

bedi Wind

BDM interface for Tornado™

PowerPC MPC8xx/MPC5xx



User Manual

Manual Version 1.38 for BDI2000



©1997-2007 by Abatron AG

1 Introduction	4
1.1 BDI2000.....	4
1.2 BDI Configuration	5
2 Installation	6
2.1 Connecting the BDI2000 to Target.....	6
2.1.1 Changing Target Processor Type	8
2.2 Connecting the BDI2000 to Power Supply.....	9
2.2.1 External Power Supply	9
2.2.2 Power Supply from Target System	10
2.3 Status LED «MODE»	11
2.4 Connecting the BDI2000 to Host.....	12
2.4.1 Serial line communication	12
2.4.2 Ethernet communication	13
2.5 Installation of the Configuration Software	14
2.6 Initial configuration of the bdiWind system	15
2.7 Testing the BDI2000 to host connection	15
2.8 TFTP server for Windows NT.....	16
3 Using bdiWind	17
3.1 Principle of operation	17
3.1.1 Gateway mode	17
3.1.2 Agent mode.....	18
3.2 Configuration File	20
3.2.1 Part [INIT].....	21
3.2.2 Part [TARGET]	22
3.2.3 Part [HOST].....	25
3.2.4 Part [FLASH]	26
3.2.5 Part [REGS]	32
3.3 BDM GATEWAY mode.....	34
3.3.1 Create a new directory for your hardware	34
3.3.2 Tornado 1.01 and Tornado II	35
3.3.3 Tornado II	36
3.3.4 Tornado 1.01	37
3.3.5 bdiWind UDP-lite packet driver	38
3.3.5.1. Transferring the communication base address.....	38
3.3.5.2. Polling for requests.....	39
3.3.5.3. Automatic change to Agent mode	39
3.3.6 PPC Interrupt Handling	40
3.4 AGENT mode	41
3.4.1 Target setup.....	41
3.4.2 Configure VxWorks	41
3.4.3 Target Server Setup	42
3.4.4 Debugging with GDB.....	42
3.4.5 PPC Interrupt Handling	43
3.5 Serial GATEWAY mode	44
3.5.1 Target connection	44
3.5.2 VxWorks configuration	45
3.6 Target serial I/O via BDI.....	46
3.7 Telnet Interface	47
4 Specifications.....	49
5 Environmental notice	50

6 Declaration of Conformity (CE)..... 50

7 Warranty 51

Appendices

A BDI2000 Setup/Update 52

B Troubleshooting 54

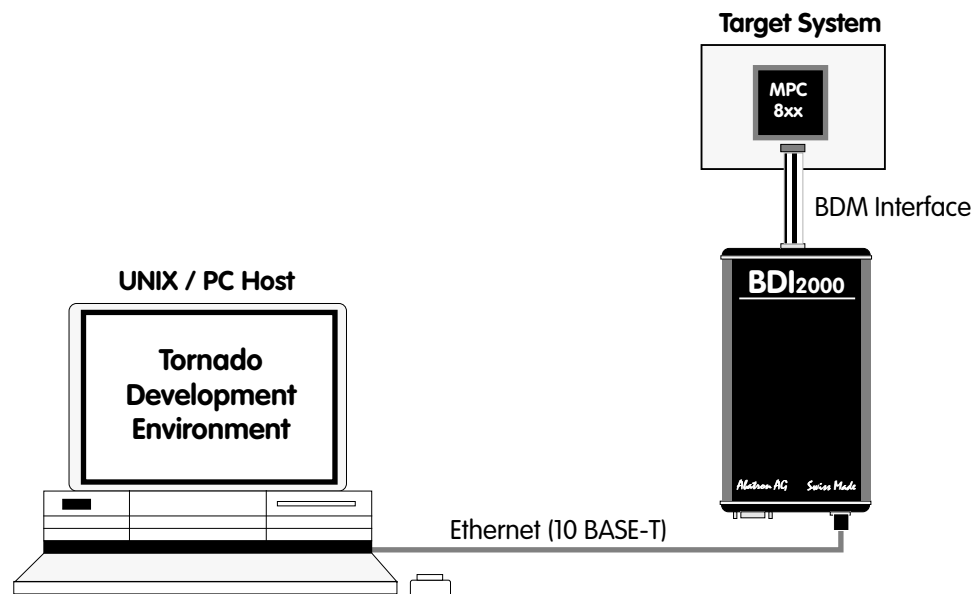
C Maintenance 55

D Trademarks 57

1 Introduction

bdiWind enhances Tornado, the latest generation of development and execution environment for embedded and real-time applications, with Background Debug Mode (BDM) debugging for MPC8xx/MPC5xx based targets. With bdiWind there is no need for Boot ROMs, because the VxWorks core is automatically loaded into the target RAM after every target restart. With the builtin Ethernet interface you get a very fast download speed of up to 150Kbytes/sec. This combination loads a typical VxWorks core in a few seconds. 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. In contrast to ICE debugging, bdiWind supports task mode and system mode debugging.

The following figure shows how the BDI2000 interface is connected between the host and the target:



1.1 BDI2000

The BDI2000 is the main part of the bdiWind system. This small box implements the interface between the BDM pins of the target CPU and a 10Base-T Ethernet connector. The firmware and the programmable logic of the BDI2000 can be updated by the user with a simple Windows based configuration program. The BDI2000 supports 1.8 – 5.0 Volts target systems (3.0 – 5.0 Volts target systems with Rev. A/B).

1.2 BDI Configuration

As an initial setup, the IP address of the BDI2000, the IP address of the host with the configuration file and the name of the configuration file is stored within the flash of the BDI2000. Every time the BDI2000 is powered on, it reads the configuration file via TFTP.

Following an example of a typical configuration file:

```

; bdiWind 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   0xFFFFFFFF88     ;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   0x8FFFE024      ;UPMA single read
WUPM    0x00000001   0x0FFFE004
WUPM    0x00000002   0x0CFE0004
WUPM    0x00000003   0x00FE0004
          .....
WUPM    0x0000003C   0x33FFC007      ;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
MEMBASE     0        ;<AGENT> base of target memory
MEMSIZE     0x400000 ;<AGENT> size of target memory
POOLBASE    0x300000 ;<AGENT> base of host controlled target memory
POOLSIZE    0x100000 ;<AGENT> size of host controlled target memory

[HOST]
IP          151.120.25.100
FILE        F:\TornadoPPC\target\config\860agent\vxworks
FORMAT      ELF
LOAD        AUTO     ;<AGENT> load VxWorks code MANUAL or AUTO after reset
DEBUGPORT   0x4321
    
```

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

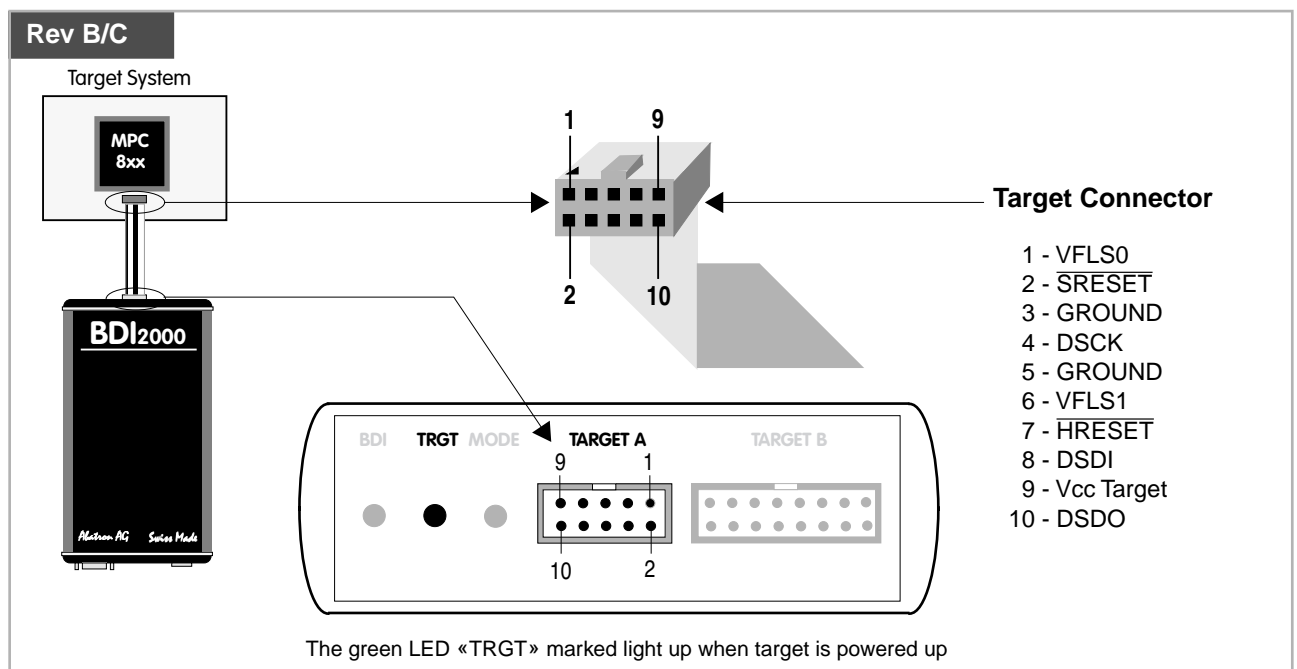
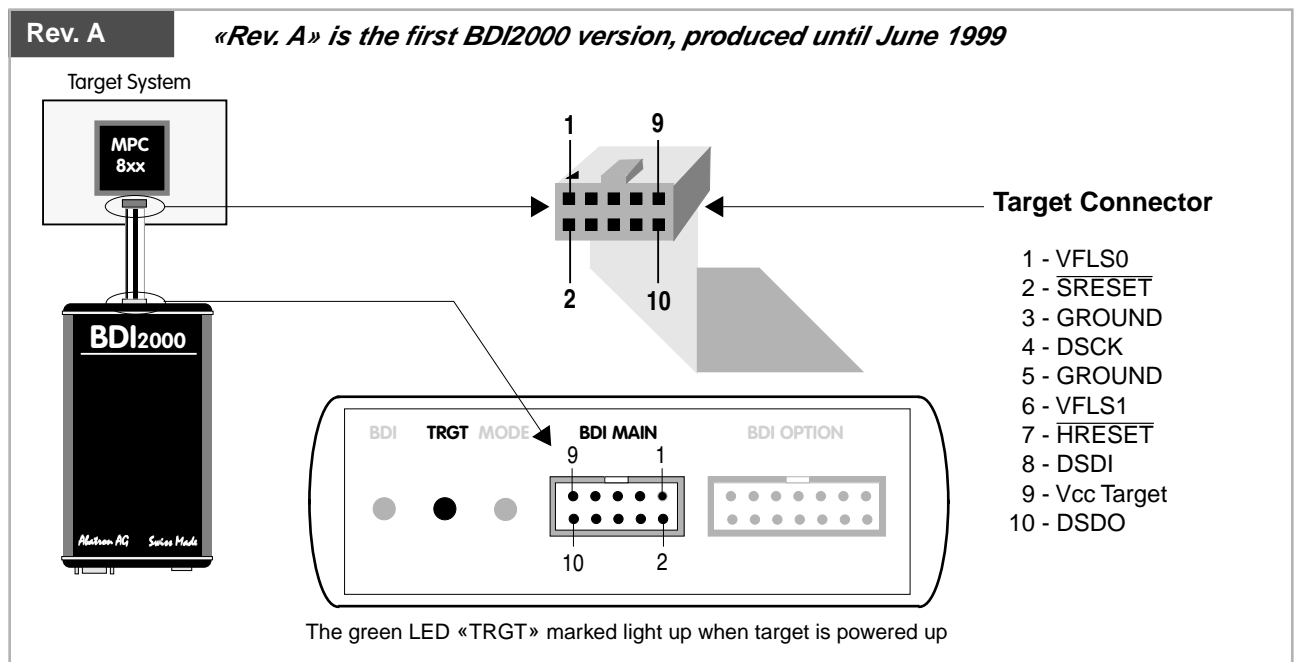
2 Installation

2.1 Connecting the BDI2000 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").



For BDI MAIN / TARGET A connector signals see table on next page.

