

*bd*NDI

JTAG debug interface for Nucleus™ Debugger

PowerPC 7440/7450/8641

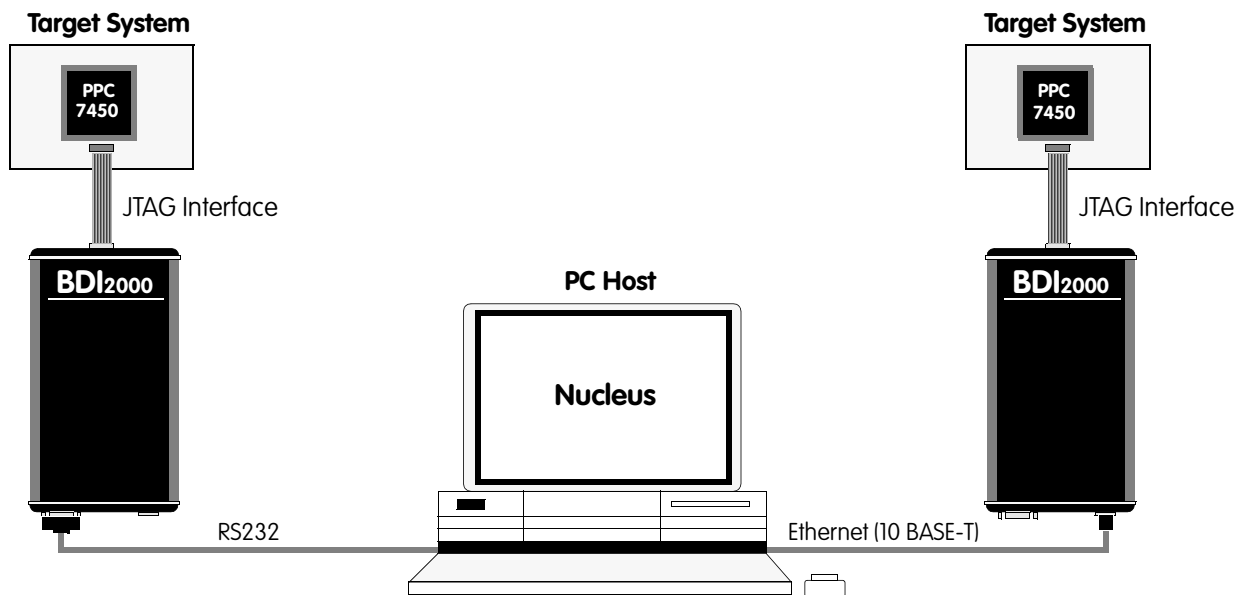


User Manual

Manual Version 1.01 for BDI2000

1 Introduction	3
1.1 BDI2000.....	3
2 Installation	4
2.1 Connecting the BDI2000 to Target	4
2.1.1 Changing Target Processor Type	6
2.2 Connecting the BDI2000 to Power Supply	7
2.3 Status LED «MODE».....	8
2.4 Connecting the BDI2000 to the Host	9
2.4.1 Serial line communication	9
2.4.2 Ethernet communication	10
2.5 Installation of the Configuration Software	11
2.6 Configuration	12
2.6.1 BDI2000 Setup/Update	12
3 Init List.....	14
4 BDI working modes.....	16
4.1 Startup Mode	17
4.1.1 Startup mode RESET.....	17
4.1.2 Startup Mode STOP.....	17
4.1.3 Startup mode RUN.....	17
4.2 Dual-Core Support for MPC8641D	18
5 Working with Nucleus.....	20
5.1 Direct Commands.....	20
5.1.1 Flash.Setup	21
5.1.2 Flash.Erase	21
5.1.3 Flash.Load	21
5.1.4 Flash.Idle.....	21
5.2 Download to Flash Memory	22
6 Telnet Interface.....	25
7 Specifications	26
8 Environmental notice.....	27
9 Declaration of Conformity (CE).....	27
10 Warranty.....	28
 Appendices	
A Troubleshooting	29
B Maintenance.....	30
C Trademarks	32

1 Introduction



The BDI2000 adds JTAG based debug features to the Nucleus debugger environment from Mentor Graphics. With the BDI2000, you control and monitor the microcontroller solely through the stable on-chip debugging services. You won't waste time and target resources with a software ROM monitor, and you eliminate the cabling problems typical of ICE's. This combination runs even when the target system crashes and allows developers to continue investigating the cause of the crash. A RS232 interface with a maximum of 115 kBaud and a 10Base-T Ethernet interface is available for the host interface.

The configuration software is used to update the firmware and to configure the BDI2000 so it works with the debugger.

1.1 BDI2000

The BDI2000 is a processor system in a small box. It implements the interface between the JTAG pins of the target CPU and a 10Base-T Ethernet / RS232 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. B).

