

*bdidGDB*

BDM debug interface for GNU Debugger

*ColdFire*



# User Manual

Manual Version 1.13 for BDI2000

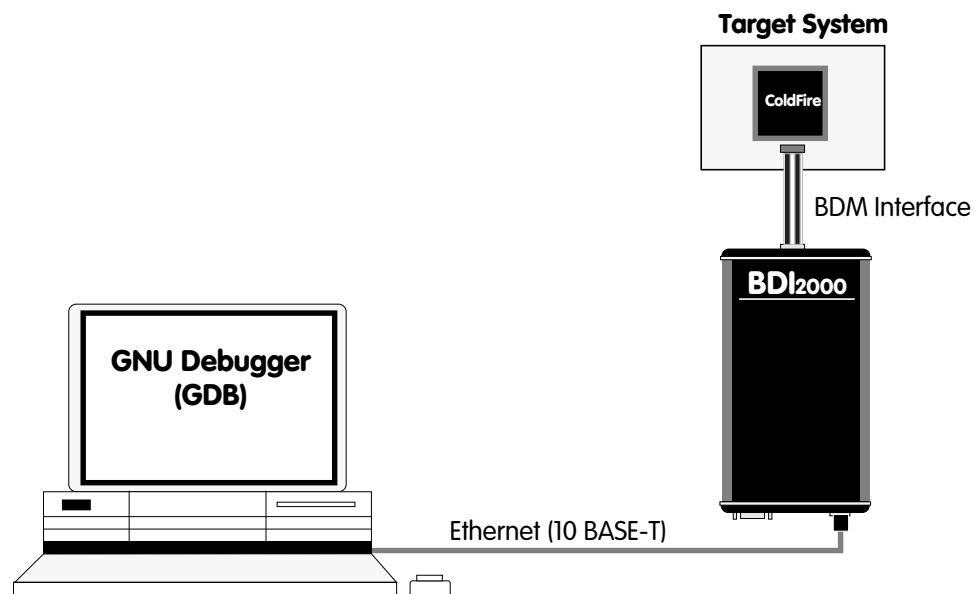
<b>1 Introduction .....</b>	<b>3</b>
1.1 BDI2000.....	3
1.2 BDI Configuration .....	4
<b>2 Installation .....</b>	<b>5</b>
2.1 Connecting the BDI2000 to Target .....	5
2.1.1 Changing Target Processor Type .....	7
2.2 Connecting the BDI2000 to Power Supply .....	8
2.3 Status LED «MODE» .....	9
2.4 Connecting 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
2.5.1 Configuration with a Linux / Unix host.....	14
2.5.2 Configuration with a Windows host.....	16
2.5.3 Recover procedure.....	18
2.6 Testing the BDI2000 to host connection.....	19
2.7 TFTP server for Windows NT .....	19
<b>3 Using bdiGDB .....</b>	<b>20</b>
3.1 Principle of operation .....	20
3.2 Configuration File.....	22
3.2.1 Part [INIT].....	22
3.2.2 Part [TARGET] .....	24
3.2.3 Part [HOST].....	26
3.2.4 Part [FLASH] .....	27
3.2.5 Part [REGS] .....	32
3.3 Debugging with GDB .....	34
3.3.1 Target setup .....	34
3.3.2 Connecting to the target.....	34
3.3.3 Breakpoint Handling.....	35
3.3.4 GDB monitor command.....	35
3.3.5 Target serial I/O via BDI.....	36
3.4 Telnet Interface.....	37
<b>4 Specifications .....</b>	<b>39</b>
<b>5 Environmental notice.....</b>	<b>40</b>
<b>6 Declaration of Conformity (CE).....</b>	<b>40</b>
<b>7 Abatron Warranty and Support Terms .....</b>	<b>41</b>
7.1 Hardware .....	41
7.2 Software .....	41
7.3 Warranty and Disclaimer .....	41
7.4 Limitation of Liability .....	41
 <b>Appendices</b>	
<b>A Troubleshooting .....</b>	<b>42</b>
<b>B Maintenance .....</b>	<b>43</b>
<b>C Trademarks .....</b>	<b>45</b>

## 1 Introduction

bdiGDB enhances the GNU debugger (GDB), with Background Debug Mode (BDM) debugging for ColdFire based targets. With the built-in Ethernet interface you get a very fast code download speed of up to 150Kbytes/sec. No target communication channel (e.g. serial line) is wasted for debugging purposes. Even better, you can use fast Ethernet debugging with target systems without network capability. The host to BDI communication uses the standard GDB remote protocol.

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

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



### 1.1 BDI2000

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

## 1.2 BDI Configuration

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

Following an example of a typical configuration file:

```
; Configuration file for a MCF5307 board
; -----
; the initialisation list used to setup the target system
[INIT]
WCREG 0xC0F 0x10000001 ;MBAR: map internal registers to 0x10000000
WCREG 0xC04 0x00800021 ;RAMBAR: map internal SRAM to 0x00800000
WM16 0x10000080 0xFFE0 ;CSAR0: Flash at 0xFFE00000
WM16 0x1000008A 0x0D80 ;CSCR0: Flash 3 waits, 16bit,
WM32 0x10000084 0x000F0001 ;CSMR0: Flash 1MB, R/W, valid
WM16 0x10000100 0x8230 ;DCR: SDRAM Trc=6, RC=48
WM32 0x10000108 0x00001300 ;DACR0: base=0x00000000; timing=2,4,2,1,-1; CBM=011
WM32 0x1000010C 0x003C0001 ;DCMR0: mask=4MB; enable
WM32 0x10000108 0x00001308 ;DACR0: Initiate Precharge All Command
WM32 0x00000400 0x00000000 ;Execute PALL command
WM32 0x10000108 0x00009300 ;DACR0: enable refresh
WM32 0x10000110 0x00401300 ;DACR1: base=0x00400000; timing=2,4,2,1,-1; CBM=011
WM32 0x10000114 0x007C0001 ;DCMR1: mask=8MB; enable
WM32 0x10000110 0x00401308 ;DACR1: Initiate Precharge All Command
WM32 0x00400400 0x00000000 ;Execute PALL command
WM32 0x10000110 0x00409300 ;DACR1: enable refresh
DELAY 20 ;Delay for Refresh
WM32 0x10000108 0x00009340 ;DACR0: Initiate Mode Register Set Command
WM32 0x00000400 0x00000000 ;Execute MRS command
WM32 0x10000110 0x00009340 ;DACR1: Initiate Mode Register Set Command
WM32 0x00400400 0x00000000 ;Execute MRS command

[TARGET]
CPUTYPE MCF5307
CPUCLOCK 90000000 ;the CPU clock rate after processing the init list
BREAKMODE SOFT ;SOFT or HARD
VECTOR CATCH ;catch unhandled exceptions

[HOST]
IP 151.120.25.115
FILE E:\cygnus\root\usr\demo\mcf5307\fibonacci.exe
FORMAT COFF
LOAD MANUAL ;load code MANUAL or AUTO after reset

[FLASH]
WORKSPACE 0x00800000 ;workspace in target RAM for fast programming algorithm
CHIPTYPE AM29F ;Flash type (AM29F | AM29BX8 | AM29BX16 | I28BX8 | I28BX16)
CHIPSIZE 0x80000 ;The size of one flash chip in bytes (e.g. AM29F010 = 0x20000)
BUSWIDTH 16 ;The width of the flash memory bus in bits (8 | 16 | 32)
FILE D:\abatron\bdi360\ColdFire\pro\sbc5307.sss
ERASE 0xFFE00000 ;erase sector 0 of flash
ERASE 0xFFE20000 ;erase sector 1 of flash

[REGS]
DMM1 0x10000000
FILE E:\cygnus\root\usr\demo\mcf5307\reg5307.def
```

