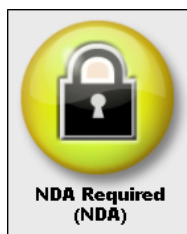


AMIBIOS8 ROM Utility User Guide

AFUBSD / AFULNX

Document Revision 1.0.0 - May 13, 2009

NDA REQUIRED



Copyright (c) 2008 American Megatrends, Inc.

All Rights Reserved.

American Megatrends, Inc.

5555 Oakbrook Parkway

Suite 200

Norcross, GA 30093

This publication contains proprietary information which is protected by copyright. No part of this publication may be reproduced, transcribed, stored in a retrieval system, translated into any language or computer language, or transmitted in any form whatsoever without the prior written consent of the publisher, American Megatrends, Inc.

American Megatrends, Inc. retains the right to update, change, modify this publication at any time, without notice.

For Additional Information

Call American Megatrends BIOS Sales Department at 1-800-828-9264 for additional information.

Limitations of Liability

In no event shall American Megatrends be held liable for any loss, expenses, or damages of any kind whatsoever, whether direct, indirect, incidental, or consequential, arising from the design or use of this product or the support materials provided with the product.

Limited Warranty

No warranties are made, either express or implied, with regard to the contents of this work, its merchantability, or fitness for a particular use. American Megatrends assumes no responsibility for errors and omissions or for the uses made of the material contained herein or reader decisions based on such use.

Trademark and Copyright Acknowledgments

All product names used in this publication are for identification purposes only and are trademarks of their respective Companies.

Disclaimer

This manual describes the operation of the AMIBIOS8 ROM Utilities. Although efforts have been made to insure the accuracy of the information contained here, American Megatrends expressly disclaims liability for any error in this information, and for damages, whether direct, indirect, special, exemplary, consequential or otherwise, that may result from such error, including but not limited to the loss of profits resulting from the use or misuse of the manual or information contained therein (even if American Megatrends has been advised of the possibility of such damages). Any questions or comments regarding this document or its contents should be addressed to American Megatrends at the address shown on the cover.

American Megatrends provides this publication "as is" without warranty of any kind, either expressed or implied, including, but not limited to, the implied warranties of merchantability or fitness for a specific purpose.

Some states do not allow disclaimer of express or implied warranties or the limitation or exclusion of liability for indirect, special, exemplary, incidental or consequential damages in certain transactions; therefore, this statement may not apply to you. Also, you may have other rights which vary from jurisdiction to jurisdiction.

This publication could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. American Megatrends may make improvements and/or revisions in the product(s) and/or the program(s) described in this publication at any time. Requests for technical information about American Megatrends products should be made to your American Megatrends authorized reseller or marketing representative.

Revision Information

| Date | Rev | Description of Changes |
|------------|-------|------------------------|
| 2009-05-13 | 1.0.0 | Initial Document. |

Overview

AFUBSD/AFULNX (AMI Firmware Update) is an updating system BIOS utility with command line interface. It has same parameters and behavior as AFUDOS.

For the convenience, AFULNX has a big change to related drivers. All necessary drivers are generated and loaded automatically since version 4.10. User can just launch the program and wait for updating job finish.

By the way, do not forget that target board MUST be **AMIBIOS** system while using this utility.

Features

This utility offers the following features:

- No need to build driver by yourself for different distributions of Linux(v4.10 or above)
- Small executable file size
- Quickly update
- Clear updating information and status
- Fully compatible with previous version (See [Appendix B AFUBSD/AFULNX v3.xx Commands](#))

Requirements

Supported Operating System

AFULNX Utility is supported in following operating system:

- Linux CORE v2.4/2.6

AFUBSD Utility is supported in following operating system:

- FreeBSD operating system

BIOS Requirements

System BIOS should have the followings:

- AMIBIOS CORE version 8.xx.xx.
- **SMIFlash** eModule with “8.00.00_SMIFlash-1.00.07” label or later.
- *Token: **SDSMGR_IN_RUNTIME** = ON.*
- *Token: **SMI_INTERFACE_FOR_SDSMGR_FUNC** = ON.*

Operating System Driver Requirements for AFULNX only

You can forget this section if you are using AFULNX v4.10 or above.