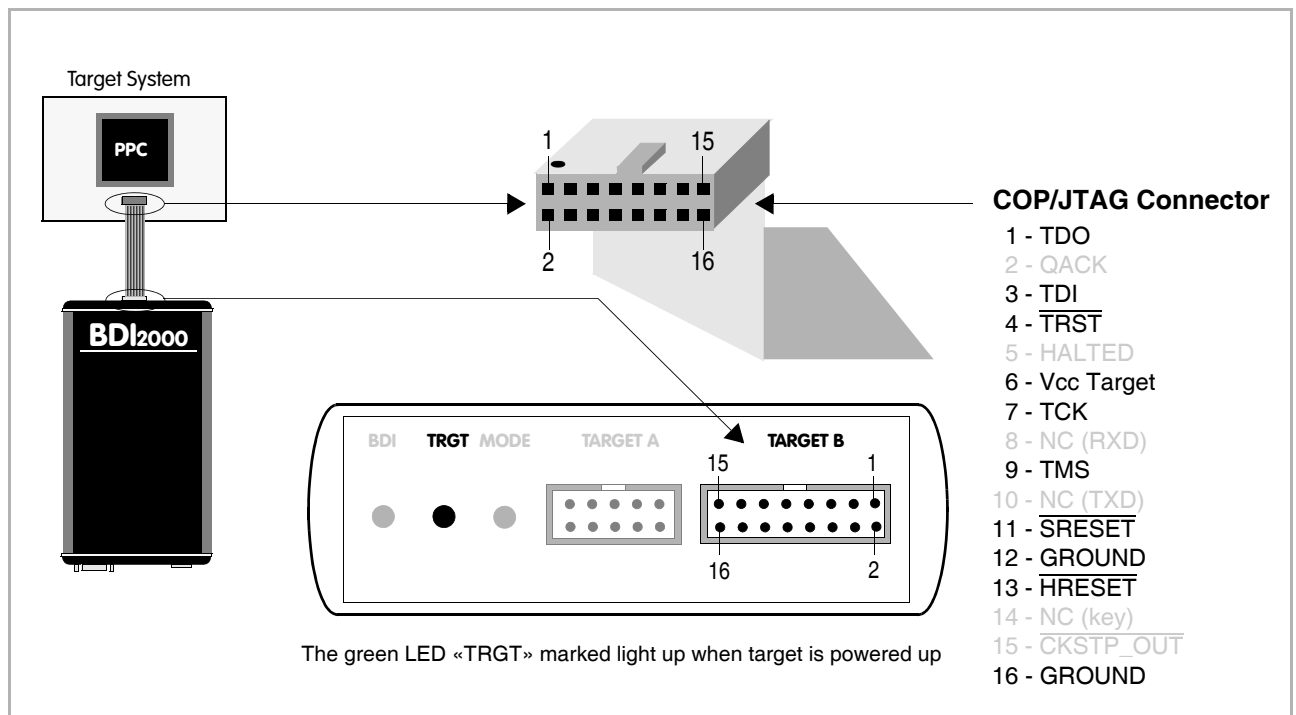
2 Installation

2.1 Connecting the BDI2000 to Target

The cable to the target system is a 16 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 PowerPC COP connector 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 TARGET B connector signals see table on next page. **BDI TARGET B Connector Signals:**

Pin	Name	Description
1	TDO	JTAG Test Data Out This input to the BDI2000 connects to the target TDO pin.
2	IO0	General purpose I/O This output of the BDI2000 connects to the target QACK pin. Currently not used.
3	TDI	JTAG Test Data In This output of the BDI2000 connects to the target TDI pin.
4	$\overline{\text{TRST}}$	JTAG Test Reset This output of the BDI2000 resets the JTAG TAP controller on the target.
5	IN0	General purpose Input This input to the BDI2000 connects to the target HALTED pin. Currently not used.
6	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. 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.
7	TCK	JTAG Test Clock This output of the BDI2000 connects to the target TCK pin.
8	<reseved>	
9	TMS	JTAG Test Mode Select This output of the BDI2000 connects to the target TMS line.
10	<reseved>	
11	$\overline{\text{SRESET}}$	Soft-Reset This open collector output of the BDI2000 connects to the target SRESET pin.
12	GROUND	System Ground
13	$\overline{\text{HRESET}}$	Hard-Reset This open collector output of the BDI2000 connects to the target HRESET pin.
14	<reseved>	
15	IN1	General purpose Input This input to the BDI2000 connects to the target CKSTP_OUT pin. Currently not used.
16	GROUND	System Ground

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 (Version A) or via the POWER connector (Version B). For more information see chapter 2.2.1 «External Power Supply».



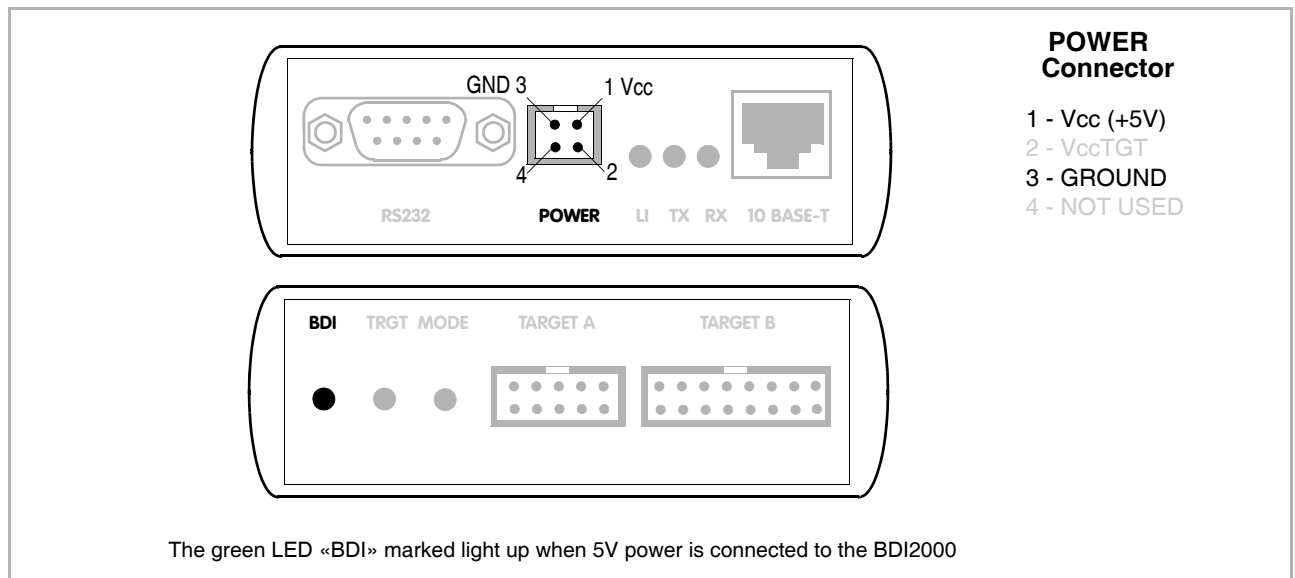
To avoid data line conflicts, the BDI2000 must be disconnected from the target system while programming the logic for an other target CPU.

