

# User's Guide



Phoenix BIOS Editor™

## **Copyright**

© Copyright 2006 by Phoenix Technologies Ltd. All rights reserved. No part of this publication may be reproduced, transmitted, transcribed, stored in a retrieval system, or translated into any language or computer language, in any form or by any means, electronic, mechanical, magnetic, optical, chemical, manual, or otherwise, without the prior written permission of Phoenix Technologies Ltd.

## **Disclaimers**

Phoenix Technologies Ltd. makes no representations or warranties with respect to the design and documentation herein described and especially disclaims any implied warranties of merchantability or fitness for a particular purpose. Further, Phoenix Technologies Ltd. reserves the right to revise this design and associated documentation and to make changes from time to time in the content without obligation of Phoenix Technologies Ltd. to notify any person of such revisions or changes.

## **Trademarks**

Phoenix, Phoenix Technologies, Phoenix Technologies Logo, TrustedCore, Phlash, AwardBIOS, MultiKey and CoreArchitect are trademarks and/or registered trademarks of Phoenix Technologies Ltd. All other marks and names are the property of their respective owners.

# Preface

---

This document describes the Phoenix BIOSEditor, its installation, and usage.

## Intended Audience

---





It is intended for OEMs, system builders, and end users and assumes that you are familiar with the Phoenix BIOSEditor.

## Conventions Used in This Manual

---

This document may use some or all of the following typographic conventions:

### Conventions Used

<b>Bold</b>	Indicates text the user must enter or select, such as menu items, buttons, and commands
<i>Italics</i>	Indicates emphasis
Courier New	Represents filename or code
	Provides users with useful tips
	Provides users with helpful suggestions
	Warns about possible loss of data and equipment damage
	Identifies important features or instructions

## Contacting Technical Support

---

Corporate Address:

Phoenix Technologies, Ltd.  
915 Murphy Ranch Road  
Milpitas, California 95035  
USA

**Web site:** [www.phoenix.com](http://www.phoenix.com)

**Customer Support:** <http://support.phoenix.com>

**Manual revision:** September 15, 2006



# Table of Contents

Preface .....	iii
Intended Audience .....	iii
Conventions Used in This Manual .....	iii
Contacting Technical Support .....	iii
<b>Chapter 1. Introduction .....</b>	<b>7</b>
Prerequisites .....	7
Installing BIOS Editor .....	7
Uninstalling BIOS Editor .....	7
Using Online Help .....	8
<b>Chapter 2. BIOS Images .....</b>	<b>9</b>
3.1 Starting the Bios Editor .....	9
3.2 Editing a BIOS Image .....	9
3.3 Building the TrustedCore CSS or BIOS .....	9
<b>Chapter 3. BIOS Editor Screens .....</b>	<b>11</b>
Navigating .....	11
BIOS Editor Window .....	11
Menus .....	11
The File Menu .....	11
The View Menu .....	11
The Edit Menu .....	12
The Windows Menu .....	12
The Help Menu .....	12
Shortcut keys .....	12
Project Panel .....	13
Core (BIOS) Organization .....	13
Project Modules .....	13
Adding/Changing a Module .....	14
Editing the ROM Flash Banks .....	14
Editing the Memory Hole .....	14
Editing the Quiet Boot Logo .....	14
The Editor Windows .....	15
The Setup Table Editor .....	15
The String Editor .....	17
The BCP Editor .....	18
The Register Editor .....	19
The DMI Editor .....	20
ROM Image DMI Editor .....	21
BIOS Project DMI Editor .....	21
The PCI Routing Table .....	21
Splash Screen .....	24
ACPI Editor .....	26



# Introduction

Phoenix BIOS Editor is a Windows application that provides an easy-to-use interface for modifying Phoenix Technologies' Core System Software, known as Phoenix TrustedCore, without having to rebuild it. With BIOS Editor, you modify the pre-built ROM image without the source code.

BIOS Editor automatically decompresses a single ROM image back to the small ROM modules that are part of a TrustedCore project. When you modify a ROM image, BIOS Editor incorporates your changes into a new BIOS image that can be written to a flash part.



In addition to Phoenix TrustedCore, BIOS Editor can also be used to modify Phoenix FirstBIOS Desktop Pro and PhoenixBIOS 4.0 Release 6 and later.

## Prerequisites

---

The following software prerequisites are required to use Phoenix BIOS Editor.

- Microsoft Windows 95/NT/2000/XP/Vista
- BIOS.ROM or Build products of Phoenix TrustedCore, Phoenix FirstBIOS Desktop Pro, or PhoenixBIOS 4 Release 6 or later.

