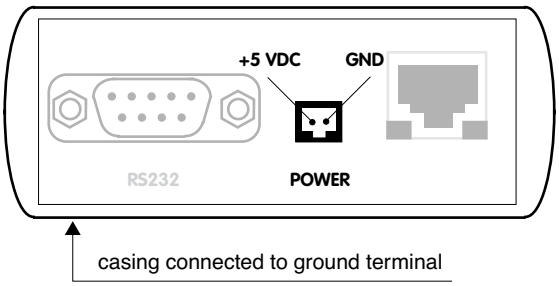
```
BOOL PPC_TargetFreezed(void) {
  if ((VFLS0 != 1) | (VFLS0 != 1)) return FALSE;
  read debug port status;
  if (status == freezed) return TRUE;
  else                   return FALSE;
```

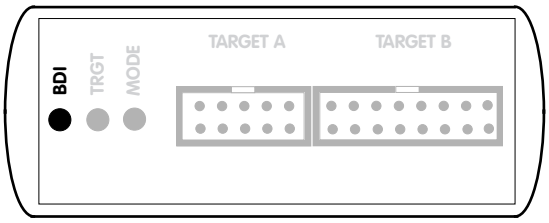

2.2 Connecting the BDI3000 to Power Supply

The BDI3000 needs to be supplied with the enclosed power supply from Abatron (5VDC).

Before use, check if the mains voltage is in accordance with the input voltage printed on power supply. Make sure that, while operating, the power supply is not covered up and not situated near a heater or in direct sun light. Dry location use only.

For error-free operation, the power supply to the BDI3000 must be between 4.75V and 5.25V DC. **The maximal tolerable supply voltage is 5.25 VDC. Any higher voltage or a wrong polarity might destroy the electronics.**





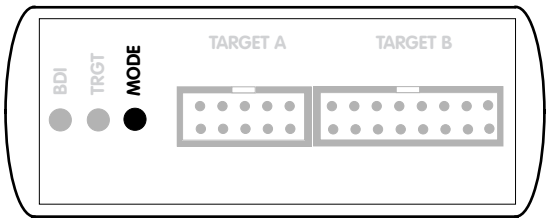
The green LED «BDI» marked light up when 5V power is connected to the BDI3000

Please switch on the system in the following sequence:

- 1 –> external power supply
- 2 –> target system

2.3 Status LED «MODE»

The built in LED indicates the following BDI states:



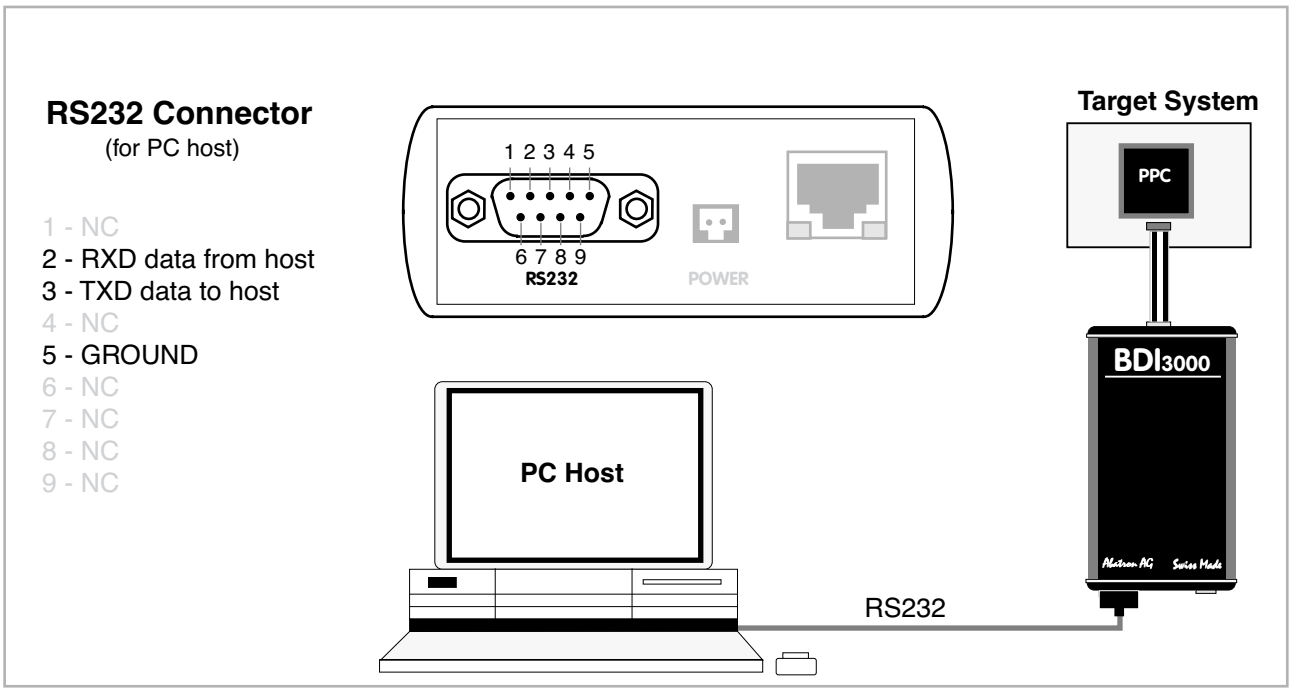
MODE LED	BDI STATES
OFF	The BDI is ready for use, the firmware is already loaded.
ON	The output voltage from the power supply is too low.
BLINK	The BDI «loader mode» is active (an invalid firmware is loaded or loading firmware is active).

2.4 Connecting the BDI3000 to Host

2.4.1 Serial line communication

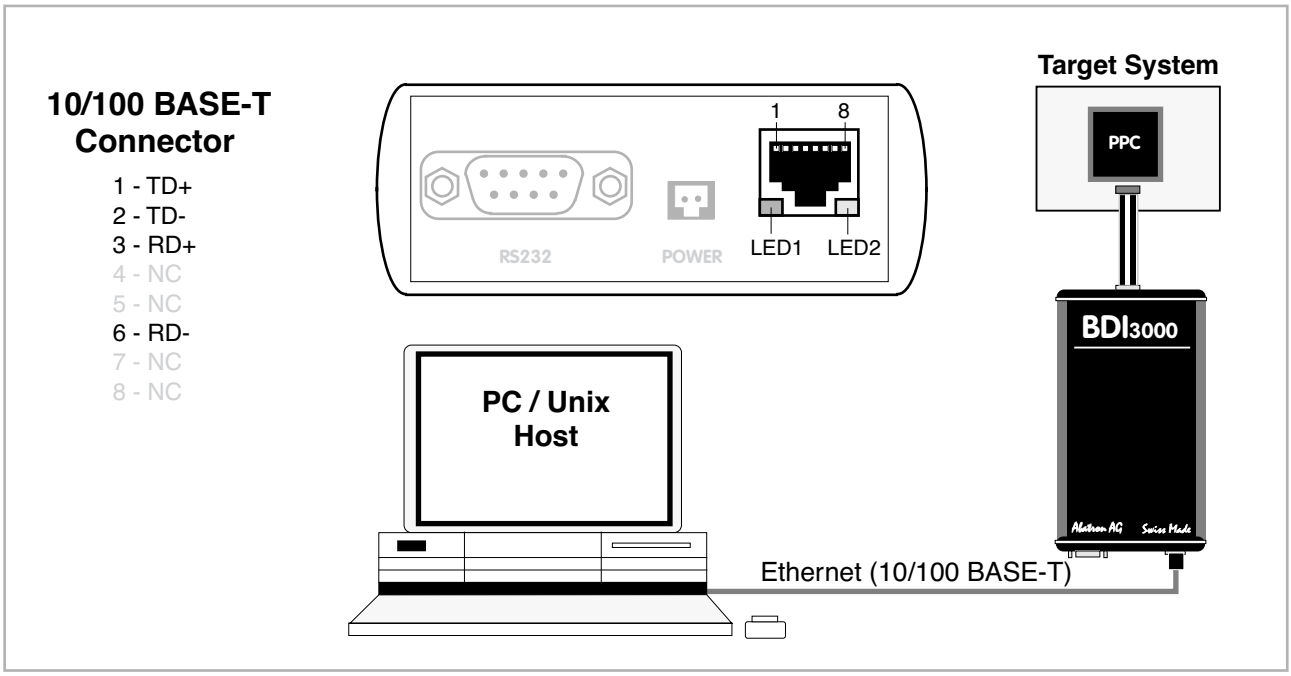
Serial line communication is only used for the initial configuration of the bdiGDB system.

The host is connected to the BDI through the serial interface (COM1...COM4). The communication cable (included) between BDI and Host is a serial cable. There is the same connector pinout for the BDI and for the Host side (Refer to Figure below).



2.4.2 Ethernet communication

The BDI3000 has a built-in 10/100 BASE-T Ethernet interface (see figure below). Connect an UTP (Unshielded Twisted Pair) cable to the BD3000. Contact your network administrator if you have questions about the network.



The following explains the meanings of the built-in LED lights:

LED	Function	Description
LED 1 (green)	Link / Activity	When this LED light is ON, data link is successful between the UTP port of the BDI3000 and the hub to which it is connected. The LED blinks when the BDI3000 is receiving or transmitting data.
LED 2 (amber)	Speed	When this LED light is ON, 100Mb/s mode is selected (default). When this LED light is OFF, 10Mb/s mode is selected

2.5 Installation of the Configuration Software

On the enclosed diskette you will find the BDI configuration software and the firmware required for the BDI3000. For Windows users there is also a TFTP server included.

