

## Chapter 6 AFUDOS (AMI Firmware Update)

### Overview

---

AFUDOS is an updating system BIOS utility with command line interface. It has no tedious and annoying parameters, just update your system BIOS. Hey!! Do not forget that target board MUST be *AMIBIOS* system.

---

### Features

---

This utility offers the following features:

- Small executable file size
  - Quickly update
  - Clear updating information and status
  - Fully compatible with previous version (See [Appendix B AFUDOS v3.xx Commands](#))
- 

### Requirements

---

#### Supported Operating System

This utility is supported by the following operating systems:

- MS-DOS environment

#### 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.*
- 

### Getting Started

---

#### Installation

Copies the *AFUDOS.EXE* executable file to any storage location accessible by the host system and then run **AFUDOS** in command prompt.

## Usage

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

**AFUDOS <BIOS ROM File Name> [Option 1] [Option 2].....**

*Or*

**AFUDOS <Output BIOS ROM File Name> <Commands>**

*Or*

**AFUDOS /M<MAC Address>**

*Or*

**AFUDOS /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 BIOS ROM file
- **/Ln** Refer to Options: /Ln
- **/M<MAC Address>** Refer to Options: /M
- **/MAI** Display current system and ROM file's MA information.
- **/HOLE** Update specific ROM Hole according to given name.
- **/HOLEOUT** Save specific Rom Hole Data according to given name.
- **/D** Verification test of given ROM File without flashing BIOS.
- **/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 Area (Refer to OFBD Spec)
- **/NCBOUT** Output NCB Data according to 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.

Rules
<ul style="list-style-type: none"> <li>▪ If [/E] present alone, there is only the Embedded Controller block to be updated.</li> <li>▪ If [/E] and [/ECUF] co-exist, [/ECUF] will be no function.</li> <li>▪ If [/Kn] present alone, there is only non-critical block to be updated.</li> <li>▪ When [/Ln] is co-exist with [/C], [/C] will be no function.</li> <li>▪ [/M] can be used as a command for backward compatible.</li> </ul>

Note: Running AFUDOS 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**  
*AFUDOS <BIOS ROM File Name> /O*
- **Get and display ROM ID from BIOS ROM file**  
*AFUDOS <BIOS ROM File Name> /U*
- **Update main BIOS image only**  
*AFUDOS <BIOS ROM File Name>*  
*Or*  
*AFUDOS <BIOS ROM File Name> /p*
- **Update Boot Block only**  
*AFUDOS <BIOS ROM File Name> /B*
- **Update NVRAM only**  
*AFUDOS <BIOS ROM File Name> /N*
- **Update Embedded Controller Block only**  
*AFUDOS <BIOS ROM File Name> /E*
- **Update Embedded Controller Block if newer version is detected**  
*AFUDOS <BIOS ROM File Name> /ECUF*
- **Update 2<sup>nd</sup> non-critical block only**  
*AFUDOS <BIOS ROM File Name> /K2*
- **Update main BIOS image, Boot Block and NVRAM at once**  
*AFUDOS <BIOS ROM File Name> /P /B /N*
- **Update whole BIOS ROM**  
*AFUDOS <BIOS ROM File Name> /P /B /N /C /E /K*
- **Update whole BIOS ROM and load current CMOS optimal settings**  
*AFUDOS <BIOS ROM File Name> /P /B /N /C /E /K /L0*
- **Update whole BIOS without checking ROM ID**  
*AFUDOS <BIOS ROM File Name> /P /B /N /C /E /K /X*
- **Update whole BIOS with quiet execution**

*AFUDOS <BIOS ROM File Name> /P /B /N /C /E /K /Q*

- **Update whole BIOS in quiet mode and REBOOT quietly**

*AFUDOS <BIOS ROM File Name> /P /B /N /C /E /K /Q /REBOOT*

- **Update BootBlock MAC address**

*AFUDOS /M<MAC Address>*

- **Update whole BIOS and BootBlock MAC address**

*AFUDOS <BIOS ROM File Name> /P /B /N /C /E /K /M<MAC Address>*

- **Update whole BIOS except existing SMBIOS structures**

*AFUDOS <BIOS ROM File Name> /P /B /N /C /E /K /R*

- **Update whole BIOS but preserve SMBIOS type 0 and 11**

*AFUDOS <BIOS ROM File Name> /P /B /N /C /E /K /R0 /R11*

- **Update dedicate ROM Hole Area**

*AFUDOS <ROM Hole File Name> /Hole:Name*

- **Update dedicate NCB Area**

*AFUDOS <NCB File Name> /NCB:Name*

- **Output dedicate ROM Hole File**

*AFUDOS <Output ROM Hole File Name> /HOLEOUT:Name*

- **Output dedicate NCB File**

*AFUDOS <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.*

*AFUDOS <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**

*AFUDOS <BIOS ROM File Name> /DeDftCfg*

---

## Chapter 7 AFULNX/AFUBSD

### Overview

---

AFULNX/AFUBSD (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 AFUDOS 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.032** Driver for 32-Bit Linux.
  - **UCORELNX.064** 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:

#### 3.1 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 'vmlinuz-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). This file must be renamed to ".config"(dot config).

#### 3.2 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.

#### 3.3 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 xvfz afulnx2.tgz
```

- The name of the archive may be different, but the overall syntax is the same.

### 3.4 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'.

### 3.5 Make Your AMI Flash Driver(UCORELNX.O32/UCORELNX.O64):

- 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
```

### 3.6 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: 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.

Please do "cat /proc/interrupts" twice and check if NMI is counting.

If it is, then boot Linux with a kernel parameter "nmi\_watchdog=N" where N is either 0, 1 or 2. Find out which configuration can halt NMI from counting by "cat /proc/interrupts" This is the configuration we should use to run AFULNX2.

## Usage & Example for command line mode

This part is same as AFUDOS.EXE but running under Linux system. So you can see [Usage](#)

[of AFUDOS](#) and [Example of AFUDOS](#) to learn more information.

## 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 another is **AFULNX2\_26**.
- Type 'uname -r' to identify kernel version. If it says 2.4.xx..., you should enter to **AFULNX2\_24** folder, otherwise, enter to **AFULNX2\_26** folder.
- To determine which version of Linux system, type 'uname -m'. under shell screen. If it says 'x86\_64', then **AFULNX\_64** should be used; otherwise, use the **AFULNX\_32**.
- Run the suitable file. AFULNX will generate necessary drivers and load it 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) must be 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, first, you can read 'Q1' and 'Q2' in 'TROUBLESHOOTING section' to shut out the kernel issues, and second, you can see 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:
  - 4.1 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 the root of the linux kernel source tree(usually it is /lib/modules/\$(uname -r)/build). This file must be renamed to ".config"(dot config).

#### 4.2 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.

#### 4.3 Check your build

- Check the version of running Linux kernel with 'uname -r'.
- Check the version of amifldr\_mod.drv with 'modinfo amifldr\_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 amifldr\_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 'amifldr\_mod.o': -1 Invalid module format".

A1: Most likely this is caused 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 amifldr\_mod.drv, type "modinfo amifldr\_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 “Generating driver file manually” section.

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: 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.

Please do "cat /proc/interrupts" twice and check if NMI is counting.

If it is, then boot Linux with a kernel parameter "nmi\_watchdog=N" where N is either 0, 1 or 2. Find out which configuration can halt NMI from counting by "cat /proc/interrupts" This is the configuration we should use to run AFULNX2.

## 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 AFUDOS.EXE. So you can see [Usage of AFUDOS](#) and [Example of AFUDOS](#) to learn more information.

## Chapter 8 AFUWIN (AMI Firmware Update)

### Overview

---

AFUWIN is an updating system BIOS utility with command line and GUI interface. It has same parameters and behavior as AFUDOS, and further, GUI feature starting from v4.10 can provide you a friendly environment to visualize BIOS update procedure. By the way, do not forget that target board **MUST** be *AMIBIOS* system while using this utility.

---

### Features

---

This utility offers the following features:

- Small executable file size
  - Quickly update
  - Clear updating information and status
  - Fully compatible with previous version (See [Appendix B AFUDOS v3.xx Commands](#))
- 

### Requirements

---

#### Supported Operating System

AFUWIN Utility is supported in following operating system:

- Microsoft® Windows® 98
- Microsoft® Windows® ME
- Microsoft® Windows® 2000
- Microsoft® Windows® NT 4.0
- Microsoft® Windows® XP/XP64
- Microsoft® Windows® PE
- Microsoft® Windows® Vista 32/64

#### 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

Following drivers for different operation system are required by this utility:

- |                              |   |
|------------------------------|---|
| • <b><i>UCOREVXD.VXD</i></b> | Driver for Microsoft® Windows® 98/ME.         |
| • <b><i>UCORESYS.SYS</i></b> | Driver for Microsoft® Windows® NT/2000/XP/PE. |
| • <b><i>UCOREW64.SYS</i></b> | Driver for Microsoft® Windows® XP64.          |
- 