## Installing BIOS Editor

---

The BIOS Editor diskette contains the BIOSEDIT.EXE file, a self-extracting file that contains the files for installation.

To install BIOS Editor, perform these steps:

1. Copy both files to a temporary directory of your hard disk
2. From Windows, run BIOSEDIT.EXE to unpack the BIOS Editor
3. From Windows, run Setup to install BIOS Editor in the directory of your choice. Follow the Setup instructions.

## Uninstalling BIOS Editor

---

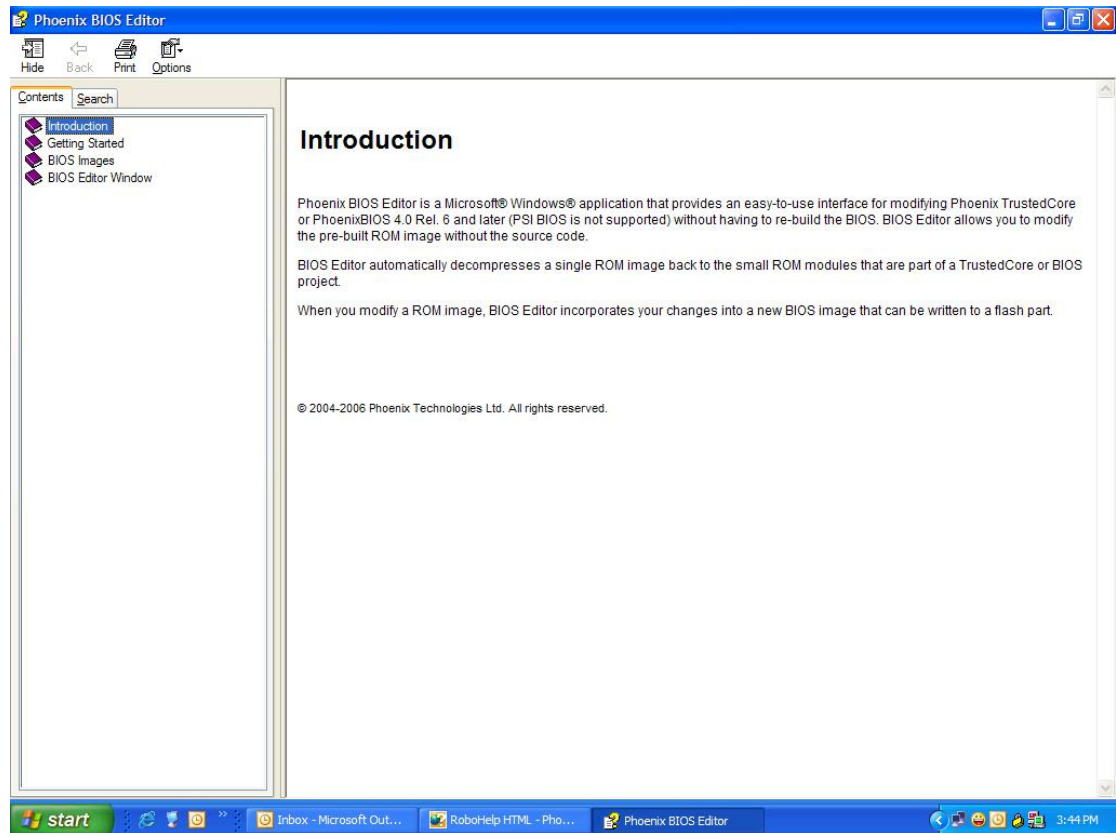
To uninstall Phoenix BIOS Editor:

1. From the **Windows Start** menu, select **Settings**.
2. Select **Control Panel**.
3. Select **Add/Remove Programs**.
4. Select **Phoenix BIOS Editor**.
5. Click **Remove**.

# Using Online Help

---

1. Press F1 after a BIOS image is opened. The corresponding topic displays in a Help window.
2. Click blue links to view additional information.
3. Click **Back** to return to the previous Help topic.





# BIOS Images

This chapter describes how to open, edit, and save a BIOS image.

## 3.1 Starting the Bios Editor

---

To run BIOS Editor in Windows, do any one of the following:

- Double-click the BIOS Editor icon in the BIOS Editor group.
- Double-click BIOSEDIT.EXE in Windows Explorer.
- Run BIOS Editor from the Run box on the Start menu (**Start>Run**).

The program opens with an empty window.