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

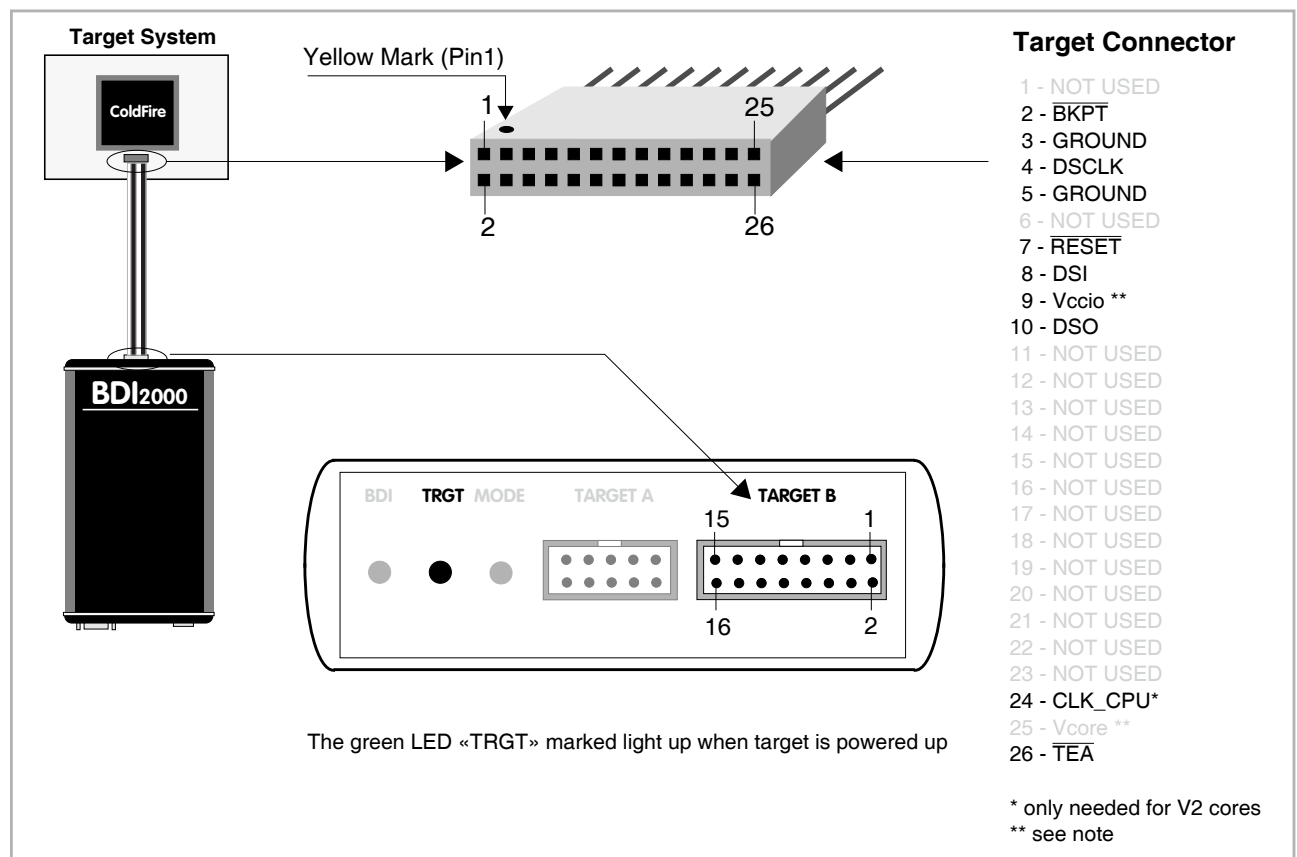
## 2 Installation

### 2.1 Connecting the BDI2000 to Target

The enclosed cable to the target system is designed for the Motorola recommended 26-pin Berg 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 Motorola specification.



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

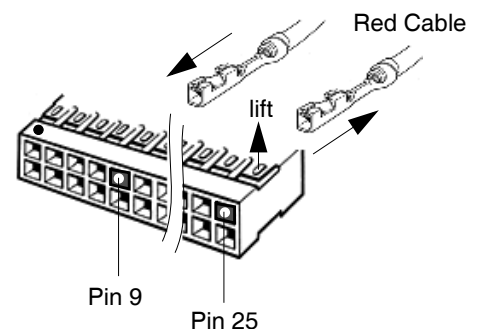


The target CPU clock is only needed for older V2 cores (MCF5204, MCF5206(e) and MCF5272) because DSCLK and DSI have to change synchronous with the CPU clock. For all other cores, BDM communication can be asynchronous to the CPU clock and therefore this signal is not needed.



#### Important note for older target cables:

The target cables delivered before October 2004 expect the target BDM reference voltage (Vccio) at pin 25 of the BDM connector. For target boards where Vccio is only routed to pin 9, remove the contact/wire from housing pin 25 and insert it into pin 9.



**BDI TARGET B Connector Signals:**

Pin	Name	Description
1	DSO	<b>DATA SERIAL OUT</b> For background debug mode, serial data output from the MCU.
2	<reseved>	
3	DSI	<b>DATA SERIAL IN</b> For background debug mode, serial data input signal to the MCU.
4	<reseved>	
5	<reseved>	
6	Vccio 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 fed from Vcc I/O on the target board.  <b>3.0 – 5.0V with Rev. 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.
7	DSCLK	<b>DEVELOPMENT SERIAL CLOCK</b> For background debug mode, serial shift clock to the MCU.
8	BKPT	<b>BREAKPOINT</b> BKPT is an active-low signal that signals a hardware breakpoint for the ColdFire core. It is used to force the ColdFire core to enter debug mode.
9	$\overline{TEA}$ (optional)	<b>TRANSFER ERROR ACKNOWLEDGE (currently not implemented)</b> Active-low open-drain signal, used to abort a bus cycle. This signal may be helpful for ColdFire devices which has no built-in bus monitor (e.g. MCF5307). The BDI is able to terminate an invalid memory access. Otherwise BDM communication may hang until a reset is applied.
10	<reseved>	
11	<reseved>	
12	GROUND	<b>System Ground</b>
13	$\overline{RESET}$	<b>RESET</b> Active-low open-drain signal, used to force a system reset.
14	CLK_CPU	<b>PROCESSOR CLOCK</b> For some cores, this clock signal is used to synchornize the BDM signals DSCLK and DSI.
15	<reseved>	
16	GROUND	<b>System Ground</b>

Pin 14 is only use for some older V2 cores (MCF5204, MCF5206(e) and MCF5272). For all other cores this signal is not used.

### 2.1.1 Changing Target Processor Type

Before you can use the BDI2000 with an other target processor type (e.g. ColdFire <--> PPC), a new setup has to be done (see chapter 2.5). 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».



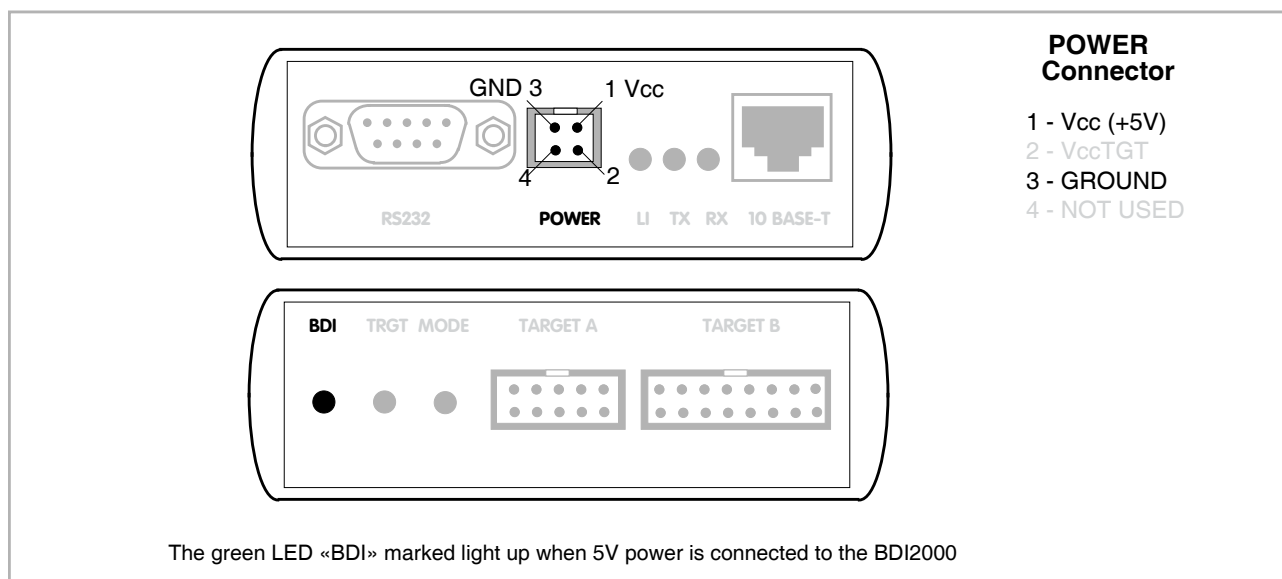
**To avoid data line conflicts, the BDI2000 must be disconnected from the target system while programming the logic for an other target CPU.**

## 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.**

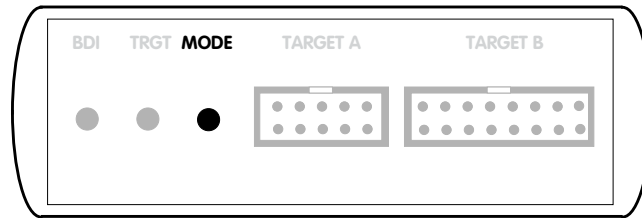


**Please switch on the system in the following sequence:**

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

### 2.3 Status LED «MODE»

The built in LED indicates the following BDI states:



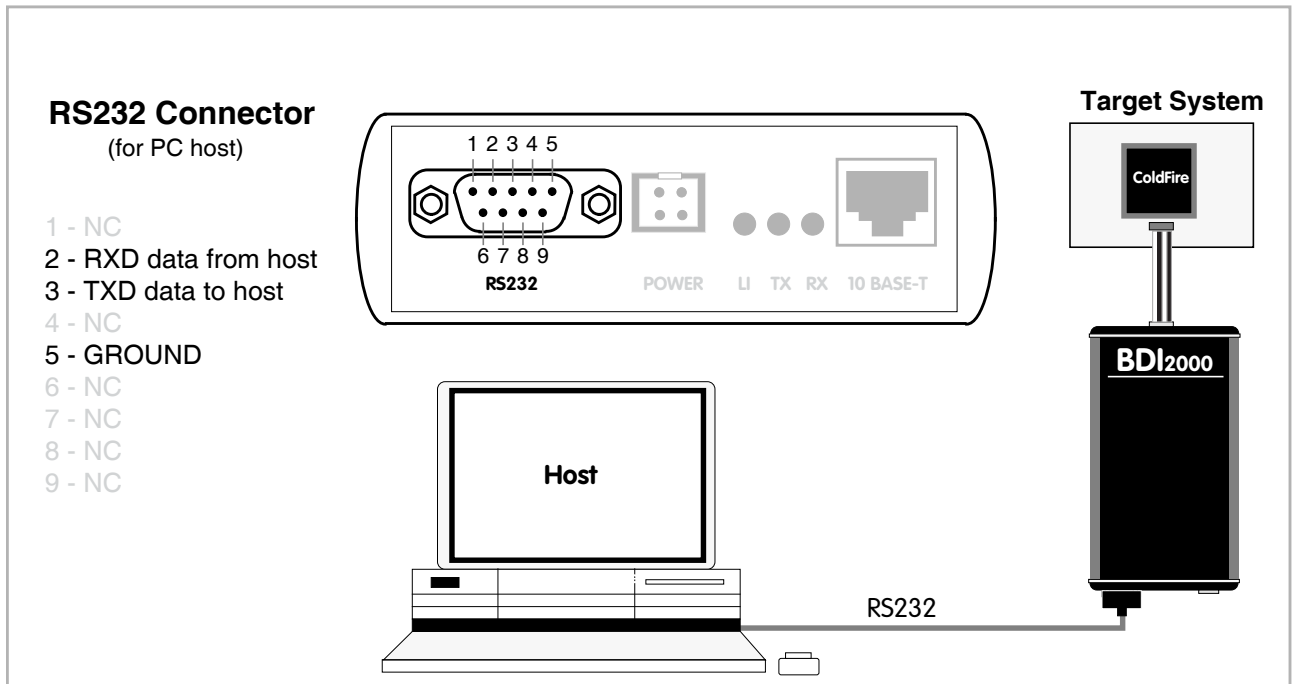
MODE LED	BDI STATES
OFF	The BDI is ready for use, the firmware is already loaded.
ON	The power supply for the BDI2000 is < 4.75VDC.
BLINK	The BDI «loader mode» is active (an invalid firmware is loaded or loading firmware is active).

## 2.4 Connecting the BDI2000 to Host

### 2.4.1 Serial line communication

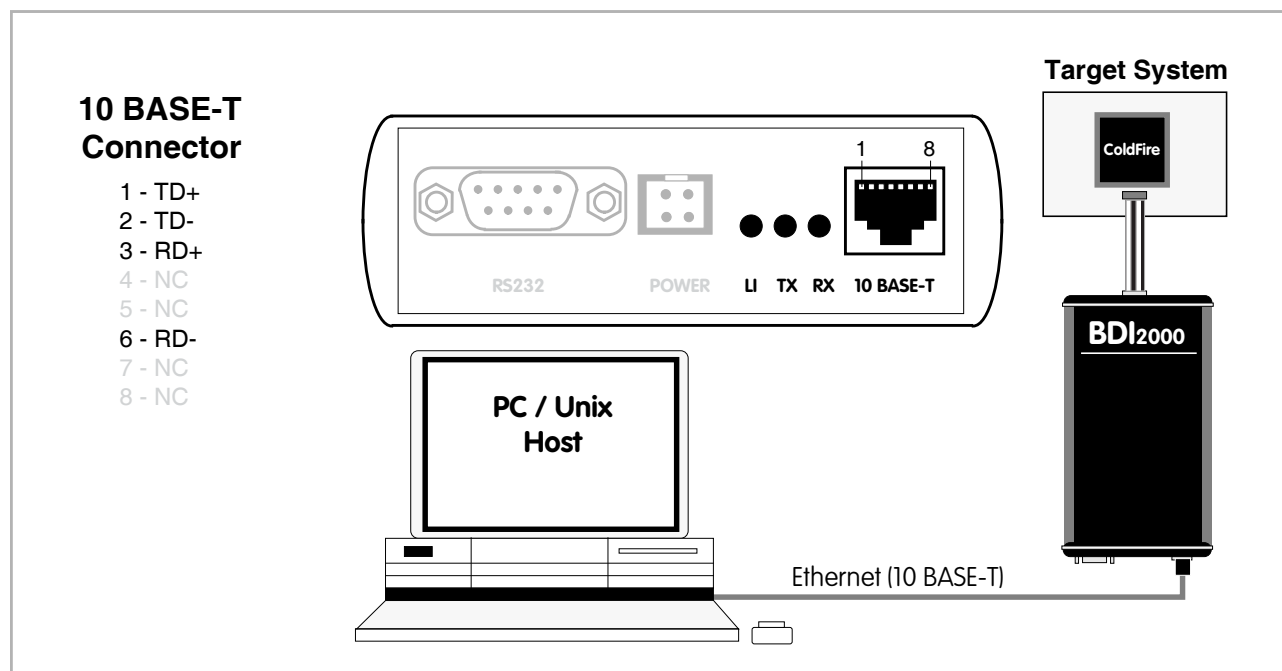
Serial line communication is only used for the initial configuration of the bdiGDB system.

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



### 2.4.2 Ethernet communication

The 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 BDI2000. For Windows users there is also a TFTP server included.

The following files are on the diskette.

b20mcfgd.exe	Configuration program (Windows application)
b20mcfgd.hlp	Windows help file for the configuration program
b20mcfgd.xxx	Firmware for the BDI2000
cf2jed20.xxx	JEDEC file for the BDI (Rev. B) logic device (for old V2 cores)
cf2jed21.xxx	JEDEC file for the BDI (Rev. C) logic device (for old V2 cores)
cf3jed20.xxx	JEDEC file for the BDI (Rev. B) logic device
cf3jed21.xxx	JEDEC file for the BDI (Rev. C) logic device
ftpsrv.exe	TFTP server for Windows (WIN32 console application)
*.cfg	Sample configuration files
*.def	Register definition files
bdisetup.zip	ZIP Archive with the Setup Tool sources for Linux / UNIX hosts.

### Overview of an installation / configuration process:

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

### Activating BOOTP:

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

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

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

The MAC address is derived from the serial number as follows:

MAC: 00-0C-01-xx-xx-xx , repare the xx-xx-xx with the 6 left digits of the serial number

Example: SN# 93123457 ==>> 00-0C-01-93-12-34

**Important:**

Because some older V2 ColdFire cores (MCF5204, MCF5206(e) and MCF5272) need synchronous signals at the BDM interface, the BDI uses a different logic for this cores. Make sure that there is the correct logic loaded for the target CPU you are using.

With the Windows setup tool, you can check a switch to select the correct logic file.

The Linux/Unix setup tool looks for a file with the name mcfjed2x.xxx. Therefore copy first the correct logic file before updating the firmware/logic.

For **MCF5204, MCF5206(e), MCF5272** select the V2 logic file (check the Sync switch).

**Linux/Unix only:**

Before you can update the firmware/logic of the BDI, copy the correct JEDEC file to mcfjed2x.xxx:

5204, 5206(e), 5272:      cf2jed20.102 -> mcfjed20.102 / cf2jed21.102 -> mcfjed21.102

All other cores:        cf3jed20.102 -> mcfjed20.102 / cf3jed21.102 -> mcfjed21.102

## 2.5.1 Configuration with a Linux / Unix host

The firmware / logic update and the initial configuration of the BDI2000 is done with a command line utility. In the ZIP Archive `bdisetup.zip` are all sources to build this utility. More information about this utility can be found at the top in the `bdisetup.c` source file. There is also a make file included. Starting the tool without any parameter displays information about the syntax and parameters.



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

Following the steps to bring-up a new BDI2000:

### 1. Build the setup tool:

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

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

### 2. Check the serial connection to the BDI:

With "`bdisetup -v`" you may check the serial connection to the BDI. The BDI will respond with information about the current loaded firmware and network configuration.