BDI MAIN / 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. This signal is in fact redundant regarding the MPC8xx debug port controller since there is a soft-reset signal integrated within the debug port protocol. However, the local debug port controller uses this signal for compatibility with MPC5xx existing boards and s/w.
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. This signal is made redundant with the MPC8xx debug port controller since there is a hard-reset command integrated within the debug port protocol.
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	1.8 – 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. 3.0 – 5.0V with Rev. A/B : This input to the BDI2000 is used to detect if the target is powered up. If there is a current limiting resistor between this pin and the target Vdd, it should be 100 Ohm or less.
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 (frozen):

```

BOOL PPC_TargetFreezed(void) {
    if ((VFLS0 != 1) | (VFLS1 != 1)) return FALSE;
    read debug port status;
    if (status == freezed) return TRUE;
    else return FALSE;
}
    
```

2.1.1 Changing Target Processor Type

Before you can use the BDI2000 with an other target processor type (e.g. CPU32 <--> PPC), a new setup has to be done (see Appendix A). During this process the target cable must be disconnected from the target system. The BDI2000 needs to be supplied with 5 Volts via the BDI OPTION connector (Rev. A) or via the POWER connector (Rev. B/C). For more information see chapter 2.2.1 «External Power Supply».



To avoid data line conflicts, the BDI2000 must be disconnected from the target system while programming the logic for an other target CPU.

2.2 Connecting the BDI2000 to Power Supply

2.2.1 External Power Supply

The BDI2000 needs to be supplied with 5 Volts (max. 1A) via the BDI OPTION connector (Rev. A) or via POWER connector (Rev. B/C). The available power supply from Abatron (option) or the enclosed power cable can be directly connected. In order to ensure reliable operation of the BDI2000, keep the power supply cable as short as possible.



For error-free operation, the power supply to the BDI2000 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.**

Rev. A

BDI OPTION Connector

- 1 - NOT USED
- 2 - GROUND
- 3 - NOT USED
- 4 - GROUND
- 5 - NOT USED
- 6 - GROUND
- 7 - NOT USED
- 8 - GROUND
- 9 - NOT USED
- 10 - GROUND
- 11 - NOT USED
- 12 - Vcc (+5V)
- 13 - Vcc Target (+5V)
- 14 - Vcc (+5V)

The green LED «BDI» marked light up when 5V power is connected to the BDI2000

Rev. B/C

POWER Connector

- 1 - Vcc (+5V)
- 2 - VccTGT
- 3 - GROUND
- 4 - NOT USED

The green LED «BDI» marked light up when 5V power is connected to the BDI2000

Please switch on the system in the following sequence:

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

2.2.2 Power Supply from Target System

The BDI2000 needs to be supplied with 5 Volts (max. 1A) via BDI MAIN target connector (Rev. A) or via TARGET A connector (Rev. B/C). This mode can only be used when the target system runs with 5V and the pin «Vcc Target» is able to deliver a current up to 1A@5V. For pin description and layout see chapter 2.1 «Connecting the BDI2000 to Target». Insert the enclosed Jumper as shown in figure below. **Please ensure that the jumper is inserted correctly.**



For error-free operation, the power supply to the BDI2000 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.**

Rev. A

The green LEDs «BDI» and «TRGT» marked light up when target is powered up and the jumper is inserted correctly

BDI OPTION Connector

- 1 - NOT USED
- 2 - GROUND
- 3 - NOT USED
- 4 - GROUND
- 5 - NOT USED
- 6 - GROUND
- 7 - NOT USED
- 8 - GROUND
- 9 - NOT USED
- 10 - GROUND
- 11 - NOT USED
- 12 - Vcc (+5V)
- 13 - Vcc Target (+5V)
- 14 - Vcc BDI2000 (+5V)

Rev. B/C

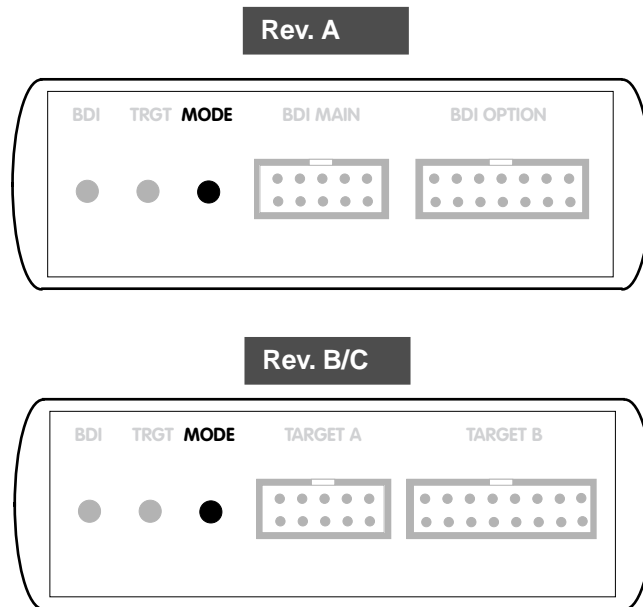
The green LEDs «BDI» and «TRGT» marked light up when target is powered up and the jumper is inserted correctly

POWER Connector

- 1 - Vcc BDI2000 (+5V)
- 2 - Vcc Target (+5V)
- 3 - GROUND
- 4 - NOT USED

2.3 Status LED «MODE»

The built in LED indicates the following BDI states:



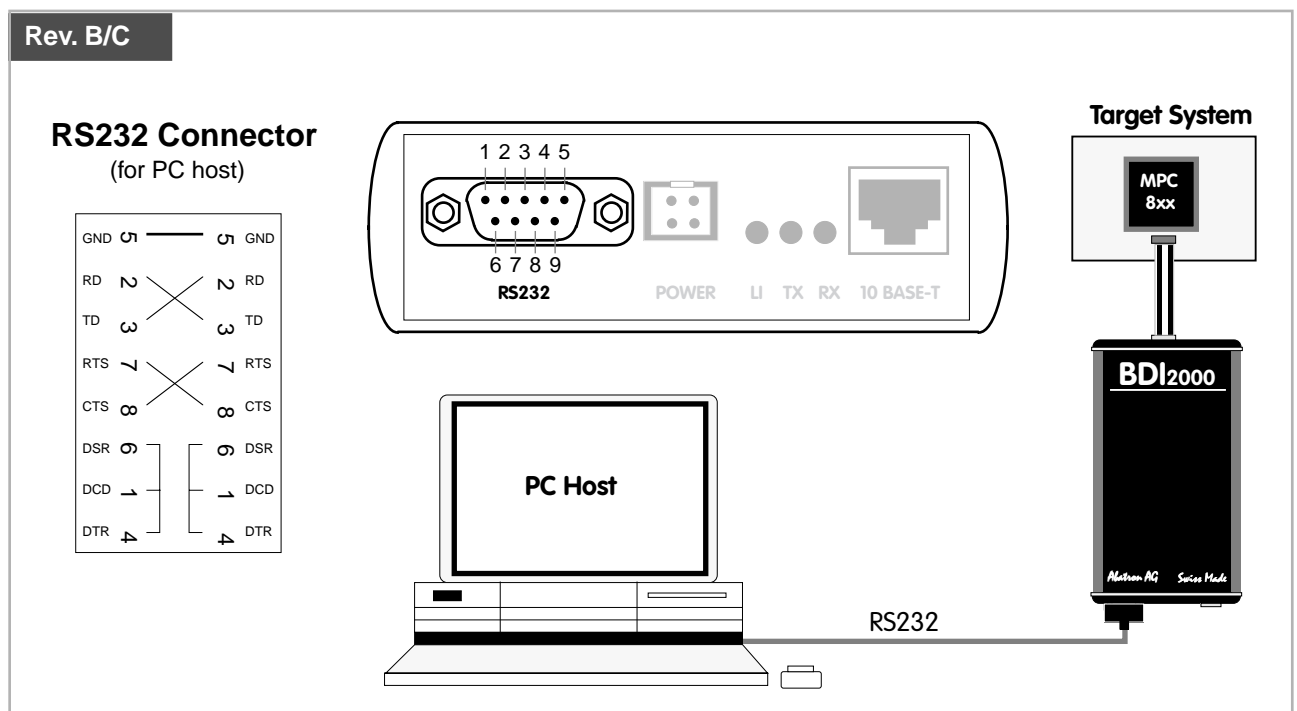
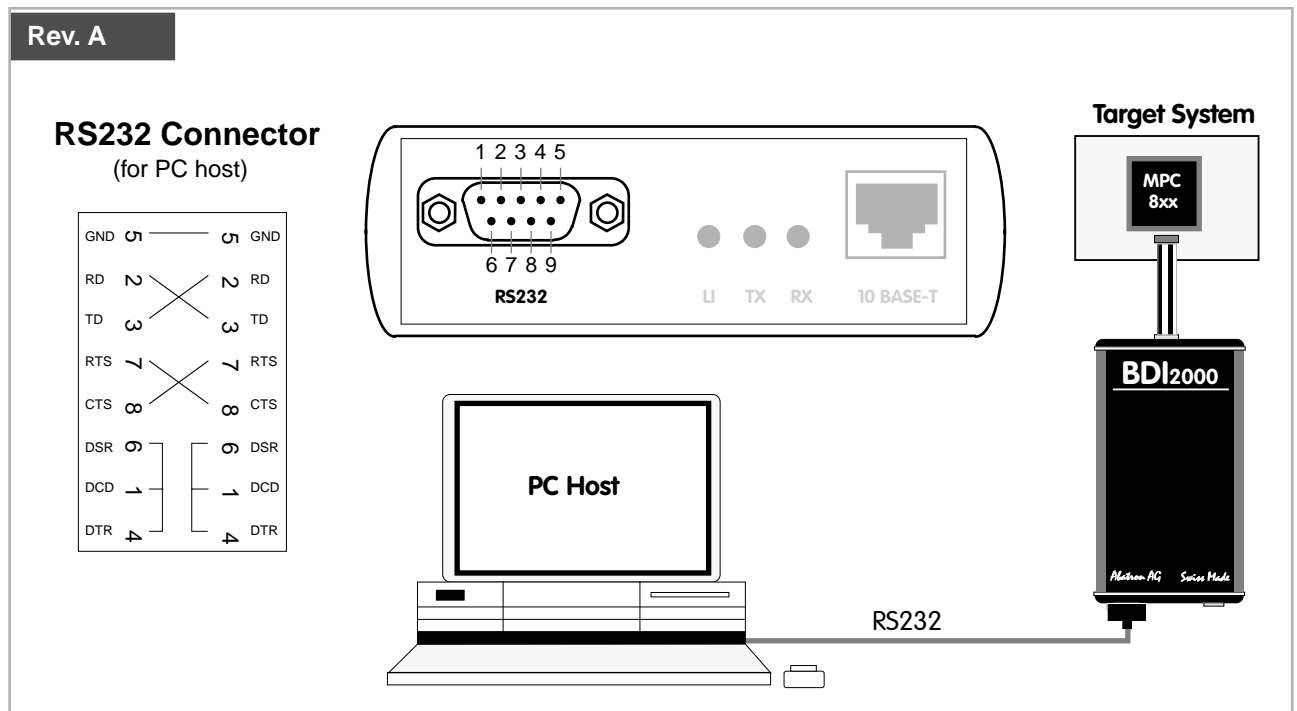
MODE LED	BDI STATES
OFF	The BDI is ready for use, the firmware is already loaded.
ON	The power supply for the BDI2000 is < 4.75VDC.
BLINK	The BDI «loader mode» is active (an invalid firmware is loaded or loading firmware is active).