Following driver for different version of Linux system are required by this utility:

- **UCORELNX.O32** Driver for 32-Bit Linux.
- **UCORELNX.O64** Driver for 64-Bit Linux.

Getting Started for AFULNX v4.06 and below

Preparing suitable driver file

1. Log in Linux as root.
2. The compiler suite (GCC) must be installed. If these packages are not installed, the driver CANNOT be built.
3. Kernel sources must be installed, **CONFIGURED**, and then compiled. Following are steps to do this:
 1. Find running kernel's configuration file:
 - To configure the sources, simply change to the kernel source directory (typically /usr/src/linux). If it doesn't exist, you need to install kernel source. Typically, the reference configuration for the kernel can be found in the /boot directory with filename '.config', 'kernel.config', or 'vmlinux-2.4.18-3.config'. Type 'uname -a' and use the configuration filename that best matches the output from 'uname -a'.
 - On some distributions, Red Hat, for instance, there is a config directory under /usr/src/linux.
 - Copy this configuration file into the root of the linux kernel source tree (usually it is /usr/src/linux). The file must be renamed to “.config” (dot config).
 2. Make the Linux kernel:
 - Under linux kernel source root (/usr/src/linux), type the command 'make' if your Linux kernel version is 2.6 or above; otherwise use 'make oldconfig dep' instead.
 - This will generate files that are required to build the driver.
 - The process of compiling the Linux kernel will take a while to accomplish
 3. Copy your AMI flash driver:
 - The AMI flash driver is distributed in a compressed TAR archive. After saving this file to your Linux system, it must be extracted so that the driver may be built.
 - First, create a directory for AFU with the command 'mkdir afu'. Change your working path into it by 'cd afu'.
 - To extract the archive, you'll need to run the shell command(as root):
tar xvzf afulnx2.tgz
 - The name of the archive may be different, but the overall syntax is the same.
 4. Determining which driver Makefile to use:
 - Due to a change made into Linux kernel 2.6. The Makefile for 2.6 is different from older kernel.
 - Type 'uname -r' to see the version.
 - If your kernel version is 2.6 or greater, copy the file 'Makefile.v26' to 'Makefile'.
 - If your kernel version is below 2.6, copy the file 'Makefile.v24' to 'Makefile'.
 - The command to do the copy is 'cp Makefile.v2x Makefile'.
 5. Make your AMI flash driver (UCORELNX.O32/UCORELNX.O64):
 - For For most distribution, the command to build the driver is:
make
 - If your linux's kernel source tree is under /usr/src/linux-2.4 instead of the default path '/usr/src/linux', add a KERNEL flag:
make KERNEL=/usr/src/linux-2.4

- For Red Hat 8.0 distribution with kernel located under /usr/src/linux-2.4, use this command:
make REDHAT9=1 KERNEL=/usr/src/linux-2.4
- If KERNEL is omitted, the default is /usr/src/linux. This should work for MOST distributions.
- To clean up object file, use the command:
make clean
Or
make REDHAT9=1 KERNEL=/usr/src/linux-2.4 clean

6. Check your build:

- Check the version of running Linux kernel with 'uname -r'.
- Check the version of UCORELNX.O32/UCORELNX.O64 with 'modinfo ucorelnx.032' and 'modinfo ucorelnx.o64'.
- If they mismatch, you will need to select the correct configuration file(.config), rebuild your kernel, and then rebuild your driver as described in (3.1), (3.2), and (3.5).

Installation

- Copies *AFULNX2_32*, *AFULNX2_64*, *UCORELNX.O32*, *UCORELNX.O64* to any storage location accessible by the host system.
- To determine which version of Linux system, type 'uname -m'. under shell screen. If it says 'x86_64', then *AFULNX2_64* should be used; otherwise, use the *AFULNX2_32*.
- Run the suitable file and remember to keep the relative driver in same directory.