**Note:** Login as root, otherwise you probably have no access to the serial port.

```
[root@LINUX_1 bdisetup]# ./bdisetup -v -p/dev/ttyS0 -b57
BDI Type : BDI2000 Rev.C (SN: 92152150)
Loader : V1.05
Firmware : unknown
Logic : unknown
MAC : ff-ff-ff-ff-ff-ff
IP Addr : 255.255.255.255
Subnet : 255.255.255.255
Gateway : 255.255.255.255
Host IP : 255.255.255.255
Config : ??????????????????
```

### 3. Load/Update the BDI firmware/logic:

With "`bdisetup -u`" the firmware is loaded and the CPLD within the BDI2000 is programmed. This configures the BDI for the target you are using. Based on the parameters `-a` and `-t`, the tool selects the correct firmware / logic files. If the firmware / logic files are in the same directory as the setup tool, there is no need to enter a `-d` parameter.

```
[root@LINUX_1 bdisetup]# cp cf3jed21.103 mcfjed21.103
[root@LINUX_1 bdisetup]# ./bdisetup -u -p/dev/ttyS0 -b57 -aGDB -tMCF
Connecting to BDI loader
Erasing CPLD
Programming firmware with ./b20mcfgd.100
Programming CPLD with ./mcfjed21.103
```

#### 4. Transmit the initial configuration parameters:

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

BDI IP Address	The IP address for the BDI2000. Ask your network administrator for assigning an IP address to this BDI2000. Every BDI2000 in your network needs a different IP address.
Subnet Mask	The subnet mask of the network where the BDI is connected to. A subnet mask of 255.255.255.255 disables the gateway feature. Ask your network administrator for the correct subnet mask. If the BDI and the host are in the same subnet, it is not necessary to enter a subnet mask.
Default Gateway	Enter the IP address of the default gateway. Ask your network administrator for the correct gateway IP address. If the gateway feature is disabled, you may enter 255.255.255.255 or any other value.
Config - Host IP Address	Enter the IP address of the host with the configuration file. The configuration file is automatically read by the BDI2000 after every start-up.
Configuration file	Enter the full path and name of the configuration file. This file is read via TFTP. Keep in mind that TFTP has it's own root directory (usual /tftpboot). You can simply copy the configuration file to this directory and the use the file name without any path. For more information about TFTP use "man tftpd".

```
[root@LINUX_1 bdiGDB]# ./bdisetup -c -p/dev/ttyS0 -b57 \  
> -i151.120.25.101 \  
> -h151.120.25.118 \  
> -fmc5307.cnf  
Connecting to BDI loader  
Writing network configuration  
Writing init list and mode  
Configuration passed
```

#### 5. Check configuration and exit loader mode:

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

```
[root@LINUX_1 bdiGDB]# ./bdisetup -v -p/dev/ttyS0 -b57 -s  
BDI Type : BDI2000 Rev.C (SN: 92152150)  
Loader : V1.05  
Firmware : V1.00 bdiGDB for ColdFire  
Logic : V1.03 ColdFire  
MAC : 00-0c-01-92-15-21  
IP Addr : 151.120.25.101  
Subnet : 255.255.255.255  
Gateway : 255.255.255.255  
Host IP : 151.120.25.118  
Config : mcf5307.cnf
```

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

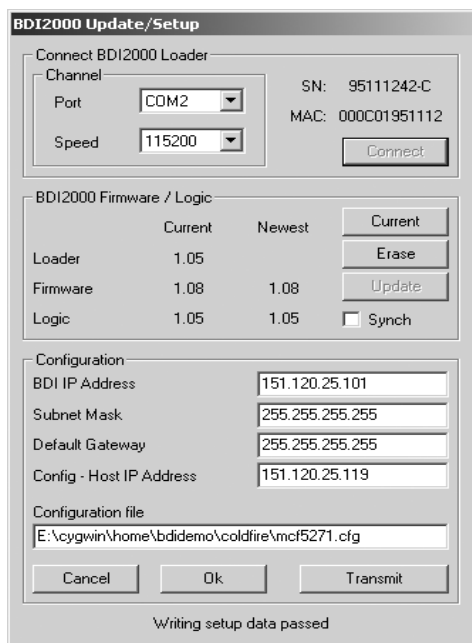
```
[root@LINUX_1 bdiGDB]# telnet 151.120.25.101
```

## 2.5.2 Configuration with a Windows host

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



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



*dialog box «BDI2000 Update/Setup»*

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

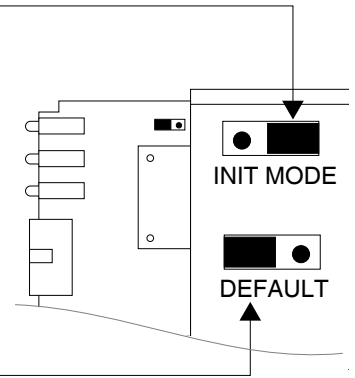
- Channel                      Select the communication port where the BDI2000 is connected during this setup session.
- Baudrate                    Select the baudrate used to communicate with the BDI2000 loader during this setup session.
- Connect                     Click on this button to establish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

Synch	Because some older V2 ColdFire cores (e.g. MCF5272) need synchronous signals at the BDM interface, the BDI uses a different logic for this cores. Check this box only if your target is a MCF5204, MCF5206, MCF5206e or MCF5272. The BDI automatically selects the correct JEDEC file. Make sure that there is the correct logic loaded for the target CPU you are using.
BDI IP Address	Enter the IP address for the BDI2000. Use the following format: xxx.xxx.xxx.xxx e.g. 151.120.25.101 Ask your network administrator for assigning an IP address to this BDI2000. Every BDI2000 in your network needs a different IP address.
Subnet Mask	Enter the subnet mask of the network where the BDI is connected to. Use the following format: xxx.xxx.xxx.xxx.e.g. 255.255.255.0 A subnet mask of 255.255.255.255 disables the gateway feature. Ask your network administrator for the correct subnet mask.
Default Gateway	Enter the IP address of the default gateway. Ask your network administrator for the correct gateway IP address. If the gateway feature is disabled, you may enter 255.255.255.255 or any other value..
Config - Host IP Address	Enter the IP address of the host with the configuration file. The configuration file is automatically read by the BDI2000 after every start-up.
Configuration file	Enter the full path and name of the configuration file. e.g. D:\gnu\config\bdi\ads860bdi.cnf For information about the syntax of the configuration file see the bdiGDB User manual. This name is transmitted to the TFTP server when reading the configuration file.
Transmit	Click on this button to store the configuration in the BDI2000 flash memory.

### 2.5.3 Recover procedure

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»



## 2.6 Testing the BDI2000 to host connection

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

- If not already done, connect the bdiGDB system to the network.
- Power-up the BDI2000.
- Start a Telnet client on the host and connect to the BDI2000 (the IP address you entered during initial configuration).
- If everything is okay, a sign on message like «BDI Debugger for ColdFire» and a list of the available commands should be displayed in the Telnet window.

## 2.7 TFTP server for Windows NT

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

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

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

The parameter [p] enables protocol output to the console window. Try it.

The parameter [w] enables write accesses to the host file system.

The parameter [d] allows to define a root directory.

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

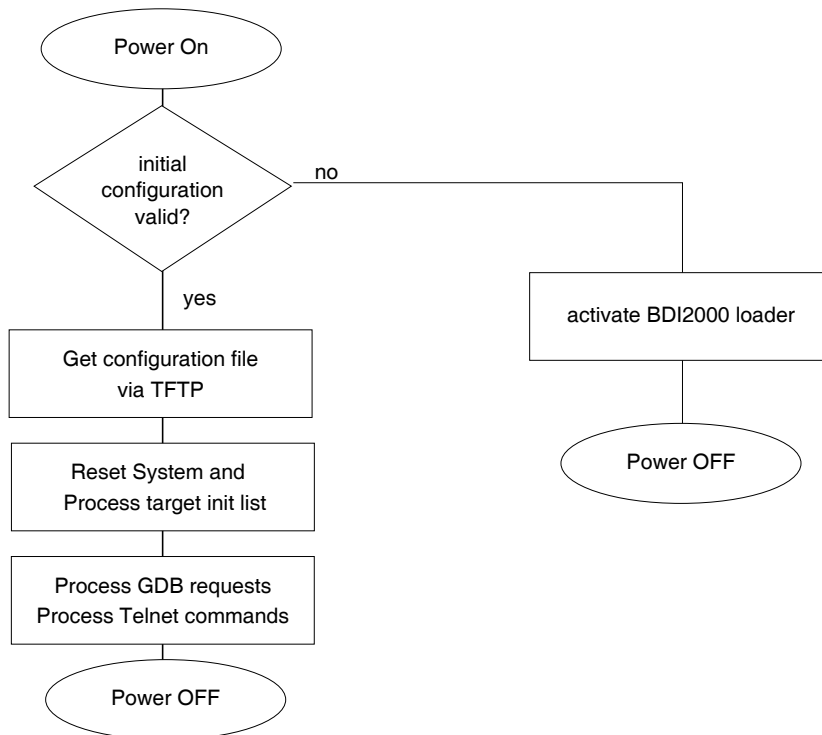
You may enter the TFTP server into the Startup group so the server is started every time you logon.