2.4 Connecting the BDI2000 to Host

2.4.1 Serial line communication

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

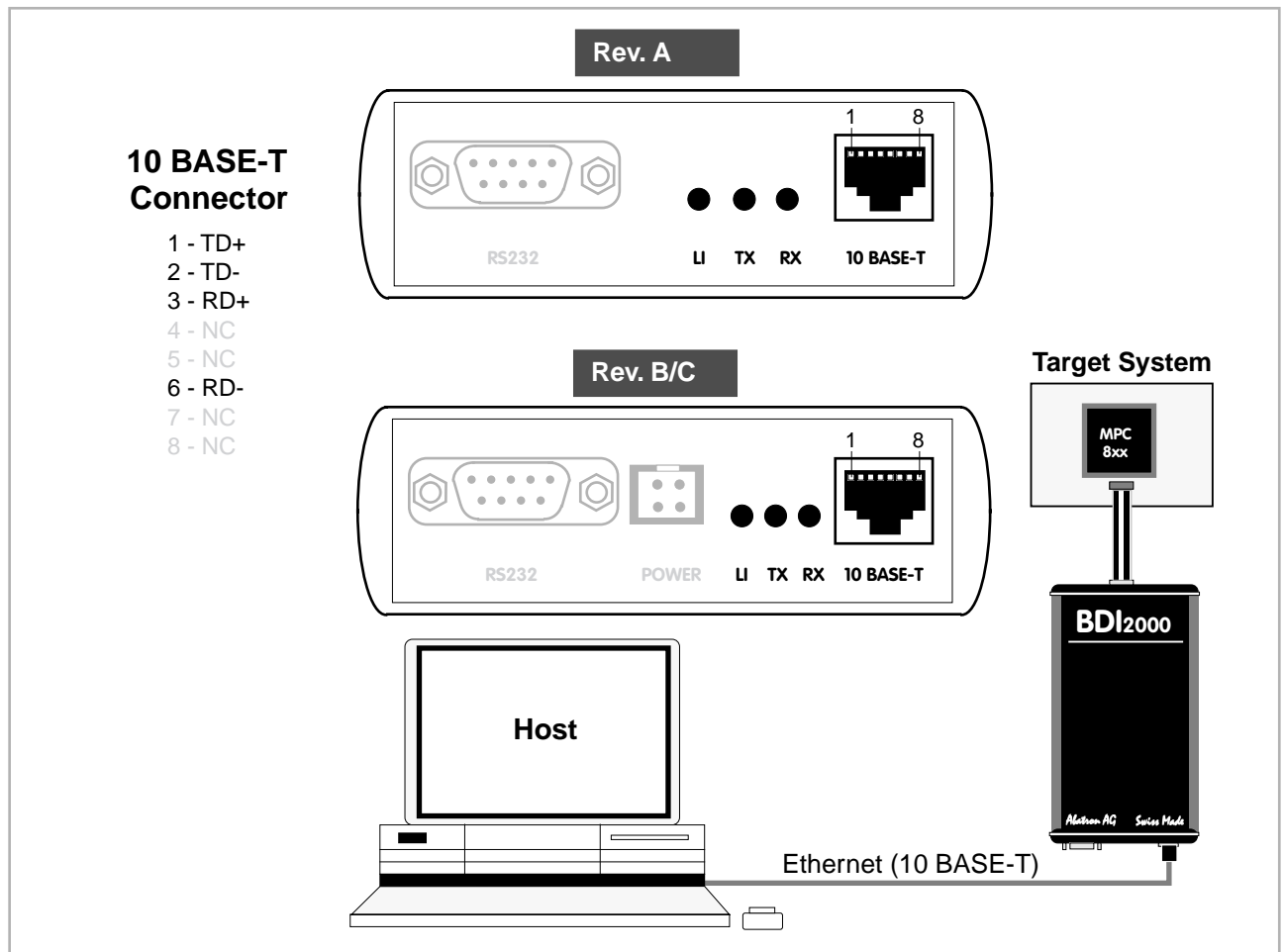
The host is connected to the BDI through the serial interface (COM1...COM4). The communication cable (included) between BDI and Host is a serial cable. There is the same connector pinout for the BDI and for the Host side (Refer to Figure below).



2.4.2 Ethernet communication

The BDI2000 has a built-in 10 BASE-T Ethernet interface (see figure below). Connect an UTP (Unshielded Twisted Pair) cable to the BDI2000. For thin Ethernet coaxial networks you can connect a commercially available media converter (BNC-->10 BASE-T) between your network and the BDI2000. Contact your network administrator if you have questions about the network.

Ethernet communication is used when communicating with the Tornado Development Environment.



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

LED	Name	Description
LI	Link	When this LED light is ON, data link is successful between the UTP port of the BDI2000 and the hub to which it is connected.
TX	Transmit	When this LED light BLINKS, data is being transmitted through the UTP port of the BDI2000
RX	Receive	When this LED light BLINKS, data is being received through the UTP port of the BDI2000

2.5 Installation of the Configuration Software

On the enclosed diskette you will find the BDI configuration software and the firmware required for the BDI2000. For Windows users there is also a TFTP server included.

For the initial configuration or to update the firmware, a PC running at least Windows 3.1 is required.

The following files are on the diskette.

b20ppcwr.exe	Configuration program (16bit Windows application)
b20ppcwr.hlp	Windows help file for the configuration program
b20ppcwr.xxx	Firmware for the BDI2000
ppcjwr20.xxx	JEDEC file for the BDI2000 (Rev. A/B) logic device
ppcjwr21.xxx	JEDEC file for the BDI2000 (Rev. C) logic device
tftpsrv.exe	TFTP server for WindowsNT/ Windows95 (WIN32 console application)
ads860.cfg	A sample configuration file for the ADS860 evaluation board
bdipkt.h	The header file for the BDM UDP-lite packet driver
bdipkt.c	The implementation file for the BDM UDP-lite packet driver
bdiSlip.h	The header file for the example SLIP UDP-lite packet driver
bdiSlip.c	The implementation file for the example SLIP UDP-lite packet driver
bdiHdlc.h	The header file for the example HDLC UDP-lite packet driver
bdiHdlc.c	The implementation file for the example HDLC UDP-lite packet driver
tornado.add	A text file with information how to extend the Tornado environment Used for updating some existing source files via copy/paste
wdbBdi.c	Tornado II : Main module for the BDI WDB communication
wdbFslip.c	Tornado II : Main module for the fast SLIP WDB communication
wdbHdlc.c	Tornado II : Main module for the HDLC WDB communication
wdbBdi.cdf	Tornado II : Component descriptions for the BDI WDB communication
*.def	Register definition files

Example of an installation process:

- Create a new directory on your hard disk, for example E:\bdi\ppc
- Copy the entire contents of the enclosed diskette into this directory
- You may create a new Windows Program Manager entry for the bdiWind configuration program.

2.6 Initial configuration of the bdiWind system

Before you can use the bdiWind system, an initial setup has to be done (see Appendix A). During this setup you define the following items and stores them in the flash memory of the BDI2000.

BDI2000 IP address	The BDI2000 is assigned an individual IP address. Ask your network administrator for a free one.
HOST IP address	The IP address of the host with the target configuration file has to be known by the BDI2000.
Configuration file name	The name (including the path) of the file with the target configuration. The string entered is used as the filename when accessing the configuration file via TFTP. Use the naming convention of the host which holds the configuration file.

For more information about using the bdiWind configuration program consult the online help.
Remark: Don't forget to press <Transmit> after you entered the configuration values.

Activating BOOTP:

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

2.7 Testing the BDI2000 to host connection

After the initial setup is done, you can test the communication between the host and the BDI2000. There is no need for a target configuration file and no TFTP server is needed on the host.

- If necessary, disconnect the BDI2000 system from the Windows PC used for the initial configuration.
- If not already done, connect the bdiWind system to the network.
- Power-up the BDI2000.
- Start a Telnet client on the host and connect to the BDI2000 (the IP address you entered during initial configuration).
- If everything is okay, a sign on message like «BDI Debugger for Embedded PowerPC» should be displayed in the Telnet window.

2.8 TFTP server for Windows NT

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

Command line syntax: `ftpsrv [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 bdiWind system needs only read access to the configuration and VxWorks 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.

<code>tftpsrv p</code>	Starts the TFTP server and enables protocol output
<code>tftpsrv p w</code>	Starts the TFTP server, enables protocol output and write accesses are allowed.
<code>tftpsrv dC:\tftp\</code>	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\mpc860.cfg" accesses "C:\tftp\bdi\mpc860.cfg"

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

3 Using bdiWind

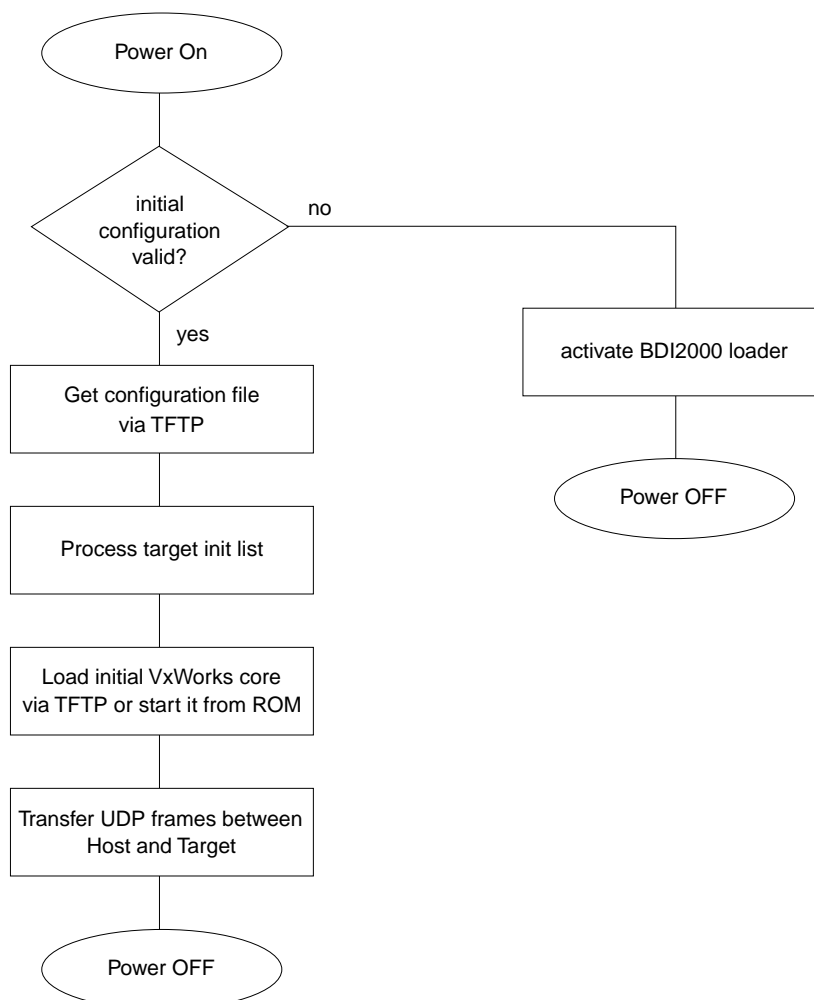
3.1 Principle of operation

To understand the principle of operation, we assume that you are familiar with the Tornado development environment. Elements of this environment like Target Server, Target Agent or VxWorks core will not be explained in this manual. See the appropriate Tornado manuals.