Troubleshooting

| | |
|-----------|---|
| Q1 | I get following error message when loading driver: "insmod: error inserting 'UCORELNX.O32' 'UCORELNX.O64': -1 Invalid module format". |
| A1 | Most likely this is cause by wrong configuration file and your kernel refuses to accept your driver because version strings(more precisely, version magic) do not match. To check the version of running Linux kernel, type "uname -r". To check the version of UCORELNX.O32/UCORELNX.O64, type "modinfo UCORELNX.O32" or "modinfo UCORELNX.O64". If they mismatch, you will need to select the correct configuration file(.config), rebuild your kernel, and then rebuild your driver as described in "preparing suitable driver file" section. |
| Q2 | When I run ./afulnx2, it says "Unable to load driver". |

| | |
|-----------|--|
| A2 | Some Linux distributions do not display driver debug messages on screen by default. Type "dmesg" to see those debug messages. This is very likely the same problem as Q1. |
| Q3 | When I run ./afulnx2, it simply freezes. |
| A3 | <p>This is caused by a Linux feature called "NMI Watchdog" which is used to debug Linux kernel. This feature must be disabled to run AFULNX2.</p> <p>Please do "cat /proc/interrupts" twice and check if NMI is counting.</p> <p>If it is, then boot Linux with a kernel parameter "nmi_watchdog=N" where N is either 0, 1 or 2.</p> <p>Find out which configuration can halt NMI from counting by "cat /proc/interrupts". This is the configuration we should use to run AFULNX2.</p> |

Usage

For previous usage, see [Appendix B AFUBSD/AFULNX v3.xx Commands](#) to know details.

AFUBSD/AFULNX <BIOS ROM File Name> [Option 1] [Option 2].....

Or

AFUBSD/AFULNX <Output BIOS ROM File Name> <Commands>

Or

AFUBSD/AFULNX /M<MAC Address>

Or

AFUBSD/AFULNX /MAI

BIOS ROM File Name

The mandatory field is used to specify path/filename of the BIOS ROM file with extension.

Commands

The mandatory field is used to select an operation mode:

- **/O** Save current ROM image to file
- **/U** Get and display ROM ID from ROM file
- **/Ln** Refer to option: **/Ln**
- **/M <MAC Address>** Refer to option: **/M**
- **/MAI** Display current system and ROM file's MA
- **/HOLE** Update specific ROM hole by given name
- **/HOLEOUT** Save specific ROM holedata by given name
- **/D** Verification test of given ROM file without flashing

- **/EC** Flash EC firmware BIOS (Refer to OFBD spec)
Path: \$BIOS/Corebin/800/ROMUtils/On Flash Block Description Specification.PDF.
Sample Code Module Path:
\$BIOS/Examples/On Flash Block Description
- **/NCB** Flash NCB data by given name
- **/NCBOUT** Output NCB data by given name
- **/C** Destroy CMOS checksum

Options

The optional field used to supply more information for flashing BIOS ROM. Following lists the supported optional parameters and format:

- **/P** Program main bios image
- **/B** Program Boot Block
- **/N** Program NVRAM
- **/C** Destroy CMOS after update BIOS done
- **/E** Program Embedded Controller block if present
- **/K** Program all non-critical blocks
- **/Kn** Program n'th non-critical block only (n=0 – 7)
- **/Q** Quiet mode enable
- **/REBOOT** Reboot after update BIOS done
- **/X** Do not check ROM ID
- **/S** Display current system's BIOS ROM ID
- **/Ln** Load CMOS default (n=0 - 1)
 - L0: Load current CMOS optimal settings
 - L1: Load current CMOS failsafe settings
 - L2: Load CMOS optimal settings from ROM file
 - L3: Load CMOS failsafe settings from ROM file
- **/M<MAC Address>** Update BootBlock MAC address if exists
- **/R** Preserve all SMBIOS structures during NVRAM programming
- **/Rn** Preserve specific SMBIOS structure during NVRAM programming
- **/ECUF** Update EC BIOS when newer version is detected.
- **/ShutDown** Shutdown system after programming.
- **/clevnlog** Clean Event Log.
- **/DeDftCfg** Delete all default settings from BIOS.
- **/-Command Name** Delete certain command's default setting.
[OEM Uses Only.]