### 3 Using bdiGDB

#### 3.1 Principle of operation

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

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



**Breakpoints:**

There are two breakpoint modes supported. One of them (SOFT) is implemented by replacing application code with a HALT instruction. The other (HARD) uses the built in breakpoint logic. If HARD is used, only up to 1 ( 4 for V4 cores ) breakpoints can be active at the same time. The following example selects SOFT as the breakpoint mode:

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

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

**Target Exceptions:**

If enabled, the BDI will catch all unhandled exceptions. This is only possible if the vector table is writable. At vector 0 the BDI writes a HALT, RTE instruction sequence and lets all other vectors point to this short exception handler. The BDI reads back the VBR after processing the initialisation list in order to get the base address of the vector table.

```
[INIT]
WCREG 0x801 0x00000000 ;set vector base

[TARGET]
VECTORCATCH ;enable vector catching
```

## 3.2 Configuration File

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

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

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

### 3.2.1 Part [INIT]

The part [INIT] defines a list of commands which should be executed every time the target comes out of reset. The commands are used to get the target ready for loading the program file. Commands in this section are processed in order from top to bottom.

- WDREG register value    Write value to the selected data register.  
                           register        the register number 0 .. 7  
                           value            the value to write into the register  
                           Example: WDREG 0 5
- WAREG register value    Write value to the selected address register.  
                           register        the register number 0 .. 7  
                           value            the value to write into the register  
                           Example: WAREG 0 5
- WCREG register value    Write value to the selected control register.  
                           register        the register number(e.g. 0x801 for VBR)  
                           value            the value to write into the register  
                           Example: WCREG 0xC0F 0x10000001 ;MBAR
- WM8 address value        Write a byte (8bit) to the selected memory place.  
                           address        the memory address  
                           value            the value to write to the target memory  
                           Example: WM8 0xFFFFFA21 0x04 ; SYPCR: watchdog disable ...
- WM16 address value       Write a half word (16bit) to the selected memory place.  
                           address        the memory address  
                           value            the value to write to the target memory  
                           Example: WM16 0x10000100 0x8230 ;DCR

WM32 address value	Write a word (32bit) to the selected memory place. address       the memory address value         the value to write to the target memory Example: WM32 0x1000010C 0x003C0001 ;DCMR0
MMAP start end	Because a memory access to an invalid memory space via BDM can lead to a deadlock, this entry can be used to define up to 32 valid memory ranges. If at least one memory range is defined, the BDI checks against this range(s) and avoids accessing of not mapped memory ranges. start         the start address of a valid memory range end          the end address of this memory range Example: MMAP 0xFFE00000 0xFFFFFFFF ;Boot ROM
DELAY value	Delay for the selected time. A delay may be necessary to let the clock PLL lock again after a new clock rate is selected. value         the delay time in milliseconds (1...30000) Example: DELAY 500 ; delay for 0.5 seconds
WTLB tag data	Only V4e cores: Adds an entry to the TLB array(s). For a detailed description of the tag/data value look at the V4e MMU description. The first WTLB entry in the init list also clears the hole TLB array. tag          virtual page number, ASID, shared and valid bit data         real page number, size, cache mode and SRWXL bits Example:     WTLB 0x00000001 0x1001025C ;SRAM 8k CB RWX

MMU setup example:

```

NOP          ;Set Memory Map
WCREG 0x0C0F 0x10000001 ;MBAR : map internal REGS to 0x10000000
WCREG 0x0008 0x11000001 ;MMUBAR : map MMU registers to 0x11000000
WCREG 0x0C04 0x20000035 ;RAMBAR0: map internal SRAM to 0x20000000
WCREG 0x0C05 0x20001035 ;RAMBAR1: map internal SRAM to 0x20001000
NOP          ;MMU : Map internal registers
WTLB 0x10000001 0x10000298 ;0x10000000 -> 0x10000000, 8k, NP, RW-
WTLB 0x10002001 0x10020298 ;0x10002000 -> 0x10002000, 8k, NP, RW-
WTLB 0x10004001 0x10040298 ;0x10004000 -> 0x10004000, 8k, NP, RW-
WTLB 0x10006001 0x10060298 ;0x10006000 -> 0x10006000, 8k, NP, RW-
WTLB 0x10008001 0x10080298 ;0x10008000 -> 0x10008000, 8k, NP, RW-
WTLB 0x1000A001 0x100A0298 ;0x1000A000 -> 0x1000A000, 8k, NP, RW-
WTLB 0x1000C001 0x100C0298 ;0x1000C000 -> 0x1000C000, 8k, NP, RW-
WTLB 0x1000E001 0x100E0298 ;0x1000E000 -> 0x1000E000, 8k, NP, RW-
NOP          ;MMU : Map 32k System RAM to 0x00000000
WTLB 0x00000001 0x1001025C ;0x00000000 -> 0x10010000, 8k, CB, RWX
WTLB 0x00002001 0x1001225C ;0x00002000 -> 0x10012000, 8k, CB, RWX
WTLB 0x00004001 0x1001425C ;0x00004000 -> 0x10014000, 8k, CB, RWX
WTLB 0x00006001 0x1001625C ;0x00006000 -> 0x10016000, 8k, CB, RWX
WM32 0x11000000 0x00000001 ;MMUCR: enable MMU

```

### 3.2.2 Part [TARGET]

The part [TARGET] defines some target specific values.

CPUTYPE type	<p>This value gives the BDI information about the connected CPU.</p> <p style="margin-left: 20px;">type      MCF5202, MCF5203, MCF5204, MCF5206, MCF5272  MCF5207, MCF5208, MCF5214, MCF5216,  MCF5249, SCF5250, MCF5251, MCF5253  MCF5230, MCF5232, MCF5233, MCF5234, MCF5235  MCF5270, MCF5271, MCF5274, MCF5275  MCF5280, MCF5281, MCF5282  MCF5210, MCF5211, MCF5212, MCF5213,  MCF5221, MCF5222, MCF5223, MCF5225, MCF5227  MCF5307, MCF5301  MCF5327, MCF5328, MCF5329, MCF5372, MCF5373  MCF5407, MCF5470, MCF5480, MCF5441, MCF5445</p> <p style="margin-left: 20px;">Example:    CPUTYPE MCF5282</p>
BDIMODE mode param	<p>This parameter selects the BDI debugging mode. The following modes are supported:</p> <p style="margin-left: 20px;">LOADONLY    Loads and starts the application core. No debugging via BDM.</p> <p style="margin-left: 20px;">AGENT        The debug agent runs within the BDI. There is no need for any debug software on the target. This mode accepts a second parameter. If RUN is entered as a second parameter, the loaded application will be started immediately, otherwise only the PC is set and BDI waits for GDB requests.</p> <p style="margin-left: 20px;">Example:     BDIMODE AGENT RUN</p>
CPUCLOCK value	<p>The BDI needs to know how fast the target CPU runs after processing the init list. The BDM communication speed is selected based on this value. If this value defines a clock rate that is higher than the real clock, BDM communication may fail. When defining a clock rate slower than possible, BDM communication still works but not as fast as possible.</p> <p>For V3/V4 cores, enter the PSTCLK value.</p> <p style="margin-left: 20px;">value        the CPU (PSTCLK) clock in hertz</p> <p style="margin-left: 20px;">Example:     CPUCLOCK 25000000 ; CPU clock is 25.0MHz</p>
STARTUP mode [runtime]	<p>This parameter selects the target startup mode. The following modes are supported:</p> <p style="margin-left: 20px;">RESET        This default mode forces the target to debug mode immediately out of reset. No code is executed after reset.</p> <p style="margin-left: 20px;">STOP         In this mode, the BDI lets the target execute code for "runtime" milliseconds after reset. This mode is useful when monitor code should initialize the target system.</p> <p style="margin-left: 20px;">RUN          After reset, the target executes code until stopped by the Telnet "halt" command.</p> <p style="margin-left: 20px;">Example:     STARTUP STOP 3000 ; let the CPU run for 3 seconds</p>

**BREAKMODE mode [NOUHE]** This parameter defines how breakpoints are implemented. The current mode can also be changed via the Telnet interface. By default the BDI sets the CSR[UHE] bit so a halt instruction in user-mode will be handled by the BDI. If the option NOUHE is present, the BDI will not set this bit and a halt in user mode will cause an invalid instruction exception that can be handled by an application debugger resident in the target system.

**SOFT** This is the normal mode. Breakpoints are implemented by replacing code with a TRAP instruction.

**HARD** In this mode, the breakpoint hardware is used. Only 1 (4) breakpoints at a time are supported.

**Example:** BREAKMODE HARD ; enable use of break hardware

**VECTOR CATCH** When this line is present, the BDI catches all unhandled exceptions. Catching exceptions is only possible if the vector table is writable.

