bdiNDI

BDM interface for Nucleus™ Debugger

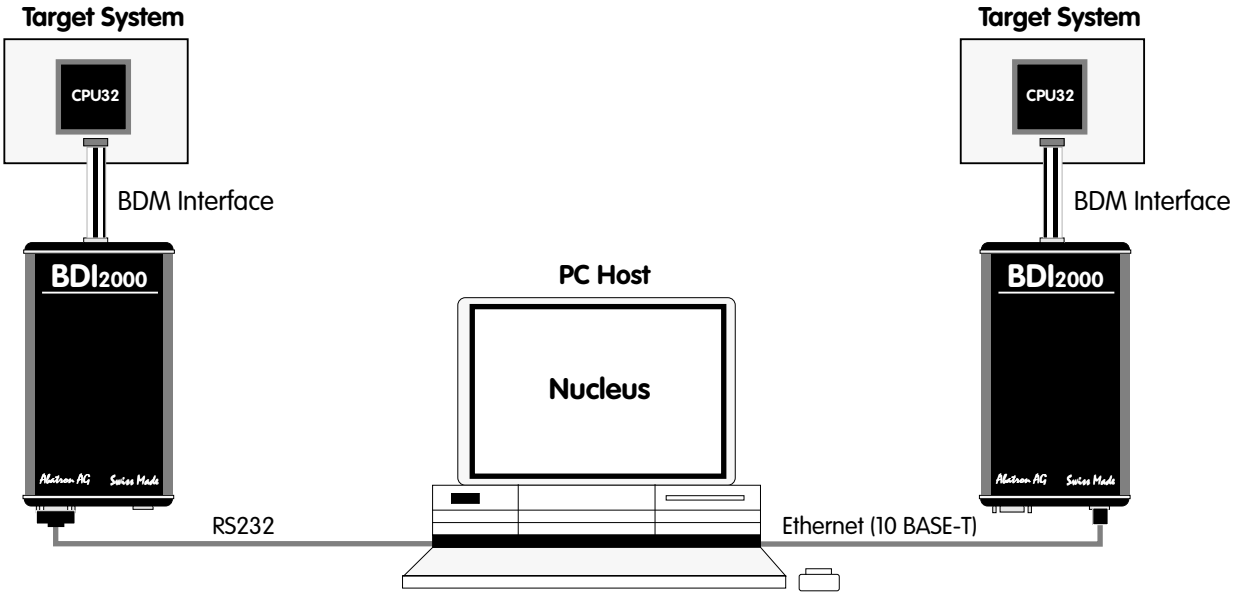
CPU32/32+

User Manual

Manual Version 1.00 for BDI2000

Appendices

1 Introduction



The BDI2000 adds Background Debug Mode features to the Nucleus debugger environment from Mentor Graphic's. 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 Nucleus 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).

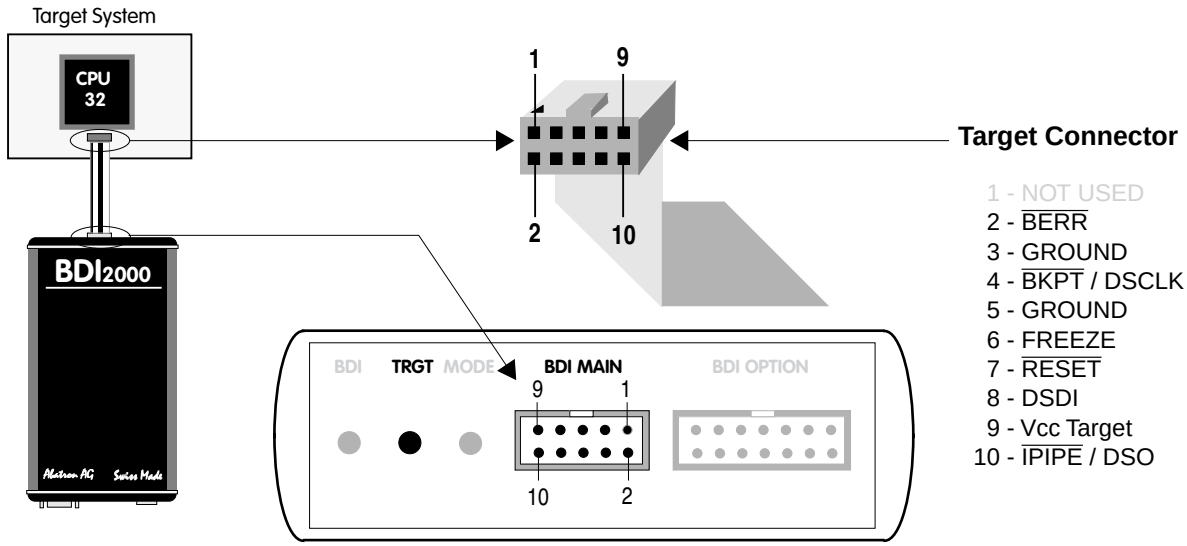
2 Installation

2.1 Connecting the BDI2000 to Target

The cable to the target system is a 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.