2.2 Connecting the BDI2000 to Power Supply

The BDI2000 needs to be supplied with 5 Volts (max. 1A) via the POWER connector. 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.**

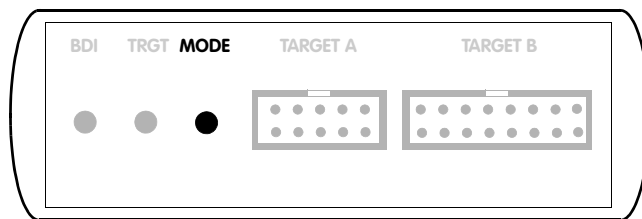


Please switch on the system in the following sequence:

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

2.3 Status LED «MODE»

The built in LED indicates the following BDI states:

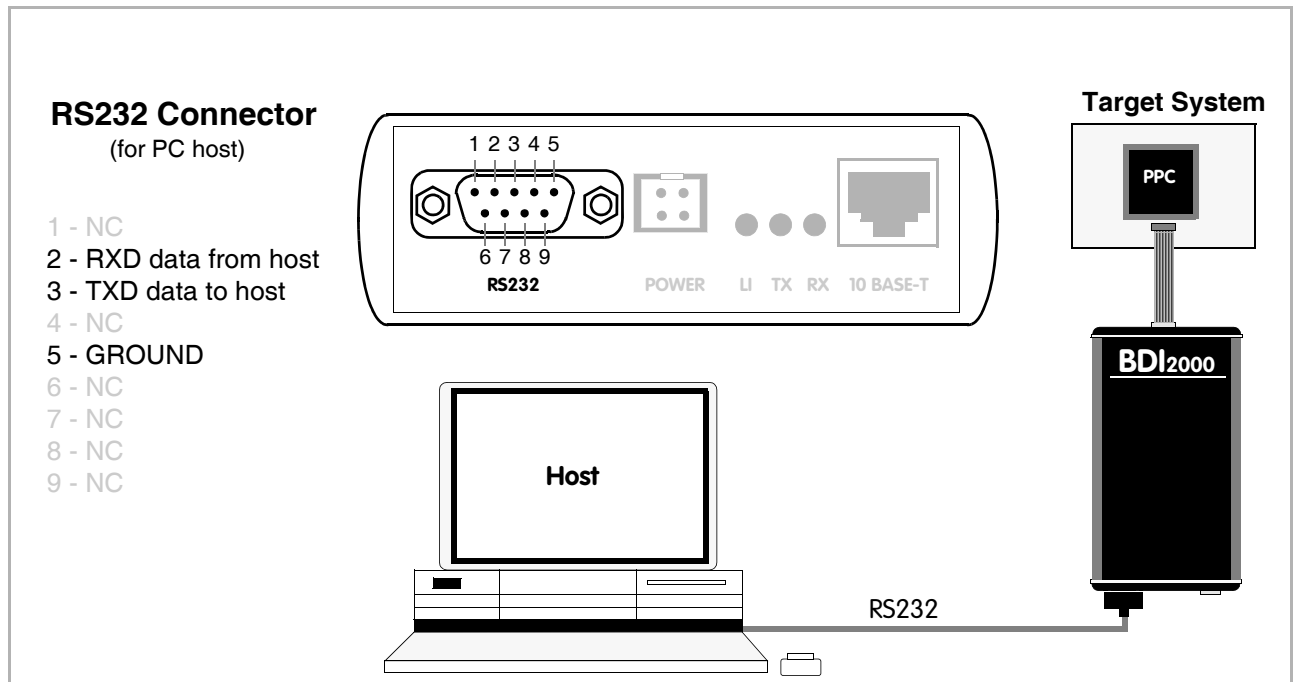


MODE LED	BDI STATES
OFF	The BDI is ready for use, the firmware is already loaded.
ON	The power supply for the BDI2000 is < 4.75VDC.
BLINK	The BDI «loader mode» is active (an invalid firmware is loaded or loading firmware is active).

2.4 Connecting the BDI2000 to the Host

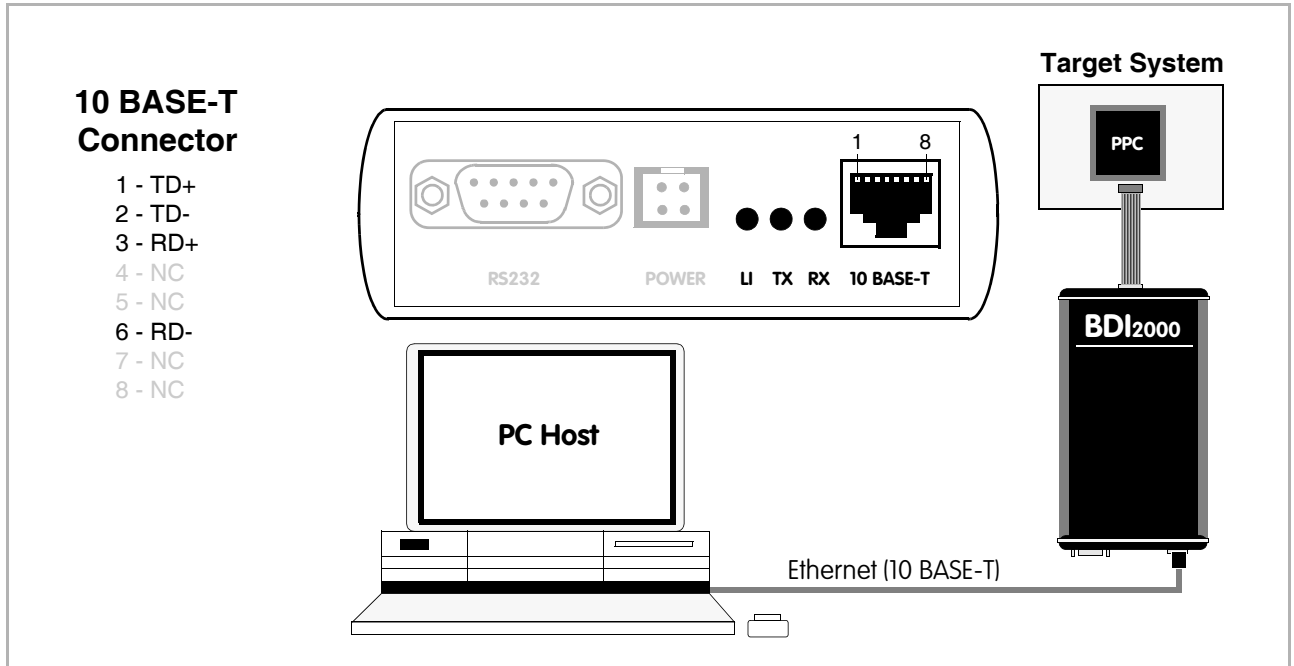
2.4.1 Serial line communication

The host is connected to the BDI through the serial interface (COM1...COM4). The communication cable between BDI and Host is a serial cable (RXD / TXD are crossed). 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.



