

*bedi*RDI

JTAG debug interface for RDI compatible debuggers

*XScale*



# User Manual

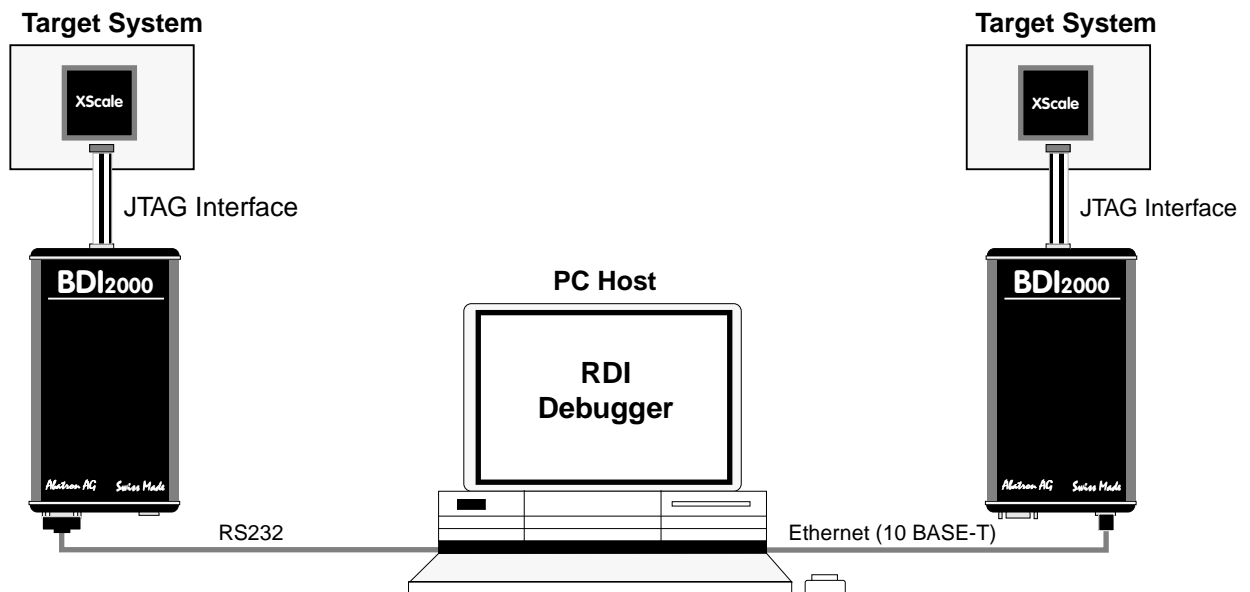
Manual Version 1.01 for BDI2000



©1999-2004 ABATRON AG

<b>1 Introduction .....</b>	<b>3</b>
1.1 BDI2000.....	3
<b>2 Installation .....</b>	<b>4</b>
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
2.2.1 External Power Supply .....	7
2.2.2 Power Supply from Target System .....	8
2.3 Status LED «MODE» .....	9
2.4 Connecting the BDI2000 to Host .....	10
2.4.1 Serial line communication .....	10
2.4.2 Ethernet communication .....	11
2.5 Installation of the Configuration Software .....	12
2.6 Configuration .....	13
2.6.1 BDI2000 Setup/Update .....	13
<b>3 Init List.....</b>	<b>15</b>
<b>4 BDI working modes.....</b>	<b>16</b>
4.1 Debug Handler.....	17
4.2 Startup Mode .....	18
4.2.1 Startup mode RESET .....	18
4.2.2 Startup Mode STOP .....	18
4.2.3 Startup mode RUN.....	18
<b>5 Working with RDI Debuggers .....</b>	<b>19</b>
5.1 ADW/AXD from ARM Ltd.....	19
5.1.1 Configuration.....	19
5.2 BDI Direct Commands.....	20
5.2.1 Target.Reset .....	20
5.2.2 Flash.Setup .....	20
5.2.3 Flash.Erase .....	21
5.2.4 Flash.Load .....	21
5.2.5 Flash.Idle.....	21
5.3 Download to Flash Memory .....	22
<b>6 Telnet Interface .....</b>	<b>24</b>
<b>7 Specifications .....</b>	<b>25</b>
<b>8 Environmental notice .....</b>	<b>26</b>
<b>9 Declaration of Conformity (CE).....</b>	<b>26</b>
<b>10 Warranty .....</b>	<b>27</b>
 <b>Appendices</b>	
<b>A Troubleshooting .....</b>	<b>28</b>
<b>B Maintenance .....</b>	<b>29</b>
<b>C Trademarks .....</b>	<b>31</b>

# 1 Introduction



The BDI2000 adds JTAG based debugging to RDI compatible debuggers (e.g. AXD from ARM Ltd). 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 RDI compatible 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. A/B).