## 3.2 Editing a BIOS Image

---

To edit a BIOS image:

1. Go to **File>Open**.
2. In the dialog box, select the ROM image you want to edit and click **Open**. The program opens a window with a number of Editor windows.
3. Select the window you want to edit and modify the **Value** field. See the following sections for descriptions on changing the **Value** field for the different windows.

The following windows are available for the ROM File:

- BCP Editor
- Register Editor
- DMI Strings
- Setup Table Editor
- String Editor
- PCI Routing Table
- Quiet Boot Logo Editor/Splash Screen Editor

4. To complete the BIOS image edit, go to **File>Build BIOS** and save the new ROM image.

## 3.3 Building the TrustedCore CSS or BIOS

---

When you finish editing the BIOS image, select Build BIOS on the File menu or click Save to save the new ROM image. A report will be generated after you build the TrustedCore CSS or BIOS. If the build process was successful, the BIOS Editor will ask you to save the new TrustedCore CSS or BIOS.



The Build button is available only when the TrustedCore CSS or BIOS image is changed.



# BIOS Editor Screens

## Navigating

---

The Editor Windows conform to standard Windows protocols. You can navigate them with the mouse or the keyboard. Use the <Tab> and <Shift-Tab> to select an area of the window. Use the direction arrows to select an item within that area.

## BIOS Editor Window

---

The BIOS Editor Window consists of these basic elements:

- The Title Bar at the top
- The Menu Bar
- The Toolbar
- The Project Panel on the left showing the project elements
- A set of Editor Windows for editing the project elements
- The Status Bar at the bottom

## Menus

---

Besides the Help menu, there are these menus on the Menu Bar:

- The File Menu
- The Edit Menu (standard Windows commands)
- The View Menu
- The Windows Menu

### The File Menu

---

The following describes the commands on this menu.

Menu Item	Action
Open <Ctrl+O>	Open a project for editing that already has been imported and saved
Close	Close the editing project
Build BIOS <Ctrl+U>	Create a new ROM image from a Project, incorporating changes
Exit	Exit BIOS Editor

### The View Menu

---

The following describes the commands on this menu.

Menu Item	Action
Tool Bar	Display or hide the Tool Bar
Status Bar	Display or hide the Status Bar
Project Panel	Display or hide the Project Panel

## The Edit Menu

The menu is available only when ACPI Editor is active.

Menu Item	Action
Cut <Ctrl+X>	Cut selected code to the Windows clipboard.
Copy <Ctrl+C>	Copy selected code to the Windows clipboard
Paste <Ctrl+V>	Paste code from the Windows clipboard to ACPI Editor
Del <Del>	Delete selected code
Find <Ctrl+F>	Display a dialog to find specifically code
Find Next <F3>	Find code specified in Find dialog
Select All	Select all content in the ACPI Editor

## The Windows Menu

Use the commands on the upper half of the screen to arrange the Editor windows and their icons. Use the commands on the lower half to open the available Editor windows. A grayed command indicates that the window not available.

Menu Item	Action
Cascade <Shift+F5>	Display the Editor Windows one behind the other.
Tile Horizontally <Shift+F4>	Tile the Editor Windows horizontally
Tile Vertically <Shift+F3>	Tile the Editor Windows vertically
Arrange Icons	Arrange the minimized icons of the Editor Windows
BCP Editor <Ctrl+B>	Open the BCP Editor
Register Editor<Ctrl+R>	Open the Register Editor
String Editor <Ctrl+S>	Open the String Editor
Setup Table Editor <Ctrl+T>	Open the Setup Table Editor
PCI IRQ Table Editor <Ctrl+I>	Open the PCI IRQ Table Editor
DMI Editor <Ctrl+D>	Open the DMI Editor
Quiet Boot Logo Editor/ Splash Screen Editor <Ctrl+Q>	Open the Quiet Boot Logo/Splash Screen Editor
ACPI Editor <Ctrl+A>	Open the ACPI Editor

## The Help Menu

To display Online Help and About dialog

Menu Item	Action
Contents<F1>	Display Online Help of BIOS Editor
About	Display Phoenix BIOS Editor information

## Shortcut keys

The following are shortcut keys which can be used instead of menu commands:

- Open file (**Ctrl+O**)

- Build BIOS (**Ctrl+U**)
- Display BIOS Configuration Parameters window (**Ctrl+B**)
- Display Register table window (**Ctrl+R**)
- Display DMI Strings window (**Ctrl+D**)
- Display String Editor (**Ctrl+S**)
- Display Setup Table (**Ctrl+T**)
- Display PCI Routing Table (**Ctrl+I**)
- Display Quite BootLogo Editor/Splash Screen Editor (**Ctrl+Q**)
- Display ACPI Editor (**Ctrl+A**)

## Project Panel

---

The Main Window displays a Project Panel with a tree of the items in the Project. This window marks the modules and other items that are included in your project with a plus sign (+). Click on the plus sign or double-click the name of the item to expand and display it.