Rules

- **Any parameter enclosed by < > is a mandatory field.**
- **Any parameter enclosed by [] is an optional field.**
- <Commands> cannot co-exist with any [Options].
- Main BIOS image is default flashing area if no any option present.
- [/C], [/Q], [/REBOOT], [/X], [/Ln] and [/S] will enable [/P] function automatically.
- If [/B] present alone, there is only the Boot Block area to be updated.
- If [/N] present alone, there is only the NVRAM area to be updated.
- If [/E] present alone, there is only the Embedded Controller block to be updated.
- If [/E] and [/ECUF] co-exist, [/ECUF] will be no function.
- If [/Kn] present alone, there is only non-critical block to be updated.
- When [/Ln] is co-exist with [/C], [/C] will be no function.
- [/M] can be used as a command for backward compatible.

Note : Running AFUBSD/AFULNX under command prompt directly will display help message.

Examples

Examples on how to update BIOS using the command prompt are shown in following:

- **Save current BIOS ROM to file**
AFUBSD/AFULNX <BIOS ROM File Name> /O
- **Get and display ROM ID from BIOS ROM file**
AFUBSD/AFULNX <BIOS ROM File Name> /U
- **Update main BIOS image only**
AFUBSD/AFULNX <BIOS ROM File Name>
Or
AFUBSD/AFULNX <BIOS ROM File Name> /p
- **Update Boot Block only**
AFUBSD/AFULNX <BIOS ROM File Name> /B
- **Update NVRAM only**
AFUBSD/AFULNX <BIOS ROM File Name> /N
- **Update Embedded Controller Block only**
AFUBSD/AFULNX <BIOS ROM File Name> /E
- **Update Embedded Controller Block if newer version is detected**
AFUBSD/AFULNX <BIOS ROM File Name> /ECUF
- **Update 2nd non-critical block only**
AFUBSD/AFULNX <BIOS ROM File Name> /K2
- **Update main BIOS image, Boot Block and NVRAM at once**
AFUBSD/AFULNX <BIOS ROM File Name> /P /B /N

- **Update whole BIOS ROM**
AFUBSD/AFULNX <BIOS ROM File Name> /P /B /N /C /E /K
- **Update whole BIOS ROM and load current CMOS optimal settings**
AFUBSD/AFULNX <BIOS ROM File Name> /P /B /N /C /E /K /L0
- **Update whole BIOS without checking ROM ID**
AFUBSD/AFULNX <BIOS ROM File Name> /P /B /N /C /E /K /X
- **Update whole BIOS with quiet execution**
AFUBSD/AFULNX <BIOS ROM File Name> /P /B /N /C /E /K /Q
- **Update whole BIOS in quiet mode and REBOOT quietly**
AFUBSD/AFULNX <BIOS ROM File Name> /P /B /N /C /E /K /Q /REBOOT
- **Update BootBlock MAC address**
AFUBSD/AFULNX /M<MAC Address>
- **Update whole BIOS and BootBlock MAC address**
AFUBSD/AFULNX <BIOS ROM File Name> /P /B /N /C /E /K /M<MAC Address>
- **Update whole BIOS except existing SMBIOS structures**
AFUBSD/AFULNX <BIOS ROM File Name> /P /B /N /C /E /K /R
- **Update whole BIOS but preserve SMBIOS type 0 and 11**
AFUBSD/AFULNX <BIOS ROM File Name> /P /B /N /C /E /K /R0 /R11
- **Update dedicate ROM Hole Area**
AFUBSD/AFULNX <ROM Hole File Name> /Hole:Name
- **Update dedicate NCB Area**
AFUBSD/AFULNX <NCB File Name> /NCB:Name
- **Output dedicate ROM Hole File**
AFUBSD/AFULNX <Output ROM Hole File Name> /HOLEOUTt:Name
- **Output dedicate NCB File**
AFUBSD/AFULNX <Output NCB File Name> /NCBOUT:Name
- **Cancel Embedded AFU default commands**
- Below sample cancels B & P commands if BIOS has embedded B & P commands in OFBD.
AFUBSD/AFULNX <BIOS ROM File Name> /-B /-P
Notice: if /p & /b are set as default command only and /-B /-P commands are issued then P command will still be issued because if none of command is issued then /p will still issue as AFU default.
- **Cancel ALL Embedded AFU default commands**
AFUBSD/AFULNX <BIOS ROM File Name> /DeDftCfg