The following files are on the diskette.

b30ppcgd.exe	Windows Configuration program
b30ppcgd.xxx	Firmware for the BDI3000
tftpsrv.exe	TFTP server for Windows (WIN32 console application)
*.cfg	Configuration files
*.def	Register definition files
bdisetup.zip	ZIP Archive with the Setup Tool sources for Linux / UNIX hosts.

Overview of an installation / configuration process:

- Create a new directory on your hard disk
- Copy the entire contents of the enclosed diskette into this directory
- Linux only: extract the setup tool sources and build the setup tool
- Use the setup tool or Telnet (default IP) to load/update the BDI firmware
  Note: A new BDI has no firmware loaded.
- Use the setup tool or Telnet (default IP) to load the initial configuration parameters
  - IP address of the BDI.
  - IP address of the host with the configuration file.
  - Name of the configuration file. This file is accessed via TFTP.
  - Optional network parameters (subnet mask, default gateway).

Activating BOOTP:
The BDI can get the network configuration and the name of the configuration file also via BOOTP. For this simple enter 0.0.0.0 as the BDI's IP address (see following chapters). If present, the subnet mask and the default gateway (router) is taken from the BOOTP vendor-specific field as defined in RFC 1533.

With the Linux setup tool, simply use the default parameters for the -c option:

```
[root@LINUX_1 bdisetup]# ./bdisetup -c -p/dev/ttyS0 -b57
```

The MAC address is derived from the serial number as follows:
MAC: 00-0C-01-xx-xx-xx , replace the xx-xx-xx with the 6 left digits of the serial number
Example: SN# 33123407 ==>> 00-0C-01-33-12-34

Default IP: 192.168.53.72
Before the BDI is configured the first time, it has a default IP of 192.168.53.72 that allows an initial configuration via Ethernet (Telnet or Setup Tools). If your host is not able to connect to this default IP, then the initial configuration has to be done via the serial connection.

2.5.1 Configuration with a Linux / Unix host

The firmware update and the initial configuration of the BDI3000 is done with a command line utility. In the ZIP Archive bdisetup.zip are all sources to build this utility. More information about this utility can be found at the top in the bdisetup.c source file. There is also a make file included.
Starting the tool without any parameter displays information about the syntax and parameters.

To avoid data line conflicts, the BDI3000 must be disconnected from the target system while programming the firmware for an other target CPU family.

Following the steps to bring-up a new BDI3000:

1. Build the setup tool:
The setup tool is delivered only as source files. This allows to build the tool on any Linux / Unix host. To build the tool, simply start the make utility.

```
[root@LINUX_1 bdisetup]# make
cc -O2   -c -o bdisetup.o bdisetup.c
cc -O2   -c -o bdicnf.o bdicnf.c
cc -O2   -c -o bdidll.o bdidll.c
cc -s bdisetup.o bdicnf.o bdidll.o -o bdisetup
```

2. Check the serial connection to the BDI:
With "bdisetup -v" you may check the serial connection to the BDI. The BDI will respond with information about the current loaded firmware and network configuration.
Note: Login as root, otherwise you probably have no access to the serial port.

```
$ ./bdisetup -v -p/dev/ttyS0 -b115
BDI Type : BDI3000 (SN: 30000154)
Loader   : V1.00
Firmware : unknown
MAC      : ff-ff-ff-ff-ff-ff
IP Addr  : 255.255.255.255
Subnet   : 255.255.255.255
Gateway  : 255.255.255.255
Host IP  : 255.255.255.255
Config   : ÿÿÿÿÿÿÿÿ........
```

3. Load/Update the BDI firmware:
With "bdisetup -u" the firmware is programmed into the BDI3000 flash memory. This configures the BDI for the target you are using. Based on the parameters -a and -t, the tool selects the correct firmware file. If the firmware file is in the same directory as the setup tool, there is no need to enter a -d parameter.

```
$ ./bdisetup -u -p/dev/ttyS0 -b115 -aGDB -tMPC800
Connecting to BDI loader
Programming firmware with ./b30ppcgd.100
Erasing firmware flash ....
Erasing firmware flash passed
Programming firmware flash ....
Programming firmware flash passed
```


4. Transmit the initial configuration parameters:
With "bdisetup -c" the configuration parameters are written to the flash memory within the BDI.
The following parameters are used to configure the BDI:

BDI IP Address	The IP address for the BDI3000. Ask your network administrator for assigning an IP address to this BDI3000. Every BDI3000 in your network needs a different IP address.
Subnet Mask	The subnet mask of the network where the BDI is connected to. A subnet mask of 255.255.255.255 disables the gateway feature. Ask your network administrator for the correct subnet mask. If the BDI and the host are in the same subnet, it is not necessary to enter a subnet mask.
Default Gateway	Enter the IP address of the default gateway. Ask your network administrator for the correct gateway IP address. If the gateway feature is disabled, you may enter 255.255.255.255 or any other value.
Config - Host IP Address	Enter the IP address of the host with the configuration file. The configuration file is automatically read by the BDI3000 after every start-up.
Configuration file	Enter the full path and name of the configuration file. This file is read via TFTP. Keep in mind that TFTP has it's own root directory (usual /tftpboot). You can simply copy the configuration file to this directory and the use the file name without any path. For more information about TFTP use "man tftpd".

```
$ ./bdisetup -c -p/dev/ttyS0 -b115 \
> -i151.120.25.102 \
> -h151.120.25.112 \
> -fe:/bdi3000/mytarget.cfg
Connecting to BDI loader
Writing network configuration
Configuration passed
```

5. Check configuration and exit loader mode:
The BDI is in loader mode when there is no valid firmware loaded or you connect to it with the setup tool. While in loader mode, the Mode LED is blinking. The BDI will not respond to network requests while in loader mode. To exit loader mode, the "bdisetup -v -s" can be used. You may also power-off the BDI, wait some time (1min.) and power-on it again to exit loader mode.

```
$ ./bdisetup -v -p/dev/ttyS0 -b115 -s
BDI Type : BDI3000 (SN: 30000154)
Loader   : V1.00
Firmware : V1.00 bdiGDB for MPC8xx/MPC5xx
MAC      : 00-0c-01-30-00-01
IP Addr  : 151.120.25.102
Subnet   : 255.255.255.255
Gateway  : 255.255.255.255
Host IP  : 151.120.25.112
Config   : /bdi3000/mytarget.cfg
```

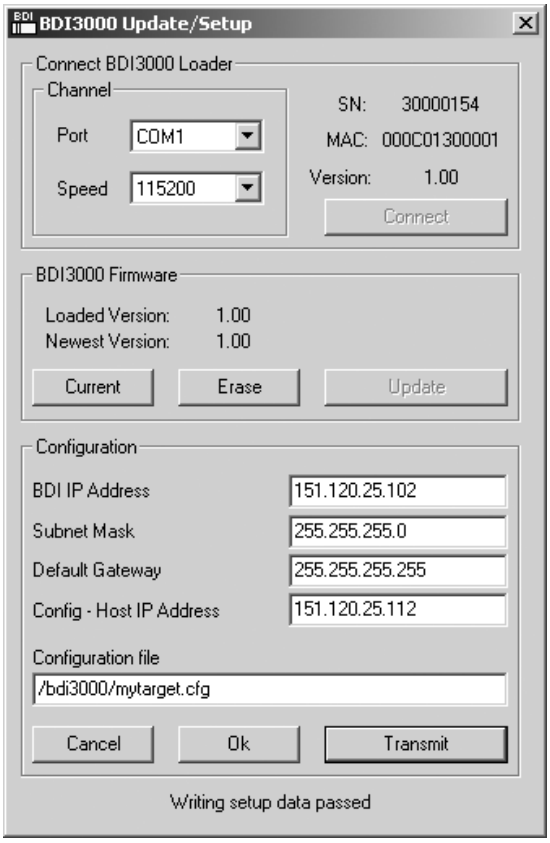
The Mode LED should go off, and you can try to connect to the BDI via Telnet.

```
$ telnet 151.120.25.102
```


2.5.2 Configuration with a Windows host

First make sure that the BDI is properly connected (see Chapter 2.1 to 2.4).

To avoid data line conflicts, the BDI3000 must be disconnected from the target system while programming the firmware for an other target CPU family.



dialog box «BDI3000 Update/Setup»

Before you can use the BDI3000 together with the GNU debugger, you must store the initial configuration parameters in the BDI3000 flash memory. The following options allow you to do this:

Port	Select the communication port where the BDI3000 is connected during this setup session. If you select Network, make sure the Loader is already active (Mode LED blinking). If there is already a firmware loaded and running, use the Telnet command "boot loader" to activate Loader Mode.
Speed	Select the baudrate used to communicate with the BDI3000 loader during this setup session.
Connect	Click on this button to establish a connection with the BDI3000 loader. Once connected, the BDI3000 remains in loader mode until it is restarted or this dialog box is closed.
Current	Press this button to read back the current loaded BDI3000 firmware version. The current firmware version will be displayed.