When you expand an item, the enabled buttons at the bottom of the window indicate whether you can Add, Remove, or Change an item. Changing some of them, like ACPI.AML/ACPI.BIN , ROMEXEC#.BIN , TEMPLATE#.BIN,STRING#.BIN, activates the related Editor window on the right. Changing others displays a special dialog box.

Briefly, the Project Items shown in this panel are:

### Core (BIOS) Organization

---

- Flash Bank Setting
- Compression Mode
- Reserved Memory Holes

### Project Modules

---

- ACPI Module
- BIOS Code
- Boot Block
- CPU Update
- Decompress Code
- Display Manager
- Option ROM
- Other Module
- Power Management
- Splash Screen
- ROM Executable Code
- ROM Pilot Module
- Setup Table
- Setup Engine

- Strings Module
- Wave Module
- OEM Module

## Adding/Changing a Module

---

If the Add or Change button is enabled when you expand an item on the Project Window, selecting it brings up a dialog box for importing a new module into the project.

Perform the following steps:

1. Check and enter the Output File Name.
2. Check and enter the Footprint Size.
3. Check to Disable Compression.
4. Select Import to open the Browse window for naming the import file.
5. Select OK.



FFS BIOS does not support Output File Name or multi-bank.

## Editing the ROM Flash Banks

---

To edit the ROM Flash Banks, select the Banks Setting on the Project Panel. Selecting the Change button displays a dialog box. From either drop-down box, pick a new value and select OK.



The BIOSEdit can only build a Multi-Bank ROM image; it doesn't support opening a Multi-Bank ROM image.

## Editing the Memory Hole

---

To edit the Reserved Memory Hole, perform the following actions:

1. In the Project Panel, select **Reserved Hole** and click **Add** or **Change**.  
If you select **Change**, a dialog box displays with the message: *A compressed Hole will be decompressed before editing. Is this Hole compressed.* If the hole is compressed, select **Yes** and if the hole is not compressed, select **No**.
2. In the **Hole Size** field, enter the size of the Memory Hole in bytes.
3. In the **Address** field, enter the location address in bytes.
4. To compress the Hole when building, check **Compress**.

When adding a Hole, you must select **Import User Data**.



Legacy BIOS does not support compressed HOLE.

## Editing the Quiet Boot Logo

---

To edit the Quiet Boot Logo, do the following:

1. In the Project panel, expand the Quiet Boot Logo folder. (If no Quiet Boot Logo file was specified in the build, this folder will not appear.)
2. Click the logo file.
3. Click the Change button.

The Quiet Boot Logo window has these commands on the command line:

Menu Item	Action
Copy	Copy the displayed picture to the clipboard
Paste	Paste the contents of the clipboard
Import	Load a Windows .bmp or .jpg file
Export	Save displayed picture
Load	Load a compiled .bin logo file
Save	Save displayed picture as a .bin logo file. Required to include modified or new logo in LOGODATA.BIN

The Paste and Import commands require that the width of the bitmap be in multiples of 8.

On the left side of the window, you can make these changes when the items are enabled:

Menu Item	Action
Logo Enable	Check to enable Quiet Boot Logo
Fade In	Check to enable logo Fade In
Fade Out	Check to enable logo Fade In. Fade Out Enable or disable logo Fade Out
Setup Prompt	Enable Setup Prompt during POST. If enabled enter X,Y coordinates (80x25) of the prompt. Select background color from color grid with right mouse button. Select text color with left mouse button
LogoLocation	Check Center if you want the logo centered. Otherwise, enter the X,Y coordinates (80x25) of the top left corner