**Example:** VECTOR CATCH ; catch unhandled exception

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

**port** The TCP/IP port used for the host communication.

**baudrate** The BDI supports 2400 ... 115200 baud

**Example:** SIO 7 9600 ;TCP port for virtual IO

**WAKEUP time** This entry in the init list allows to define a delay time (in ms) the BDI inserts between releasing the RESET line and starting communicating with the target. This init list entry may be necessary if RESET is delayed on its way to the processors reset pin. If not defined a delay of 1 ms is used.

**time** the delay time in milliseconds (10 ... 60'000)

**Example:** WAKEUP 3000 ; insert 3sec wake-up time

**RESET time** This entry in the init list allows to define the time (in ms) the BDI asserts the RESET signal. If not defined, reset is asserted for 1 ms.

**time** the reset time in milliseconds (10 ... 60'000)

**Example:** RESET 500 ; assert RESET for 500 ms

### 3.2.3 Part [HOST]

The part [HOST] defines some host specific values.

IP	ipaddress	The IP address of the host. ipaddress the IP address in the form xxx.xxx.xxx.xxx Example: IP 151.120.25.100
FILE	filename	The default name of the file that is loaded into RAM using the Telnet 'load' command. This name is used to access the file via TFTP. If the filename starts with a \$, this \$ is replace with the path of the configuration file name. filename the filename including the full path or \$ for relative path. Example: FILE F:\gnu\demo\mcf\test.elf FILE \$test.elf
FORMAT	format [offset]	The format of the program file and an optional load address offset. Currently binary, S-record, a.out, ELF and COFF formats are supported. If the code is already stored in ROM on the target, select ROM as the format. The optional parameter "offset" is added to any load address read from the image file. format BIN, SREC, AOUT, ELF, COFF or ROM Example: FORMAT COFF FORMAT COFF 0x10000
LOAD	mode	In Agent mode, this parameters defines if the code is loaded automatically after every reset. mode AUTO, MANUAL Example: LOAD MANUAL
START	address	The address where to start the program file. If this value is not defined and the core is not in ROM, the address is taken from the code file. If this value is not defined and the core is already in ROM, the PC will not be set before starting the program file. This means, the program starts at the normal reset address (0x0100). address the address where to start the program file Example: START 0x1000
DEBUGPORT	port [RECONNECT]	The TCP port GDB uses to access the target. If the RECONNECT parameter is present, an open TCP/IP connection (Telnet/GDB) will be closed if there is a connect request from the same host (same IP address). port the TCP port number (default = 2001) Example: DEBUGPORT 2001
PROMPT	string	This entry defines a new Telnet prompt. The current prompt can also be changed via the Telnet interface. Example: PROMPT 5307>
DUMP	filename	The default file name used for the Telnet DUMP command. filename the filename including the full path Example: DUMP dump.bin

### 3.2.4 Part [FLASH]

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

CHIPTYPE type	<p>This parameter defines the type of flash used. It is used to select the correct programming algorithm.</p> <p>format            AM29F, AM29BX8, AM29BX16, I28BX8, I28BX16, AT49, AT49X8, AT49X16, STRATAX8, STRATAX16, MIRROR, MIRRORX8, MIRRORX16, S29M64X8, S29M32X16, AM29DX16, AM29DX32</p> <p>Example:        CHIPTYPE AM29F</p>
CHIPSIZE size	<p>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.</p> <p>size              the size of one flash chip in bytes</p> <p>Example:        CHIPSIZE 0x80000</p>
BUSWIDTH width	<p>Enter the width of the memory bus that leads to the flash chips. Do not enter the width of the flash chip itself. The parameter CHIPTYPE carries the information about the number of data lines connected to one flash chip. For example, enter 16 if you are using two AM29F010 to build a 16bit flash memory bank.</p> <p>with              the width of the flash memory bus in bits (8   16   32)</p> <p>Example:        BUSWIDTH 16</p>
FILE filename	<p>The default name of the file that is programmed into flash using the Telnet 'prog' command. This name is used to access the file via TFTP. If the filename starts with a \$, this \$ is replace with the path of the configuration file name. This name may be overridden interactively at the Telnet interface.</p> <p>filename         the filename including the full path or \$ for relative path.</p> <p>Example:        FILE F:\gnu\mcf\bootrom.hex FILE \$bootrom.hex</p>
FORMAT format [offset]	<p>The format of the file and an optional address offset. The optional parameter "offset" is added to any load address read from the program file.</p> <p>format            SREC, BIN, AOUT, ELF or COFF</p> <p>Example:        FORMAT SREC FORMAT ELF 0x10000</p>
WORKSPACE address	<p>If a workspace is defined, the BDI uses a faster programming algorithm that runs out of RAM on the target system. Otherwise, the algorithm is processed within the BDI. The workspace is used for a 1kByte data buffer and to store the algorithm code. There must be at least 2kBytes of RAM available for this purpose.</p> <p>address          the address of the RAM area</p> <p>Example:        WORKSPACE 0x00000000</p>

ERASE addr [increment count] [mode [wait]]

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

- address      Address of the flash sector, block or chip to erase
- increment    If present, the address offset to the next flash sector
- count        If present, the number of equal sized sectors to erase
- mode         BLOCK, CHIP, UNLOCK  
Without this optional parameter, the BDI executes a sector erase. If supported by the chip, you can also specify a block or chip erase. If UNLOCK is defined, this entry is also part of the unlock list. This unlock list is processed if the Telnet UNLOCK command is entered without any parameters.  
**Note:** Chip erase does not work for large chips because the BDI time-outs after 3 minutes. Use block erase.
- wait         The wait time in ms is only used for the unlock mode. After starting the flash unlock, the BDI waits until it processes the next entry.

Example: ERASE 0xff040000 ;erase sector 4 of flash  
 ERASE 0xff060000 ;erase sector 6 of flash  
 ERASE 0xff000000 CHIP ;erase whole chip(s)  
 ERASE 0xff010000 UNLOCK 100 ;unlock, wait 100ms  
 ERASE 0xff000000 0x10000 7 ; erase 7 sectors

Example for the MCF5307 evaluation board flash memory:

```
[FLASH]
WORKSPACE 0x00800000 ;workspace in target RAM for fast programming algorithm
CHIPTYPE AM29F ;Flash type (AM29F | AM29BX8 | AM29BX16 | I28BX8 | I28BX16)
CHIPSIZE 0x80000 ;The size of one flash chip in bytes (e.g. AM29F010 = 0x20000)
BUSWIDTH 16 ;The width of the flash memory bus in bits (8 | 16 | 32)
FILE D:\abatron\bdi360\ColdFire\pro\sbc5307.sss
ERASE 0xFFE00000 ;erase sector 0 of flash
ERASE 0xFFE20000 ;erase sector 1 of flash
ERASE 0xFFE40000 ;erase sector 1 of flash
ERASE 0xFFE60000 ;erase sector 1 of flash
```

the above erase list maybe replaces with:

```
ERASE 0xFFE00000 0x20000 4 ;erase 4 sectors
```

**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

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

The AMD and AT49 algorithm are almost the same. The only difference is, that the AT49 algorithm does not check for the AMD status bit 5 (Exceeded Timing Limits).

Only the AMD and AT49 algorithm support chip erase. Block erase is only supported with the AT49 algorithm. If the algorithm does not support the selected mode, sector erase is performed. If the chip does not support the selected mode, erasing will fail. The erase command sequence is different only in the 6th write cycle. Depending on the selected mode, the following data is written in this cycle (see also flash data sheets): 0x10 for chip erase, 0x30 for sector erase, 0x50 for block erase.

To speed up programming of Intel Strata Flash and AMD MirrorBit Flash, an additional algorithm is implemented that makes use of the write buffer. This algorithm needs a workspace, otherwise the standard Intel/AMD algorithm is used.

The following table shows some examples:

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

**Note:**

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

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

or use the Telnet "unlock" command:

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

**addr** This is the address of the sector (block) to unlock

**delay** A delay time in milliseconds the BDI waits after sending the unlock command to the flash. For example, clearing all lock-bits of an Intel J3 Strata flash takes up to 0.7 seconds.

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

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

```
BDI> unlock 0xFFF000000 1000
```

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

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

**addr** This is the address of the first sector to erase or unlock.

**step** This value is added to the last used address in order to get to the next sector. In other words, this is the size of one sector in bytes.