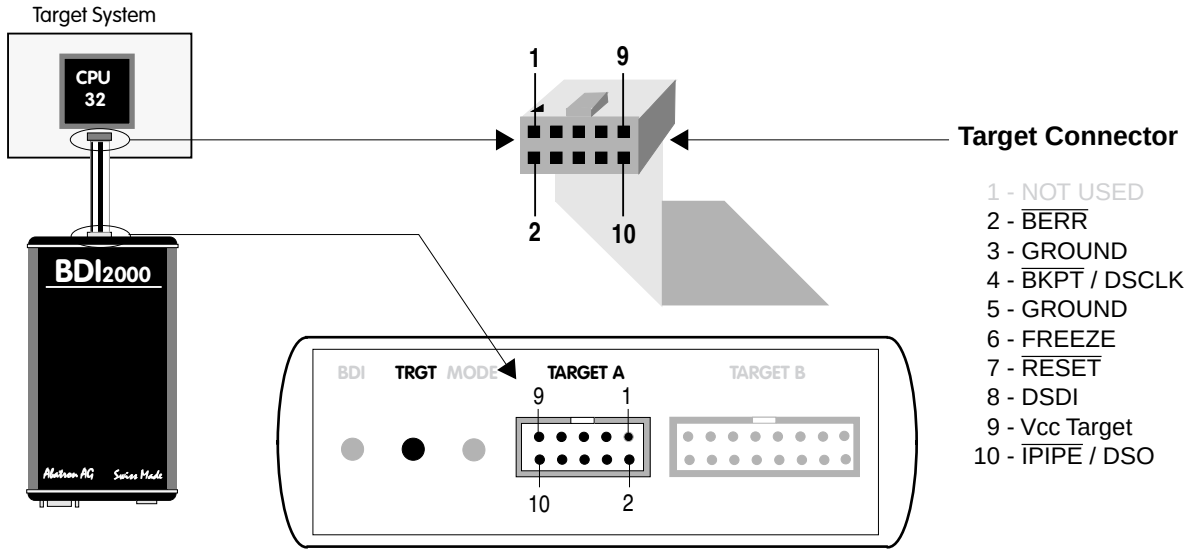
In order to ensure reliable operation of the BDI (EMC, runtimes, etc.) the target cable length must not exceed 20 cm (8").

Rev. A



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

Reb. B/C



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

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

BDI MAIN / TARGET A Connector Signals:

Pin	Name	Describtion
1	---	Not used.
2	$\overline{\text{BERR}}$	**BUS ERROR** Active-low input to the MCU. Signals an invalid bus operation attempt.
3+5	GROUND	**SYSTEM GROUND**
4	$\overline{\text{BKPT}}$ / DSCLK	**BREAKPOINT** For normal modes, active-low input to the MCU. Signals a hardware breakpoint. **DEVELOPMENT SERIAL CLOCK** For background debug mode, serial input clock signal to the MCU.
6	FREEZE	**FREEZE** Active-high output from the MCU. Indicates that the MCU has acknowledged a breakpoint and that it has entered background debug mode.
7	$\overline{\text{RESET}}$	**RESET** Active-low, open-drain, signal to start a system reset.
8	$\overline{\text{IFETCH}}$ / DSI	**INSTRUCTION FETCH** For normal modes, output signal from the MCU. Indicates instruction pipeline activity. **DATA SERIAL IN** For background debug mode, serial data input signal to the MCU.
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	$\overline{\text{IPIPE}}$ / DSO	**INSTRUCTION PIPE** For normal modes, output signal from the MCU. Indicates instruction pipeline activity. **DATA SERIAL OUT** For background debug mode, serial data output from the MCU.

All the pins except pin 1need to be connected to the target system for the debug operation.

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.

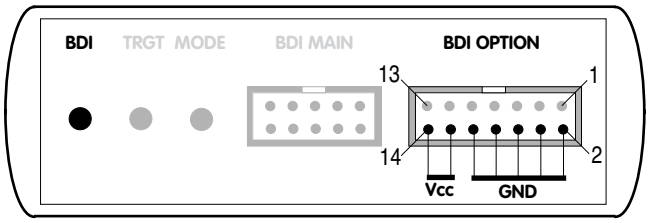
2.2 Connecting the BDI2000 to Power Supply

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

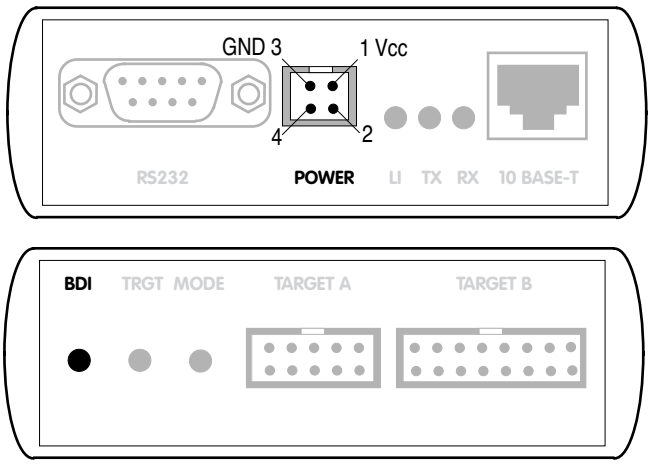


The green LED «BDI» marked light up when 5V power is connected to the BDI2000

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)

Rev. B/C



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

POWER Connector

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

Please switch on the system in the following sequence:

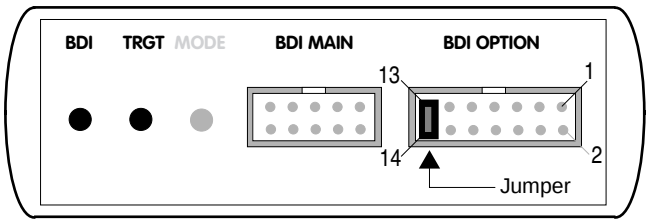
- 1 --> external power supply
- 2 --> target system