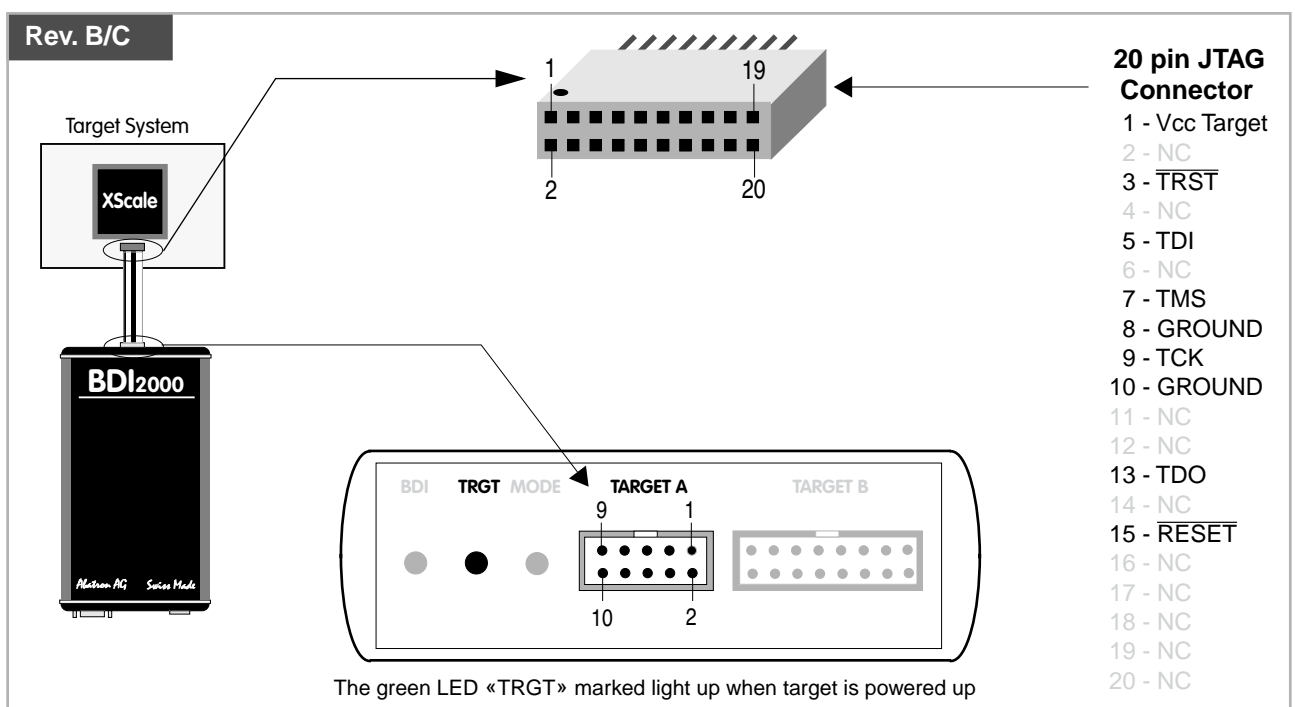
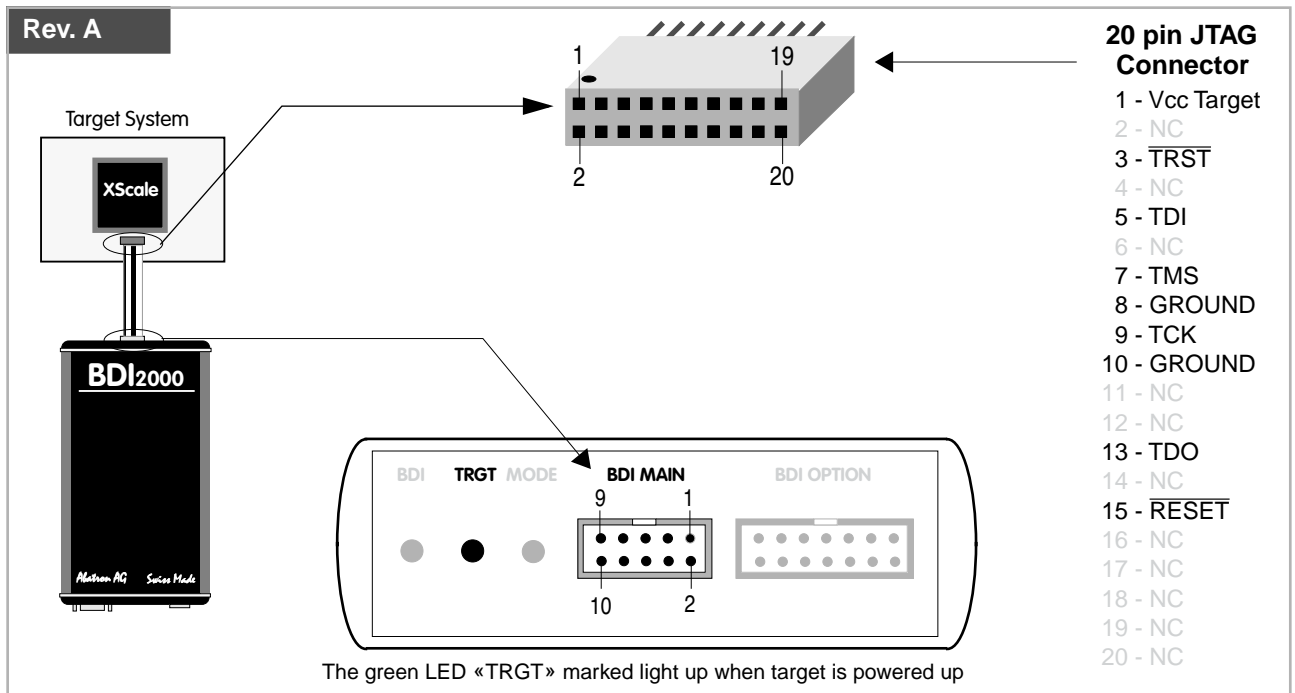
## 2 Installation

### 2.1 Connecting the BDI2000 to Target

The enclosed target cable is designed for the Intel recommended 20pin JTAG connector. In case where the target system has an appropriate connector, the cable can be directly connected. The pin assignment is in accordance with the Intel specification.