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 BDI. Copy all these files to a directory on your hard disk.

The following files are on the diskette:

b20pws.exe	Configuration program
b20pws.hlp	Helpfile for the configuration program
b20pws.cnt	Help contents file
b20pwsfw.xxx	Firmware for BDI2000 for COP targets (PPC7450)
copjed20.xxx	JEDEC file for BDI2000 (Rev. B) logic device programming
copjed21.xxx	JEDEC file for BDI2000 (Rev. C) logic device programming
bdiifc32.dll	BDI Interface DLL
*.bdi	Configuration Examples

Example of an installation process:

- Copy the entire contents of the enclosed diskette into a directory on the hard disk.
- You may create a new shortcut to the b20cop.exe configuration program.

2.6 Configuration

Before you can use the BDI together with the debugger, the BDI must be configured. Use the *SETUP* menu and follow the steps listed below:

- Load or update the firmware / logic, store IP address --> *Firmware*
- Set the communication parameters between Host and BDI --> *Communication*
- Setup an initialization list for the target processor --> *Initlist*
- Select the working mode --> *Mode*
- Transmit the configuration to the BDI --> *Mode Transmit*

For information about the dialogs and menus use the help system (F1).

2.6.1 BDI2000 Setup/Update

First make sure that the BDI is properly connected (see Chapter 2.1 to 2.4). The BDI must be connected via RS232 to the Windows host.



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).

The following dialogbox is used to check or update the BDI firmware and logic and to set the network parameters.

dialog box «BDI2000 Update/Setup»

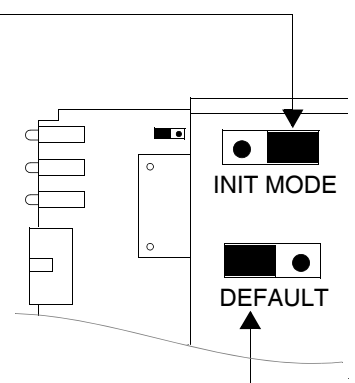
The following options allow you to check or update the BDI firmware and logic and to set the network parameters:

- 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.

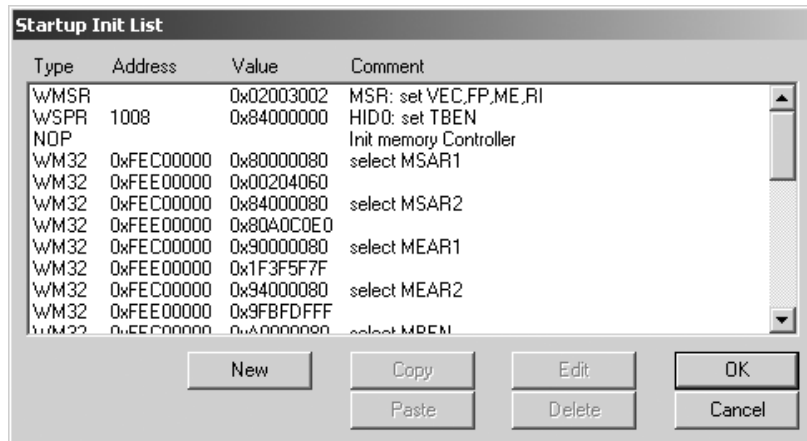
Connect	Click on this button to establish a connection with the BDI2000 loader. Once connected, the BDI2000 remains in loader mode until it is restarted or this dialog box is closed.
Current	Press this button to read back the current loaded BDI2000 software and logic versions. The current loader, firmware 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 BDI setup software. Press this button to write the new firmware and/or logic into the BDI2000 flash memory / programmable logic.
IP Address	Enter the IP address for the BDI2000. Use the following format: xxx.xxx.xxx.xxe.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.xxe.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..
Transmit	Click on this button to store the network 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»



3 Init List



dialog box «Startup Init List»

In order to prepare the target for debugging, you can define an Initialization List. This list is stored in the Flash memory of the BDI2000 and worked through every time the target comes out of reset. Use it to get the target operational after a reset. The memory system is usually initialized through this list. After processing the init list, the RAM used to download the application must be accessible.

Use on-line help (F1) and the supplied configuration examples on the distribution disk to get more information about the init list.

You may also use the debuggers feature to setup the hardware (chip initialization file).

Special BDI Configuration Registers:

In order to change some special configuration parameters of the BDI, the SPR entry in the init list is used. Normal PPC SPR's covers a range from 0 to 1023. Other SPR's are used to set BDI internal registers:

- 8001 For slow memory it may be necessary to increase the number of clocks used to execute a memory access cycle. If for example you cannot access boot ROM content with the default configuration of your memory controller, define additional memory access clocks with this SPR number in the init list. Usual values are in the range 1000 - 4000 (0x400 - 0x1000).
- 8002 Defines an alternate boot address. Normally a PPC boots from 0xFFF00100. A MPC8260 has also the option to boot from 0x00000100. The BDI needs to know the boot address in order to set the correct hardware breakpoint during startup.
- 8003 Defines the base address of the L3 cache private memory. Because L3 cache private memory cannot be accessed directly via JTAG, the BDI loads some support code into the workspace and uses it to access this memory range. Therefore a workspace is necessary to access this memory range.
- 8004 Defines the size of the L3 cache private memory in bytes (e.g. 0x100000 for 1Mbyte).

