BLUE TECHNIX

Software

Developer Guide

www.tinyboards.com

Maximum Power at Minimum Size

Contact

Bluetechnix Mechatronische Systeme GmbH
Waidhausenstr. 3/19
A-1140 Vienna
AUSTRIA/EUROPE
office@bluetechnix.at
http://www.bluetechnix.at
http://www.tinyboards.com

Version 1.1

2005-03-07

Document No. 099-0001-11

Table of Contents

The information herein is given to describe certain components and shall not be considered as a guarantee of characteristics.
Terms of delivery and rights of technical change reserved.

We hereby disclaim any warranties, including but not limited to warranties of non-infringement, regarding circuits, descriptions and charts stated herein.

Bluetechnix makes and you receive no warranties or conditions, express, implied, statutory or in any communication with you. Bluetechnix specifically disclaims any implied warranty of merchantability or fitness for a particular purpose.

Bluetechnix takes no liability for any damages and errors causing of the usage of this board. The user of this board is responsible by himself for the functionality of his application. He is allowed to use the board only if he has the qualification. More information is found in the General Terms and Conditions (AGB).

Information
For further information on technology, delivery terms and conditions and prices please contact Bluetechnix (http://www.bluetechnix.at).

Warnings
Due to technical requirements components may contain dangerous substances

The Core Boards and Development systems contain ESD (electrostatic discharge) sensitive devices. Electro-static charges readily accumulate on the human body and equipment and can discharge without detection. Permanent damage may occur on devices subjected to high-energy discharges. Proper ESD precautions are recommended to avoid performance degradation or loss of functionality. Unused core boards and development boards should be stored in the protective shipping package.

1 Developing projects with the Blackfin Core Modules

1.1 Developing projects using a JTAG device:

Our core modules works with every JTAG-device provided by ANLOG DEVICES INC. Please contact Bluetechnix for purchasing a JTAG device or look at www.tinyboards.com for further information.
If you have a JTAG devices connected to the core module create an appropriate target platform with the VDSP++ configurator, depending on the core module you are using. Use this target device to connect your JTAG with the core module. Refer to the VDSP++ Manuals to see how to use the configurator.

You can develop projects and download the executable via the JTAG device. You have full control over the blackfin processor and full debug capabilities. Refer to the Compiler and Linker Manuals and the User Guides from Analog devices to learn more about developing projects.
Creating a loader file with the VDSP++ you can flash this file, using the integrated flash programming tool to create standalone applicatons. As flash driver you have to use the appropriate *.dxe flash driver file located on your support CD in the directory *"VDSPFlashToolDriver*", depending on witch core module you use. Refer to the VDSP++ manual, how to use the flashing tool.

If you are using a JTAG devices it is important that you have the correct register reset settings for the EBIU registers after starting the VDSP++. The VDSP++ development environment initializes this registers once started, according to the definitions in the xml-files located in the *"System/ArchDef"* directory in the installation path of the VDSP++. The relevant files are named "ADSP-BF5xx-proc.xml". We provide files with the correct settings for our core modules on the support CDs within the directory *"Common\DevelopmentTools\config-files"*. Save the old files within the directory and overwrite the files with the files on your support CD.
Be aware that if you make an update of the VDSP++ development environment also the xml-files will be overwritten by the installation process, so you have to copy the files again, after the installation process has finished.

1.1.1 Valid linker description files (*.ldf)

Using a JTAG device you are completely free in using customized linker description files. The only things to take in consider is the sd-ram size of the core modules, 32Mb starting at 0x00000000. We provide sample linker files for our core modules on the support CD within the directory *"Common\DevelopmentTools\ldf"*

1.1.2 Valid loader files (*.ldr)

Refer to the VDSP++ Linker and Loader Manual to see how to create loader files that can be booted by the blackfin processor.

If you have sections in the sd-ram that has to be initialized during the boot process you have to use an initialisation file. This initialisation function sets up the sd-ram prior to boot the application. As the sd-ram settings depends on the sd-ram type used it is important to use the appropriate initialisation file that matches the sd-ram used. For each core module we provide appropriate initialisation files (*.dxe) on the support CD within the directory "*Common\DevelopmentTools\initcode*". Select the appropriate *.dxe file that correspond to your core module as initialisation file within the loader settings.
On the CM-BF561 you have to use the file "bf561_prom16.dxe" as boot loader (second stage loader) not as initialisation file! This is necessary because the BF561 is a dual core and so a second stage loader is needed to boot both cores, otherwise only one core is booted. If you don't use a boot file on a BF561 project for creating a loader file the elfloader is reporting a warning.