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



### BDI MAIN / TARGET A Connector Signals

Pin	Name	Description
1	reserved	This pin is currently not used.
2	TRST	<b>JTAG Test Reset</b> This output of the BDI2000 resets the JTAG TAP controller on the target.
3+5	GND	<b>System Ground</b>
4	TCK	<b>JTAG Test Clock</b> This output of the BDI2000 connects to the target TCK line.
6	TMS	<b>JTAG Test Mode Select</b> This output of the BDI2000 connects to the target TMS line.
7	RESET	This open collector output of the BDI2000 is used to reset the target system.
8	TDI	<b>JTAG Test Data In</b> This output of the BDI2000 connects to the target TDI line.
9	Vcc Target	<b>1.8 – 5.0V:</b> 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 connected to Vdd I/O on the target board.  <b>3.0 – 5.0V with Rev. A/B :</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	TDO	<b>JTAG Test Data Out</b> This input to the BDI2000 connects to the target TDO line.

**Note:**

The BDI actively drives TRST. It does not require any special power-up circuitry. Simply, the requirement is that TRST is weakly pulled down at the processor. It is suggested that the value of the pull-down resistor is 10k or greater.

### **2.1.1 Changing Target Processor Type**

Before you can use the BDI2000 with an other target processor type (e.g. ARM <--> 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 - BREAK-ENA
- 6 - GROUND
- 7 - NOT USED
- 8 - GROUND
- 9 - BREAK-A
- 10 - GROUND
- 11 - BREAK-B
- 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 - BREAK-ENA
- 6 - GROUND
- 7 - NOT USED
- 8 - GROUND
- 9 - BREAK-A
- 10 - GROUND
- 11 - BREAK-B
- 12 - Vcc (+5V)
- 13 - Vcc Target (+5V)
- 14 - Vcc BDI2000 (+5V)

**Rev. B/C**

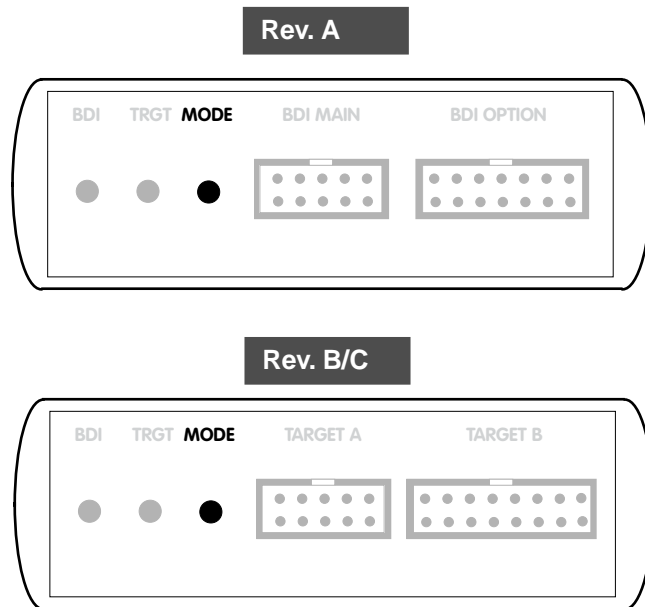
The green LEDs «BDI» and «TRGT» marked light up when target is powered up and the jumper is inserted correctly