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

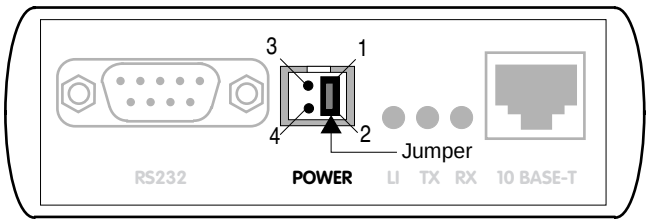


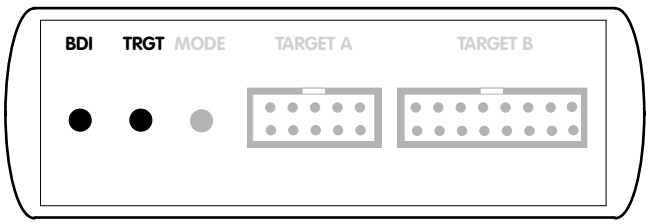
The green LEDs «BDI» and «TRGT» marked light up when target is powered up and the jumper is inserted correctly

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





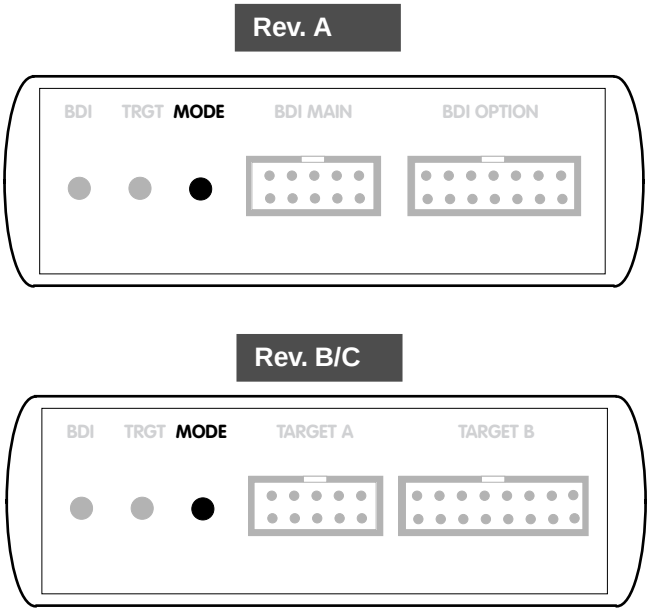
The green LEDs «BDI» and «TRGT» marked light up when target is powered up and the jumper is inserted correctly

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

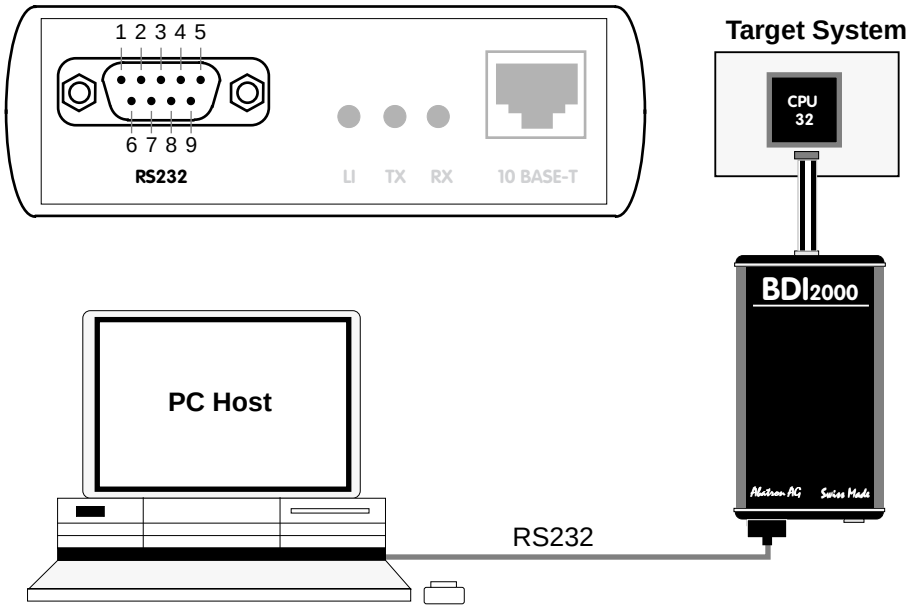
2.4.1 Serial line communication

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).

Rev. A

RS232 Connector
(for PC host)

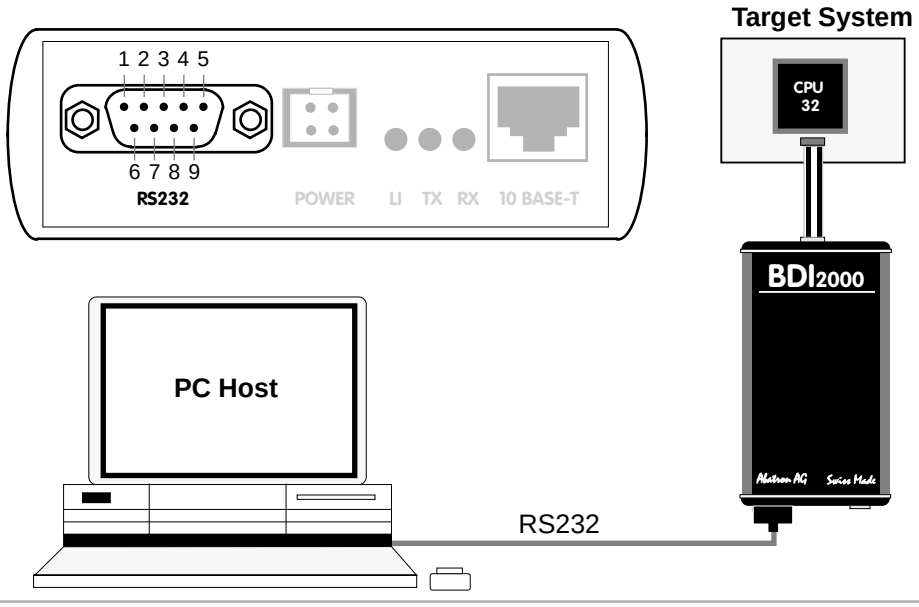
1 - NC
2 - RXD data from host
3 - TXD data to host
4 - NC
5 - GROUND
6 - NC
7 - NC
8 - NC
9 - NC