## Getting Started

---

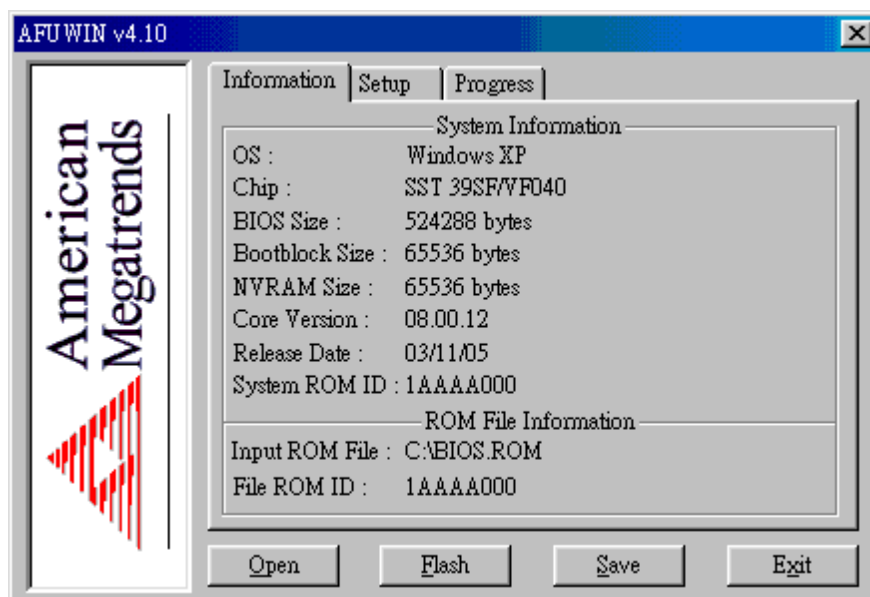
### Installation

Copies ***AFUWIN.EXE***, ***UCOREVXD.VXD***, ***UCORESYS.SYS*** and ***UCOREW64.SYS*** to any storage location accessible by the host system and then run **AFUWIN** in command prompt. Remember that three files MUST be in same directory. For launching GUI mode, you can just double-click on the icon.

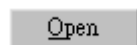
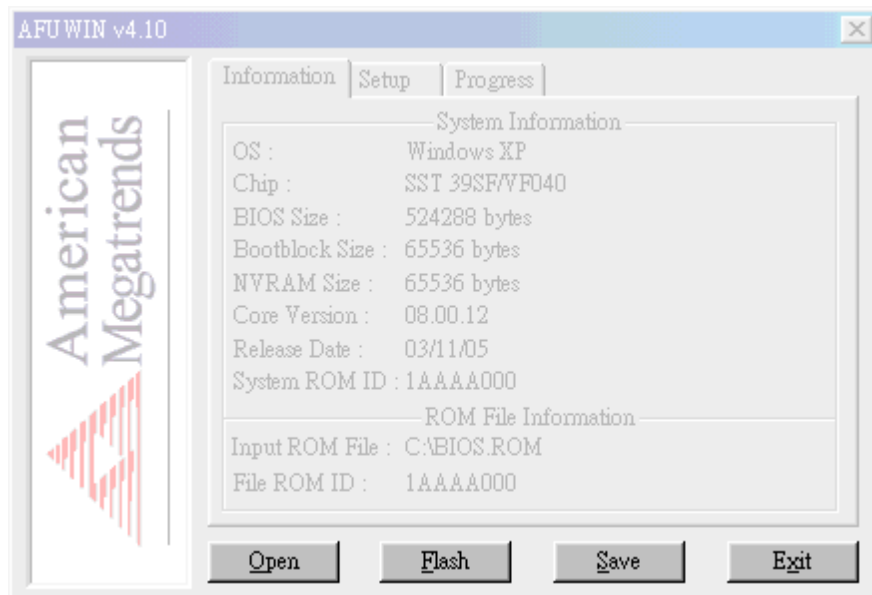
### Usage & Example for command line mode

This part is same as AFUDOS.EXE but running under Microsoft® Windows®. So you can see [Usage of AFUDOS](#) and [Example of AFUDOS](#) to learn more information.

### Main Window



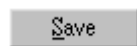
## Buttons



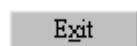
Click this button to search for BIOS ROM file from any disk drive.