**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:

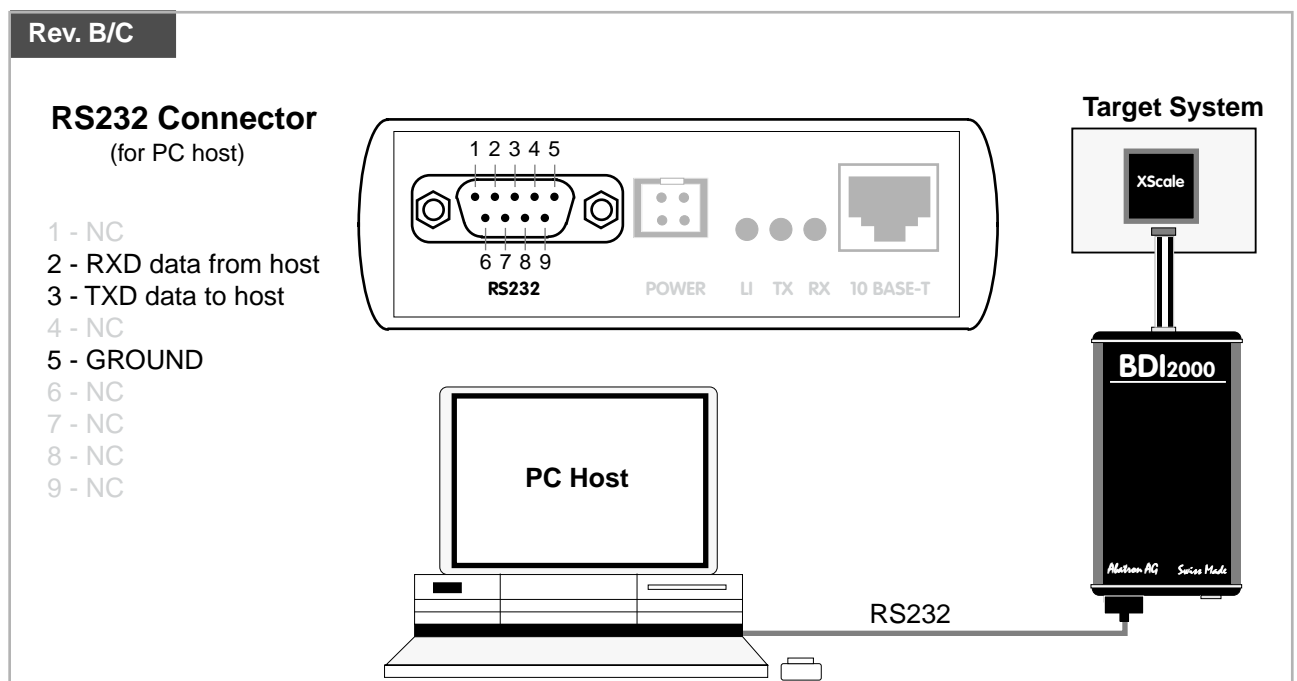
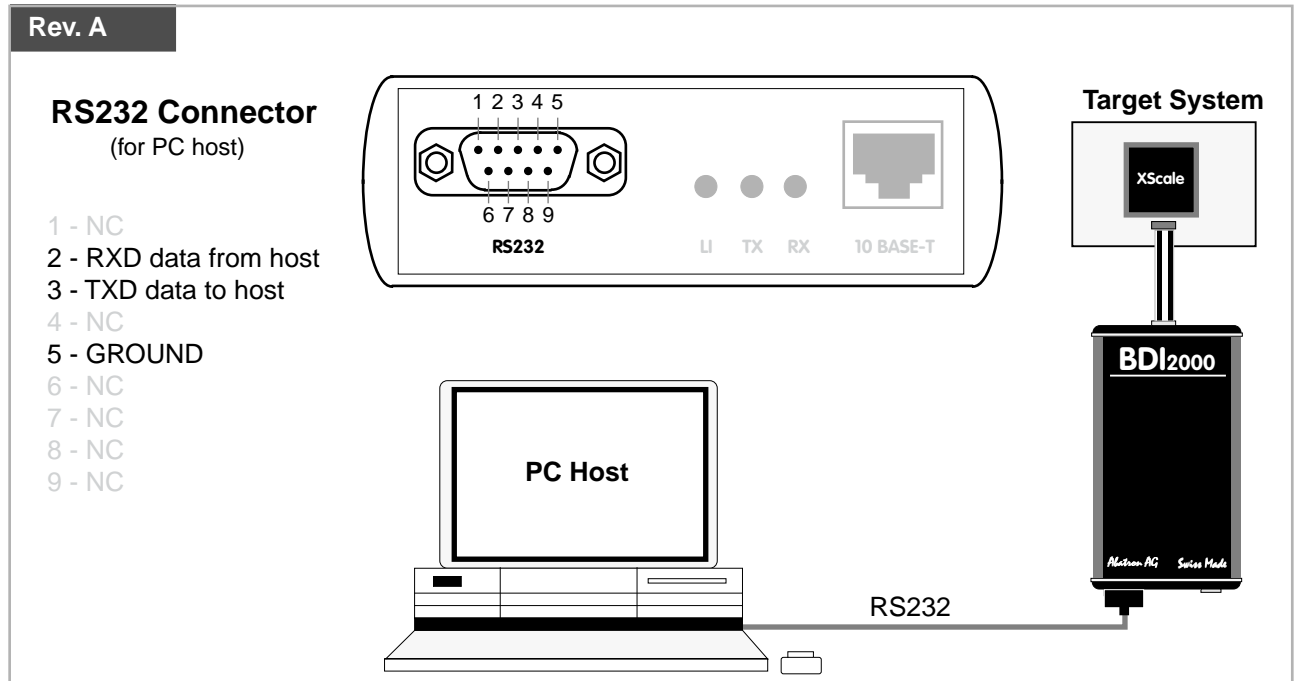


MODE LED	BDI STATES
OFF	The BDI is ready for use, the firmware is already loaded.
ON	The power supply for the BDI2000 is < 4.75VDC.
BLINK	The BDI «loader mode» is active (an invalid firmware is loaded or loading firmware is active).