Rev. B/C

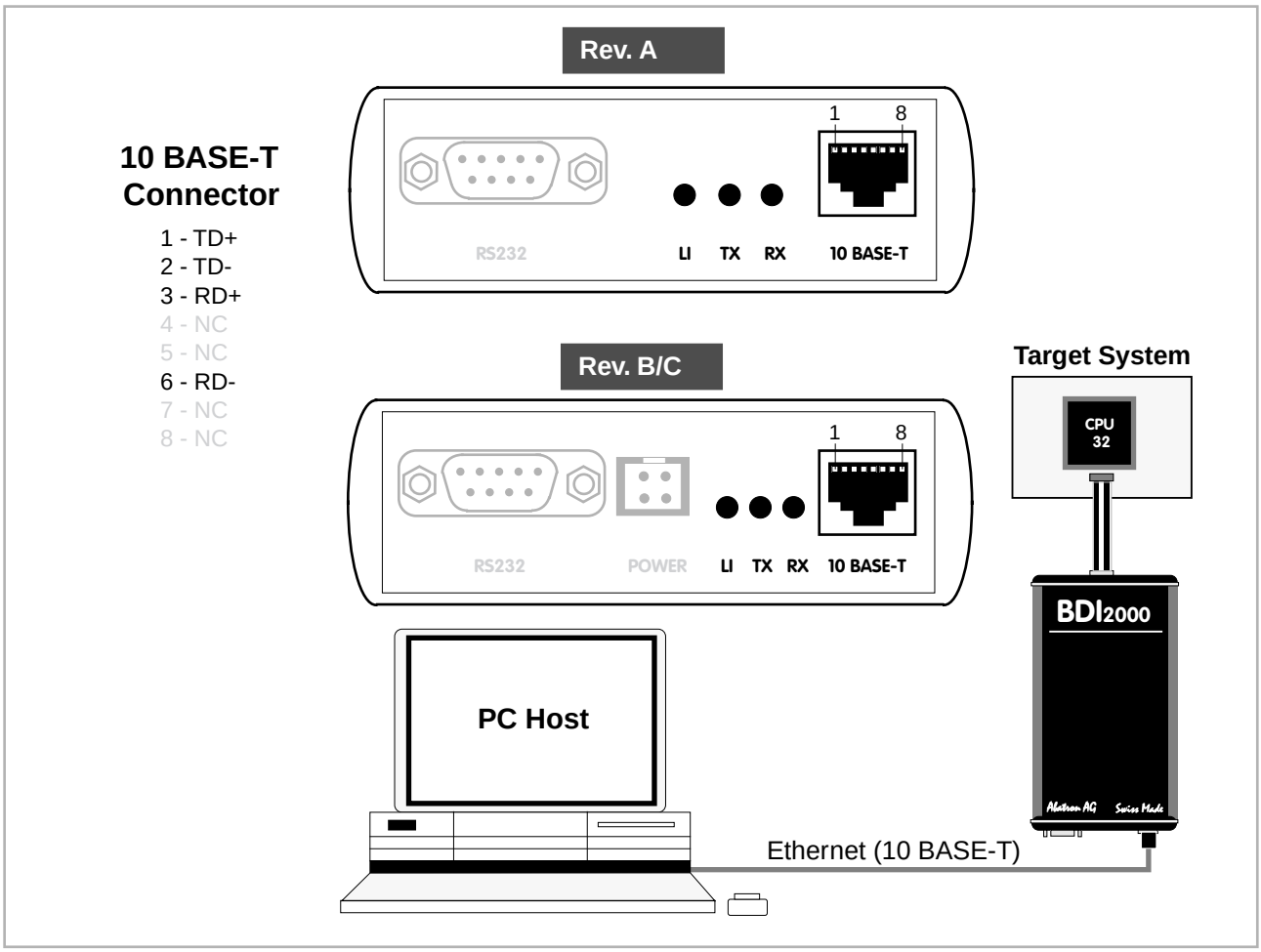
RS232 Connector
(for PC host)

1 - NC
2 - RXD data from host
3 - TXD data to host
4 - NC
5 - GROUND
6 - NC
7 - NC
8 - NC
9 - NC



2.4.2 Ethernet communication

The BDI2000 has a built-in 10 BASE-T Ethernet interface (see figure below). Connect an UTP (Unshilded Twisted Pair) cable to the BD2000. 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:

b20c32.exe	Configuration program
b20c32.hlp	Helpfile for the configuration program
b20c32.cnt	Help contents file
b20c32fw.xxx	Firmware for BDI2000 for CPU32 targets
c32jed20.xxx	JEDEC file for the BDI2000 (Rev. A/B) logic device programming
c32jed21.xxx	JEDEC file for the BDI2000 (Rev. A/B) logic device programming
bdiifc32.dll	BDI Interface DLL for configuration program
*.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 b20c32.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*

