BDI Configuration Guide

1. Overview

When starting out with the BDI2000/BDI3000 (BDI), generating a correct configuration file can be a difficult process at times. A configuration depends on several things; the processor used, the devices present on the board, and mainly what you are planning to do. For example, a configuration to program the boot flash is not the same as one used to debug boot ROM code starting from the reset vector. A developer needs to have very detailed knowledge of the processor used and the board (especially relating to the FLASH and SDRAM subsystems) in order to successfully create a BDI configuration file.

The configuration files provided on the distribution CD or on ftp://94.230.212.16/bdigdb/config/ are only examples and can be used as a reference point. They mostly cover instances where you want to load code directly into RAM and then debug it. A developer needs to understand what every single line of the configuration file is doing in order to resolve any problems. This document gives information on how to create configuration files for different needs.

Note: The BDI itself does not need a register definition file (*.def). The information in this file is only used to allow users to access memory or registers via Telnet by their names instead of addresses or numbers.

2. Create a BDI Configuration File

As mentioned before, a configuration depends mainly on what you are planning to do, so this is covered in several different chapters.

2.1. Configuration for debugging boot code

If there is already boot code present in the boot flash and you want to debug it from the reset vector, then the configuration is straight forward. The boot code expects a virgin system so you must not modify the state of the system with [INIT] list entries. The [INIT] list is empty or almost empty depending on the processor. What you may need in the [INIT] list are entries to disable the watchdog and setup the initial TLB for some CPUs such as the Applied Micro PPC440/460 processors.
The BREAKMODE should be set to HWBP because we cannot replace code in flash.

Start with this type of configuration also for the initial connection to a new board. Once you get control of the processor you can add starting [INIT] list entries.

The following processors need additional [INIT] list entries:

MPC5xx/8xx: Add entry to disable the watchdog

PPC44x/46x: Add entry to setup the boot page TLB

2.2. Configuration for programming NOR flash

For a configuration that allows programming NOR flash, add [INIT] list entries to setup the memory controller. For faster programming with a WORKSPACE, also add entries that enable (if present) internal SRAM or setup SDRAM. There are many configuration examples that show this type of configuration. If there is working boot code already on the board, let it first setup the system and then read back the relevant register values. You can then use these values to create the BDI configuration file.

The process is an engineering effort. Every processor and board combination is different. Take a look at one of the example configuration files found in the distribution CD. Walk through each line in the [INIT] section with the corresponding processor User's Manual and other chip documentation and determine what each step is doing to the CPU/board. That will give you a good head start on what you will need to write one from scratch.

Hint: With "reset run" entered at the Telnet you can reset the system and execute the boot code without processing any [INIT] list entries.

Important: The BDI needs access to the whole flash in order to successfully erase/program the device, even when you only plan to erase/program the top part of the flash. This is because the BDI calculates the base address of the flash based on the CHIPSIZE. The unlock sequences are written relative to this base address. If this part of the flash is not accessible, erase/program will fail. In this case either setup the memory controller correctly or reduce CHIPSIZE in the [FLASH] section to the visible flash chip size.

2.3. Configuration for loading and debugging code in RAM/SDRAM

For loading and debugging code directly with GDB in RAM/SDRAM, the memory controller needs to be setup so the RAM/SDRAM can be used. The configuration is very similar to the one used for programming flash with a workspace. Most of the example configurations cover this case. This is the most complex configuration because the correct execution environment needs to be setup for the application you plan to load and execute. Additional TLB's may need to be setup to allow access to peripheral devices from within the loaded application.

Creating the correct configuration could be a complex task, but the designer/user of a system needs an in-depth understanding of the processor and board in order to develop code for it. So creating a BDI configuration file is one step in learning more about the processor and board.

Additional information is found in:

- bdiGDB User's Manual
- Processor Reference Manual
- Board Reference Manual (if available)
- Example Configurations on the distribution CD
- Example Configurations on ftp://94.230.212.16/bdigdb/config/