## 2.4 Connecting the BDI2000 to Host

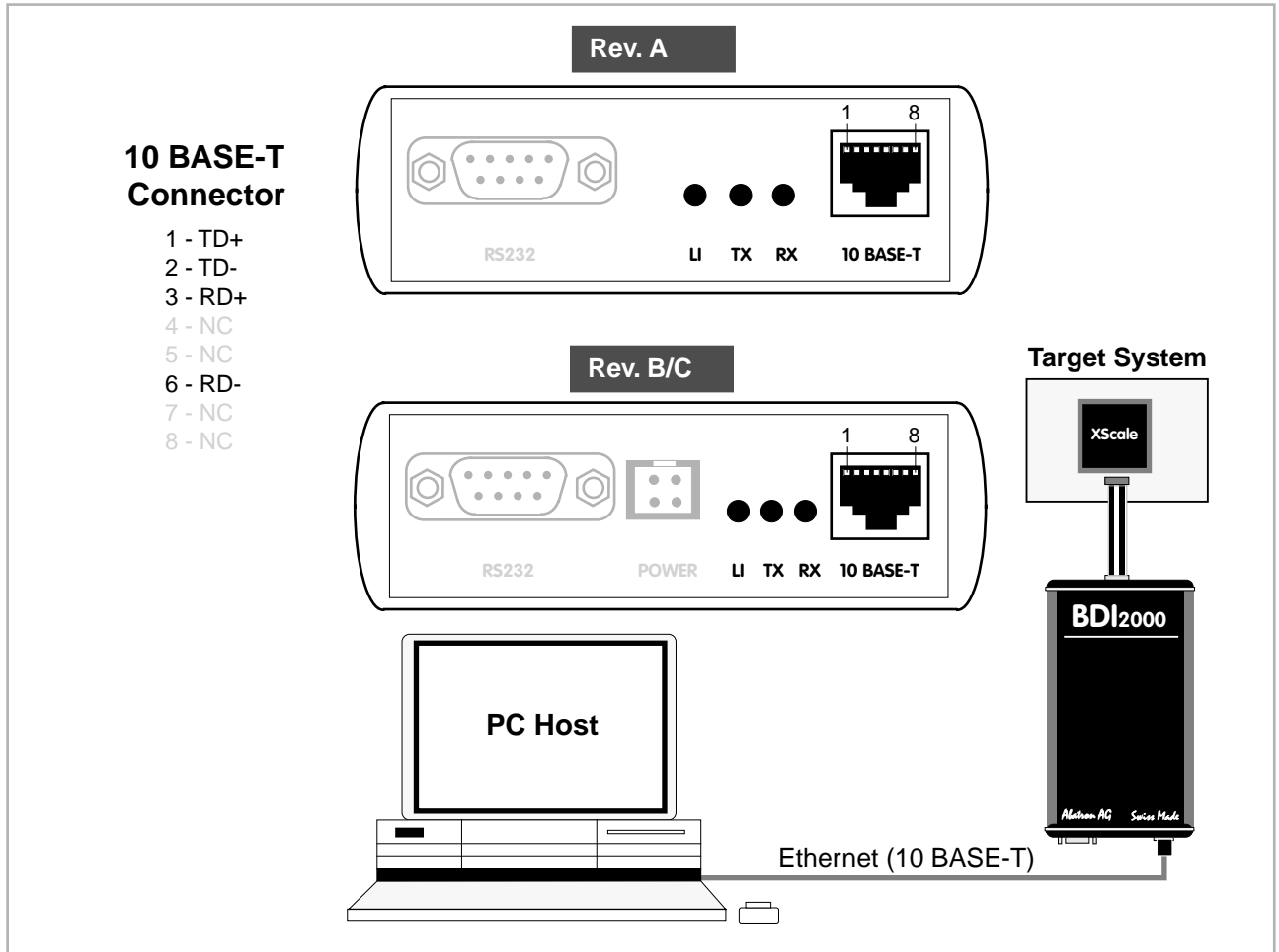
### 2.4.1 Serial line communication

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).



### 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 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:

b20xsc.exe	Configuration program
b20xsc.hlp	Helpfile for the configuration program
b20xscfw.xxx	Firmware for BDI2000 for ARM targets
xscjed20.xxx	JEDEC file for the BDI2000 (Rev. A/B) logic device programming
xscjed21.xxx	JEDEC file for the BDI2000 (Rev. C) logic device programming
bdiifc32.dll	BDI Interface DLL for configuration program
bdiridi.dll	RDI 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 b20arm.exe configuration program.
- The RDI interface DLL has to be copied to the appropriate debugger directory

## 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).

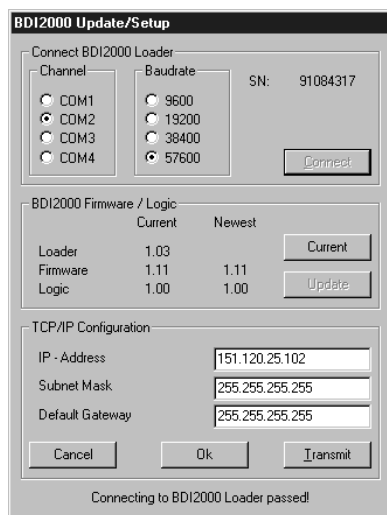
### 2.6.1 BDI2000 Setup/Update

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»*

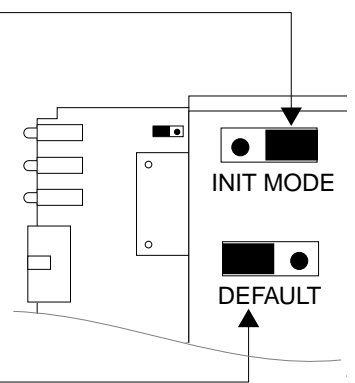
The following options allow you to check or update the BDI firmware and logic and to set the network parameters:

- 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.