Click this button to starting update BIOS.



Click this button to save BIOS ROM image to disk drive.

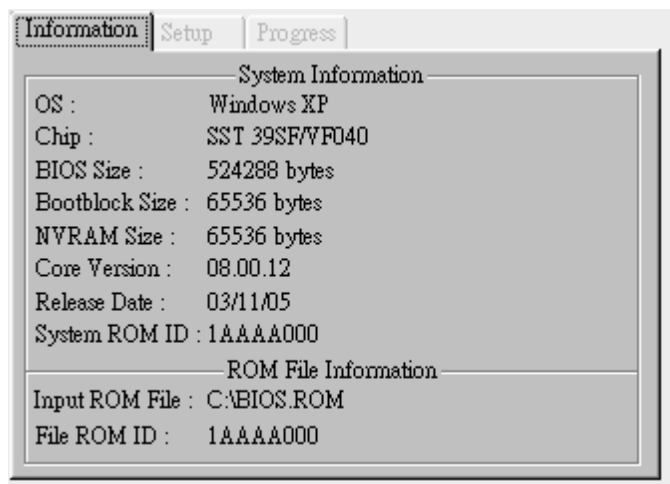


Click this button to exit this program.

## Function Frame

### Information Tab

This tab displays system BIOS information for your reference before flashing BIOS.



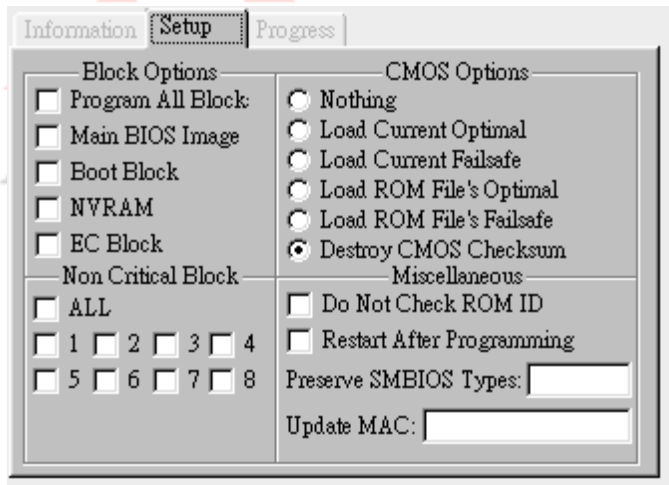


## Field

Name	Description
OS	This field displays current O/S version.
Chip	This field displays current flash part on the system.
BIOS Size	This field displays current BIOS ROM size.
BootBlock Size	This field displays current BIOS BootBlock size.
NVRAM Size	This field displays current BIOS NVRAM size.
Core Version	This field displays current AMIBIOS CORE version.
Release Date	This field displays current BIOS release date.
System ROM ID	This field displays current system BIOS ROM ID.
Input ROM File	This field displays BIOS ROM image file name/path where will be used to instead of old one.
File ROM ID	This field displays ROM ID in given BIOS ROM image file.

## Setup Tab

This tab allows you to change the settings for flashing BIOS.



## Field

Block Options	
Name	Description
Program All Block	This option is used to enable all programmable blocks.
Main BIOS Image	This option is used to determine if Main BIOS Image needs to update.
Boot Block	This option is used to determine if Boot Block needs to update.
NVRAM	This option is used to determine if NVRAM needs to update.
EC Block	This option is used to determine if EC Block needs to update.

## CMOS Options

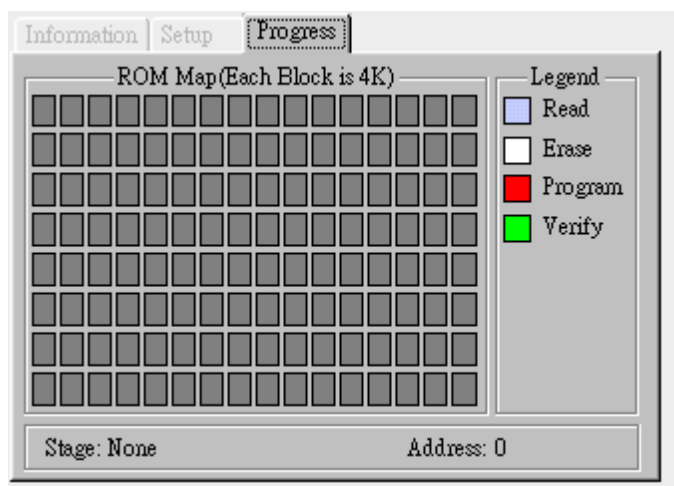
Name	Description
Nothing	Enable if you want to do nothing for CMOS after BIOS updated.
Load Current Optimal	Enable if you do like to load CMOS optimal settings from current system after BIOS updated.
Load Current Failsafe	Enable if you do like to load CMOS failsafe settings from current system after BIOS updated.
Load ROM File's Optimal	Enable if you do like to load CMOS optimal settings from current system after BIOS updated.
Load ROM File's Failsafe	Enable if you do like to load CMOS failsafe settings from current system after BIOS updated.
Destroy CMOS Checksum	Enable if you do like to destroy CMOS checksum after BIOS updated. This is default setting in CMOS Options block.