Erase	Press this button to erase the current loaded firmware.
Update	This button is only active if there is a newer firmware version present in the execution directory of the bdiGDB setup software. Press this button to write the new firmware into the BDI3000 flash memory.
BDI IP Address	Enter the IP address for the BDI3000. Use the following format: xxx.xxx.xxx.xxx e.g.151.120.25.101 Ask your network administrator for assigning an IP address to this BDI3000. Every BDI3000 in your network needs a different IP address.
Subnet Mask	Enter the subnet mask of the network where the BDI is connected to. Use the following format: xxx.xxx.xxx.xxxe.g.255.255.255.0 A subnet mask of 255.255.255.255 disables the gateway feature. Ask your network administrator for the correct subnet mask.
Default Gateway	Enter the IP address of the default gateway. Ask your network administrator for the correct gateway IP address. If the gateway feature is disabled, you may enter 255.255.255.255 or any other value.
Config - Host IP Address	Enter the IP address of the host with the configuration file. The configuration file is automatically read by the BDI3000 after every start-up.
Configuration file	Enter the full path and name of the configuration file. This name is transmitted to the TFTP server when reading the configuration file.
Transmit	Click on this button to store the configuration in the BDI3000 flash memory.

Note:
Using this setup tool via the Network channel is only possible if the BDI3000 is already in Loader mode (Mode LED blinking). To force Loader mode, enter "boot loader" at the Telnet. The setup tool tries first to establish a connection to the Loader via the IP address present in the "BDI IP Address" entry field. If there is no connection established after a time-out, it tries to connect to the default IP (192.168.53.72).

2.5.3 Configuration via Telnet / TFTP

The firmware update and the initial configuration of the BDI3000 can also be done interactively via a Telnet connection and a running TFTP server on the host with the firmware file. In cases where it is not possible to connect to the default IP, the initial setup has to be done via a serial connection.

To avoid data line conflicts, the BDI3000 must be disconnected from the target system while programming the firmware for an other target CPU family.

Following the steps to bring-up a new BDI3000 or updating the firmware.
Connect to the BDI Loader via Telnet.
If a firmware is already running enter "boot loader" and reconnect via Telnet.

```
$ telnet 192.168.53.72
or
$ telnet <your BDI IP address>
```

Update the network parameters so it matches your needs:

```
LDR>network
    BDI MAC     : 00-0c-01-30-00-01
    BDI IP      : 192.168.53.72
    BDI Subnet  : 255.255.255.0
    BDI Gateway : 255.255.255.255
    Config IP   : 255.255.255.255
    Config File :

LDR>netip 151.120.25.102
LDR>nethost 151.120.25.112
LDR>netfile /bdi3000/mytarget.cfg


LDR>network
    BDI MAC     : 00-0c-01-30-00-01
    BDI IP      : 151.120.25.102
    BDI Subnet  : 255.255.255.0
    BDI Gateway : 255.255.255.255
    Config IP   : 151.120.25.112
    Config File : /bdi3000/mytarget.cfg

LDR>network save
saving network configuration ... passed
    BDI MAC     : 00-0c-01-30-00-01
    BDI IP      : 151.120.25.102
    BDI Subnet  : 255.255.255.0
    BDI Gateway : 255.255.255.255
    Config IP   : 151.120.25.112
    Config File : /bdi3000/mytarget.cfg
```

In case the subnet has changed, reboot before trying to load the firmware

```
LDR>boot loader
```


Connect again via Telnet and program the firmware into the BDI flash:

```
$ telnet 151.120.25.102

LDR>info
    BDI Firmware: not loaded
    BDI CPLD ID : 01285043
    BDI CPLD UES: ffffffff
    BDI MAC     : 00-0c-01-30-00-01
    BDI IP      : 151.120.25.102
    BDI Subnet  : 255.255.255.0
    BDI Gateway : 255.255.255.255
    Config IP   : 151.120.25.112
    Config File : /bdi3000/mytarget.cfg

LDR>fwload e:/temp/b30ppcgd.100
erasing firmware flash ... passed
programming firmware flash ... passed

LDR>info
    BDI Firmware: 13 / 1.00
    BDI CPLD ID : 01285043
    BDI CPLD UES: ffffffff
    BDI MAC     : 00-0c-01-30-00-01
    BDI IP      : 151.120.25.102
    BDI Subnet  : 255.255.255.0
    BDI Gateway : 255.255.255.255
    Config IP   : 151.120.25.112
    Config File : /bdi3000/mytarget.cfg
LDR>
```

To boot now into the firmware use:

```
LDR>boot
```

The Mode LED should go off, and you can try to connect to the BDI again via Telnet.

```
telnet 151.120.25.102
```


2.6 Testing the BDI3000 to host connection

After the initial setup is done, you can test the communication between the host and the BDI3000. There is no need for a target configuration file and no TFTP server is needed on the host.

- If not already done, connect the BDI3000 system to the network.
- Power-up the BDI3000.
- Start a Telnet client on the host and connect to the BDI3000 (the IP address you entered during initial configuration).
- If everything is okay, a sign on message like «BDI Debugger for Embedded PowerPC» and a list of the available commands should be displayed in the Telnet window.

2.7 TFTP server for Windows

The bdiGDB system uses TFTP to access the configuration file and to load the application program. Because there is no TFTP server bundled with Windows, Abatron provides a TFTP server application **tftpsrv.exe**. This WIN32 console application runs as normal user application (not as a system service).

Command line syntax: tftpsrv [p] [w] [dRootDirectory]

Without any parameter, the server starts in read-only mode. This means, only read access request from the client are granted. This is the normal working mode. The bdiGDB system needs only read access to the configuration and program files.

The parameter [p] enables protocol output to the console window. Try it.
The parameter [w] enables write accesses to the host file system.
The parameter [d] allows to define a root directory.

`tftpsrv p`	Starts the TFTP server and enables protocol output
`tftpsrv p w`	Starts the TFTP server, enables protocol output and write accesses are allowed.
`tftpsrv dC:\tftp\`	Starts the TFTP server and allows only access to files in C:\tftp and its subdirectories. As file name, use relative names. For example "bdi\mpc750.cfg" accesses "C:\tftp\bdi\mpc750.cfg"

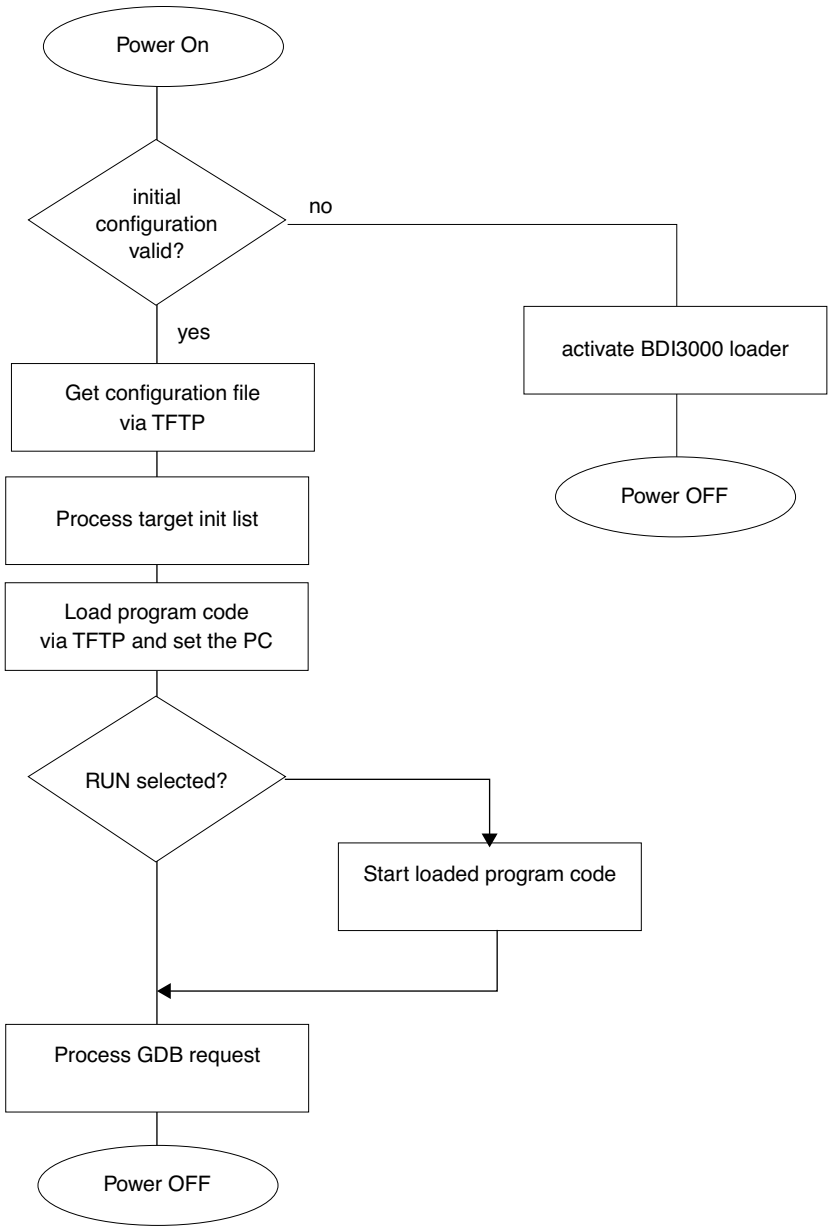
You may enter the TFTP server into the Startup group so the server is started every time you login.