- 8006 Write to this special register a value of 1 if the BDI should use the alternate single step mode. The alternate mode does not use the trace bit (MSR[SE]) to implement single stepping. It uses always a hardware breakpoint (via IABR) on the next instruction to implement single stepping.
- 8007 Write to this special register a value of 1 if the BDI must not use burst reads when reading memory via COP. This will slow down memory read performance dramatically. Disabling burst reads maybe necessary if the memory controller does not support misaligned burst accesses.
- 8009 This entry in the init list allows to define a delay time (in ms) the BDI inserts between releasing the COP-HRESET line and starting communicating with the target. This init list entry may be necessary if COP-HRESET is delayed on its way to the PowerPC reset pin.
- 8010 If the BDI should forces the QACK pin (pin 2) on the COP connector low, write to this special register a value of 1. By default the QACK pin is not driven by the BDI. This option maybe useful for PPC750 and PPC7400 targets.

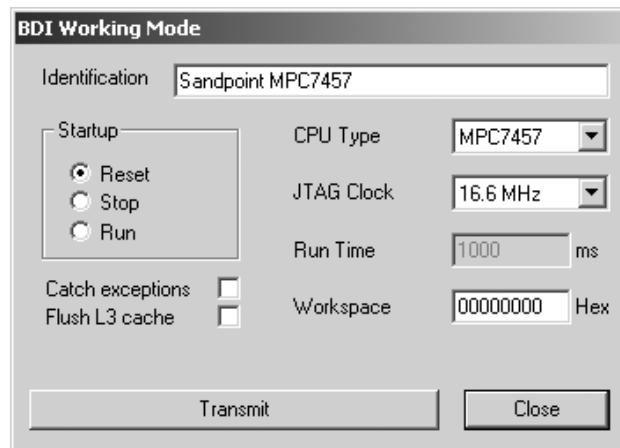
In order to support the dual core MPC8641D processor the following special configuration registers has been added. Read also the chapter about MPC8641D support.

- 8015 Selects the core (0 or 1) for the following init list entries.
- 8016 Defines the startup mode for the second core:
 - 0 = NONE Even when present, the second core is not handled by the BDI
 - 1 = HALT The second core is enabled and halted at the boot vector
 - 2 = STOP Then second core is stopped after the run time
 - 3 = RUN Then is assume to be present and let running after reset

The BDI can also handle systems with multiple devices connected to the JTAG scan chain. In order to put the other devices into BYPASS mode and to count for the additional bypass registers, the BDI needs some information about the scan chain layout. Enter the number and total instruction register (IR) length of the devices present before the PowerPC chip. Enter the appropriate information also for the devices following the PowerPC chip.

- 8011 Number of JTAG devices connected before the PowerPC chip.
- 8012 Total IR length of the JTAG devices connected before the PowerPC chip
- 8013 Number of JTAG devices connected after the PowerPC chip
- 8014 Total IR length of the JTAG devices connected after the PowerPC chip

4 BDI working modes



dialog box «BDI Working Mode»

With this dialog box you can define how the BDI interacts with the target system.

Identification	Enter a text to identify this setup.
Startup	Startup mode defines how the BDI interacts with the target processor after reset or power up. The options RESET, STOP or RUN can be selected.
CPU Type	Select the CPU family type of the target system.
JTAG Clock	This option allows to select the used JTAG clock rate.
Run Time	When startup mode STOP is selected, this option allows to set the run time after reset in milliseconds until the target CPU is stopped. Values from 100 (0.1 sec) till 32000 (32 sec) are accepted.
Workspace	In order to speed up code download, enter the address of a free 256 byte RAM area. The BDI will install there some code that supports faster program download. A value of 0xFFFFFFFF disables the workspace. The BDI also needs this workspace to flush the data cache and to access L2 private memory.
Catch exceptions	Check this switch if the BDI should catch unhandled exception. Catching exceptions is only possible if the memory at address 0x00000100 to 0x00001FFF is writable and the vector table is mapped to 0x00000000 (MSR[IP] = 0).
Flush L3 cache	Check this switch if the BDI should flush the target cache before accessing memory. This is mainly useful if there is an enabled L3 cache. If this switch is not set, the BDI uses L1/L2 cache coherent read and write accesses to target memory. Coherent access means, that the L1/L2 cache is directly read or written via COP if the appropriate cache line is valid. In order to flush the cache, the BDI needs some workspace in target RAM to execute the flush code.
Transmit	Click on this button to send the initialization list and the working mode to the BDI. This is normally the last step done before the BDI can be used with the debugging system.

4.1 Startup Mode

Startup mode defines how the BDI interacts with the target system after a reset or power up sequence.

4.1.1 Startup mode RESET

In this mode no ROM is required on the target system. The necessary initialization is done by the BDI with the programmed init list. The following steps are executed by the BDI after system reset or system power up:

- HRESET is activated on the target system.
- HRESET is deactivated and the target is forced into debug mode.
- The BDI works through the initialization list.

The RESET mode is the standard working mode. Other modes are used in special cases (i.e. applications in ROM, special requirements on the reset sequence...).

4.1.2 Startup Mode STOP

In this mode the initialization code is in a ROM on the target system. The code in this ROM handles base initialization. At the end of the code, the initialization program enters an endless loop until it is interrupted by the BDI. This mode is intended for special requirements on the reset sequence (e.g. loading a RAM based programmable logic device).

In this mode the following steps are executed by the BDI after system reset or power up:

- HRESET is activated on the target system.
- HRESET is deactivated and the target begins executing application code.
- After a delay (Run Time), the target is forced into debug mode.
- The BDI works through the initialization list.

4.1.3 Startup mode RUN

This mode is used to debug an application which is already stored in ROM. The application is started normally and will be stopped when the debugger is started.

In this mode, the following steps are executed by the BDI after system reset or power up:

- HRESET is activated on the target system.
- HRESET is deactivated and the target begins executing application code.
- The application runs until it is stopped by the debugger.

4.2 Dual-Core Support for MPC8641D

The BDI system supports debugging of the two e600 cores present in the MPC8641D. In the Telnet you switch between the cores with the command "select {0 | 1}".