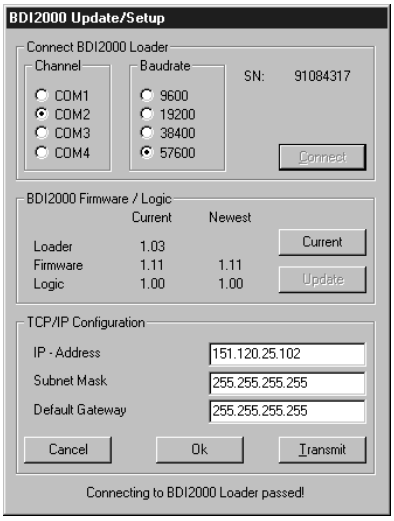
For information about the dialogs and menus use the help system (F1).

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:

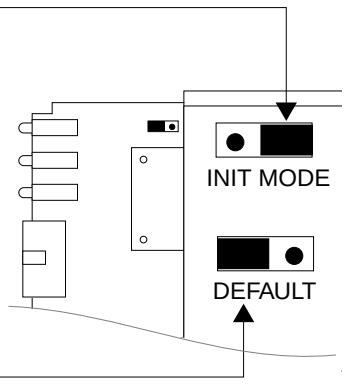
Channel — Select the communication port where the BDI2000 is connected during this setup session.

Baudrate — Select the baudrate used to communicate with the BDI2000 loader during this setup session.

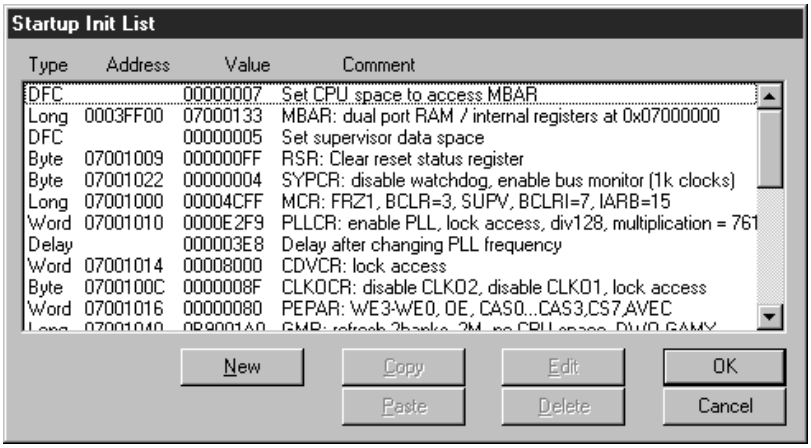
Connect	Click on this button to establish a connection with the BDI2000 loader. Once connected, the BDI2000 remains in loader mode until it is restarted or this dialog box is closed.
Current	Press this button to read back the current loaded BDI2000 software and logic versions. The current loader, firmware 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.xxxe.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.xxxe.g.255.255.255.0 A subnet mask of 255.255.255.255 disables the gateway feature. Ask your network administrator for the correct subnet mask.
Default Gateway	Enter the IP address of the default gateway. Ask your network administrator for the correct gateway IP address. If the gateway feature is disabled, you may enter 255.255.255.255 or any other value..
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

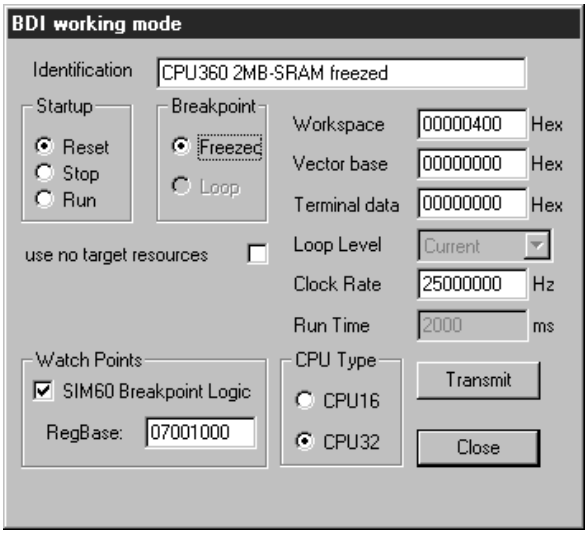


dialog box «Startup Init List»

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.

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. This text can be read by the debugger with the appropriate Command.
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.
Breakpoint	Breakpoint mode defines how breakpoints are processed. The target processor may be frozen (FREEZED option) or may be set to loop in an exception procedure (LOOP option) while the application software is halted.
CPU Type	Select the appropriate CPU type. For CPU32+ select CPU32. For Nucleus only CPU32/32+ is supported.
CPU Clock Rate ...	Enter the clock rate which the target CPU runs, after BDI has worked through the init list. BDI selects the BDM communication speed based on this parameter. If this parameter selects a CPU clock rate that is higher than the real clock, BDM communication may fail. When selecting a clock rate slower than possible, BDM communication still works but not as fast as possible.
Workspace	In all configurations except when «use no target resources» is activated, BDI needs some target memory space. Enter here the start address of this memory area. A maximum of 512 bytes is needed.
Vector base	The BDI needs to know where the vector table is located. Enter here the start address of the vector table. This address is automatically loaded into the VBR register at startup time. The application should not change the VBR unless «use no target resources» is selected.
Terminal Data	Together with Nucleus, the terminal function is not supported. Enter a value of 00000000 in any case. This disables the terminal function.