3 Using bdiGDB

3.1 Principle of operation

The firmware within the BDI handles the GDB request and accesses the target memory or registers via the BDM interface. There is no need for any debug software on the target system. After loading the code via TFTP debugging can begin at the very first assembler statement.

Whenever the BDI system is powered-up the following sequence starts:



Breakpoints:
There are two breakpoint modes supported. One of them (SOFT) is implemented by replacing application code with a TRAP instruction. The other (HARD) uses the built in breakpoint logic. If HARD is used, only up to 4 breakpoints can be active at the same time.
The following example selects SOFT as the breakpoint mode:

```
BREAKMODE   SOFT      ;<AGENT> SOFT or HARD, HARD uses PPC hardware breakpoints
```

All the time the application is suspended (i.e. caused by a breakpoint) the target processor remains freezed.

3.2 Configuration File

The configuration file is automatically read by the BDI after every power on.
The syntax of this file is as follows:

```
; comment
[part name]
identifier parameter1 parameter2 ..... parameterN  ; comment
identifier parameter1 parameter2 ..... parameterN
.....
[part name]
identifier parameter1 parameter2 ..... parameterN
identifier parameter1 parameter2 ..... parameterN
.....
              etc.
```

Numeric parameters can be entered as decimal (e.g. 700) or as hexadecimal (0x80000).

3.2.1 Part [INIT]

The part [INIT] defines a list of commands which should be executed every time the target comes out of reset. The commands are used to get the target ready for loading the program file. The SIM registers (chip select, clock, ...) are usually initialized with this command list.

WGPR register value	Write value to the selected general purpose register. register the register number 0 .. 31 value the value to write into the register Example: WGPR 0 5
WSPR register value	Write value to the selected special purpose register. register the register number value the value to write into the register Example: WSPR 27 0x00001002 ; SRR1 : ME,RI
WREG name value	Write value to the selected CPU register by name name the register name (MSR,CR,XER,LR,CTR,DSISR,...) value the value to write into the register Example: WREG MSR 0x00001002
WM8 address value	Write a byte (8bit) to the selected memory place. address the memory address value the value to write to the target memory Example: WM8 0xFFFFFA21 0x04 ; SYPCR: watchdog disable ...
WM16 address value	Write a half word (16bit) to the selected memory place. address the memory address value the value to write to the target memory Example: WM16 0x02200200 0x0002 ; TBSCR
WM32 address value	Write a word (32bit) to the selected memory place. address the memory address value the value to write to the target memory Example: WM32 0x02200000 0x01632440 ; SIUMCR
SUPM cmdaddr dataaddr	Starts a sequence of writes to the UPM RAM array. cmdaddr the address of the UPM command register dataaddr the address of the UPM data register Example: SUPM 0x02200168 0x0220017c
WUPM command data	Write indirect to the UPM RAM array. The data is always written first. command this value is written to the UPM command register data this value is written to the UPM data register Example: WUPM 0x00000001 0x0FFFEC04
DELAY value	Delay for the selected time. A delay may be necessary to let the clock PLL lock again after a new clock rate is selected. value the delay time in milliseconds (1...30000) Example: DELAY 500 ; delay for 0.5 seconds

3.2.2 Part [TARGET]

The part [TARGET] defines some target specific values.

CPUTYPE type
This value gives the BDI information about the connected CPU:
- type — The CPU type from the following list: MPC500 or MPC800
- Example: CPUTYPE MPC500

CPUCLOCK value
The BDI needs to know how fast the target CPU runs after processing the init list. The BDM communication speed is selected based on this value. If this value defines a clock rate that is higher than the real clock, BDM communication may fail. When defining a clock rate slower than possible, BDM communication still works but not as fast as possible.
Important: When programming the MPC555 internal flash, this value is used to calculate the appropriate timing parameters.
- value — the CPU clock in hertz
- Example: CPUCLOCK 25000000 ; CPU clock is 25.0MHz

BDIMODE mode param
This parameter selects the BDI debugging mode. The following modes are supported:
- LOADONLY — Loads and starts the application core. No debugging via BDM.
- AGENT — The debug agent runs within the BDI. There is no need for any debug software on the target. This mode accepts a second parameter. If RUN is entered as a second parameter, the loaded application will be started immediately, otherwise only the PC is set and BDI waits for GDB requests.
- Example: BDIMODE AGENT RUN

STARTUP mode [runtime]
This parameter selects the target startup mode. The following modes are supported:
- RESET — This default mode forces the target to debug mode immediately out of reset. No code is executed after reset.
- STOP — In this mode, the BDI lets the target execute code for "runtime" milliseconds after reset. This mode is useful when monitor code should initialize the target system.
- RUN — After reset, the target executes code until stopped by the Telnet "halt" command.
- Example: STARTUP STOP 3000 ; let the CPU run for 3 seconds

WORKSPACE address
In order to access the floating-point registers of a MPC5xx microprocessor, the BDI needs a workspace of 8 bytes in target RAM. Enter the base address of this RAM area.
- address — the address of the RAM area
- Example: WORKSPACE 0x00000000

BREAKMODE mode [op] This parameter defines how breakpoints are implemented. The current mode can also be changed via the Telnet interface.

SOFT	This is the normal mode. Breakpoints are implemented by replacing code with a TRAP (default) or ILLEGAL instruction. The optional [op] parameter defines if a trap or an illegal instruction is used.
HARD	In this mode, the PPC breakpoint hardware is used. Only 4 breakpoints at a time are supported.
Example:	BREAKMODE HARD ; use hardware breakpoints BREAKMODE SOFT ILLEGAL

STEPMODE mode This parameter defines how single step (instruction step) is implemented. Use thealternate step mode (HWBP) if the default step mode (MSR[SE] bit) causes problems.

TRACE	This is the default mode. Single step is implemented by setting the SE bit in MSR.
HWBP	In this mode, one or two hardware breakpoints are used to implement single stepping.
Example:	STEPMODE HWBP

MMU XLAT [kb] In order to support Linux kernel debugging when MMU is on, the BDI translates effective (virtual) to physical addresses. This translation is done based on the current MMU configuration. Currently only the Linux model with 4k pages is supported. If this configuration line is present and address relocation active (MSR bits IR/DR), the BDI translates the addresses received from GDB before it accesses physical memory. The optional parameter defines the kernel virtual base address (default is 0xC0000000) and is used for default address translation. For more information see also chapter "Embedded Linux MMU Support". Addresses entered at the Telnet are never translated. Translation can be probed with the Telnet command PHYS.
If kb is defined as 0x00000000 then the BDI uses only the current MPC8xx TLB's to translate a virtual address, there is no page table search in this case. Useful for systems where a fixed MMU mapping is used.

kb	The kernel virtual base address (KERNELBASE) or 0x00000000 for a translation based only on the current TLB's.
Example:	MMU XLAT ;enable address translation

PTBASE addr This parameter defines the physical memory address where the BDI looks for the address of the array with the two page table pointers. For more information see also chapter "Embedded Linux MMU Support".

addr	Physical address of the memory used to store the virtual address of the array with the two page table pointers.
Example:	PTBASE 0xf0

REGLIST list

With GDB version 5.0, the number of registers read from the target has been increased. Additional registers like SR's, BAT's and SPR's are requested when you select a specific PowerPC variant with the "set processor" command (see GDB source file rs6000-tdep.c). In order to be compatible with older GDB versions and to optimize the time spent to read registers, this parameter can be used. You can define which register group is really read from the target. By default only STD are read and transferred. This default is compatible with older GDB versions. The following names are use to select a register group:

STD	The standard (old) register block. The FPR registers are not read from the target but transferred. You can't disable this register group.
FPR	The floating point registers are read and transferred.
SR	not available for MPC8xx/5xx targets.
BAT	not available for MPC8xx/5xx targets
SPR	Some additional special purpose registers
AUX	The debug module special purpose registers
ALL	Include all register groups
Example:	REGLIST STD ; only standard registers REGLIST STD FPR SPR ; all except SR and BAT

SIO port [baudrate]

When this line is present, a TCP/IP channel is routed to the BDI's RS232 connector. The port parameter defines the TCP port used for this BDI to host communication. You may choose any port except 0 and the default Telnet port (23). On the host, open a Telnet session using this port. Now you should see the UART output in this Telnet session. You can use the normal Telnet connection to the BDI in parallel, they work completely independent. Also input to the UART is implemented.

port	The TCP/IP port used for the host communication.
baudrate	The BDI supports 2400 ... 115200 baud
Example:	SIO 7 9600 ;TCP port for virtual IO

3.2.3 Part [HOST]

The part [HOST] defines some host specific values.