Non Critical Block	
Name	Description
All	Enable if you want to update all Non Critical Blocks.
1 – 8	Enable one of Non Critical Blocks if it needs to update.

Miscellaneous	
Name	Description
Do Not Check ROM ID	Enable if you do not want to check ROM ID before updating BIOS.
Restart after Programming	Enable if you want to restart system after BIOS updated.
Preserve SMBIOS Type	This field allows you to preserve SMBIOS types while BIOS updating. The types string must be decimal-digit and separated by a space(' ') character. For convenience, you can strike 'A' key as first character to select all SMBIOS structures at once.
Update MAC	This field is used to change BootBlock MAC address. It MUST be hexadecimal-digit string.

### Progress Tab

This tab displays the updating status.

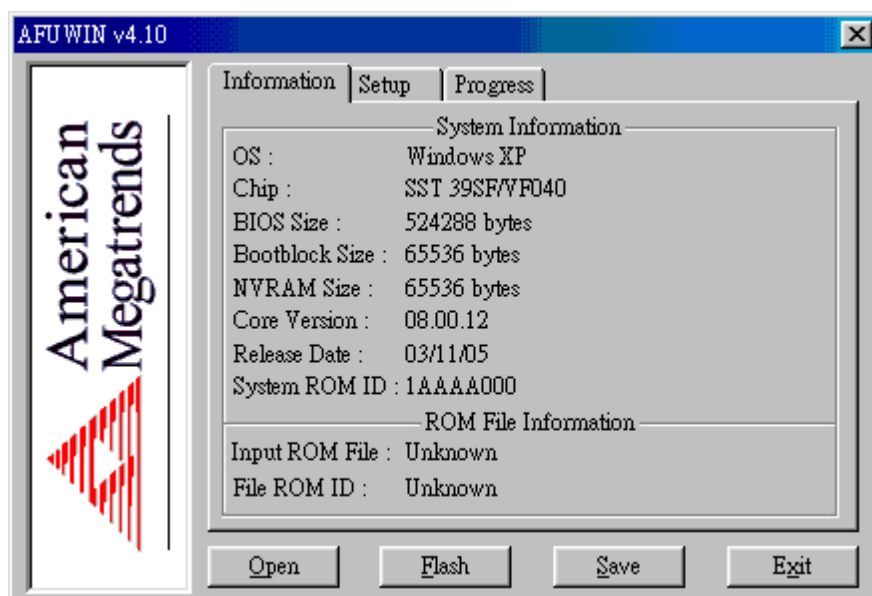


### Field

Name	Description
ROM Map	This area displays current updating status.
Legend	This area illustrates the meaning of color in ROM MAP area.
Stage	This field displays the stage of updating BIOS.
Address	This field display the address where block is under working.



## Functions

To launch into AFUWIN with GUI mode, you can double-click the executable file icon to open the operating window:






Usually, system BIOS information will be displayed first, but you may see a pop-up dialog if the system does not support AMIBIOS update function. After open this program successfully, you can refer to following steps to finish the operation what you need:

## Saving system BIOS ROM image to file

1. Press  button to open file dialog box.
2. Select path and input a file name.
3. Click on OK button to save system BIOS ROM image into specific file.
4. Press  button to exit this program.

## Flashing system BIOS with given file

1. Press  button to search for BIOS ROM image file from any disk driver and load it into memory.
2. Switch to *Setup Tab* to check and change necessary settings.
3. Press  button to start the operation.
4. *Progress Tab* will be switched automatically and display the programming status.
5. After BIOS updated, you can press  button to exit this program or system will restart automatically if the **Restart After Programming** option enabled.