Getting Started for AFULNX v4.10 or above

Installation

- Copies **AFULNX2.TGZ** to any storage location accessible by the host system.
- Extracts the contents to same directory. You will get two folders, one is **AFULNX2_24** and the other is **AFULNX2_26**.
- Type 'uname -r' to identify kernel version. If it says 2.4.xx..., you should enter **AFULNX2_24** folder, otherwise enter **AFULNX2_26** folder.

- To determine which version of Linux system, type 'uname -m' under shell screen. If it say 'x86_64', then **AFULNX_64** should be used; otherwise use **AFULNX_32**.
- Run the suitable file. AFULNX will generate necessary drivers and load ot automatically. If AFULNX cannot work well, please refer to “Generating driver file manually” or “Troubleshooting” section to get help.

Generating driver file manually

1. Log in Linux as root.
2. The compiler suite (gcc) mustbe installed. If these packages are not installed, the driver **CANNOT** be built
3. For most distributions, **AFULNX2** will generate AMI Flash Driver file automatically without notification. Certainly, the driver file may NOT be generated in some specific case and the loading driver failure message will be displayed. If you get this error, the first step is to read 'Q1' and 'Q2' in 'Troubleshooting' section to shut out the kernel issues, and second step is to check Point.4 below to create driver file by yourself and launch AFULNX2 again.
4. Kernel sources must be installed, **CONFIGURED**, and then compiled. Following are steps to do this:
 1. Find running kernel's configuration file
 - To configure the sources, simply change the kernel source directory (typically /lib/module/\$(uname -r)/build). If it doesn't exist, you need to install kernel source. Typically, the reference configuration for the kernel can be found in the /boot directory with filename '.config', 'kernel.config', or 'vmlinux-2.4.18-3.config'. Type 'uname -a' and use the configuration filename that best matches the output from 'uname -a'.
 - On some distributions, Red Hat, for instance, there is a config directory under /lib/modules/\$(uname -r)/build.
 - Copy this configuration file into root of the linux kernel source tree (usually it is /lib/modules/\$(uname -r)/build). This file must be renamed to '.config' (dot config).
 2. Make your AMI flash driver (amifldr_mod.drv)
 - For most distribution, the command to build the driver is:
AFULNX_32 /MAKEDRV
Or
AFULNX_64 /MAKEDRV
 - If your linux's kernel source tree is under /lib/modules/\$(uname -r)/build instead of the default path '/lib/modules/\$(uname -r)/build', add a KERNEL flag:
AFULNX_32 /MAKEDRV KERNEL=/lib/modules/\$(uname -r)/build
Or
AFULNX_64 /MAKEDRV KERNEL=/lib/modules/\$(uname -r)/build
 - If KERNEL is omitted, the default is /lib/modules/\$(uname -r)/build.

- This should work for MOST distributions.
- 5. Check your build
 - Check the version of running Linux kernel with 'uname -r'.
 - Check the version of amifdrv_mod.drv with 'modinfo amifdrv_mod.drv'.
 - If they mismatch, you will need to select the correct configuration file(.config), rebuild your kernel, and then rebuild your driver as described in (4.1), (4.2), and (4.3).
 - The amifdrv_mod.drv must be in same directory with afulnx_32 (afulnx_64). If they match, continue on to the 'AFULNX2' section to run afulnx2.