3.1.1 Gateway mode

For ease of understanding, you can look at the bdiWind system as an alternative communication channel between the Target Server running on the host and the Target Agent running on the target. This mode supports anything the original WindRiver target agent supports (task specific breakpoints, dynamically loading of new modules, and so on).

Whenever the bdiWind system is started (target is powered on) the following sequence starts:

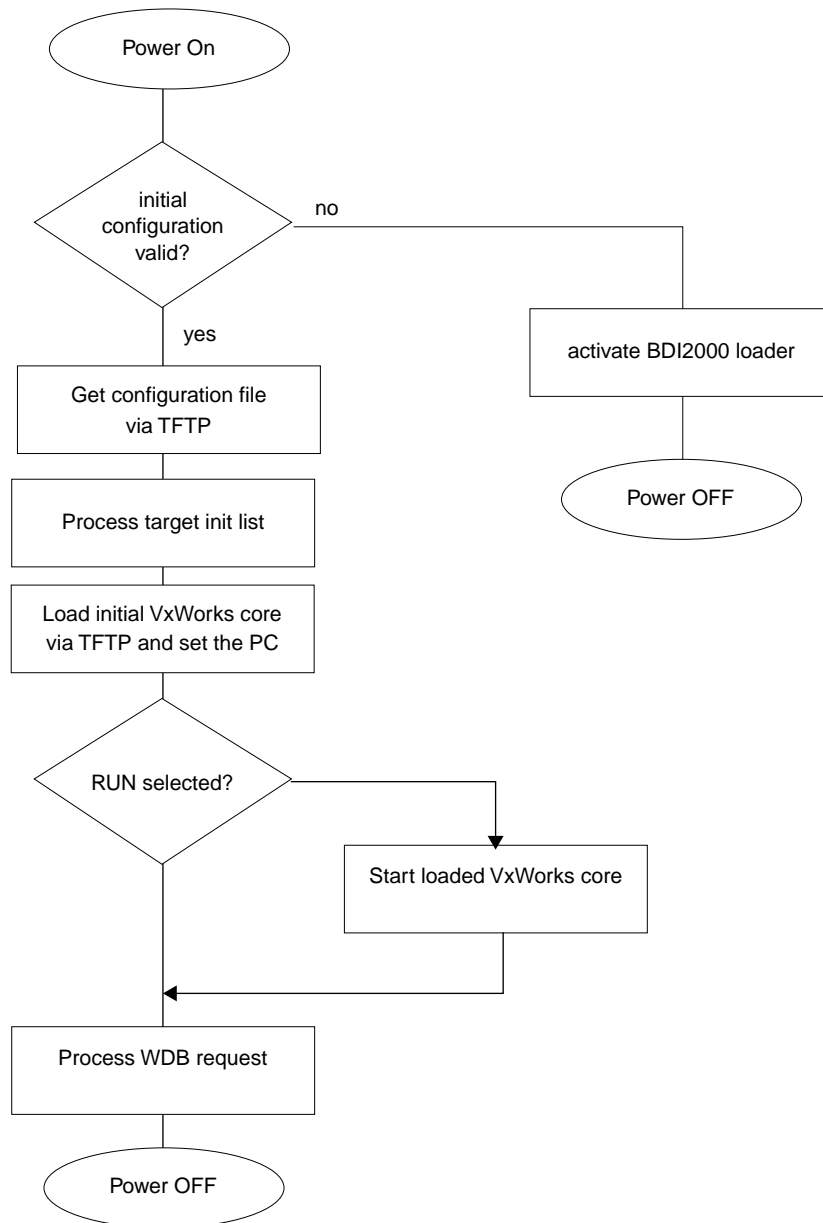


In BDM Gateway mode, the UDP frames are transferred via the target BDM interface. In Serial Gateway mode, the UDP frames are transferred via a serial connection (UART or HDLC) between the BDI2000 and the target (e.g. via a SMC channel).

3.1.2 Agent mode

In this mode, the target agent runs within the BDI. There is no need for any debug software on the target system. After loading the VxWorks core (or even any other fully linked executable) debugging can begin at the very first statement (e.g. syslnit). This mode is useful to get an initial VxWorks code running or in the final state of a development, when no debug support is linked into the application. This mode also supports the PowerPC built in breakpoint logic. It's possible to debug ROM resident applications.

Whenever the bdiWind system is started (target is powered on) 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 frozen.

Restrictions when using Agent mode:

This mode supports only system level debugging. Only fully linked systems can be debugged. Loading of modules is not supported because an agent not running within the target cannot create VxWorks tasks. In general only the system context is supported.

Following a list of unsupported WDB requests and WDB requests with restrictions:

bkendModeSet	Only WDB_MODE_EXTERN is accepted
bkendMemFill	Supported, but large memory blocks may cause a target agent time-out
bkendMemMove	Supported, but large memory blocks may cause a target agent time-out
bkendMemChecksum	Supported, but large memory blocks may cause a target agent time-out
bkendMemProtect	Returns OKAY but does nothing
bkendCacheTextUpdate	Returns OKAY but does nothing
bkendMemScan	Supported, but large memory blocks may cause a target agent time-out
bkendVIOWrite	not supported
bkendFuncCall	not supported
bkendDirectCall	not supported

For large VxWorks core, the target server time-out value may be increased.

Tornado II:

Hardware breakpoints are also supported with Tornado II. There are 4 instruction and 2 data hardware breakpoints available. Hardware breakpoints can be set with the Tornado Shell (enter help).

The following example sets a hardware breakpoint on writing to the variable "loopCount":

```
bh &loopCount,3
```

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 VxWorks core program. 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 0x0FFFE04
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	<p>This value gives the BDI information about the connected CPU:</p> <p style="padding-left: 20px;">type The CPU type from the following list: MPC500 or MPC800</p> <p style="padding-left: 20px;">Example: CPUTYPE MPC500</p>
CPUCLOCK value	<p>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.</p> <p>Important: When programming the MPC555 internal flash, this value is used to calculate the appropriate timing parameters.</p> <p style="padding-left: 20px;">value the CPU clock in hertz</p> <p style="padding-left: 20px;">Example: CPUCLOCK 25000000 ; CPU clock is 25.0MHz</p>
BDIMODE mode [param]	<p>This parameter selects the BDI debugging mode. The following modes are supported:</p> <p style="padding-left: 20px;">LOADONLY Loads and starts the VxWorks core. No debugging via BDM.</p> <p style="padding-left: 20px;">GATEWAY After loading and starting the VxWorks core. The BDI establishes a communication channel between the target server on the host and the target agent on the target. The second parameter (UART HDLC) defines the serial communication protocol. If no second parameter is present, normal BDM communication is used. This mode supports task level debugging.</p> <p style="padding-left: 20px;">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 WDB requests.</p> <p style="padding-left: 20px;">Example: BDIMODE AGENT RUN BDIMODE GATEWAY HDLC</p>
STARTUP mode [runtime]	<p>This parameter selects the target startup mode. The following modes are supported:</p> <p style="padding-left: 20px;">RESET This default mode forces the target to debug mode immediately out of reset. No code is executed after reset.</p> <p style="padding-left: 20px;">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.</p> <p style="padding-left: 20px;">RUN After reset, the target executes code until stopped by the Telnet "halt" command.</p> <p style="padding-left: 20px;">Example: STARTUP STOP 3000 ; let the CPU run for 3 seconds</p>

The following item is only used if BDI mode is AGENT:

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 the alternate 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

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

The following items are only used if BDI mode is AGENT. The values are used as answers to the WDB request `bkendTgtConnect` (see Tornado documentation):

- MEMBASE value** The base address of the target memory.
- MEMSIZE value** The size of the target memory.
- POOLBASE value** The base address of host controlled target memory.
- POOLSIZ value** The size of host controlled target memory.

The following items are only used if BDI mode is GATEWAY:

- INITTIME** timeout By default, the BDI assumes that the communication base address is transferred within 10 seconds after starting the VxWorks core. You can increase this timeout value if your application needs more time to startup.
- timeout the timeout value in seconds
- Example: INITTIME 20 ; wait 20 seconds for startup
-
- BAUDRATE** rate This parameter defines the used baudrate for the serial connection between the BDI2000 and the target. See the chapter "Serial GATEWAY mode" for more information. Rates above 500kb should only be used with the HDLC protocol.
- rate the baudrate to use. Following a list with the baudrates the BDI2000 can support:
9600, 19200, 38400, 57600, 115200
122kb, 130kb, 139kb, 149kb, 160kb, 174kb, 189kb,
208kb, 232kb, 260kb, 298kb, 347kb, 417kb, 521kb,
693kb, 1042kb
- Example: BAUDRATE 260000

The following items are only used if BDI mode is not GATEWAY UART or GATEWAY HDLC:

- VIO port [baudrate]** When this line is present and the optional Rx/Tx pins of the second BDI connector are routed to a UART, the serial IO of this UART can be accessed from the host via a Telnet session. 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. See the chapter "Serial GATEWAY mode" for more information about the BDI connector pins used for this serial link.
- Note:** You cannot use SIO and VIO at the same time.
- port The TCP/IP port used for the host communication.
- baudrate The BDI supports 2400 ... 115200 baud
- Example: VIO 7 ;TCP port for virtual IO
-
- 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.
- Note:** You cannot use SIO and VIO at the same time.
- 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	<p>The IP address of the host.</p> <p>ipaddress the IP address in the form xxx.xxx.xxx.xxx</p> <p>Example: IP 151.120.25.100</p>
FILE filename	<p>The file name of the VxWorks core. This name is used to access the core file via TFTP. If the filename starts with a \$, this \$ is replace with the path of the configuration file name.</p> <p>filename the filename including the full path</p> <p>Example: FILE F:\tornado\target\config\ads860\vxworks \$vxworks</p>
FORMAT format [offset]	<p>The format of the VxWorks core file. Currently binary, S-record, a.out and ELF file formats are supported. If the core 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 core file.</p> <p>format BIN, SREC, AOUT, ELF or ROM</p> <p>Example: FORMAT ELF</p>
LOAD mode	<p>In Agent mode, this parameters defines if the code is loaded automatically after every reset.</p> <p>mode AUTO, MANUAL</p> <p>Example: LOAD MANUAL</p>
START address	<p>The address where to start the VxWorks core. 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 VxWorks core. This means, the program starts at the normal reset address (0x0100).</p> <p>address the address where to start the VxWorks core</p> <p>Example: START 0x1000</p>
DEBUGPORT port	<p>The UDP port the target server uses to access the target agent.</p> <p>port the UDP port number (default = 0x4321)</p> <p>Example: DEBUGPORT 2001</p>
PROMPT string	<p>This entry defines a new Telnet prompt. The current prompt can also be changed via the Telnet interface.</p> <p>Example: PROMPT MPC860></p>
DUMP filename	<p>The default file name used for the Telnet DUMP command.</p> <p>filename the filename including the full path</p> <p>Example: DUMP dump.bin</p>
TELNET mode	<p>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.</p> <p>mode ECHO (default), NOECHO or LINE</p> <p>Example: TELNET NOECHO ; use old line mode</p>

3.2.4 Part [FLASH]

The Telnet interface supports programming and erasing of flash memories. The bdiWind 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 name of the file to program into the flash. 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\mpc\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

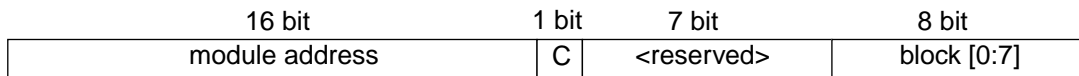
Without this optional parameter, the BDI executes a sector erase. If supported by the chip, you can also specify a block or chip erase. If UNLOCK is defined, the entry is also part of the unlock list. This unlock list is processed if the Telnet UNLOCK command is entered without any parameters.

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:

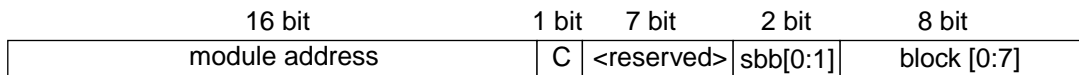
For the MPC555 internal flash, the BDI assumes the following structure of the address:



- 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

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  0xFFFF00000  0x0060  unlock block 0
WM16  0xFFFF00000  0x00D0
WM16  0xFFFF10000  0x0060  unlock block 1
WM16  0xFFFF10000  0x00D0
      . . .
WM16  0xFFFF00000  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 BDM GATEWAY mode

The bdiWind UDP-lite packet driver has to be linked to the initial VxWorks core. In order to work with your version of the Tornado environment, no Tornado files are delivered with bdiWind. Do not add the bdiWind UDP-lite packet driver before you are sure the VxWorks core runs on your hardware. For more information see VxWorks's Programmer's Guide, chapter Configuration.

3.3.1 Create a new directory for your hardware

Create a new directory in `tornado\target\config\` (e.g. `.....\myTarget`).

From an existing BSP directory (e.g. `tornado\target\config\ads860`) copy the following files into the new created directory or build them from scratch. Change the files so they work with your hardware.

```

config.h          ->  config.h
sysLib.c          ->  sysLib.c
sysALib.s        ->  sysALib.s
romInit.s        ->  romInit.s
Makefile         ->  Makefile
    
```

From the distribution disk copy the following files into the new directory:

```

bdiPkt.h          ->  wdbBdiPktDrv.h
bdiPkt.c          ->  wdbBdiPktDrv.c
tornado.add       ->  tornado.add
ads860.cnf        ->  ads860.cnf
    
```

Tornado II : From the distribution disk copy:

```

wdbBdi.c          ->  ../target/config/comps/src/
wdbBdi.cdf        ->  ../target/config/comps/vxWorks/
    
```

The file `tornado.add` includes the text which must be added to some of the existing tornado source files.

3.3.2 Tornado 1.01 and Tornado II

The BDI UDP-Lite packed driver needs some assembly support functions. This assembly functions can be added to the file sysALib.s in your BSP directory.

tornado\target\config\myTarget\sysALib.s:

Add the following lines to the list of exported symbols:

```
/* externals */
.globl      _usrInit      /* system initialization routine */

.globl      bdiInit      /* bdi communication setup */
.globl      bdiCall      /* enter debug mode */
```

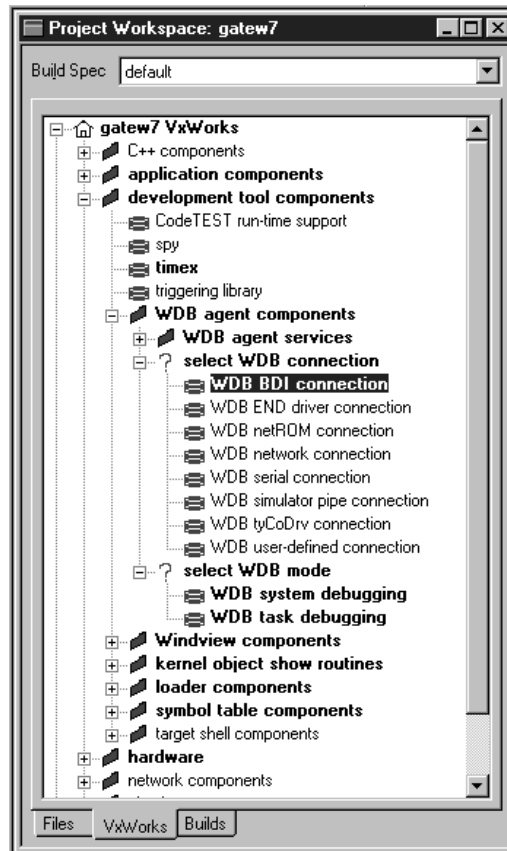
Add the following routines to the end of the file:

```
/******
*
* bdiInit - send BDI communication base address to BDI
*
* bdiInit
* (
* void* baseAddr    /* the base address of the communication structure */
* )
*/
bdiInit:
    sc          /* enter debug mode */
    blr

/******
*
* bdiCall - enter debug mode
*
* bdiCall(void)
*/
bdiCall:
    sc          /* enter debug mode */
    blr
```

3.3.3 Tornado II

For Tornado II the VxWorks configuration utility can be used to select the BDI as the WDB connection.



For more information about VxWorks configuration, see Tornado User's Guide, chapter Projects.

Summary for Tornado II:

- Copy wdbBdiPktDrv.c to the BSP directory
- Copy wdbBdiPktDrv.h to the BSP directory
- Copy wdbBdi.c to ../target/config/comps/src/
- Copy wdbBdi.cdf to ../target/config/comps/vxWorks/
- Edit sysALib.s as defined above (use tornado.add)
- Use the VxWorks configuration utility to select BDI as the WDB connection

3.3.4 Tornado 1.01

Change the following source files.

tornado\target\src\config\usrWdb.c:

Add the following line to the list of header files:

```
#include "drv/wdb/wdbUlripPktDrv.h"
#include "drv/wdb/wdbNetromPktDrv.h"
#include "wdbBdiPktDrv.h"
```

Add the following lines to the routine wdbCommIfInit:

```
    }
#endif /* (WDB_COMM_TYPE == WDB_COMM_CUSTOM) */

#if (WDB_COMM_TYPE == WDB_COMM_BDI)
{
/* bdi packet driver - supports task or external agent */

static WDB_BDI_PKT_DEV wdbBdiPktDev; /* BDI packet device */

wdbBdiPktDevInit (&wdbBdiPktDev, udpRcv);
udpCommIfInit (pCommIf, &wdbBdiPktDev.wdbDrvIf);
}
#endif /* (WDB_COMM_TYPE == WDB_COMM_BDI) */

/*
* Install the agents communication interface and RPC transport handle
```

tornado\target\config\all\configAll.h:

Add the following line to the list of communication paths:

```
#define WDB_COMM_NETROM 4 /* netrom packet device - bimodal */
#define WDB_COMM_CUSTOM 5 /* custom packet device - bimodal */
#define WDB_COMM_BDI 6 /* bdi packet device - bimodal */
```

tornado\target\config\myTarget\Makefile:

Add the following lines to the VxWorks Makefile:

```
MACH_EXTRA = wdbBdiPktDrv.o
```

tornado\target\config\myTarget\config.h:

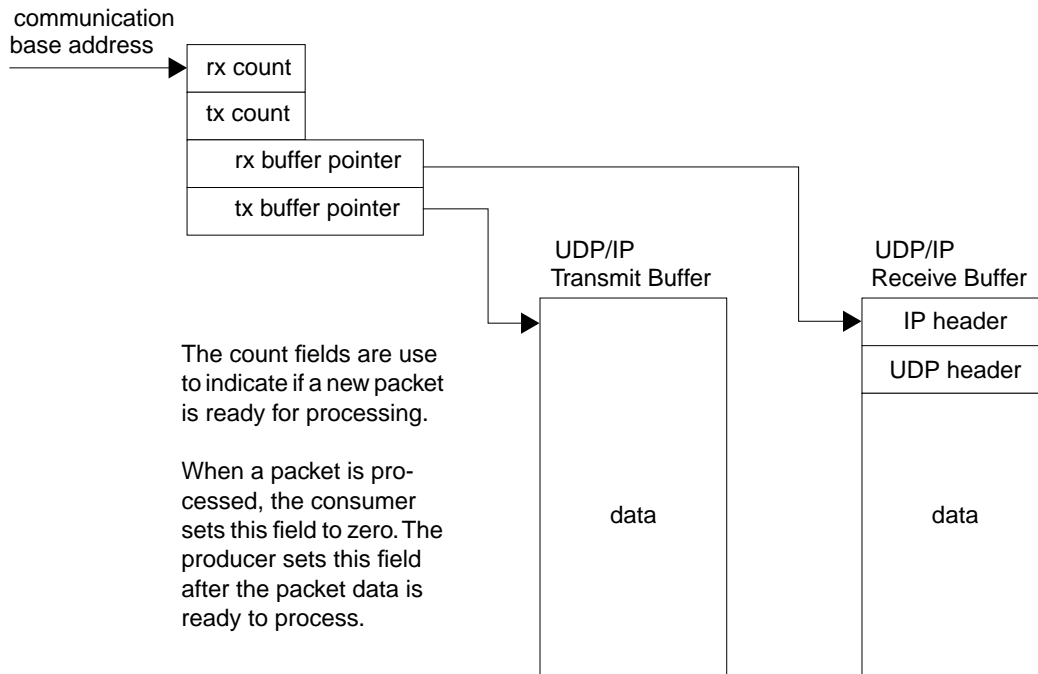
Change this file for your need. Configure the WDB agent so that the BDI packet driver will be used. The following lines show a possible WDB agent configuration:

```
/* agent mode */
#undef WDB_MODE
#define WDB_MODE WDB_MODE_BI /* WDB_MODE_[BI|TASK|EXTERN] */

/* agent communication path */
#undef WDB_COMM_TYPE
#define WDB_COMM_TYPE WDB_COMM_BDI
```

3.3.5 bdiWind UDP-lite packet driver

The bdiWind UDP-lite packet driver communicates with the BDI over the debug interface. Communication is established through buffers in the target memory. One for receiving packets and one for outgoing packets.