In the configuration, you can switch between the cores with the special "WSPR 8015 n" init list entry. Then the following init list entries are directed to the selected core.

```

WSPR 8009          0x00000200  Wakeup Delay
WSPR 8016          0x00000001  Core#1: Startup Mode HALT
WSPR 8015          0x00000000  Select Core#0
WMSR              0x02003002  #0 MSR: set VEC,FP,ME,RI
WSPR 1008          0x84000000  #0 HID0: set TBEN
WSPR 8015          0x00000001  Select Core#1
WMSR              0x02003002  #1 MSR: set VEC,FP,ME,RI
WSPR 1008          0x84000000  #1 HID0: set TBEN
WSPR 8015          0x00000000  Select Core#0
WM32 0xFF700000    0x000F8000  CCSRBAR to 0xf8000000
NOP
WM32 0xF8000C08    0x00000000  LAWBAR0:
WM32 0xF8000C10    0x00000000  LAWBAR0 :
WM32 0xF8000C28    0x00000000  LAWBAR1: @0x00000000
WM32 0xF8000C30    0x80F0001D  LAWBAR1 : DDR/SDRAM 1024MB
. . . .

```

The BDI supports different startup modes. The startup mode for Core #0 is defined in the Mode dialog box. For Core#1 the special "WSPR 8016 mode" init list entry is used.

Because after reset Core#1 is disabled, the BDI writes to the MCMPCR and enables it in cases where HALT is selected as startup mode for Core#1. Following some examples:

HALT HALT

The second core will be enabled via MCMPCR and both core are halted at the reset vector via an IABR breakpoint.

RUN RUN

Both core are let running after reset. You can halt them individually via the Telnet "halt" command. The BDI does not write to MCMPCR. Halting the second core will only succeed if it has been enabled form the code running on the first core. The init list is not processed in this case.

STOP 4000 HALT

The second core will be enabled via MCMPCR and halted at the reset vector via an IABR breakpoint. The first core is let running for 4 seconds and then halted.

STOP 4000 STOP

Both core are let running after reset for 4 seconds and then halted. The BDI does not write to MCMPCR. Halting the second core will only succeed if it has been enabled form the code running on the first core during this 4 second runtime. After halting, the init list is processed.

HALT RUN

Useful if you want to debug boot code on core#0 but want to be able to access core#1 later. The BDI does not write to MCMPCR in this case.

RUN HALT

The first core is let running while the second core will be enabled via MCMPCR and halted at the reset vector via an IABR breakpoint.

Note about memory accesses via JTAG:

On MPC8641 targets, memory accesses are done via the so called "System Access Port" (SAP). The SAP is like an additional bus master. You can access memory while the core(s) are running and memory coherency is maintained because SAP accesses are snooped. This has the side effect that for example cache lines are flushed when memory is accessed via the BDI. Debugging code where data/stack is only present in the cache without real memory behind it becomes almost impossible.

The following Telnet sequence shows the effect on reading cached data via the BDI.

```
8641>dcache 0
W0 : 0_0010f000 VD 0010f000 0010f004 0010f008 0010f00c
                        0010f010 0010f014 0010f018 0010f01c
W1 : 0_0010b000 VD 0010b000 0010b004 0010b008 0010b00c
                        0010b010 0010b014 0010b018 0010b01c
W2 : 0_00111000 VD 00111000 00111004 00111008 0011100c
                        00111010 00111014 00111018 0011101c
W3 : 0_0010d000 VD 0010d000 0010d004 0010d008 0010d00c
                        0010d010 0010d014 0010d018 0010d01c

8641>md 0x00111000
0_00111000 : 00111000 00111004 00111008 0011100c .....
0_00111010 : 00111010 00111014 00111018 0011101c .....
.....
0_001110e0 : 001110e0 001110e4 001110e8 001110ec .....
0_001110f0 : 001110f0 001110f4 001110f8 001110fc .....

8641>dcache 0
W0 : 0_0010f000 VD 0010f000 0010f004 0010f008 0010f00c
                        0010f010 0010f014 0010f018 0010f01c
W1 : 0_0010b000 VD 0010b000 0010b004 0010b008 0010b00c
                        0010b010 0010b014 0010b018 0010b01c
W2 : 0_00111000 V- 00111000 00111004 00111008 0011100c
                        00111010 00111014 00111018 0011101c
W3 : 0_0010d000 VD 0010d000 0010d004 0010d008 0010d00c
                        0010d010 0010d014 0010d018 0010d01c
```

5 Working with Nucleus

For information about using the Nucleus debugger look at the appropriate Nucleus user's manual.

5.1 Direct Commands

For special functions (mainly for flash programming) the BDI supports so called «Direct Commands». This commands can be entered in a codelet file (e.g. PRELOAD.CDL) or directly executed in the Nucleus Debugger Command Line Window. This Direct Commands are not interpreted by the Nucleus Debugger but directly sent to the BDI. After processing the command the result is displayed in the Nucleus Debugger Command Line Window.

Direct Commands are ASCII - Strings with the following structure:

`<Object>.<Action> [<ParName>=<ParValue>]...`

Example:

`flash.erase addr=0x02800000`

All names are case insensitive. Parameter values are numbers or strings. Numeric parameters can be entered as decimal (e.g. 700) or as hexadecimal (0x80000) values.

If the commands are directly entered in the Nucleus Debugger Command Line Window, use the following syntax:

`bdi "direct-command"`

Example:

`bdi "flash.erase addr=0x02800000"`

5.1.1 Flash.Setup

In order to support loading into flash memory, the BDI needs some information about the used flash devices. Before any other flash related command can be used, this direct command must be executed.

Syntax:

```
flash.setup type=am29f size=0x80000 bus=32 workspace=0x1000
```

type	This parameter defines the type of flash used. It is used to select the correct programming algorithm. The following flash types are supported: AM29F, AM29BX8, AM29BX16, I28BX8, I28BX16, AT49, AT49X8, AT49X16, STRATAX8, STRATAX16, MIRROR, MORRORX8, MIRRORX16, I28BX32, AM29DX16, AM29DX32
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.
bus	The width of the memory bus that leads to the flash chips. Do not enter the width of the flash chip itself. The parameter TYPE 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.
workspace	If a workspace is defined, the BDI uses a faster programming algorithm that run 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.

5.1.2 Flash.Erase

This command allows to erase one flash sector, block or chip.