IP ipaddress

The IP address of the host.

ipaddress	the IP address in the form xxx.xxx.xxx.xxx
Example:	IP 151.120.25.100

FILE filename

The default name of the file that is loaded into RAM using the Telnet 'load' command. This name is used to access the file via TFTP. If the filename starts with a $, this $ is replace with the path of the configuration file name.

filename	the filename including the full path or $ for relative path.
Example:	FILE F:\gnu\demo\mpc\test.elf FILE $test.elf

FORMAT format [offset]

The format of the image file and an optional load address offset. Currently binary, S-record, a.out and ELF formats are supported. If the image is already stored in ROM on the target, select ROM as the format. The optional parameter "offset" is added to any load address read from the image file.

format	BIN, SREC, AOUT, ELF, IMAGE* or ROM
Example:	FORMAT ELF FORMAT ELF 0x10000

LOAD mode

In Agent mode, this parameters defines if the code is loaded automatically after every reset.

mode	AUTO, MANUAL
Example:	LOAD MANUAL

START address

The address where to start the program file. If this value is not defined and the core is not in ROM, the address is taken from the code file. If this value is not defined and the core is already in ROM, the PC will not be set before starting the program file. This means, the program starts at the normal reset address (0x0100).

address	the address where to start the program file
Example:	START 0x1000

* Special IMAGE load format:
The IMAGE format is a special version of the ELF format used to load a Linux boot image into target memory. When this format is selected, the BDI loads not only the loadable segment as defined in the Program Header, it also loads the rest of the file up to the Section Header Table. The relationship between load address and file offset will be maintained throughout this process. This way, the compressed Linux image and a optional RAM disk image will also be loaded.

DEBUGPORT port [RECONNECT]

The TCP port GDB uses to access the target. If the RECONNECT parameter is present, an open TCP/IP connection (Telnet/GDB) will be closed if there is a connect request from the same host (same IP address).

port the TCP port number (default = 2001)

Example: DEBUGPORT 2001

PROMPT string

This entry defines a new Telnet prompt. The current prompt can also be changed via the Telnet interface.

Example: PROMPT MPC860>

DUMP filename

The default file name used for the Telnet DUMP command.

filename the filename including the full path

Example: DUMP dump.bin

TELNET mode

By default the BDI sends echoes for the received characters and supports command history and line editing. If it should not send echoes and let the Telnet client in "line mode", add this entry to the configuration file.

mode ECHO (default), NOECHO or LINE

Example: TELNET NOECHO ; use old line mode

3.2.4 Part [FLASH]

The Telnet interface supports programming and erasing of flash memories. The bdiGDB system has to know which type of flash is used, how the chip(s) are connected to the CPU and which sectors to erase in case the ERASE command is entered without any parameter.

CHIPTYPE type

This parameter defines the type of flash used. It is used to select the correct programming algorithm.
Note: A workspace is necessary for STRATA, MIRROR, AT29, MPC5xx.

format AM29F, AM29BX8, AM29BX16, I28BX8, I28BX16, AT49, AT49X8, AT49X16, STRATAX8, STRATAX16, AT29X8, AT29X16, MIRROR, MIRRORX8, MIRRORX16, M58X32, AM29DX16, AM29DX32 AM29BDDX16, AM29BDDX32 MPC555, MPC555SHD, MPC565, MPC565SHD

Example: CHIPTYPE AM29F

CHIPSIZE size

The size of **one** flash chip in bytes (e.g. AM29F010 = 0x20000). This value is used to calculate the starting address of the current flash memory bank. For MPC5xx internal flash, this parameter is not used.

size the size of one flash chip in bytes

Example: CHIPSIZE 0x80000

BUSWIDTH width

Enter the width of the memory bus that leads to the flash chips. Do not enter the width of the flash chip itself. The parameter CHIPTYPE carries the information about the number of data lines connected to one flash chip. For example, enter 16 if you are using two AM29F010 to build a 16bit flash memory bank.
For MPC5xx internal flash, this parameter is not used.

with the width of the flash memory bus in bits (8 | 16 | 32)

Example: BUSWIDTH 16

FILE filename

The default name of the file that is programmed into flash using the Telnet 'prog' command. This name is used to access the file via TFTP. If the filename starts with a $, this $ is replace with the path of the configuration file name. This name may be overridden interactively at the Telnet interface.

filename the filename including the full path or $ for relative path.

Example: FILE F:\gnu\xscale\bootrom.hex
FILE $bootrom.hex

FORMAT format [offset]

The format of the file and an optional address offset. The optional parameter "offset" is added to any load address read from the program file.

format SREC, BIN, AOUT, ELF or IMAGE

Example: FORMAT SREC
FORMAT ELF 0x10000

WORKSPACE address

If a workspace is defined, the BDI uses a faster programming algorithm that runs out of RAM on the target system. Otherwise, the algorithm is processed within the BDI. The workspace is used for a 1kByte data buffer and to store the algorithm code. There must be at least 2kBytes of RAM available for this purpose. Programming MPC5xx internal flash also needs a workspace in target RAM. A workspace is also required for the AT29 and STRATA algorithm.

address	the address of the RAM area
Example:	WORKSPACE 0x00000000

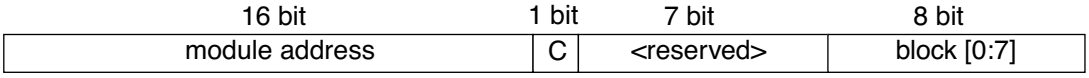
ERASE addr [mode [wait]]

The flash memory may be individually erased or unlocked via the Telnet interface. In order to make erasing of multiple flash sectors easier, you can enter an erase list. All entries in the erase list will be processed if you enter ERASE at the Telnet prompt without any parameter. This list is also used if you enter UNLOCK at the Telnet without any parameters.

address	Address of the flash sector, block or chip to erase
mode	BLOCK, CHIP, UNLOCK Without this optional parameter, the BDI executes a sector erase. If supported by the chip, you can also specify a block or chip erase. If UNLOCK is defined, the entry is also part of the unlock list. This unlock list is processed if the Telnet UNLOCK command is entered without any parameters.
wait	The wait time in ms is only used for the unlock mode. After starting the flash unlock, the BDI waits until it processes the next entry.
Example:	ERASE 0xff040000 ;erase sector 4 of flash ERASE 0xff060000 ;erase sector 6 of flash ERASE 0xff000000 CHIP ;erase whole chip(s) ERASE 0xff010000 UNLOCK 100 ;unlock, wait 100ms

MPC555 Internal Flash:

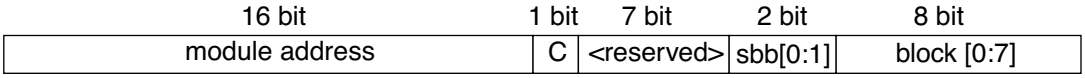
For the MPC555 internal flash, the BDI assumes the following structure of the address:



module address	The 16 most significant bits of the flash module address.
C	The censor bit. If this bit is set, the censor information is erased.
block	The bit mask to select the flash block to erase. Bit ordering is the same as in the CMFCTL register (see MPC555 manual).

MPC565 Internal Flash:

For the MPC565 internal flash, the BDI assumes the following structure of the address:



module address	The 16 most significant bits of the flash module address.
C	The censor bit. If this bit is set, the censor information is erased.
sbb*	The bit mask to select the small blocks to erase. Bit ordering is the same as in the UC3FCTL register (see MPC565 manual).
block	The bit mask to select the flash block to erase. Bit ordering is the same as in the UC3FCTL register (see MPC565 manual).

* The BDI does not write implicit any value to the UC3FMCRE registers. If small blocks are used, the appropriate value has to be written to the UC3FMCRE registers via the BDI initialization list or via the connected debugger.

Supported Flash Memories:
There are currently 3 standard flash algorithm supported. The AMD, Intel and Atmel AT49 algorithm. Almost all currently available flash memories can be programmed with one of this algorithm. The flash type selects the appropriate algorithm and gives additional information about the used flash.

For 8bit only flash:	AM29F (MIRROR), I28BX8, AT49
For 8/16 bit flash in 8bit mode:	AM29BX8 (MIRRORX8), I28BX8 (STRATAX8), AT49X8
For 8/16 bit flash in 16bit mode:	AM29BX16 (MIRRORX16), I28BX16 (STRATAX16), AT49X16
For 16bit only flash:	AM29BX16, I28BX16, AT49X16
For 16/32 bit flash in 16bit mode:	AM29DX16, AM29BDDX16
For 16/32 bit flash in 32bit mode:	AM29DX32, AM29BDDX32
For 32bit only flash:	M58X32

Some newer Spansion MirrorBit flashes cannot be programmed with the MIRRORX16 algorithm because of the used unlock address offset. Use S29M32X16 for these flashes.

The AMD and AT49 algorithm are almost the same. The only difference is, that the AT49 algorithm does not check for the AMD status bit 5 (Exceeded Timing Limits).
Only the AMD and AT49 algorithm support chip erase. Block erase is only supported with the AT49 algorithm. If the algorithm does not support the selected mode, sector erase is performed. If the chip does not support the selected mode, erasing will fail. The erase command sequence is different only in the 6th write cycle. Depending on the selected mode, the following data is written in this cycle (see also flash data sheets): 0x10 for chip erase, 0x30 for sector erase, 0x50 for block erase.
To speed up programming of Intel Strata Flash and AMD MirrorBit Flash, an additional algorithm is implemented that makes use of the write buffer. This algorithm needs a workspace, otherwise the standard Intel/AMD algorithm is used.