Correct settings to create valid loader files:
(You can find the settings under the "Project options→Load")

For the CM-BF533:

Boot mode	**Boot format**	**Output width**	
PROM	Intel hex	16 bit	Default start address

As initialisation file use:
Common\DevelopmentTools\initcode\CM-BF533\Debug\Init_code.dxe
No Boot Kernel
No ROM Splitter

For the CM-BF537:

Boot mode	**Boot format**	**Output width**	
PROM	Intel hex	16 bit	Default start address

As initialisation file use:
Common\DevelopmentTools\initcode\CM-BF537\Debug\Init_code.dxe
No Boot Kernel
No ROM Splitter

For the CM-BF561:

Boot mode	**Boot format**	**Output width**	
PROM	Intel hex	16 bit	Default start address

No intialisation file
As Boot Kernel use:
Common\DevelopmentTools\initcode\CM-BF561\Release\ bf561_prom16.dxe
No ROM Splitter

Be sure to have selected the correct boot mode "Boot from 16 bit Flash/PROM" on your target board. Refer to the appropriate "Hardware User Manual" to see how to set boot modes.

1.2 Developing projects without JTAG device:

If you don't have a JTAG device you can use the simulator from the VDSP++ development environment to develop your application. Create a loader file from your project. This loader file (*.ldr) can be downloaded to the core module using the download features from the BLACKSheep. Depending on the BLACKSheep and core modul version you have different download capabilities either over the UART with the xmodem protocol or over a standard http browser. Refer to the *"BLACKSheep Command Reference"* to see how to transfer files with the "*xmt*" command.

With the flash programming tool integrated to the BLACKSheep you can flash the application either as application booted by the BLACKSheep or as standalone application.
Be aware that flashing a program as standalone application overwrites the BLACKSheep code and the only way to reflash the core module is to use a JTAG device.

If you have flashed a program as application you can start it by using the "load" command from the BLACKSheep. After booting prior to enter to the command shell, the BLACKSheep looks if there is a valid application in the flash and boots the application if no key is pressed.

There are no debug features included in the BLACKSheep, but you can debug the application in the simulator of the VDSP++.

Refer to the *"BLACKSheep Command Reference"* for a description of the relevant commands.

1.2.1 Application boot process

The application boot process of the BLACKSheep is similar to that of the bootloader from the blackfin located in the boot rom of the processor.
The BLACKSheep application loader looks for a valid dxe header in the *.ldr file or the flash and copies each section defined in the loader file into the appropriate memory section.
After the copy process has finished it clears all interrupts and jumps to the start address of the internal code memory (0xFFA00000).
The code for the application loader resides in the first 8kb of the sd-ram memory. So the developer has to take care to avoid sections that have to be initialized at boot time that are located within this first 8kb. You can easily do this by using appropriate linker files. We provide valid linker files on the support CD within the directory *"Common\DevelopmentTools\ldf"*. This linker files avoids having code or data in this memory sections and provides correct sd-ram size for our core modules.

1.2.2 Flash usage

The command *"flash –ff"* of the BLACKSheep shows you the flash usage of the BLACKSheep. The BLACKSheep resides on bottom of the flash beginning with address 0x20000000. Applications will be stored on the top of the BLACKSheep code, starting on the first free sector. After the BLACKSheep and the application has booted, the flash is not used anymore and is free to be used by the user application.

The flash size needed by the BLACKSheep code is around 384kb, depending on the BLACKSheep version.

Be aware to don't write in sectors where the application or the BLACKSheep code is stored, otherwise the BLACKSheep and or the application will not start anymore. It is recommended to start at the top of the flash to store user data.

1.2.3 Valid linker description files (*.ldf)

As described above you are free to use your own linker file as long you be aware to **don't have any sections that has to be initialized at boot time within the first 8kb** (code of the application loader), and you take in consider the sd-ram size of 32Mb starting at 0x00000000. We provide sample linker files for our core modules on the support CD within the directory *"Common\DevelopmentTools\ldf"*

1.2.4 Valid loader files (*.ldr)

Refer to section 1.1.2 to see how to create valid loader files. If you want to flash your application with the "–b" option (Refer to the *"BLACKSheep Command Reference"*) to substitute the BLACKSheep the same things as described in the above section are valid. If you want to create loader files to be started by the BLACKSheep either from flash or with the *"exec"* command, there are two things you have to take in consider.

On **CM-BF533** and **CM-BF537 don't** use the initialisation file as the sd-ram is already initialized by the BLACKSheep!

On **CM-BF561** don't use the boot kernel as the BLACKSheep acts as second stage loader!

2 Revision History

2005 05 03	First release V1.0 of the Document
2006 03 07	Update to V1.1

A List of Figures and Tables