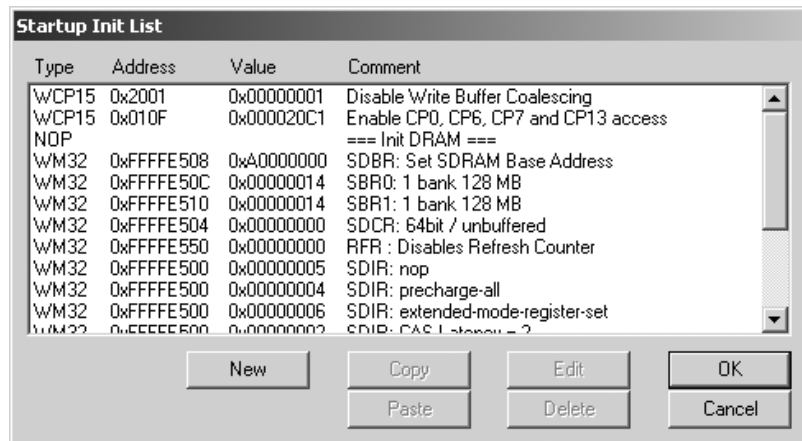
Connect	Click on this button to establish a connection with the BDI2000 loader. Once connected, the BDI2000 remains in loader mode until it is restarted or this dialog box is closed.
Current	Press this button to read back the current loaded BDI2000 software and logic versions. The current loader, firmware 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.xx.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.xx.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..
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.

#### CPx register number:

The register number is used to build the appropriate MCR or MRC instruction.

```
+-----+-----+-----+-----+
|opc_2|0| CRm |opc_1|0| nbr |
+-----+-----+-----+-----+
```

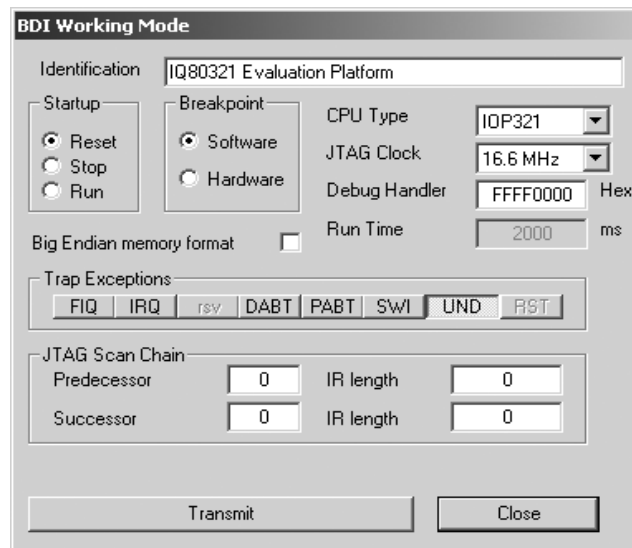
- CP15 : ID register (CRn = 0, opcode\_2 = 0) 0x0000
- CP15 : Cache Type (CRn = 0, opcode\_2 = 1) 0x2000
- CP15 : Invalidate I cache line (CRn = 7, opcode\_2 = 1, CRm = 5) 0x2507

#### Special BDI Configuration Registers:

In order to change some special configuration parameters of the BDI, the GPR entry in the init list is used. Normal ARM GPR's covers a range from 0 to 15. Other GPR's are used to set BDI internal registers:

- 8006 When this entry is present, the BDI fills the default vector table in the Mini IC (except the debug/reset vector) with the requested opcode. Also the vector table in the Mini IC will not be updated each time before the target is restarted.:
- 8007 When this entry is present, the BDI fills the relocated vector table in the Mini IC (except the debug/reset vector) with the requested opcode. Also the vector table in the Mini IC will not be updated each time before the target is restarted.

## 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 implemented. When Software is selected (default), Breakpoints are set by replacing program code in target memory. When Hardware is selected, the built-in breakpoint logic of the target CPU is used to implement Breakpoints. In this mode, only up to 2 Breakpoints are available. This mode may be used when debugging code already stored in a ROM.
CPU Type	Select the CPU type of the target system.
JTAG Clock	This option allows to select the used JTAG clock rate.
Debug Handler	This parameter defines the base address of the debug handler. The debug handler code (and the override vector tables) are loaded into the mini instruction cache during reset processing. See also XScale core manual chapter "Software Debug". The entered value has to be 2k aligned in the range 0x00000000 ... 0x01FEF800 or 0xFE000800 ... 0xFFFFF800.
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.
Trap Exceptions	Select the exceptions that should lead to debug mode entry instead of entering the normal exception handler. Maybe overridden by the debugger.
Big Endian...	Check this switch if the target memory uses Big Endian format.

JTAG Scan Chain	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 XScale chip (Predecessor). Enter the appropriate information also for the devices following the XScale chip (Successor).
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 Debug Handler

The XScale variant of debugging via JTAG depends on a debug handler running on the target. This handler communicates with the BDI via the JTAG interface. This debug handler is loaded into the mini instruction cache via JTAG during reset processing. Please read also the chapter "Software Debug" in the XScale manual. It is also necessary that the reset vectors at 0x00000000 and 0xFFFF0000 are overridden.