Loop Level	Selects the priority level (interrupt priority mask) the application uses when halted in LOOP mode. A level of 7 disables all interrupts when the application is halted. The value CURRENT(default) means, the application loops with the level currently active at the point where it is stopped.
use no target res...	Check this switch if the BDI should not use any RAM or vectors in the target system. This option is only enabled when FREEZED is selected as breakpoint mode. This mode is suitable for testing new hardware or debugging custom exception routines.
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.
SIM60 Breakpoint	The internal breakpoint logic in a CPU with a SIM60 module (e.g. MC68360) is supported and can be used to set Breakpoints on variable accesses or to set code breakpoints even when the code runs out of a read only memory (e.g. Flash Device). Check this switch, if the SIM60 breakpoint logic is present and should be used to support hardware breakpoints. Only one hardware breakpoint can be set at the same time. In the special case, when also «use no target resources» is checked, every code breakpoint uses the SIM60 hardware. In this mode it is easy to work through the code (even when stored in ROM) by simple mouse clicks to the source code.
RegBase	Enter the base address of the SIM60 registers. For example, enter 101000 if MBAR points to 100000 (all numbers in hexadecimal).

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:

- RESET and BKPT are activated on the target system.
- RESET is deactivated and the target system changes to background debug mode. If the target system does not change to background debug mode, the BDI waits for a short time and then activates the BERR line.
- Register DFC and SFC are set to supervisor data space.
- The BDI works through the initialization list and writes to the corresponding addresses.
- Depending on the break mode, the necessary vectors are set and help code is written into the RAM on the target system.

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 and sets the stackpointer. 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 or, if, for example, separate hardware needs to be initialized immediately.

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

- RESET and BKPT are activated on the target system.
- RESET is deactivated and the target system changes to background debug mode. If the target system does not change to background debug mode, the BDI waits for a short time and then activates the BERR line.
- Register DFC and SFC are set to supervisor data space.
- The target CPU is started (the target starts at the address fetched when reading the start vector at address 0).
- The target system is working through the application code.
- After the programmed run time, BKPT is activated and the target system changes to background debug mode. If the target system does not change to background debug mode, the BDI waits for a short time and then activates the BERR line.
- The BDI works through the initialization list and writes the corresponding addresses.
- Depending on the break mode, the necessary vectors are set and support code is written into the RAM on the target system.

4.1.3 Startup mode RUN

This mode is used to debug applications which are already stored in ROM. The application is started normally and is stopped by the BDI as soon as the debugger connects to the BDI.
In this mode, the following steps are executed by the BDI after system reset or power up:

- RESET and BKPT are activated on the target system.
- RESET is deactivated and the target system changes to background debug mode. If the target system does not change to background debug mode, the BDI waits for a short time and then activates the BERR line.
- Register DFC and SFC are set to supervisor data space.
- The target CPU is started (the target starts at the address fetched when reading the start vector at address 0).
- The target system is executing the application code.
- The application runs until the debugger stops the execution.
- BKPT is activated on the target system, and the target system changes to background debug mode. If the target system does not change to background debug mode, the BDI waits for a short time and then activates the BERR line.
- Depending on the break mode, the necessary vectors are set and help code is written into the RAM on the target system.

4.2 Breakpoint Mode

The use of breakpoints is only possible if the application code is stored in RAM (not in ROM) on the target system. Depending on the selected breakpoint mode, breakpoint and single step functions are implemented total differently.

4.2.1 Breakpoint Mode FREEZED

In this mode breakpoints are implemented by replacing application code with the BGND instruction. All the time the application is halted (i.e. caused by a breakpoint) the target processor remains freezed.

Single step is implemented by starting the target processor and immediately activating the BKPT line. So the target processor will be frozen after executing exactly one instruction. All interrupts are disabled during the execution of this single instruction. The status register (SR) is restored to the correct value after single stepping an instruction. The BDI takes care of instructions that changes the SR (e.g. RTE, MOVE to SR, ANDI to SR, ...) or writes the SR value to memory / register (e.g. MOVE from SR, TRAP, ...).

Note when using SIM60 breakpoint logic:
If the SIM60 breakpoint hardware is enabled and also «use no target resources» is checked, only hardware breakpoints will be used. There is no code replacement and therefore, the code may be stored in read only memory. Only one breakpoint can be active at the same time.