The following table shows some examples:

Flash	x 8	x 16	x 32	Chipsize
Am29F010	AM29F	-	-	0x020000
Am29F800B	AM29BX8	AM29BX16	-	0x100000
Am29DL323C	AM29BX8	AM29BX16	-	0x400000
Am29PDL128G	-	AM29DX16	AM29DX32	0x01000000
Intel 28F032B3	I28BX8	-	-	0x400000
Intel 28F640J3A	STRATAX8	STRATAX16	-	0x800000
Intel 28F320C3	-	I28BX16	-	0x400000
AT49BV040	AT49	-	-	0x080000
AT49BV1614	AT49X8	AT49X16	-	0x200000
M58BW016BT	-	-	M58X32	0x200000
SST39VF160	-	AT49X16	-	0x200000
Am29LV320M	MIRRORX8	MIRRORX16	-	0x400000

Note:
Some Intel flash chips (e.g. 28F800C3, 28F160C3, 28F320C3) power-up with all blocks in locked state. In order to erase/program those flash chips, use the init list to unlock the appropriate blocks:

```
WM16    0xFFF00000    0x0060        unlock block 0
WM16    0xFFF00000    0x00D0
WM16    0xFFF10000    0x0060        unlock block 1
WM16    0xFFF10000    0x00D0
        ....
WM16    0xFFF00000    0xFFFF        select read mode
```

or use the Telnet "unlock" command:

```
UNLOCK [<addr> [<delay>]]
```

addr	This is the address of the sector (block) to unlock
delay	A delay time in milliseconds the BDI waits after sending the unlock command to the flash. For example, clearing all lock-bits of an Intel J3 Strata flash takes up to 0.7 seconds.

If "unlock" is used without any parameter, all sectors in the erase list with the UNLOCK option are processed.

To clear all lock-bits of an Intel J3 Strata flash use for example:

```
BDI> unlock 0xFF000000 1000
```

To erase or unlock multiple, continuous flash sectors (blocks) of the same size, the following Telnet commands can be used:

```
ERASE  <addr> <step> <count>
UNLOCK <addr> <step> <count>
```

addr	This is the address of the first sector to erase or unlock.
step	This value is added to the last used address in order to get to the next sector. In other words, this is the size of one sector in bytes.
count	The number of sectors to erase or unlock.

The following example unlocks all 256 sectors of an Intel Strata flash (28F256K3) that is mapped to 0x00000000. In case there are two flash chips to get a 32bit system, double the "step" parameter.

```
BDI> unlock 0x00000000 0x20000 256
```


Examples:

ADS860 flash memory:

```
[FLASH]
CHIPTYPE       AM29F          ;Flash type (AM29F | AM29BX8 | AM29BX16 | I28BX8 | I28BX16)
CHIPSIZE       0x80000        ;The size of one flash chip in bytes (e.g. AM29F010 = 0x20000)
BUSWIDTH       32             ;The width of the flash memory bus in bits (8 | 16 | 32)
FILE           E:\ada\demo\mpc860\bootrom.hex  ;The file to program
ERASE          0x02800000     ;erase sector 0 of flash SIMM (MCM29F040)
ERASE          0x02840000     ;erase sector 1 of flash SIMM
ERASE          0x02880000     ;erase sector 2 of flash SIMM
ERASE          0x028C0000     ;erase sector 3 of flash SIMM
ERASE          0x02900000     ;erase sector 4 of flash SIMM
ERASE          0x02940000     ;erase sector 5 of flash SIMM
ERASE          0x02980000     ;erase sector 6 of flash SIMM
ERASE          0x029C0000     ;erase sector 7 of flash SIMM
```

MPC555 internal flash:

```
[INIT]
...
WSPR    638    0x00000802     ;IMMR: InternalRegs to 0x00400000, Flash enabled
...

[TARGET]
CPUTYPE     MPC500      ;CPU type (MPC800 | MPC500)
CPUCLOCK    20000000    ;the CPU clock rate, used for flash timing calculation
...

[FLASH]
CHIPTYPE    MPC555      ;Select MPC555 internal CDR MoneT Flash
WORKSPACE   0x007FC000  ;use internal SRAM array B for workspace
FORMAT      SREC
FILE        D:\abatron\bdi360\ppc\pro\mpc555.sss  ;The file to program
ERASE       0x004000FF  ;Erase module A all sectors
ERASE       0x004400FC  ;Erase module B all sectors
```

MPC565 internal flash:

```
[INIT]
...
WSPR    638    0x00000802     ;IMMR: InternalRegs to 0x00400000, Flash enabled
...

[FLASH]
CHIPTYPE    MPC565      ;Select MPC565 internal CDR3 Flash
WORKSPACE   0x007F8000  ;use CALRAM A for workspace
FORMAT      SREC
FILE        D:\abatron\bdi360\ppc\pro\mpc565.sss  ;The file to program
ERASE       0x004000FF  ;Erase module A all sectors
ERASE       0x004800FF  ;Erase module B all sectors
```


3.2.5 Part [REGS]

In order to make it easier to access target registers via the Telnet interface, the BDI can read in a register definition file. In this file, the user defines a name for the register and how the BDI should access it (e.g. as memory mapped, memory mapped with offset, ...). The name of the register definition file and information for different registers type has to be defined in the configuration file.
The register name, type, address/offset/number and size are defined in a separate register definition file. This way, you can create one register definition file for the MPC860 that can be used for all possible positions of the internal memory map. You only have to change one entry in the configuration file.

An entry in the register definition file has the following syntax:

```
name    type    addr    size
```

name	The name of the register (max. 12 characters)
type	The register type GPR — General purpose register SPR — Special purpose register MM — Absolute direct memory mapped register DMM1...DMM4 — Relative direct memory mapped register IMM1...IMM4 — Indirect memory mapped register
addr	The address, offset or number of the register
size	The size (8, 16, 32) of the register

The following entries are supported in the [REGS] part of the configuration file:

FILE filename	The name of the register definition file. This name is used to access the file via TFTP. The file is loaded once during BDI startup. filename — the filename including the full path Example: FILE C:\bdi\regs\mpc8260.def
DMMn base	This defines the base address of direct memory mapped registers. This base address is added to the individual offset of the register. base — the base address Example: DMM1 0x01000
IMMn addr data	This defines the addresses of the memory mapped address and data registers of indirect memory mapped registers. The address of a IMMn register is first written to "addr" and then the register value is access using "data" as address. addr — the address of the Address register data — the address of the Data register Example: DMM1 0x02200000

Example for a register definition (MPC860):

Entry in the configuration file:

```
[REGS]
DMM1    0x02200000                      ;Internal Memory Map Base Address
FILE    E:\bdi\mpc860\reg860.def        ;The register definition file
```

The register definition file:

```
;name          type    addr          size
;-----------------------------------------
;
gpr0           GPR     0
sp             GPR     1
;
pc             SPR     26                  ; is SRR0
xer            SPR     1
lr             SPR     8
ctr            SPR     9
sprg0          SPR     272
sprg1          SPR     273
sprg2          SPR     274
sprg3          SPR     275
;
;
; DMM1 must be set to the internal memory map base address
;
siumcr         DMM1    0x0000        32
sypcr          DMM1    0x0004        32
;
mstat          DMM1    0x0178        16
padir          DMM1    0x0950        16
papar          DMM1    0x0952        16
paodr          DMM1    0x0954        16
padat          DMM1    0x0956        16
```

Now the defined registers can be accessed by name via the Telnet interface:

BDI> rd siumcr
BDI>rm padir 0xFF00

3.3 Debugging with GDB

Because the target agent runs within BDI, no debug support has to be linked to your application. There is also no need for any BDI specific changes in the application sources. Your application must be fully linked because no dynamic loading is supported.

3.3.1 Target setup

Target initialization may be done at two places. First with the BDI configuration file, second within the application. The setup in the configuration file must at least enable access to the target memory where the application will be loaded. Disable the watchdog and setting the CPU clock rate should also be done with the BDI configuration file. Application specific initializations like setting the timer rate are best located in the application startup sequence.

3.3.2 Connecting to the target

As soon as the target comes out of reset, BDI initializes it and loads your application code. If RUN is selected, the application is immediately started, otherwise only the target PC is set. BDI now waits for GDB request from the debugger running on the host.

After starting the debugger, it must be connected to the remote target. This can be done with the following command at the GDB prompt:

```
(gdb)target remote bdi3000:2001
```

bdi3000	This stands for an IP address. The HOST file must have an appropriate entry. You may also use an IP address in the form xxx.xxx.xxx.xxx
2001	This is the TCP port used to communicate with the BDI

If not already suspended, this stops the execution of application code and the target CPU changes to background debug mode.

Remember, every time the application is suspended, the target CPU is freezed. During this time no hardware interrupts will be processed.

Note: For convenience, the GDB detach command triggers a target reset sequence in the BDI.

```
(gdb)...
(gdb)detach
... Wait until BDI has resetet the target and reloaded the image
(gdb)target remote bdi3000:2001
```

Note:
After loading a program to the target you cannot use the GDB "*run*" command to start execution. You have to use the GDB "*continue*" command.

3.3.3 Breakpoint Handling

GDB versions before V5.0:
GDB inserts breakpoints by replacing code via simple memory read / write commands. There is no command like "Set Breakpoint" defined in the GDB remote protocol. When breakpoint mode HARD is selected, the BDI checks the memory write commands for such hidden "Set Breakpoint" actions. If such a write is detected, the write is not performed and the BDI sets an appropriate hardware breakpoint. The BDI assumes that this is a "Set Breakpoint" action when memory write length is 4 bytes and the pattern to write is 0x7D821008 (tw 12,r2,r2).

GDB version V5.x:
GDB version 5.x uses the Z-packet to set breakpoints (watchpoints). For software breakpoints, the BDI replaces code with 0x7D821008 (tw 12,r2,r2). When breakpoint mode HARD is selected, the BDI sets an appropriate hardware breakpoint.