When a target server sends a request to the target agent, this request is sent to the BDI. When it receives this request, the BDI places it in the receive buffer of the target and sets the rx count field accordingly. A VxWorks task polls the rx count field to retrieve such requests, and passes them on to the agent.

When the agent has data to send, the data is copied into the transmit buffer and the tx count field is set. Then the target CPU changes to debug mode by executing a system call instruction (sc). The BDI detects this change to debug mode, reads the data from the transmit buffer and sends the packet to the requesting target server.

Important note:

Communication between the BDI and the target CPU is only possible while the target CPU is frozen. In this state, no interrupts are processed. To transfer a complete IP frame the target CPU is frozen during a time of up to 5ms.

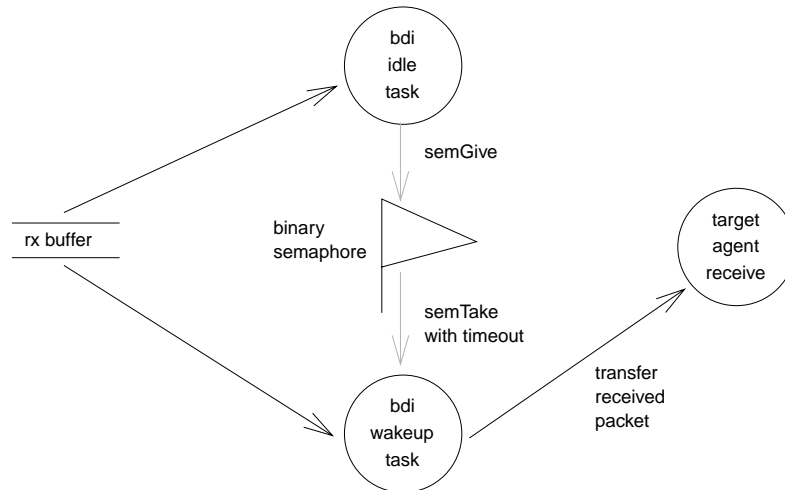
3.3.5.1. Transferring the communication base address

The address of the communication structure is not fixed in memory and may change after every new build of the VxWorks core. On the other side, the BDI has to know where this structure lies in memory. To transfer this information to the BDI, the initializing routine (wdbBdiPktDevInit) copies the address of the communication structure into register GPR3 and enters debug mode by executing a SC assembler instruction. The BDI detects this change to debug mode and reads the register GPR3. To support this transfer, an assembly routine is added to the module sysALib.c.

```
bdiInit:
    sc          /* enter debug mode      */
    blr
```

3.3.5.2. Polling for requests

Because no interrupt can be generated via debug interface, polling is required. To optimize communication performance and CPU load, two VxWorks task are generated to handle the polling task. The following figure shows the relationship of this two tasks.



The BDI Idle Task polls the rx count field without delay. The task priority has to be 255 because it consumes all remaining CPU time. Every time it finds the rx count field not zero, it signal this by giving a binary semaphore.

The BDI Wakeup Task should have a priority equal to the target agent. It waits with time-out at the semaphore. Then it checks the rx count field and sends the packet to the target agent if one is available. Time-out when waiting at the semaphore is used because it's possible that the BDI Idle Task gets no CPU time. This may be the case when an faulty application task loops without releasing the CPU.

3.3.5.3. Automatic change to Agent mode

The BDI may change automatically to Agent mode in case of a fatal error or when HALT is entered at the Telnet prompt. In this case, restart the target server on the host and attach the debugger to the system. You can now analyze the crashed application (e.g. inspect variables).

3.3.6 PPC Interrupt Handling

In Gateway mode, only a few PPC interrupts causes an entry into debug mode. Most of the interrupts are handled by the application. By default, the Debug Enable Register (DER) is set as follows:

Debug Enable Register

Bit	Mnemonic	State	Description
0	-		
1	RSTE	enabled	Reset Interrupt
2	CHSTPE	enabled	Check Stop
3	MCIE		Machine Check Interrupt
4-5	-		
6	EXTIE		External Interrupts
7	ALIE		Alignment Interrupt
8	PRIE		Program Interrupt
9	FPUVIE		Floating-Point Unavailable Interrupt
10	DECIE		Decrementer Interrupt
11-12	-		
13	SYSIE	enabled	System Call Interrupt (used to signal a pending WDB answer)
14	TRE		Trace Interrupt
15-16	-		
17	SEIE		Software Emulation Interrupt
18	ITLBMSE		Implementation Specific Instruction TLB Miss
19	ITLBERE		Implementation Specific Instruction TLB Error
20	DTLBMSE		Implementation Specific Data TLB Miss
21	DTLBERE		Implementation Specific Data TLB Error
22-27	-		
28	LBRKE		Load/Store Breakpoint Interrupt
29	IBRKE		Instruction Breakpoint Interrupt
30	EBRKE	enabled	External Breakpoint Interrupt
31	DPIE	enabled	Development 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 AGENT mode

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

3.4.1 Target setup

Target initialization may be done at two places. First with the BDI configuration file, second within the VxWorks routine "sysHwInit". The setup in the configuration file must at least enable access to the target memory where the VxWorks core 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 routine "sysHwInit".

3.4.2 Configure VxWorks

Tornado 1.01:

Use the VxWorks file "config.h" to scale your VxWorks core. Undefine INCLUDE_WDB and statically link all your modules with the VxWorks core. Don't forget to create your tasks because they can't be created dynamically.

Add the following lines to the VxWorks Makefile:

```
ADDED_CFLAGS      = -g
MACH_EXTRA        = myModule.o
```

For more information about building a bootable VxWorks application, see Tornado User's Guide chapter 11.6 "Creating Bootable Applications".

Tornado II:

Use the VxWorks configuration utility and exclude all "development tool components".

Note:

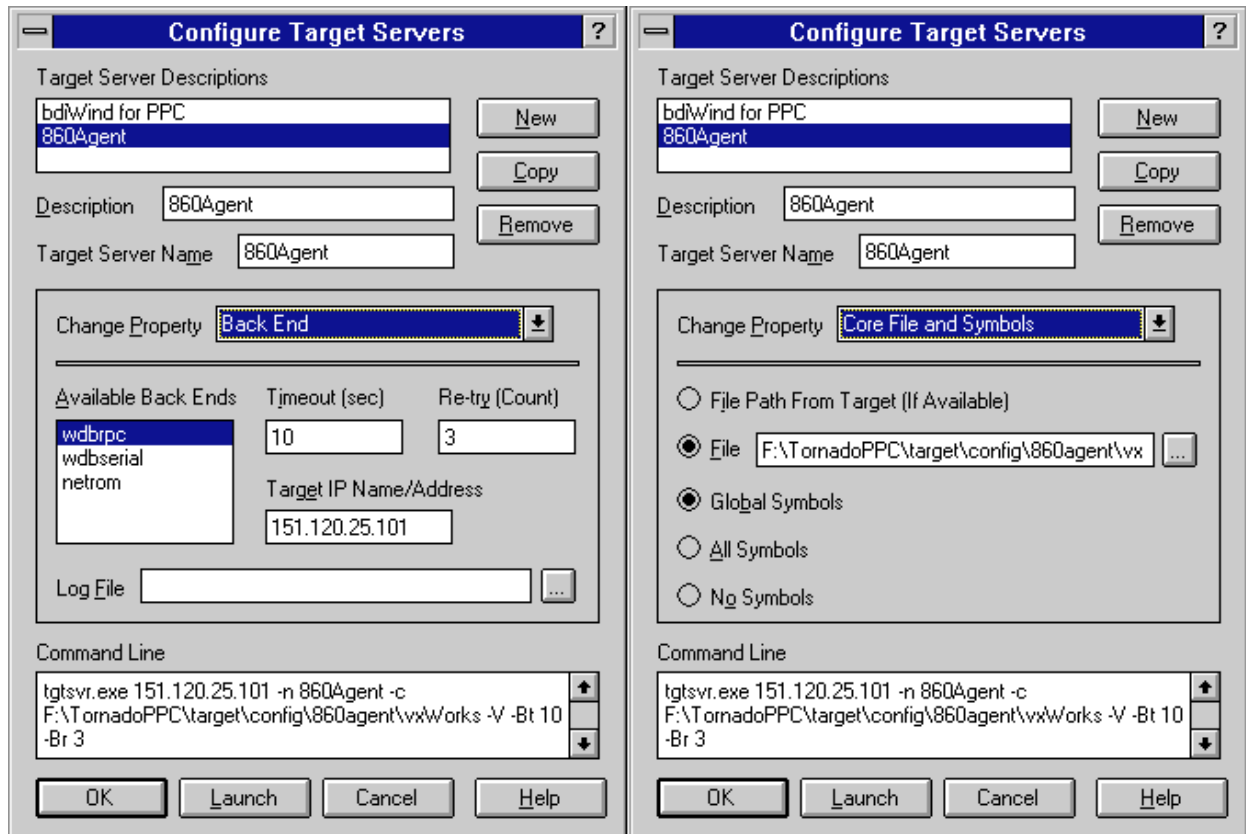
If your vxWorks application needs information from the boot line (e.g. if you are using networking), enter the following statement at the end of *sysHwInit()*.

```
/* init boot line for BDM debugging */
strcpy(sysBootLine, DEFAULT_BOOT_LINE);
```

This is necessary because there is no Boot-ROM which initializes the boot line variable in RAM.

3.4.3 Target Server Setup

Setup the target server as follows:



3.4.4 Debugging with GDB

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

After starting the debugger, enter "attach system" at the GDB **prompt**, do not use the "Attach" menu entry because the task list cannot be read at this time. If not already suspended, this stops the execution of application code and the target CPU changes to debug mode.

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

Accessing target resources (e.g. inspecting a variable) is only possible during the time the application is halted (e.g. cause by breakpoint).

Note:

If you would like to debug code before the VxWorks **taskLib** has been initialized, enter the GDB command **tasking-off** to disable fetching of the thread list (Tornado Release Notes 1.0.1 chapter 6.4 GDB).