Because it is not possible to override only the reset vector, all vectors are overridden by a valid entry in the mini IC. A code fetch always accesses the vector table in the mini IC loaded via JTAG. The mini IC is never updated from memory, it can only be loaded via JTAG.

Always before the target exits debug mode, the BDI reads back the vector tables from memory and updates the mini IC accordingly. This works fine when the vector table is only updated while the target is in debug mode (e.g. via application download). If the application updates the vector table on the fly, this will not change the mini IC and the old vectors will still be used until the target enters/exits debug mode at least once. Therefore it is recommended that the vector table itself is not change dynamically. A second table outside the mini IC addresses may be used to change exception handler addresses on the fly.

In order to force an enter/exit debug mode sequence, a "bkpt #1" instruction maybe added after the vector tables are updated. The BDI recognizes this "bkpt #1" instruction and immediately restarts the target.

## **4.2 Startup Mode**

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

### **4.2.1 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 is activated on the target system and the mini IC is loaded with the debug handler.
- RESET is deactivated and the target is forced into debug mode.
- The BDI works through the initialization list and writes to the corresponding addresses.

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...).

### **4.2.2 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:

- RESET is activated on the target system and the mini IC is loaded with the debug handler.
- RESET is deactivated and the target starts executing application code.
- After a delay of RUNTIME seconds, the target is forced into debug mode.
- The BDI works through the initialization list and writes the corresponding addresses.

### **4.2.3 Startup mode RUN**

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

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

- RESET is activated on the target system and the mini IC is loaded with the debug handler.
- RESET is deactivated and the target starts executing application code.
- The application runs until it is stopped by the debugger.

## 5 Working with RDI Debuggers

### 5.1 ADW/AXD from ARM Ltd.

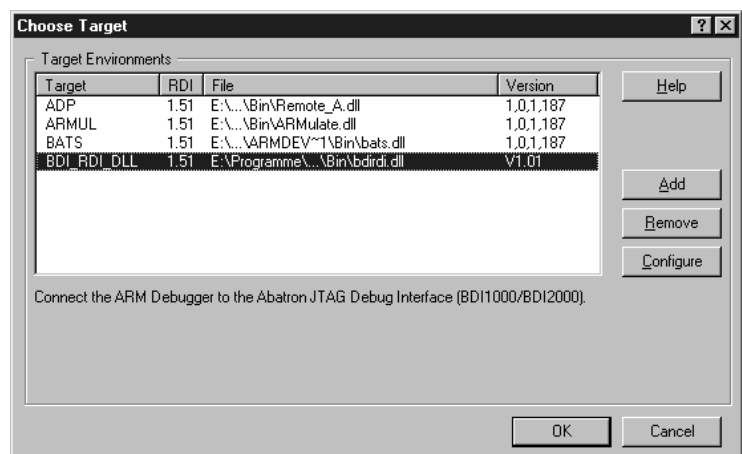
#### 5.1.1 Configuration

In order work with ADW/AXD from ARM Ltd. copy the RDI interface DLL (bdirdi.dll) to the \bin subdirectory of your ARM Software Development Toolkit directory.

Start the ARM debugger and select "Options >> Configure Debugger..." / "Options >> Configure Target...". This opens the "Debugger Configuration" / "Choose Target" dialog box. Add BDIRDI to the list of "Target Environment".

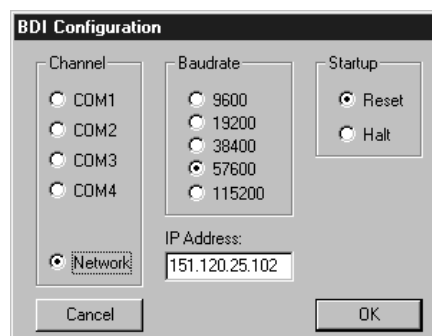


ADW



AXD

To setup the communication parameters, press button "Configure...". This opens the "BDI Communication Setup" dialog box. Enter the appropriate communication parameters.



The Startup mode defines how the BDI interacts with the target when the debugger starts. If Reset is selected (default), the BDI forces a hardware reset of the target when the debugger attaches. The Halt mode is useful when you would like to connect to a running target. The BDI simply stops the target when the debugger attaches. This way you can investigate the current status of the target.

#### Implementation note:

Interactive input of "Direct commands" is not supported, but download to flash is possible with the appropriate command files in the working directory. The execution of the command files can be observed in the RDI Log window. See also chapter "BDI Direct Commands".

## 5.2 BDI Direct Commands

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

### 5.2.1 Target.Reset

This direct command executes a real physical reset of the target system.

### 5.2.2 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 <b>one</b> 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.

### 5.2.3 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)

### 5.2.4 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

### 5.2.5 Flash.Idle

This command disables loading to flash memory.

Syntax:

```
flash.idle
```

### 5.3 Download to Flash Memory

The BDI supports download and debugging of code that runs out of flash memory. To automate the process of downloading to flash memory, the BDI looks for two command files in the working directory.

PRELOAD.CMD            This command file is executed just before download begins

POSTLOAD.CMD          This command file is executed after download is terminated.

Following is an example used to download into the flash memory of the Cogent CSB266 board.