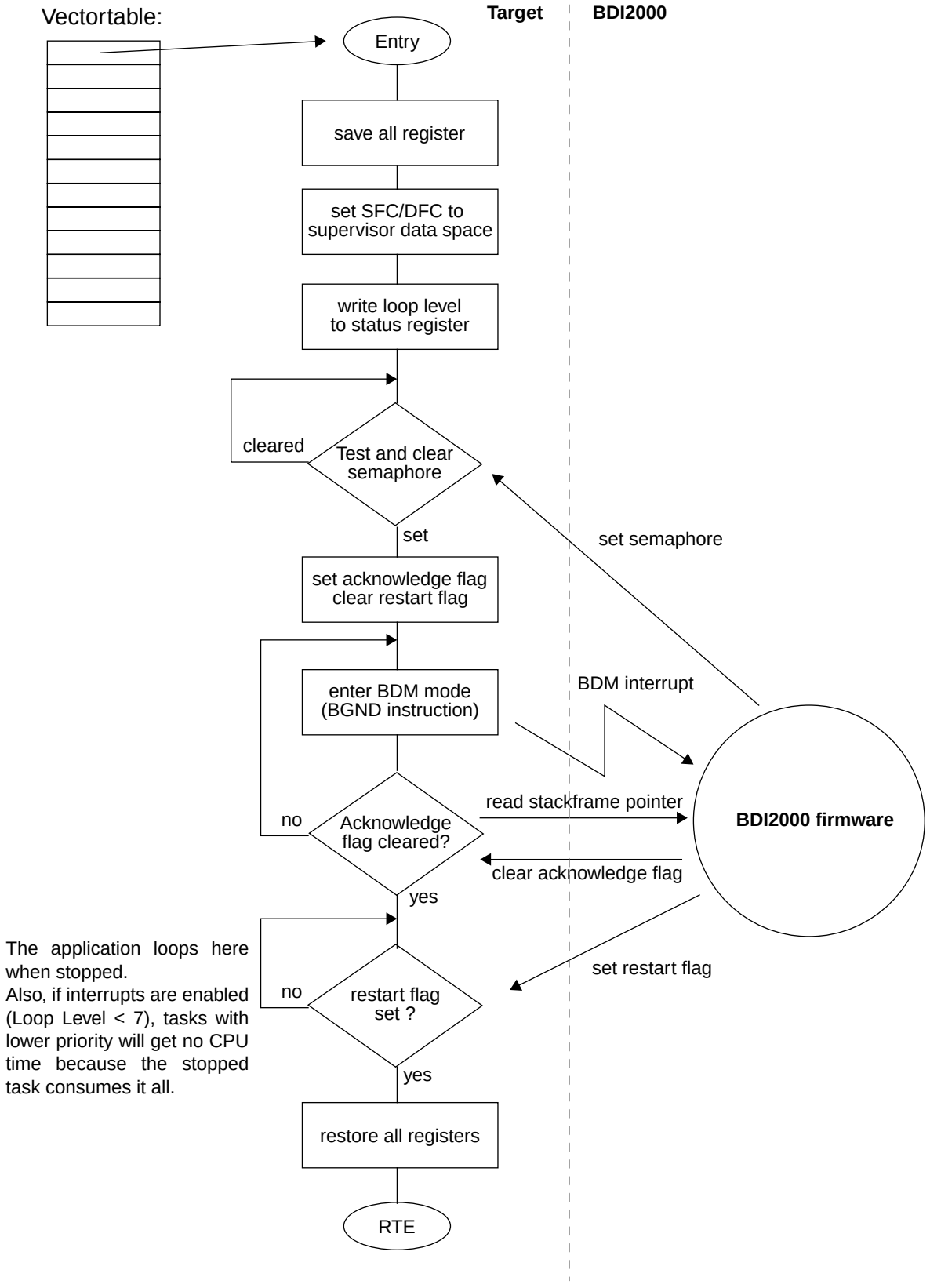
4.2.2 Breakpoint Mode LOOP

In this mode breakpoints are implemented by replacing application code with a TRAP#15 instruction. A stopped application loops within an exception procedure. The target processor is freezed only for short time periods (max 70us). The priority level used when looping in the exception procedure can be defined using the option «Loop Level». If you want to stop the hole application use Loop Level 7. If only the current task should be stopped, use Loop Level CURRENT.

Single step is implemented by setting the Trace bit in the processor status register. So a single step steps always over the current instruction. If interrupts are pending, they are served first without stopping the target processor.

In order to halt the application TRAP #1 is used.

The following diagram shows the used universal exception procedure for CPU32 targets. May be this helps you to understand how BDI works in LOOP mode.



4.3 Workspace

Depending on the working mode, the BDI needs some RAM in the target system. The following table shows how much RAM is used in the different modes.

Mode	Workspace (Bytes)	Remark
RESET/FREEZED	4	needed to trap the exceptions
STOP/FREEZED	4	needed to trap the exceptions
RUN/FREEZED	4	needed to trap the exceptions
RESET/LOOP	512	used for standard exception procedure and the initial (supervisor) stack
STOP/LOOP	256	used for standard exception procedure
RUN/LOOP	256	used for standard exception procedure

Vectors in RUN mode:

When RUN mode is selected, only specific vectors are initialized when the application is halted for the first time .

CPU32 0...9, 12...24 (and 33, 47 in LOOP mode)

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 (see also appendix C): AM29F, AM29BX8, AM29BX16, I28BX8, I28BX16
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.

Syntax:

flash.erase addr=0x02800000

addr	The start address of the flash sector to 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=0xfff00000 mode=sector", output,256);
 printf("%s\n", output);
 printf("Erasing the second sector...");
 command("bdi flash.erase addr=0xfff10000 mode=sector", output,256);
 printf("%s\n", output);
 printf("Erasing the third sector...");
 command("bdi flash.erase addr=0xfff20000 mode=sector", output,256);
 printf("%s\n", output);
 printf("Setting load address...");
 command("bdi flash.load addr=0xfff00000 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 2 standard flash algorithm supported. The AMD and Intel 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, select: AM29F, I28BX8

For 8/16 bit flash in 8bit mode, select: AM29BX8, I28BX8

For 8/16 bit flash in 16bit mode, select: AM29BX16, I28BX16

For 16bit only flash, select: AM29BX16, I28BX16

The following table shows some examples:

Flash	x 8	x 16	Chipsize
Am29F010	AM29F	-	0x020000
Am29F800B	AM29BX8	AM29BX16	0x100000
Am29DL323C	AM29BX8	AM29BX16	0x400000
Intel 28F032B3	I28BX8	-	0x400000
Intel 28F640J3A	I28BX8	I28BX16	0x800000
Intel 28F320C3	-	I28BX16	0x400000