Troubleshooting

| | |
|-----------|--|
| Q1 | I get following error message when loading driver: "insmod: error inserting 'amifdrv_mod.o': -1 Invalid module format". |
| A1 | <p>Most likely this is cause by wrong configuration file and your kernel refuses to accept your driver because version strings (more precisely, version magic) do not match.</p> <p>To check the version of running Linux kernel, type "uname -r". To check the version of amifdrv_mod.drv, type "modinfo amifdrv_mod.drv"</p> <p>If they mismatch, you will need to select the correct configuration file (.config), rebuild your kernel, and then rebuild your driver as described in "Generating driver file manually" section.</p> |
| Q2 | When I run ./afulnx_32(./afulnx_64), it says "Unable to load driver". |
| A2 | Some Linux distributions do not display driver debug messages on screen by default. Type "dmesg" to see those debug messages. This is very likely the same problem as Q1. |
| Q3 | When I run ./afulnx_32(./afulnx_64), it simply freezes. |
| A3 | <p>This is caused by a Linux feature called "NMI Watchdog" which is used to debug Linux kernel. This feature must be disabled to run AFULNX2.</p> <p>Please do "cat /proc/interrupts" twice and check if NMI is counting.</p> <p>If it is, then boot Linux with a kernel parameter "nmi_watchdog=N" where N is either 0, 1 or 2.</p> <p>Find out which configuration can halt NMI from counting by "cat /proc/interrupts" This is the configuration we should use to run AFULNX2.</p> |

Error Code List

| Error Number | Description |
|--------------|--|
| 00h | No error |
| 01h | Unknown command |
| 02h | Can't open ROM ID file |
| 03h | ROM ID file is not a ROM file |
| 04h | Invalid MAC address |
| 05h | Invalid retry count |
| 06h | System doesn't support MAC programming |
| 07h | This program can not run under this operating system |
| 08h | Flash part is not supported |
| 09h | Problem extracting module from ROM file |
| 0Ah | Can not analyze ROM file. ROM file may be corrupted |
| 0Bh | NCB error |
| 0Ch | Invalid option |
| 0Dh | BIOS does not support AFU |
| 0Eh | ROM file size incorrect |
| 0Fh | File ROM ID incorrect |
| 10h | Bootblock error |
| 11h | Loading driver |
| 12h | Unloading driver |
| 13h | Invalid NCB |
| 14h | Closing memory manager |
| 15h | Mapping BIOS data buffer error |
| 16h | Problem allocating memory |
| 17h | Problem freeing memory |
| 18h | Problem allocating BIOS buffer |
| 19h | Problem freeing BIOS buffer |

Getting Started for AFUBSD v2.00 or above

Installation

- Copies **AFUBSD.TGZ** to any storage location accessible by the host system.
- Extracts the contents to same directory. You will get a folder named **AFUBSD**.
- Run **AFUBSD** in command prompt.

Usage & Example for command line mode

For AFULNX v4.10, we have added a new command:

AFULNX/MAKEDRV <Kernel Path>

This command can help user to build driver manually. Please see “Generating driver file manually” section to know detail. In addition to this command, other behaviors are same as AFULNX.EXE. So you can see [Usage of AFULNX](#) and [Example of AFULNX](#) to learn more information.

Error Code List

See AFULNX [Error Code List table](#) for detail.

Appendix B : AFUBSD/AFULNX v3.xx Commands

Usage : AFUBSD/AFULNX /i<ROM File Name> [/o<Save ROM File Name>] [/n] [/p[b]
[n][c][e]] [/s] [kN] [/c[N]] [/q] [/h] [/t] [/u[ROM File Name]]

Following table lists the description of previous version of AFUBSD/AFULNX commands.

| Command | Description |
|-------------------|--|
| /n | Do not check ROM ID |
| /pbnce | p – Program main BIOS b – Program boot block n – Program NVRAM c – Destroy system CMOS e – Program embedded controller block |
| /k | Program all Non-Critical Blocks only |
| /kN | Program N'th Non-Critical Block only (N = 0 – 7) |
| /s | Leaves signature in BIOS |
| /q | Silent execution |
| /h | Print help |
| /t | Display current system's ROM ID |
| /c | Program main BIOS and all Non-Critical Blocks |
| /cN | Program main BIOS and N'th Non-Critical Block (N = 0 – 7) |
| /srb | Force REBOOT after programming done |
| /d | Compare ROM file (Skip flashing) |
| /o<ROM File Name> | Save current system BIOS ROM into disk |
| /u<ROM File Name> | Display ROM file's ROM ID |