**count** The number of sectors to erase or unlock.

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

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

**MCF52xxx internal flash (CFM):**

To erase and program the ColdFire Flash Module (CFM) you have to access it via the backdoor addresses (IPSBAR + 0x04000000). This backdoor address has to be used for erase and program commands. Following an example how to setup for CFM programming when IPSBAR is at the default address off 0x40000000.

```
[FLASH]
WORKSPACE 0x20000000 ;workspace in target SRAM for fast programming algorithm
CHIPTYPE   CFM       ;MCF52xxx internal flash
BUSWIDTH   32        ;The width of the flash memory bus in bits (8 | 16 | 32)
FILE       E:\cygwin\home\bdidemo\coldfire\evb5282_cmf.bin
FORMAT     BIN 0x44000000
ERASE      0x44000000 MASS
ERASE      0x44040000 MASS
```

Before you can erase/program the CFM, the CFM Clock Divider needs to be setup via an init list entry. Check the MCF52xxx user's manual about how to setup the CFMCLKD.

```
WM8 0x401D0002 0x54 ;CFMCLKD : Flash clock divider for 64MHz
```

### 3.2.5 Part [REGS]

In order to make it easier to access target registers via the Telnet interface, the BDI can read in a register definition file. In this file, the user defines a name for the register and how the BDI should access it (e.g. as memory mapped, memory mapped with offset, ...). The name of the register definition file and information for different registers type has to be defined in the configuration file.

The register name, type, address/offset/number and size are defined in a separate register definition file. This way, you can create one register definition file for the MCF5307 that can be used for all possible positions of the internal memory map. You only have to change one entry in the configuration file.

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

name	type	addr	size
name	The name of the register (max. 12 characters)		
type	The register type		
	DREG	Data register	
	AREG	Address register	
	CREG	Control register	
	MM	Absolute direct memory mapped register	
	DMM1...DMM4	Relative direct memory mapped register	
	IMM1...IMM4	Indirect memory mapped register	
	CMM1...CMM4	Control register based memory mapped register	
addr	The address, offset or number of the register		
size	The size (8, 16, 32) of the register		

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

FILE filename	The name of the register definition file. This name is used to access the file via TFTP. The file is loaded once during BDI startup.		
	filename	the filename including the full path	
	Example:	FILE C:\bdi\regs\mcf5485.def	
DMMn base	This defines the base address of direct memory mapped registers. This base address is added to the individual offset of the register.		
	base	the base address	
	Example:	DMM1 0x01000	
IMMn addr data	This defines the addresses of the memory mapped address and data registers of indirect memory mapped registers. The address of a IMMn register is first written to "addr" and then the register value is access using "data" as address.		
	addr	the address of the Address register	
	data	the address of the Data register	
	Example:	DMM1 0x02200000	

CMMn addr mask            This defines the control register that holds the base address of control register based memory mapped registers. The base address is added to the individual offset of the register.

                              addr            the address of the Control register

                              mask            the mask applied to the control register value before it is used as base address.

                              Example:        CMM2 0x008 0xFFFF0000 ; MMUBAR

**Note:**

The following register names are already predefined:

d0 .. d7, a0 .. a7, fp, sp, vbr, sr, pc, cacr, acr0, acr1, rambar, rombar, mbar

**Example for a register definition (MCF5485):**

Entry in the configuration file:

```
[REGS]
CMM1 0xC0F 0xfffc0000 ; MBAR
CMM2 0x008 0xffff0000 ; MMUBAR
FILE $reg5485.def
```

The register definition file:

```
;name            type    addr            size
;-----
;
; Additional Control Register
;
cacr            CREG   0x002            32
asid            CREG   0x003            32
acr0            CREG   0x004            32
.....
;
; CMM2 must point to MMUBAR
;
mmucr           CMM2   0x000            32
mmuor           CMM2   0x004            32
.....
;
; CMM1 must point to MBAR
;
;    System Integration Unit (SIU)
sdramds1        CMM1   0x004            32
sbcr            CMM1   0x010            32
cs0cfg01        CMM1   0x020            32
.....
```

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

```
BDI> rd mmucr
BDI> rm csar0 0xFFE0
```

### 3.3 Debugging with GDB

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

#### 3.3.1 Target setup

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

#### 3.3.2 Connecting to the target

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

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

```
(gdb)target remote bdi2000:2001
```

bdi2000                      This stands for an IP address. The HOST file must have an appropriate entry. You may also use an IP address in the form xxx.xxx.xxx.xxx

2001                         This is the TCP port used to communicate with the BDI

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

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

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

```
(gdb)...
```

```
(gdb)detach
```

... Wait until BDI has resetet the target and reloaded the image

```
(gdb)target remote bdi2000:2001
```

### 3.3.3 Breakpoint Handling

#### GDB versions before V5.0:

GDB inserts breakpoints by replacing code via simple memory read / write commands. There is no command like "Set Breakpoint" defined in the GDB remote protocol. When breakpoint mode HARD is selected, the BDI checks the memory write commands for such hidden "Set Breakpoint" actions. If such a write is detected, the write is not performed and the BDI sets an appropriate hardware breakpoint. The BDI assumes that this is a "Set Breakpoint" action when memory write length is 2 bytes and the pattern to write is a trap instruction (0x4E4?).

#### GDB version V5.x:

GDB version 5.x uses the Z-packet to set breakpoints (watchpoints). For software breakpoints, the BDI replaces code with a HALT instruction. When breakpoint mode HARD is selected, the BDI sets an appropriate hardware breakpoint.

#### User controlled hardware breakpoints:

The ColdFire has a special watchpoint / breakpoint hardware integrated. Normally the BDI controls this hardware in response to Telnet commands (BI, BDx) or when breakpoint mode HARD is selected. Via the Telnet commands BI and BDx, you cannot access all the features of the breakpoint hardware. Therefore the BDI assumes that the user will control / setup this breakpoint hardware as soon as TDR is written to with a value that is not zero. This way the debugger or the user via Telnet has full access to all features of this watchpoint / breakpoint hardware. Clearing TDR gives control back to the BDI.

### 3.3.4 GDB monitor command

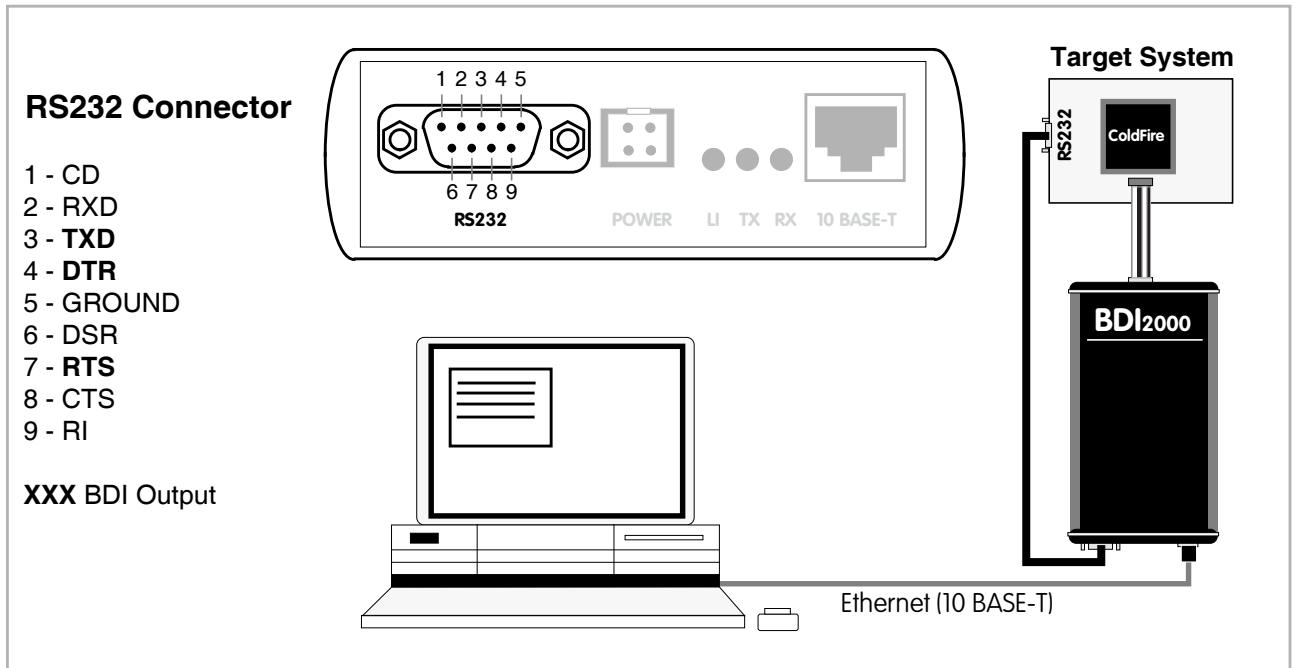
The BDI supports the GDB V5.x "monitor" command. Telnet commands are executed and the Telnet output is returned to GDB.

```
(gdb) target remote bdi2000:2001
Remote debugging using bdi2000:2001
0x10b2 in start ()
(gdb) mon break
Breakpoint mode is SOFT
(gdb) mon break hard

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

### 3.3.5 Target serial I/O via BDI

A RS232 port of the target can be connected to the RS232 port of the BDI2000. This way it is possible to access the target's serial I/O via a TCP/IP channel. For example, you can connect a Telnet session to the appropriate BDI2000 port. Connecting GDB to a GDB server (stub) running on the target should also be possible.



The configuration parameter "SIO" is used to enable this serial I/O routing. The BDI asserts RTS and DTR when a TCP connection is established.