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

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

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 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

7 Environmental notice

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

8 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

9 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 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

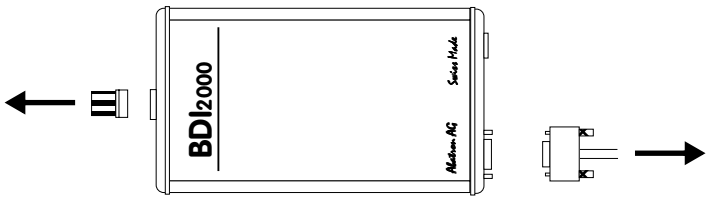
The BDI needs no special maintenance. Clean the housing with a mild detergent only. Solvents such as gasoline may damage it.

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).

1

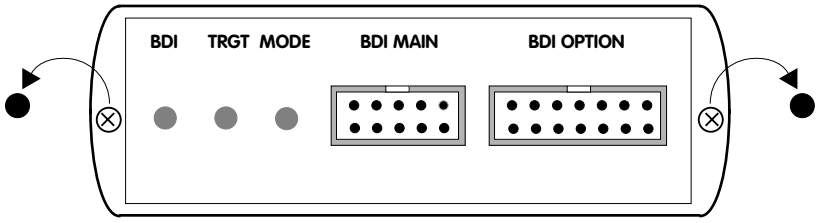
1.1 Unplug the cables



2

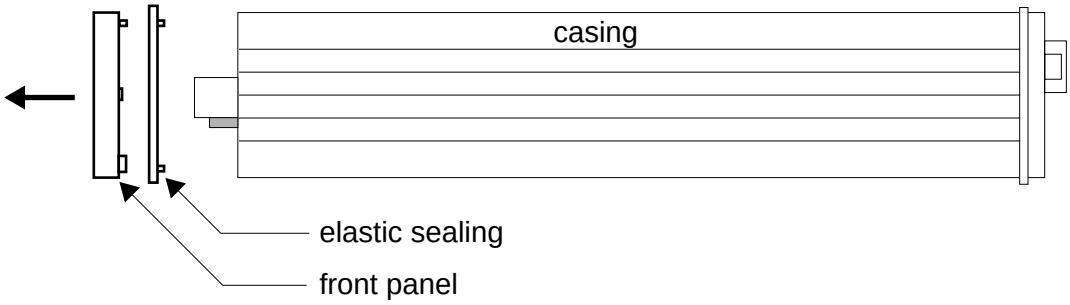
2.1 Remove the two plastic caps that cover the screws on target front side (e.g. with a small knife)

2.2 Remove the two screws that hold the front panel



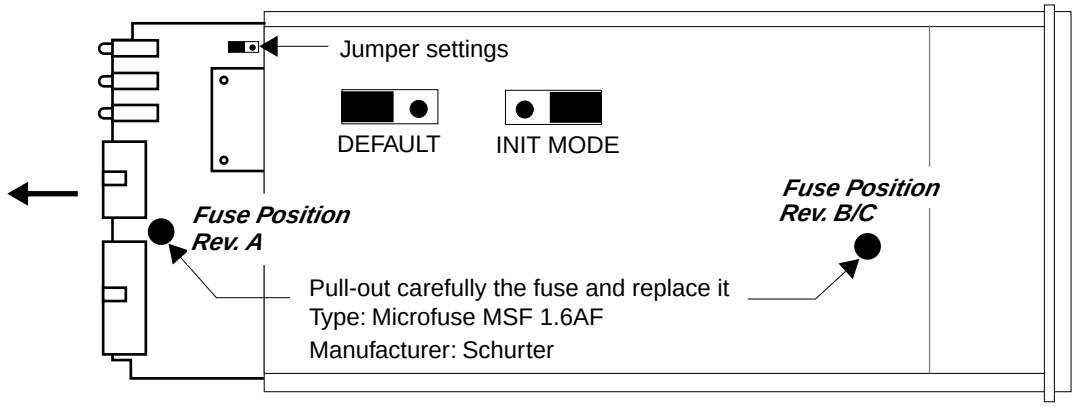
3

3.1 While holding the casing, remove the front panel and the red elastig sealing



4

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

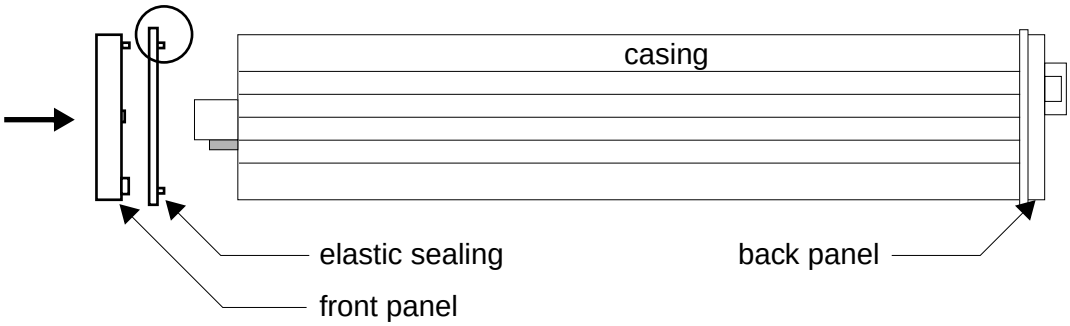


5

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 elastig 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.