**PRELOAD.CMD:**

```

;
;Define used flash memory: Intel Strata 28F128J3A
flash.setup type=stratax16 size=0x1000000 bus=32 workspace=0xa0000000
;
;Erase sector 0 to 4 of flash memory bank
flash.erase addr=0x00000000
flash.erase addr=0x00040000
flash.erase addr=0x00080000
flash.erase addr=0x000c0000
;
;Enable loading into flash
flash.load addr=0x00000000 size=0x00100000

```

**POSTLOAD.CMD:**

```
flash.idle
```

**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

```

**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

## 6 Telnet Interface

A Telnet server is integrated within the BDI that can be accessed via the network connection. The Telnet is used by the BDI to output additional error messages and other information. The Telnet is useful during initial installation and to bring up new hardware. During normal debugging it is not necessary to connect to the Telnet.

Telnet Debug features:

- Display and modify memory locations
- Display and modify registers
- Single step a code sequence
- Set hardware breakpoints (for code and data accesses)
- Start / Stop program execution
- Execute BDI Direct Commands interactively

### Telnet Command List:

```
"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)",
"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   <addr> <count>         memory test",
"MC   [<address>] [<count>]  calculates a checksum over a memory range",
"MV                                       verifies the last calculated checksum",
"RD                                       display general purpose registers",
"RDALL                                     display all ARM registers ",
"RDCP [<cp>] <number>         display CP register, default is CP15",
"RDACC [<number>]             display an internal accumulator",
"RM   <number> <value>        modify general purpose or user defined register",
"RMCP [<cp>] <number><value>  modify CP register, default is CP15",
"RMACC [<nbr>] <high> <low>  modify an internal accumulator",
"BOOT                                     reset the BDI",
"RESET                                    reset the target system",
"GO   [<pc>]                       set PC and start target system",
"TI   [<pc>]                       single step an instruction",
"HALT                                     force target to enter debug mode",
"BI   <addr>                         set instruction breakpoint",
"CI   [<id>]                         clear instruction breakpoint(s)",
"BD   [R|W] <addr> [<mask>]        set data breakpoint",
"CD   [<id>]                         clear data watchpoint(s)",
"TRACE [<OFF|FILL|WRAP>]          display trace buffer or change trace mode",
"INFO                                     display information about the current state",
"DCMD <direct command>           execute a BDI direct command (see manual)",
"HELP                                     display command list",
"QUIT                                    terminate the Telnet session"
```

## 7 Specifications

Operating Voltage Limiting	5 VDC $\pm$ 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

## 8 Environmental notice



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

## 9 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

## **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 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)

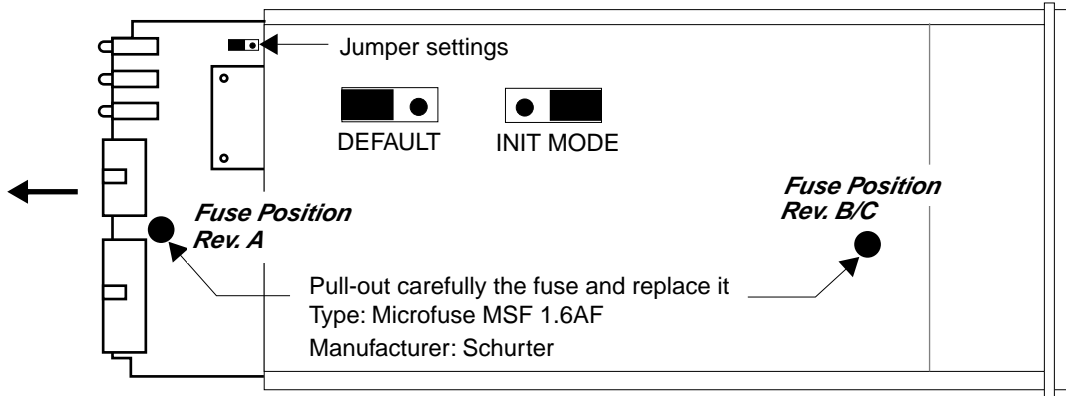
2.2 Remove the two screws that hold the front panel

3

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

**4**

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

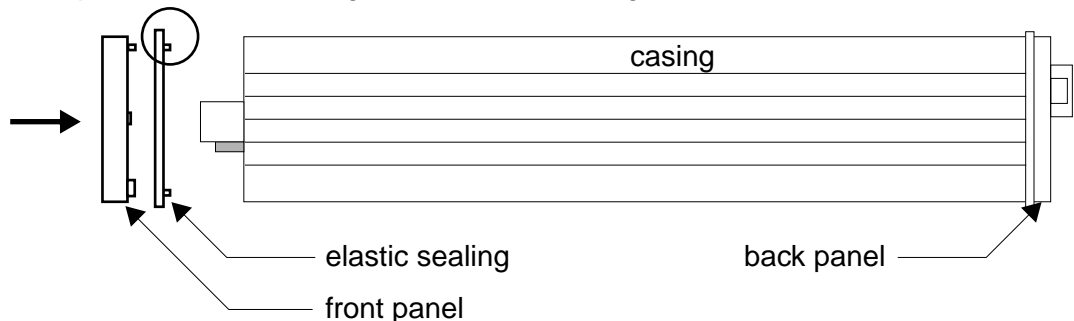


**5**

**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 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.