3.4.5 PPC Interrupt Handling

In Agent mode, 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	Description
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 Instruction TLB Miss
19	ITLBERE		Implementation Specific Instruction 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	Development Port Nonmaskable Request

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

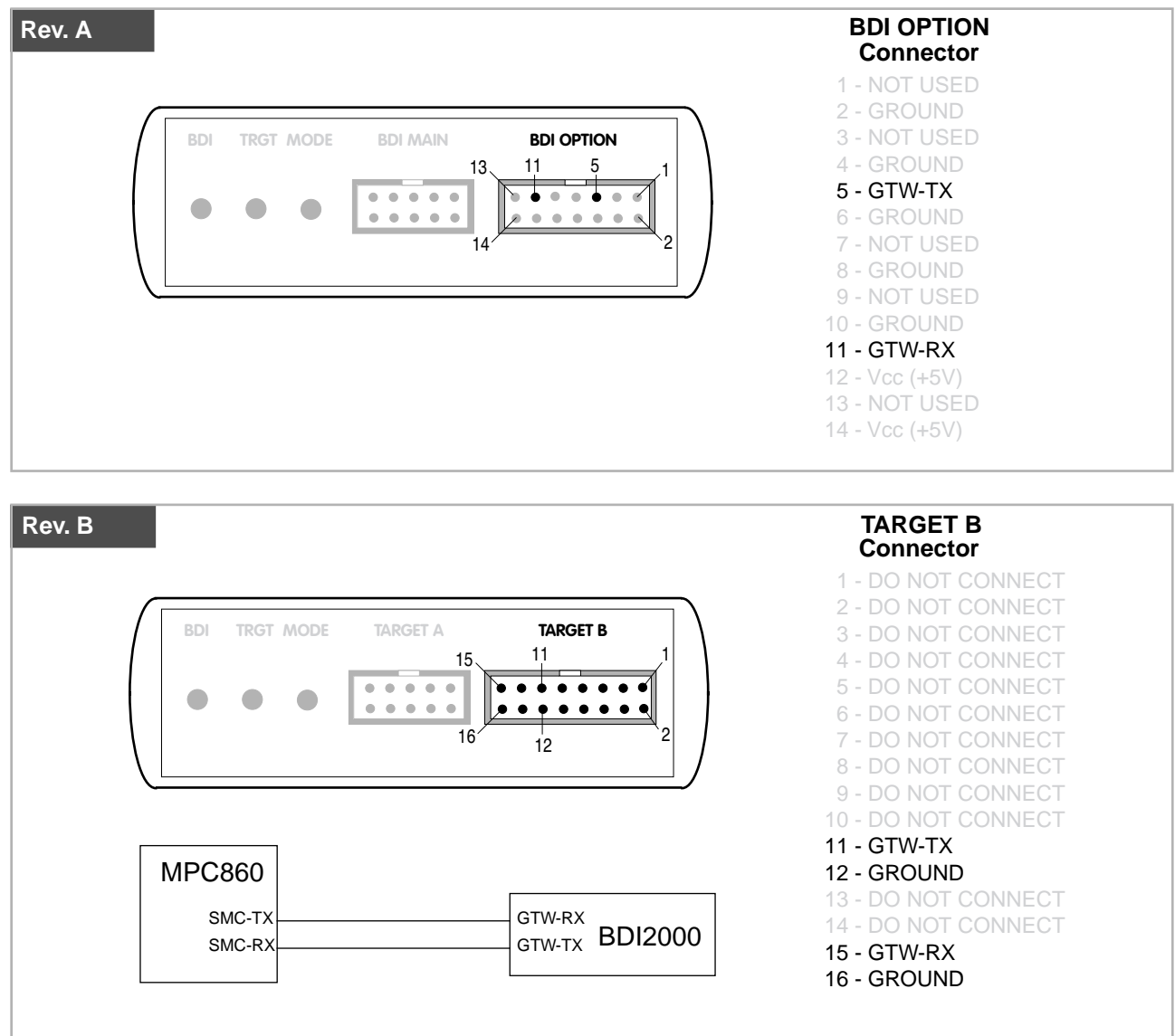
```
WSPR 149 0xFFE7400F ;DER: set debug enable register
```

3.5 Serial GATEWAY mode

This mode can be used, if the time the target is frozen in BDM Gateway mode is not acceptable (e.g. interrupt response time). In order to use this mode, there must be a free serial channel on the target (e.g. a SCC channel). The processors or UART's RX/TX pins can be directly used because the BDI2000 needs and drives Vdd I/O levels (Rev. A/B TTL levels). There is no need for a RS232 level converter.

3.5.1 Target connection

The two communication lines (RX/TX) and optional an additional GROUND has to be connected between the BDI2000 and the target. The cable length should not exceed 50 cm (20").



3.5.2 VxWorks configuration

Configure VxWorks for a serial, fast SLIP or HDLC WDB connection. You may use the standard serial connection which selects the VxWorks SLIP UDP-lite packet driver but this is a very slow connection. If your BSP does not support a WDB serial connection, or if you would like to select a baudrate greater than 38'400, then you have to write your own fast SLIP or HDLC UDP-lite packet driver. There are examples on the diskette for a fast SLIP and a HDLC UDP-Lite packet driver running on a MBX860 board.

If you would like to use the example driver, copy the following files from the distribution disk:

```

bdiSlip.h      ->  ../target/config/your_bsp/wdbFslipPktDrv.h
bdiSlip.c      ->  ../target/config/your_bsp/wdbFslipPktDrv.c
bdiHdlc.h      ->  ../target/config/your_bsp/wdbHdlcPktDrv.h
bdiHdlc.c      ->  ../target/config/your_bsp/wdbHdlcPktDrv.c
wdbFslip.c     ->  ../target/config/comps/src/
wdbHdlc.c      ->  ../target/config/comps/src/
wdbBdi.cdf     ->  ../target/config/comps/vxWorks/
    
```

Use the VxWorks configuration utility to select the fast SLIP or HDLC WDB connection. Change the example driver so it runs on your hardware.

BDI2000 @33MHz	MPC8xx @25MHz	MPC8xx @33MHz	MPC8xx @40MHz	MPC8xx @50MHz
122'000		122'000		
130'000	130'000	130'000		130'000
139'000		139'000	139'000	
149'000		149'000		149'000
160'000		160'000		
174'000	174'000	174'000		174'000
189'000		189'000		
208'000		208'000	208'000	208'000
232'000		232'000		
260'000	260'000*	260'000		260'000
298'000		298'000*		
347'000		347'000*		347'000
417'000		417'000*	417'000*	
521'000	521'000*	521'000*		521'000*
1'042'000		1'042'000*		1'042'000*

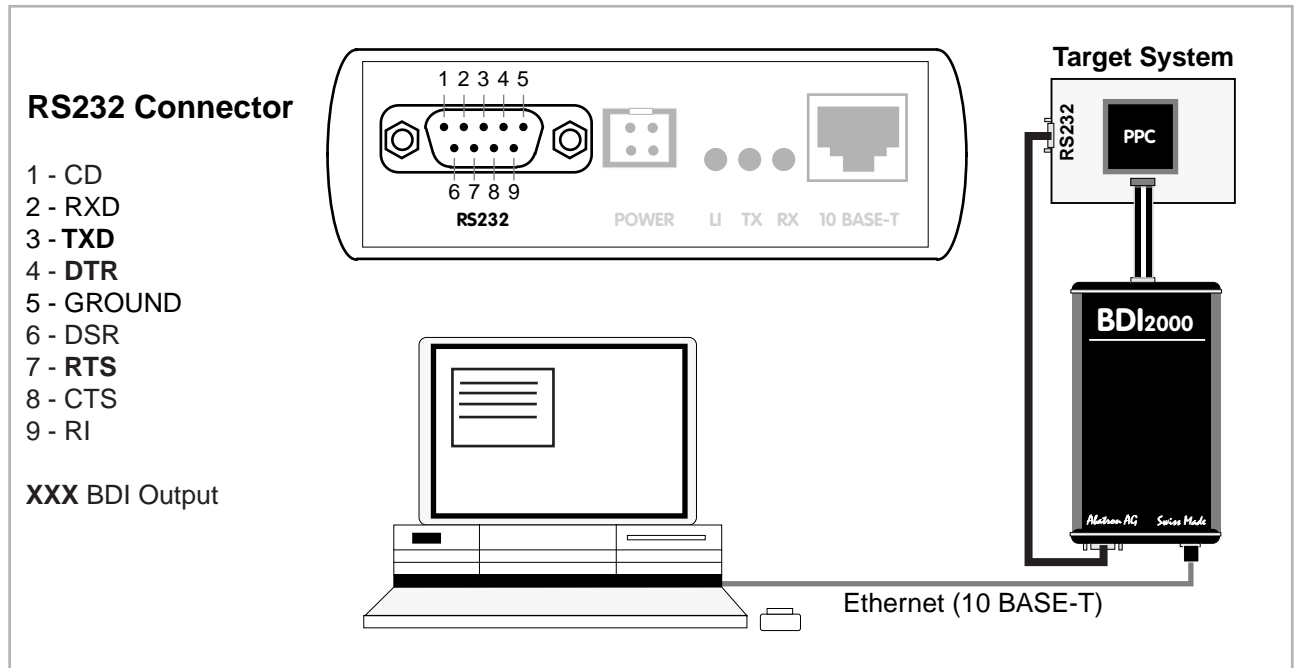
* Only with SCC channel, not with SMC because of CPM performance

IMPORTANT:

Select WDBRPC (not WDBSERIAL) as the backend because the target server communicates with the BDI2000 via the ethernet connection.

3.6 Target serial I/O via BDI

A RS232 port of the target can be connected to the RS232 port of the BDI2000. 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 BDI2000 port. This way you can route vxWorks console I/O to a TCP/IP channel in order to completely remote control a vxWorks system.



The configuration parameter "SIO" is used to enable this serial I/O routing. The baudrate can be selected with in the configuration file (see "SIO" parameter). The framing is fix, 8 data, 1 stop, no parity. The BDI asserts RTS and DTR when a TCP connection is established.

```
[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.

3.7 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 tasks may be done by using this interface. Enter help at the Telnet command prompt to get a list of the available commands.

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 normal debugging with Tornado, there is no need for connecting to the Telnet server. The Telnet interface may be useful during the first installation of the bdiWind system or in case of special debug needs (e.g. setting breakpoints on variable access).

Note:

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.

Note:

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:

```
"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",
"BOOT                               reset the BDI and reload the configuration",
"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> [<bdIP> [<gateway> [<mask>]]]",
"HELP                               display command list",
"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 BASE-T
Serial Transfer Rate between BDI and Target	up to 16 Mbit/s
Supported target voltage	1.8 – 5.0 V (3.0 – 5.0 V with Rev. A/B)
Operating Temperature	+ 5 °C ... +60 °C
Storage Temperature	-20 °C ... +65 °C
Relative Humidity (noncondensing)	<90 %rF
Size	190 x 110 x 35 mm
Weight (without cables)	420 g
Host Cable length (RS232)	2.5 m

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: BDI2000

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:


EN 50081-2
EN 50082-2

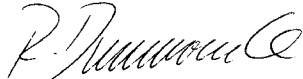
This declaration of conformity is based on the test report no. QNL-E853-05-8-a of QUINEL, Zug, accredited according to EN 45001.

Manufacturer:

ABATRON AG
Stöckenstrasse 4
CH-6221 Rickenbach

Authority:


Max Vock
Marketing Director


Ruedi Dummermuth
Technical Director

Rickenbach, May 30, 1998

7 Warranty

ABATRON Switzerland warrants the physical diskette, cable, BDI2000 and physical documentation to be free of defects in materials and workmanship for a period of 24 months 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 diskette, cable, BDI2000 or documentation. The remedy for breach of this warranty shall be limited to replacement and shall not encompass any other damages, including but not limited to 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 diskette, cable, BDI2000 and documentation, 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 except exchanging the fuse.