User controlled hardware breakpoints:
The MPC8xx/5xx has a special watchpoint / breakpoint hardware integrated. Normally the BDI controls this hardware in response to Telnet commands (BI, BDx) or when breakpoint mode HARD is selected. Via the Telnet commands BI and BDx, you cannot access all the features of the breakpoint hardware. Therefore the BDI assumes that the user will control / setup this breakpoint hardware as soon as ICTRL, LCTRL1 or LCTRL2 is written to. This way the debugger or the user via Telnet has full access to all features of this watchpoint / breakpoint hardware. A hardware breakpoint set via BI or BDx gives control back to the BDI.

3.3.4 GDB monitor command

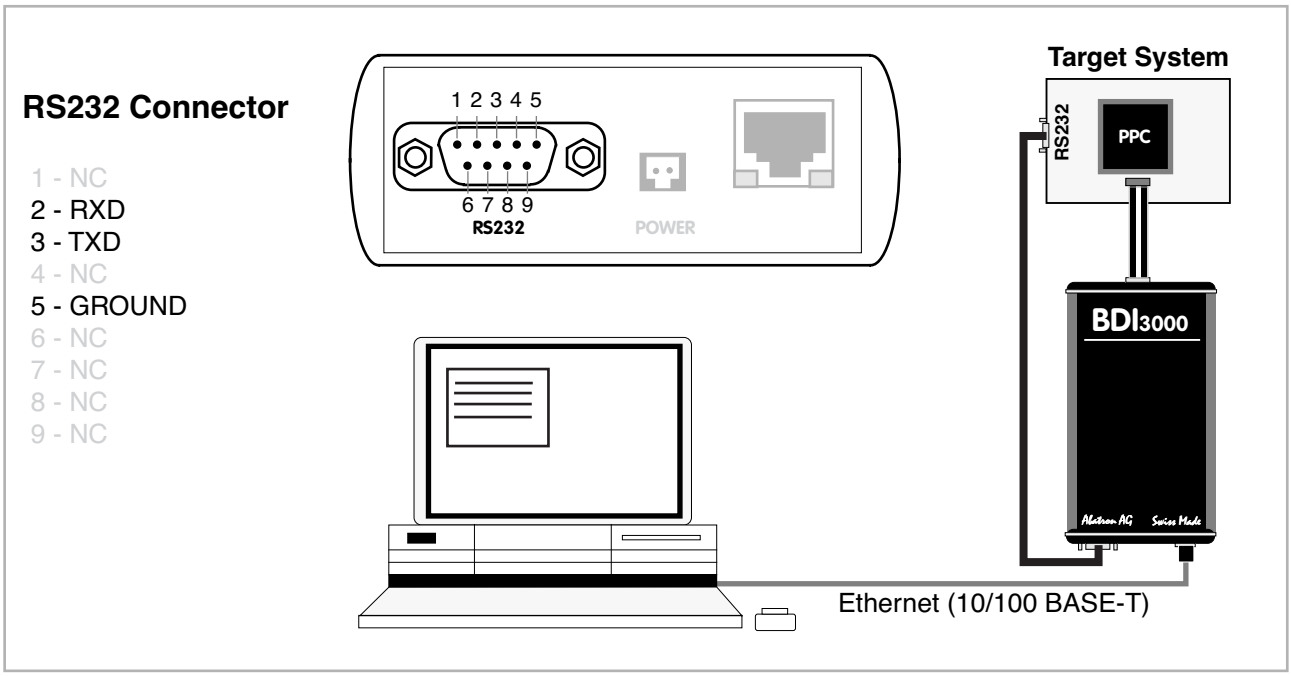
The BDI supports the GDB V5.x "monitor" command. Telnet commands are executed and the Telnet output is returned to GDB. This way you can for example switch the BDI breakpoint mode from within your GDB session.

```
(gdb) target remote bdi3000:2001
Remote debugging using bdi3000:2001
0x10b2 in start ()
(gdb) monitor break
Breakpoint mode is SOFT
(gdb) mon break hard

(gdb) mon break
Breakpoint mode is HARD
(gdb)
```


3.3.5 Target serial I/O via BDI

A RS232 port of the target can be connected to the RS232 port of the BDI3000. This way it is possible to access the target's serial I/O via a TCP/IP channel. For example, you can connect a Telnet session to the appropriate BDI3000 port. Connecting GDB to a GDB server (stub) running on the target should also be possible.



The configuration parameter "SIO" is used to enable this serial I/O routing.
The used framing parameters are 8 data, 1 stop and not parity.

```
[TARGET]
....
SIO     7       9600            ;Enable SIO via TCP port 7 at 9600 baud
```

Warning!!!
Once SIO is enabled, connecting with the setup tool to update the firmware will fail. In this case either disable SIO first or disconnect the BDI from the LAN while updating the firmware.

3.3.6 Embedded Linux MMU Support

The bdiGDB system supports Linux kernel debugging when MMU is on. The MMU configuration parameter enables this mode of operation. In this mode, all addresses received from GDB are assumed to be virtual. Before the BDI accesses memory, it translates this address into a physical one based on information found in the TLB's or kernel/user page table. Default address translation is used if address relocation is currently not active (MSR[DR] bit cleared) or the kernel page table pointer is still zero.

In order to search the page tables, the BDI needs to know the start address(es) of the first level page table(s). The configuration parameter PTBASE defines the physical address where the BDI looks for the address of an array with two addresses of first level page tables. The first one points normally to the kernel page table, the second one can point to the current user page table. As long as the base pointer or the first entry is zero, the BDI does only TLB and default translation.

If PTBASE is not defined in the configuration file, the BDI does only TLB and default translation.

Default translation maps addresses in the range KERNELBASE...(KERNELBASE + 0x0FFFFFFF) to 0x00000000...0x0FFFFFFF. The second page table is only searched if its address is not zero and there was no match in the first one.

The pointer structure is as follows:

```
PTBASE (physical address) ->
       PTE pointer pointer(virtual or physical address) ->
              PTE kernel pointer (virtual or physical address)
              PTE user pointer (virtual or physical address)
```

Newer versions of "arch/ppc/kernel/head_8xx.S" support the automatic update of the BDI page table information structure. Search "head_8xx.S" for "abatron" and you will find the BDI specific extensions.

Extract from the configuration file:

```
[INIT]
......
WM32    0x000000f0     0x00000000     ;invalidate page table base

[TARGET]
....
MMU            XLAT           ;translate effective to physical address
PTBASE      0x000000f0  ;here is the pointer to the page table pointers
```

Note:
The BDI can also handle L1 page tables where the entries are physical addresses instead of virtual ones as used in Linux 2.4.x. For example Linux 2.6.x and NetBSD uses physical L1 page table entries.

To debug the Linux kernel when MMU is enabled you may use the following load and startup sequence:

- Load the compressed linux image
- Set a hardware breakpoint with the Telnet at a point where MMU is enabled. This can be easily achieved with the following hardware range breakpoint
  BDI> BI 0xC0000000 0xC00FFFFF
- Start the code with GO at the Telnet
- The Linux kernel is decompressed and started
- The system should stop as soon as address translation is enabled (normally at start_here)
- Disable the hardware breakpoint with the Telnet command CI.
- Start GDB with vmlinux as parameter
- Attach to the target
- Now you should be able to debug the Linux kernel

There are of course other ways to begin kernel debugging. You may set a hardware breakpoint directly at a point of interest (e.g. start_kernel).

Note:
If PTBASE is used you should use a kernel that stores the virtual address of the first level page table(s) to the appropriate place in memory. Of course this can be done manually, but then, set a hardware breakpoint at "start_kernel" and use the Telnet to write the address of "swapper_pg_dir" to the appropriate place.

```
BDI>bi 0xc0061550            /* set breakpoint at start_kernel */
BDI>go
..                           /* target stops at start_kernel */
BDI>ci
BDI>mm 0xf0 0xc00000f8       /* Let PTBASE point to an array of two pointers*/
BDI>mm 0xf8 0xc0057000       /* write address of swapper_pg_dir to first pointer */
BDI>mm 0xfc 0x00000000       /* clear second (user) pointer */
```

Note:
The default value of DER is not suitable for Linux kernel debugging because almost all exceptions lead to debug mode entry. Use the configuration file to set an appropriate value:

```
WSPR    149     0x0082000F     ;DER: enable PRIE,TRE,LBRK,IBRK,EBRK,DPI
```


3.3.7 PPC Interrupt Handling

Almost all PPC interrupts causes an entry into debug mode. By default, the Debug Enable Register (DER) is set as follows:

Debug Enable Register

Bit	Mnemonic	State	Describtion
0	-		
1	RSTE	enabled	Reset Interrupt
2	CHSTPE	enabled	Check Stop
3	MCIE	enabled	Maschine Check Interrupt
4-5	-		
6	EXTIE		External Interrupts
7	ALIE	enabled	Alignment Interrupt
8	PRIE	enabled	Program Interrupt
9	FPUVIE	enabled	Floating-Point Unavailable Interrupt
10	DECIE		Decrementer Interrupt
11-12	-		
13	SYSIE	enabled	System Call Interrupt
14	TRE	enabled	Trace Interrupt
15-16	-		
17	SEIE	enabled	Software Emulation Interrupt
18	ITLBMSE		Implementation Specific Instructuction TLB Miss
19	ITLBERE		Implementation Specific Instructuction TLB Error
20	DTLBMSE		Implementation Specific Data TLB Miss
21	DTLBERE		Implementation Specific Data TLB Error
22-27	-		
28	LBRKE	enabled	Load/Store Breakpoint Interrupt
29	IBRKE	enabled	Instruction Breakpoint Interrupt
30	EBRKE	enabled	External Breakpoint Interrupt
31	DPIE	enabled	Developement Port Nonmaskable Request

If this is not appropriate for the application the default initialisation may be change with an entry in the configuration file.

```
WSPR    149     0xFFE7400F     ;DER: set debug enable register
```


3.4 Telnet Interface

A Telnet server is integrated within the BDI. The Telnet channel is used by the BDI to output error messages and other information. Also some basic debug commands can be executed.