Syntax:

```
flash.erase addr=0x02800000 mode=chip
```

addr	The start address of the flash sector to erase.
mode	This parameter defines the erase mode. The following modes are supported: CHIP, BLOCK and SECTOR (default is sector erase)

5.1.3 Flash.Load

This command enables loading to flash memory. If the address of a data block is within the given flash range, the BDI automatically uses the appropriate programming algorithm. This command must be executed before downloading is started.

Syntax:

```
flash.load addr=0x02800000 size=0x200000
```

addr	The start address of the flash memory
size	The size of the flash memory

5.1.4 Flash.Idle

This command disables loading to flash memory.

Syntax:

```
flash.idle
```

5.2 Download to Flash Memory

The BDI supports programming flash memory. To automate the process of downloading to flash memory a codelet can be used. Following an example of such a codelet:

```
void flash_load(int coreId)
{
    char output[256];

    printf("Specifying the flash type...");
    command("bdi flash.setup type=AM29F size=0x00800000 bus=8",output, 256);
    printf("%s\n", output);
    printf("Erasing the first sector...");
    command("bdi flash.erase addr=0xffff0000 mode=sector", output,256);
    printf("%s\n", output);
    printf("Erasing the second sector...");
    command("bdi flash.erase addr=0xffff10000 mode=sector", output,256);
    printf("%s\n", output);
    printf("Erasing the third sector...");
    command("bdi flash.erase addr=0xffff20000 mode=sector", output,256);
    printf("%s\n", output);
    printf("Setting load address...");
    command("bdi flash.load addr=0xffff0000 size=0x00020000",output, 256);
    printf("%s\n", output);
    printf("Loading the image...");
    command("load C:\\MGC\\embedded\\Nucleus\\demo\\out\\plus_demo.out", output, 256);
    printf("%s\n", output);
    printf("Taking the BDI out of Flashing mode...");
    command("bdi flash.idle", output, 256);
    printf("%s\n", output);
}
```

A user who needs to reflash often can just call such a codelet from the Nucleus Debugger command view by typing `flash_load(1)` at the command prompt. For this to work two steps are required:

1. The codelet file must first be loaded into EGDE.
 - From the Run Menu select "Codelet Composer"
 - On the Codelet Composer dialog click the Load button.
 - Browse to and select your *.cdl file.
 - To complete the operation click the Open button.

Alternatively, any *.cdl file that is simply imported into one of the user's projects will be identified by Nucleus Debugger.

2. Since the flashing commands are issued over the debug connection, this of course requires that a connection to already been established to the target.

In addition, the contents of the codelet can be placed in the user's initialization codelet and thus be called automatically after connect.

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
- For 16/32 bit flash in 32bit mode: AM29DX32
- For 32bit only flash: I28BX32

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	-	-	I28BX32	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  0xFFFF0000  0x0060      unlock block 0
WM16  0xFFFF0000  0x00D0
WM16  0xFFFF1000  0x0060      unlock block 1
WM16  0xFFFF1000  0x00D0
      . . . .
WM16  0xFFFF0000  0xFFFF      select read mode
```

Not all flash chips support a chip erase command. Also if a chip erase takes too long, the BDI communication layer may time-out. In this case, use multiple sector erase commands

6 Telnet Interface

A Telnet server is integrated within the BDI that can be accessed when the BDI is connected via ethernet to the host. It may help to investigate problems and allows access to target resources that can not directly be accessed by the debugger.

The following commands are available:

```
"MD      [<address>] [<count>]  display target memory as word (32bit)",
"MDD     [<address>] [<count>]  display target memory as double word (64bit)",
"MDH     [<address>] [<count>]  display target memory as half word (16bit)",
"MDB     [<address>] [<count>]  display target memory as byte (8bit)",
"MM      <addr> <value> [<cnt>]  modify word(s) (32bit) in target memory",
"MMD     <addr> <value> [<cnt>]  modify double word(s) (64bit) 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",
"MC      [<address>] [<count>]  calculates a checksum over a memory range",
"MV      verifies the last calculated checksum",
"RD      display general purpose registers",
"RDSPR <number>                display special purpose register",
"RDSR    <number>                display segment register",
"RDVR    [<number>]              display vector register",
"RM      <number> <value>        modify general purpose or user defined register",
"RMSPR   <number> <value>        modify special purpose register",
"RMSR    <number> <value>        modify segment register",
"RMVR    <nbr><val val val val>  modify vector register (four 32bit values)",
"DCACHE <addr | set>            display L1 data cache content",
"L2CACHE <addr | set>          display L2 cache content",
"DTAG    <from> [<to>]           display L1 DTAG values",
"ITAG    <from> [<to>]           display L1 ITAG values",
"BOOT    reset the BDI and reload the configuration",
"RESET   reset the target system",
"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      <addr> [v]              set instruction hardware breakpoint",
"CI      [<id>]                  clear instruction hardware breakpoint(s)",
"BD      [R|W] <addr>            set data watchpoint via DABR",
"BDT     [R|W] <addr>            set data watchpoint via DABR (DABR[BT]=1)",
"CD      [<id>]                  clear data watchpoint(s)",
"INFO    display information about the current state",
"DCMD    <direct command>       execute a BDI direct command (see manual)",
"SELECT  <core>                 change the current core",
"HELP    display command list",
"QUIT    terminate the Telnet session"
```

7 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. 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

8 Environmental notice

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

9 Declaration of Conformity (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

10 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 36 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/repair 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 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 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)

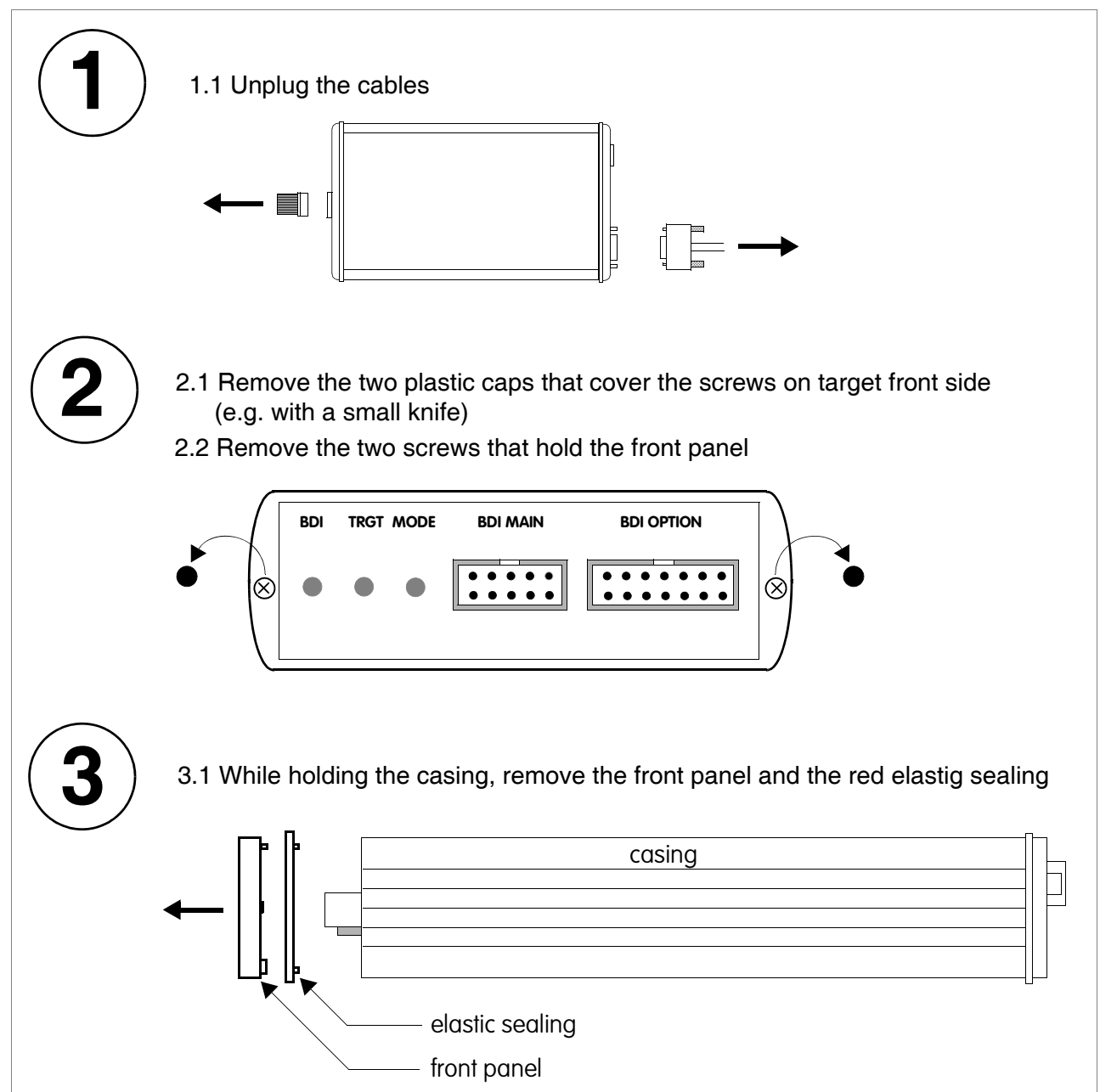
B Maintenance

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:

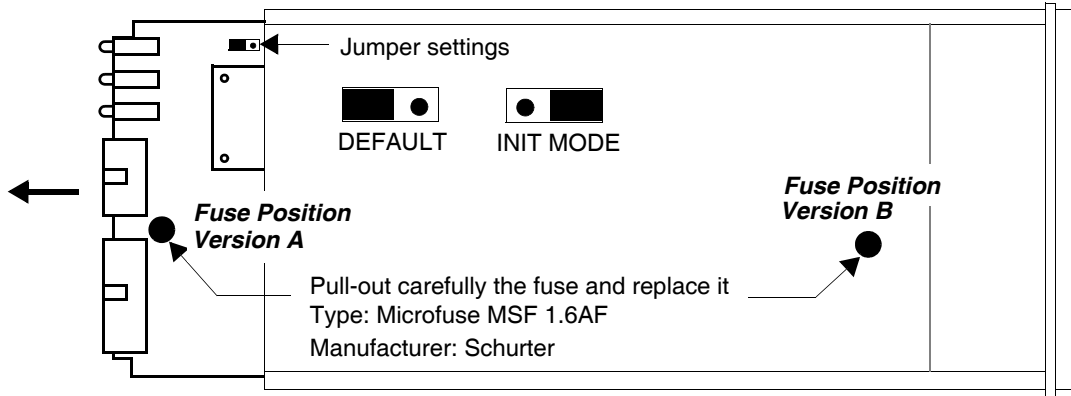


Observe precautions for handling (Electrostatic sensitive device)
Unplug the cables before opening the cover.
Use exact fuse replacement (Microfuse MSF 1.6 AF).



4

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

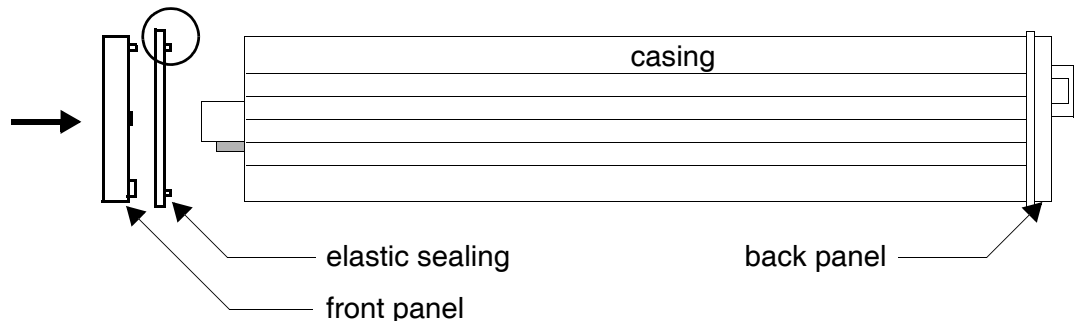


5

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).**

C Trademarks

All trademarks are property of their respective holders.