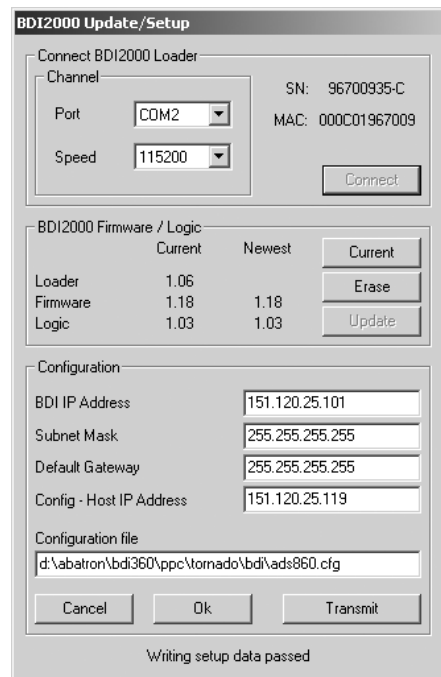
Appendices

A BDI2000 Setup/Update

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



To avoid data line conflicts, the BDI2000 must be disconnected from the target system while programming the logic for an other target CPU (see Chapter 2.1.1).



dialog box «BDI2000 Update/Setup»

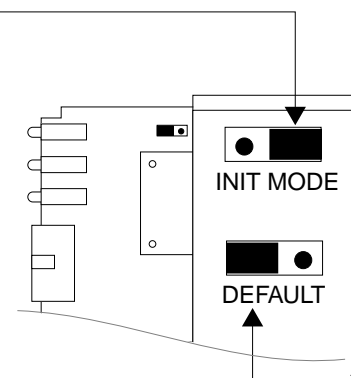
Before you can use the BDI2000 together with the Tornado development environment, you must store the initial configuration parameters in the BDI2000 flash memory. The following options allow you to do this:

- Channel Select the communication port where the BDI2000 is connected during this setup session.
- Baudrate Select the baudrate used to communicate with the BDI2000 loader during this setup session.
- Connect Click on this button to establish a connection with the BDI2000 loader. Once connected, the BDI2000 remains in loader mode until it is restarted or this dialog box is closed.
- Current Press this button to read back the current loaded BDI2000 software and logic versions. The current loader, firmware and logic version will be displayed.
- Update This button is only active if there is a newer firmware or logic version present in the execution directory of the bdiWind setup software. Press this button to write the new firmware and/or logic into the BDI2000 flash memory / programmable logic.

BDI IP Address	Enter the IP address for the BDI2000. 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 BDI2000. Every BDI2000 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.xxx e.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 BDI2000 after every start-up.
Configuration file	Enter the full path and name of the configuration file. e.g. D:\tornado\target\config\bdi\ads860bdi.cnf For information about the syntax of the configuration file see the bdiWind User manual. This name is transmitted to the TFTP server when reading the configuration file.
Transmit	Click on this button to store the configuration in the BDI2000 flash memory.

In rare instances you may not be able to load the firmware in spite of a correctly connected BDI (error of the previous firmware in the flash memory). **Before carrying out the following procedure, check the possibilities in Appendix «Troubleshooting».** In case you do not have any success with the tips there, do the following:

- Switch OFF the power supply for the BDI and open the unit as described in Appendix «Maintenance»
- Place the jumper in the «**INIT MODE**» position
- Connect the power cable or target cable if the BDI is powered from target system
- Switch ON the power supply for the BDI again and wait until the LED «MODE» blinks fast
- Turn the power supply OFF again
- Return the jumper to the «**DEFAULT**» position
- Reassemble the unit as described in Appendix «Maintenance»



B Troubleshooting

Problem

The firmware can not be loaded.

Possible reasons

- The BDI is not correctly connected with the target system (see chapter 2).
- The power supply of the target system is switched off or not in operating range (4.75 VDC ... 5.25 VDC) --> MODE LED is OFF or RED
- The built in fuse is damaged --> MODE LED is OFF
- The BDI is not correctly connected with the Host (see chapter 2).
- A wrong communication port (Com 1...Com 4) is selected.

Problem

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

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 (BDI2000 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 BDI2000 is not connected or not correctly connected to the network (LAN cable or media converter)
- An incorrect IP address was entered (BDI2000 configuration)

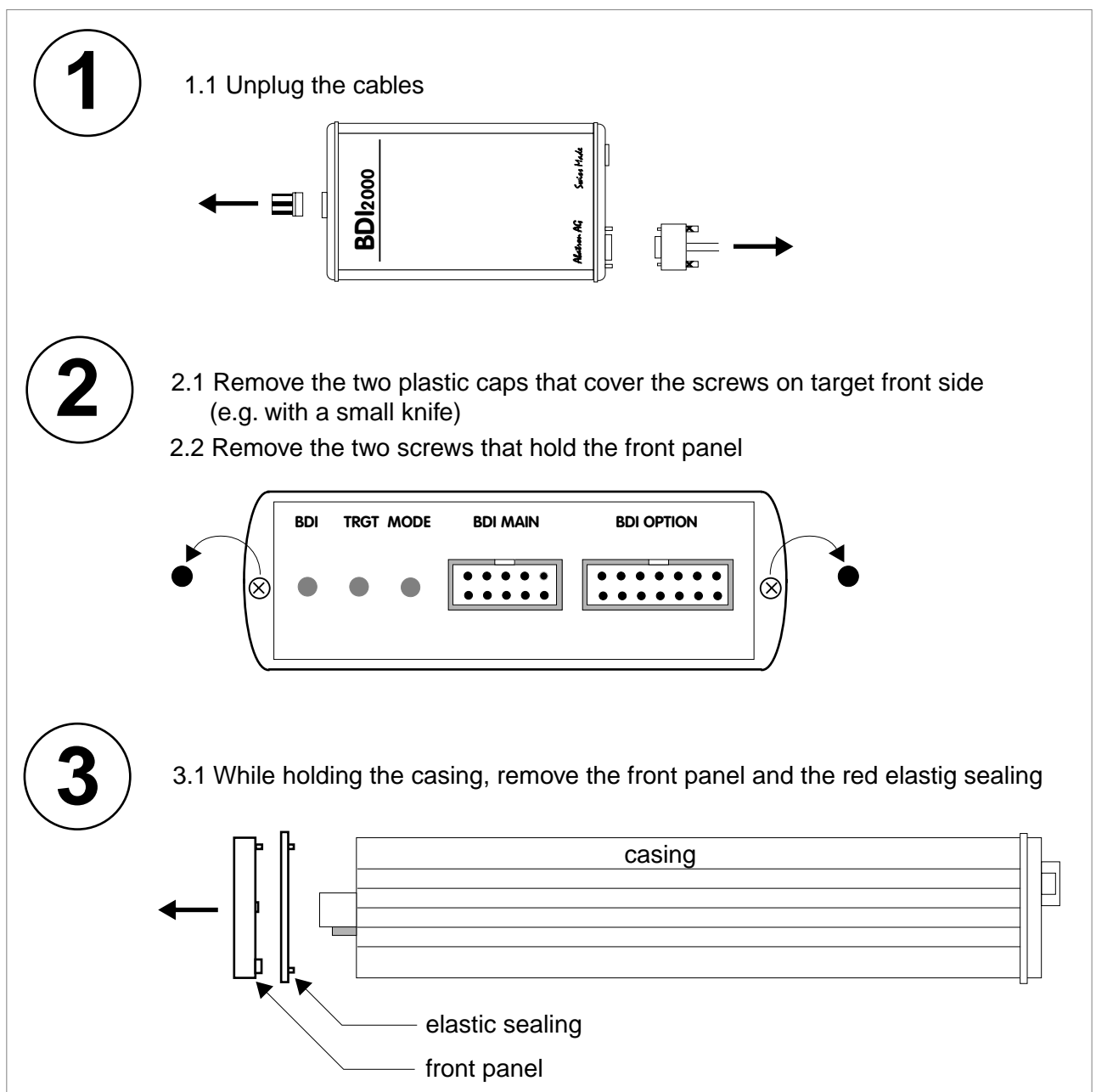
C Maintenance

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

If the BDI is connected correctly and it is still not responding, then the built in fuse might be damaged (in cases where the device was used with wrong supply voltage or wrong polarity). To exchange the fuse or to perform special initialization, please proceed according to the following steps:

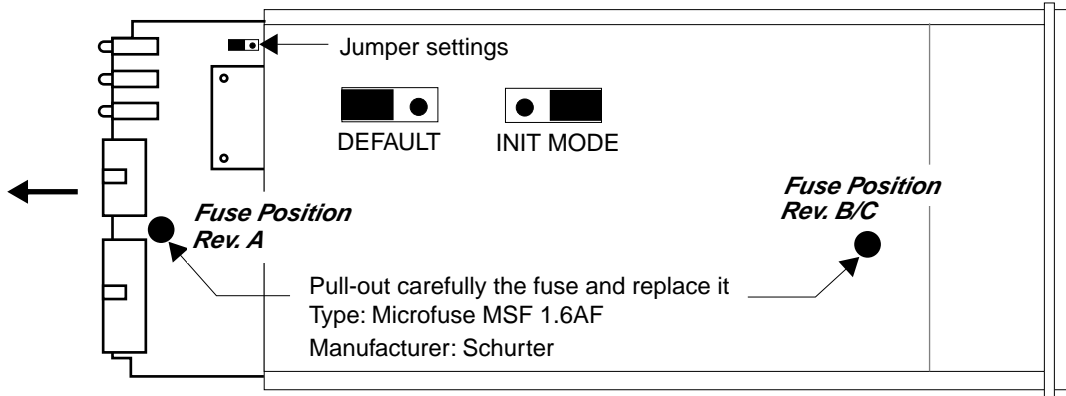


**Observe precautions for handling (Electrostatic sensitive device)
Unplug the cables before opening the cover.
Use exact fuse replacement (Microfuse MSF 1.6 AF).**



4

4.1 While holding the casing, slide carefully the print in position as shown in figure below

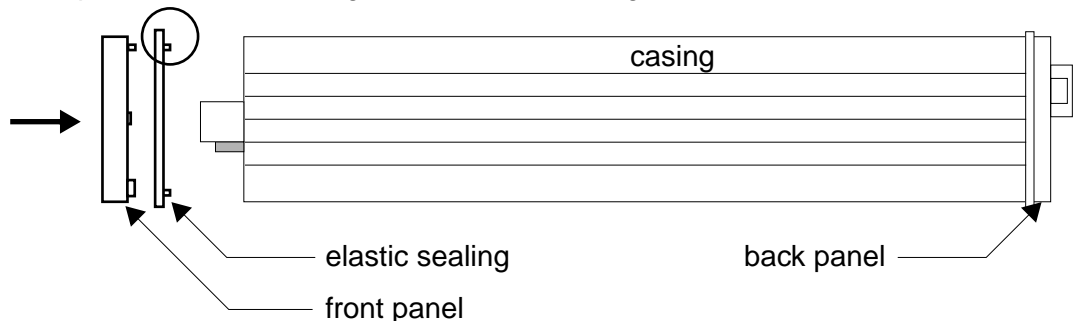


5

Reinstallation

5.1 Slide back carefully the print. Check that the LEDs align with the holes in the back panel.

5.2 Push carefully the front panel and the red elastic sealing on the casing. Check that the LEDs align with the holes in the front panel and that the position of the sealing is as shown in the figure below.



5.3 Mount the screws (do not overtighten it)

5.4 Mount the two plastic caps that cover the screws

5.5 Plug the cables



**Observe precautions for handling (Electrostatic sensitive device)
Unplug the cables before opening the cover.
Use exact fuse replacement (Microfuse MSF 1.6 AF).**

D Trademarks

All trademarks are property of their respective holders.