Telnet Debug features:

- Display and modify memory locations
- Display and modify general and special purpose registers
- Single step a code sequence
- Set hardware breakpoints (for code and data accesses)
- Load a code file from any host
- Start / Stop program execution
- Programming and Erasing Flash memory

During debugging with GDB, the Telnet is mainly used to reboot the target (generate a hardware reset and reload the application code). It may be also useful during the first installation of the bdiGDB system or in case of special debug needs (e.g. setting breakpoints on variable access).

Multiple commands separated by a semicolon can be entered on one line.

Notes:
The Telnet command RESET does only reset the target system. The configuration file is not loaded again. If the configuration file has changed, use the Telnet command BOOT to reload it.

The DUMP command uses TFTP to write a binary image to a host file. Writing via TFTP on a Linux/ Unix system is only possible if the file already exists and has public write access. Use "man tftpd" to get more information about the TFTP server on your host.

Following a list of the available Telnet commands:

```
"PHYS  <address>              converts an effective to a physical address",
"MD    [<address>] [<count>]  display target memory as word (32bit)",
"MDH   [<address>] [<count>]  display target memory as half word (16bit)",
"MDB   [<address>] [<count>]  display target memory as byte (8bit)",
"DUMP  <addr> <size> [<file>] dump target memory to a file",
"MM    <addr> <value> [<cnt>] modify word(s) (32bit) in target memory",
"MMH   <addr> <value> [<cnt>] modify half word(s) (16bit) in target memory",
"MMB   <addr> <value> [<cnt>] modify byte(s) (8bit) in target memory",
"MT    <address>   <count>    single word (32bit) memory test",
"MTH   <address>   <count>    single half word (16bit) memory test",
"MTB   <address>   <count>    single byte (8bit) memory test",
"MC    [<address>] [<count>]  calculates a checksum over a memory range",
"MV                           verifies the last calculated checksum",
"RD    [<name>]               display general purpose or user defined register",
"RDUMP [<file>]               dump all user defined register to a file",
"RDS   <number>               display special purpose register",
"RM    {<nbr>=<name>} <value> modify general purpose or user defined register",
"RMS   <number>   <value>     modify special purpose register",
"UPMS  <MCR-addr> <MDR-addr>  set address of register MCR and MDR",
"UPMA                         display UPMA setup",
"UPMB                         display UPMB setup",
"DTLB <from> [<to>]           display data TLB entry",
"ITLB <from> [<to>]           display inst TLB entry",
"DTAG  <from> [<to>]          display data cache tags",
"CBB                          display copyback buffer",
"RESET [HALT | RUN [time]]    reset the target system, change startup mode",
"BREAK [SOFT | HARD]          display or set current breakpoint mode",
"GO    [<pc>]                 set PC and start target system",
"TI    [<pc>]                 trace on instuction (single step)",
"TC    [<pc>]                 trace on change of flow",
"HALT                         force target to enter debug mode",
"BI  <from> [<to>] [<count>]  set instruction hardware breakpoint",
"CI  [<id>]                   clear instruction hardware breakpoint(s)",
"BD  [R|W] <addr> [<count>] [<data>] set data breakpoint (32bit access)",
"BDH [R|W] <addr> [<count>] [<data>] set data breakpoint (16bit access)",
"BDB [R|W] <addr> [<count>] [<data>] set data breakpoint ( 8bit access)",
"BDR [R|W] <from> <to> [<count>]     set data breakpoint on a range",
"CD [<id>]                    clear data breakpoint(s)",
"INFO                         display information about the current state",
"LOAD   [<offset>] [<file> [<format>]] load program file to target memory",
"VERIFY [<offset>] [<file> [<format>]] verify a program file to target memory",
"PROG   [<offset>] [<file> [<format>]] program flash memory",
"                                      <format> : SREC or BIN or AOUT or ELF",
"ERASE  [<address> [<mode>]]  erase a flash memory sector, chip or block",
"                  <mode>  : CHIP, BLOCK or SECTOR (default is sector)",
"ERASE  <addr> <step> <count> erase multiple flash sectors",
"UNLOCK [<addr> [<delay>]]    unlock a flash sector",
"UNLOCK <addr> <step> <count> unlock multiple flash sectors",
"FLASH  <type> <size> <bus>   change flash configuration",
"DELAY  <ms>                  delay for a number of milliseconds",
"HOST   <ip>                  change IP address of program file host",
"PROMPT <string>              defines a new prompt string",
"CONFIG                       display or update BDI configuration",
"CONFIG <file> [<hostIP> [<bdiIP> [<gateway> [<mask>]]]]",
"HELP                         display command list",
"BOOT  [loader]               reboot the BDI and reload the configuration",
"QUIT                         terminate the Telnet session"
```


4 Specifications

Operating Voltage Limiting	5 VDC ± 0.25 V
Power Supply Current	typ. 500 mA max. 1000 mA
RS232 Interface: Baud Rates	9’600,19’200, 38’400, 57’600,115’200
Data Bits	8
Parity Bits	none
Stop Bits	1
Network Interface	10/100 BASE-T
BDM/JTAG clock	up to 32 MHz
Supported target voltage	1.2 – 5.0 V
Operating Temperature	+ 5 °C ... +60 °C
Storage Temperature	-20 °C ... +65 °C
Relative Humidity (noncondensing)	<90 %rF
Size	160 x 85 x 35 mm
Weight (without cables)	280 g
Host Cable length (RS232)	2.5 m
Electromagnetic Compatibility	CE compliant
Restriction of Hazardous Substances	RoHS 2002/95/EC compliant

Specifications subject to change without notice

5 Environmental notice

Disposal of the equipment must be carried out at a designated disposal site.

6 Declaration of Conformity (CE)

CE

DECLARATION OF CONFORMITY

This declaration is valid for following product:

Type of device: BDM/JTAG Interface
Product name: BDI3000

The signing authorities state, that the above mentioned equipment meets the requirements for emission and immunity according to

EMC Directive 89/336/EEC

The evaluation procedure of conformity was assured according to the following standards:

IEC 61000-6-2: 1999, mod. EN61000-6-2: 2001
IEC 61000-6-3: 1996, mod. EN61000-6-2: 2001

This declaration of conformity is based on the test report no. E1087-05-7a of Quinel, Zug, Swiss Testing Service, accreditation no. STS 037

Manufacturer:

ABATRON AG
Lettenstrasse 9
CH-6343 Rotkreuz

Authority:

Max Vock
Marketing Director

Ruedi Dummermuth
Technical Director

Rotkreuz, 7/18/2007

7 Warranty

ABATRON Switzerland warrants the physical CD, cable and BDI3000 to be free of defects in materials and workmanship for a period of 3 years following the date of purchase when used under normal conditions.

In the event of notification within the warranty period of defects in material or workmanship, ABATRON will replace defective CD, cable or BDI3000. The remedy for breach of this warranty shall be limited to replacement and shall not encompass any other damages, including but not limited loss of profit, special, incidental, consequential, or other similar claims.
ABATRON Switzerland specifically disclaims all other warranties - expressed or implied, including but not limited to implied warranties of merchantability and fitness for particular purposes - with respect to defects in the CD, cable and BDI3000, and the program license granted herein, including without limitation the operation of the program with respect to any particular application, use, or purposes. In no event shall ABATRON be liable for any loss of profit or any other commercial damage, including but not limited to special, incidental, consequential, or other damages.

Failure in handling which leads to defects are not covered under this warranty. The warranty is void under any self-made repair operation.

Appendices

A Troubleshooting

Problem
The firmware can not be loaded.

Possible reasons

- The BDI is not correctly connected with the Host (see chapter 2).
- A wrong communication port is selected (Com 1...Com 4).
- The BDI is not powered up

Problem
No working with the target system (loading firmware is okay).

Possible reasons

- Wrong pin assignment (BDM/JTAG connector) of the target system (see chapter 2).
- Target system initialization is not correctly –> enter an appropriate target initialization list.
- An incorrect IP address was entered (BDI3000 configuration)
- BDM/JTAG signals from the target system are not correctly (short-circuit, break, ...).
- The target system is damaged.

Problem
Network processes do not function (loading the firmware was successful)

Possible reasons

- The BDI3000 is not connected or not correctly connected to the network (LAN cable or media converter)
- An incorrect IP address was entered (BDI3000 configuration)

B Maintenance

The BDI needs no special maintenance. Clean the housing with a mild detergent only. Solvents such as gasoline may damage it.

C Trademarks

All trademarks are property of their respective holders.