```
[TARGET]
....
SIO    7    19200    ;Enable SIO via TCP port 7 at 9600 baud
```

#### Warning!!!

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

### 3.4 Telnet Interface

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

Telnet Debug features:

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

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

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

Following a list of the available Telnet commands:

```
"MD [<address>] [<count>] display target memory as word (32bit)",
"MDH [<address>] [<count>] display target memory as half word (16bit)",
"MDB [<address>] [<count>] display target memory as byte (8bit)",
"DUMP <addr> <size> [<file>] dump target memory to a file",
"MM <addr> <value> [<cnt>] modify word(s) (32bit) in target memory",
"MMH <addr> <value> [<cnt>] modify half word(s) (16bit) in target memory",
"MMB <addr> <value> [<cnt>] modify byte(s) (8bit) in target memory",
"MT <addr> <count> memory test",
"MC [<address>] [<count>] calculates a checksum over a memory range",
"MV verifies the last calculated checksum",
"RD [<name>] display CPU or user defined register",
"RDFP display floating point registers",
"RDUMP [<file>] dump all user defined register to a file",
"RM <name> <value> modify CPU or user defined register",
"TLB <from> [<to>] display TLB entry (only V4e cores)",
"WTLB <idx> <epn> <rp> write TLB entry (only V4e cores)",
"RESET reset the target system",
"BREAK [SOFT | HARD] display or set current breakpoint mode",
"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 hardware breakpoint",
"CI [<id>] clear instruction hardware breakpoint(s)",
"BD [R|W] <addr> set data watchpoint (32bit access)",
"BDH [R|W] <addr> set data watchpoint (16bit access)",
"BDB [R|W] <addr> set data watchpoint ( 8bit access)",
"CD [<id>] clear data breakpoint(s)",
"INFO display information about the current state",
"LOAD [<offset>] [<file> [<format>]] load program file to target memory",
"VERIFY [<offset>] [<file> [<format>]] verify a program file to target memory",
"PROG [<offset>] [<file> [<format>]] program flash memory",
" <format> : SREC or BIN or AOUT or ELF",
"ERASE [<address> [<mode>]] erase a flash memory sector, chip or block",
" <mode> : CHIP, BLOCK or SECTOR (default is sector)",
"ERASE <addr> <step> <count> erase multiple flash sectors",
"UNLOCK [<addr> [<delay>]] unlock a flash sector",
"UNLOCK <addr> <step> <count> unlock multiple flash sectors",
"DELAY <ms> delay for a number of milliseconds",
"HOST <ip> change IP address of program file host",
"PROMPT <string> defines a new prompt string",
"CONFIG display or update BDI configuration",
"CONFIG <file> [<hostIP> [<bdiIP> [<gateway> [<mask>]]]]",
"HELP display command list",
"BOOT [loader] reboot the BDI and reload the configuration",
"QUIT terminate the Telnet session"
```

**Note:**

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

**Note:**

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

## 4 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. B)
Operating Temperature	+ 5 °C ... +60 °C
Storage Temperature	-20 °C ... +65 °C
Relative Humidity (noncondensing)	<90 %rF
Size	190 x 110 x 35 mm
Weight (without cables)	420 g
Host Cable length (RS232)	2.5 m

Specifications subject to change without notice

## 5 Environmental notice

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

## 6 Declaration of Conformity (CE)



**DECLARATION OF CONFORMITY**

This declaration is valid for following product:

**Type of device: BDM/JTAG Interface**  
**Product name: BDI2000**

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

**EMC Directive 89/336/EEC**

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



**EN 50081-2**  
**EN 50082-2**

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

Manufacturer:

**ABATRON AG**  
**Stöckenstrasse 4**  
**CH-6221 Rickenbach**

Authority:

 Max Vock Marketing Director	 Ruedi Dummermuth Technical Director
---	--

Rickenbach, May 30, 1998

## 7 Abatron Warranty and Support Terms

### 7.1 Hardware

ABATRON Switzerland warrants that the Hardware shall be free from defects in material and workmanship for a period of 3 years following the date of purchase when used under normal conditions. Failure in handling which leads to defects or any self-made repair attempts are not covered under this warranty. In the event of notification within the warranty period of defects in material or workmanship, ABATRON will repair or replace the defective hardware. The customer must contact the distributor or Abatron for a RMA number prior to returning.

### 7.2 Software

#### License

Against payment of a license fee the client receives a usage license for this software product, which is not exclusive and cannot be transferred.

#### Copies

The client is entitled to make copies according to the number of licenses purchased. Copies exceeding this number are allowed for storage purposes as a replacement for defective storage mediums.

#### Update and Support

The agreement includes free software maintenance (update and support) for one year from date of purchase. After this period the client may purchase software maintenance for an additional year.

### 7.3 Warranty and Disclaimer

ABATRON AND ITS SUPPLIERS HEREBY DISCLAIMS AND EXCLUDES, TO THE EXTENT PERMITTED BY APPLICABLE LAW, ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION, ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT.

### 7.4 Limitation of Liability

IN NO EVENT SHALL ABATRON OR ITS SUPPLIERS BE LIABLE TO YOU FOR ANY DAMAGES, INCLUDING, WITHOUT LIMITATION, ANY SPECIAL, INDIRECT, INCIDENTAL OR CONSEQUENTIAL DAMAGES, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THE HARDWARE AND/OR SOFTWARE, INCLUDING WITHOUT LIMITATION, LOSS OF PROFITS, BUSINESS, DATA, GOODWILL, OR ANTICIPATED SAVINGS, EVEN IF ADVISED OF THE POSSIBILITY OF THOSE DAMAGES.

The hardware and software product with all its parts, copyrights and any other rights remain in possession of ABATRON. Any dispute, which may arise in connection with the present agreement shall be submitted to Swiss Law in the Court of Zug (Switzerland) to which both parties hereby assign competence.

## 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)

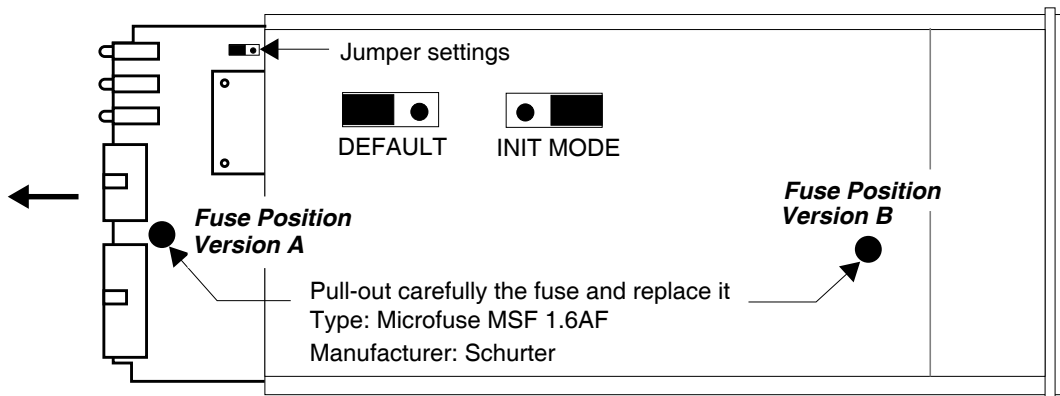
2.2 Remove the two screws that hold the front panel

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

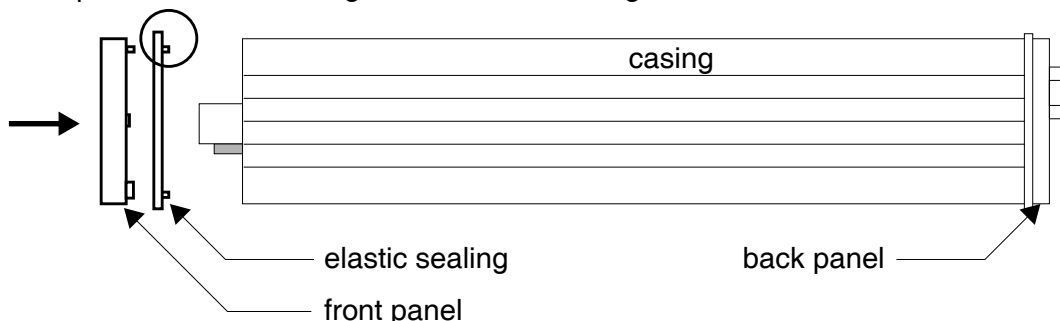


**5**

**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.