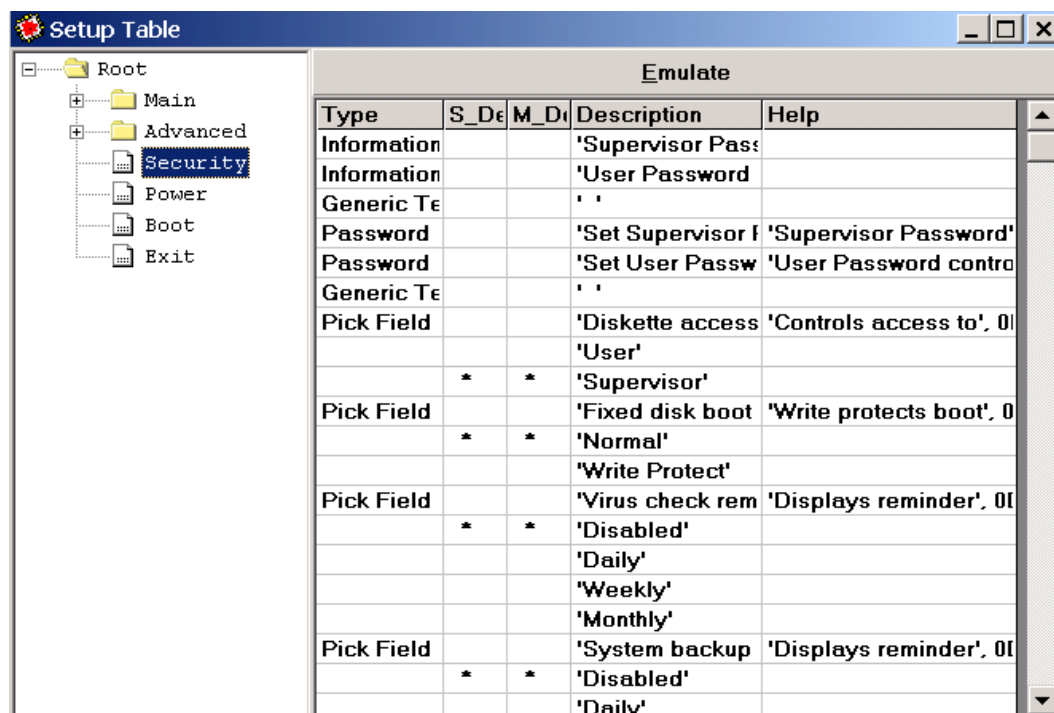
## The Editor Windows



All the Editor Windows are very platform specific. Some windows may be missing from a particular platform or build. In each window, the items displayed can also vary considerably depending on the devices installed on the platform and the features installed for the build.

### The Setup Table Editor

The Setup Table Editor can edit the Phoenix Setup Utility.



Selecting the Setup Table Editor displays a window that is divided into two panels. The left panel displays groups of Setup Nodes, as organized in the file TEMPLATE.ASM. The right panel shows the current settings for each mode. Each line in the panel displays the editable elements of the Node:

- Type (read-only)
- System Default (S\_Default)
- Manufacturer's Default (M\_Default)
- Description
- Help Text

You can edit all settings except Type.

## Changing a System and Manufacturing Default settings

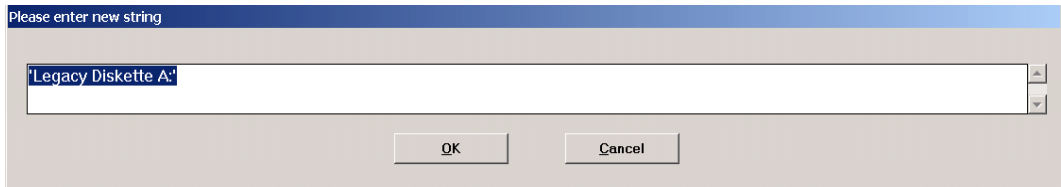
Select the value you want to change. Double-click on it or select <Enter>. An asterisk (\*) indicates the field is turned on.

Pick Field			'Legacy Diskette A:'	'S'
			'Disabled'	
			'360 Kb	5', 00h, '
			'1.2 MB	5', 00h, '
		*	'720 Kb	3', 00h, '
	*		'1.44/1.25 MB	3', 00
			NULL	
			'2.88 MB	3', 00h, '



## Changing the Description and Help

1. Selecting the value which you want to change and double-click on it or select <Enter>. It displays a dialog: "Please enter a new string."



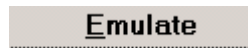
2. Key in the new description or help and press <OK>.

## Editing the settings

1. In the left panel, click the plus sign beside the group you want to edit. The right panel changes. You can scroll down to view additional lines.
2. In the right panel, double-click the text you want to change.
3. Edit the string that displays. Enclose text strings in single quotes: 'changed text'
4. Click OK.
5. Click the Emulate bar to view an emulation of the edited setup screen.

## Viewing an emulation of a screen

Click the Emulate button.



On the emulated screen, you can:

- Select the blue text on the emulated screen to display item-specific Help in the right area of the screen.
- Double-click a bulleted item to change to another screen emulation.



Use the right or left arrow to browse through screen emulations.

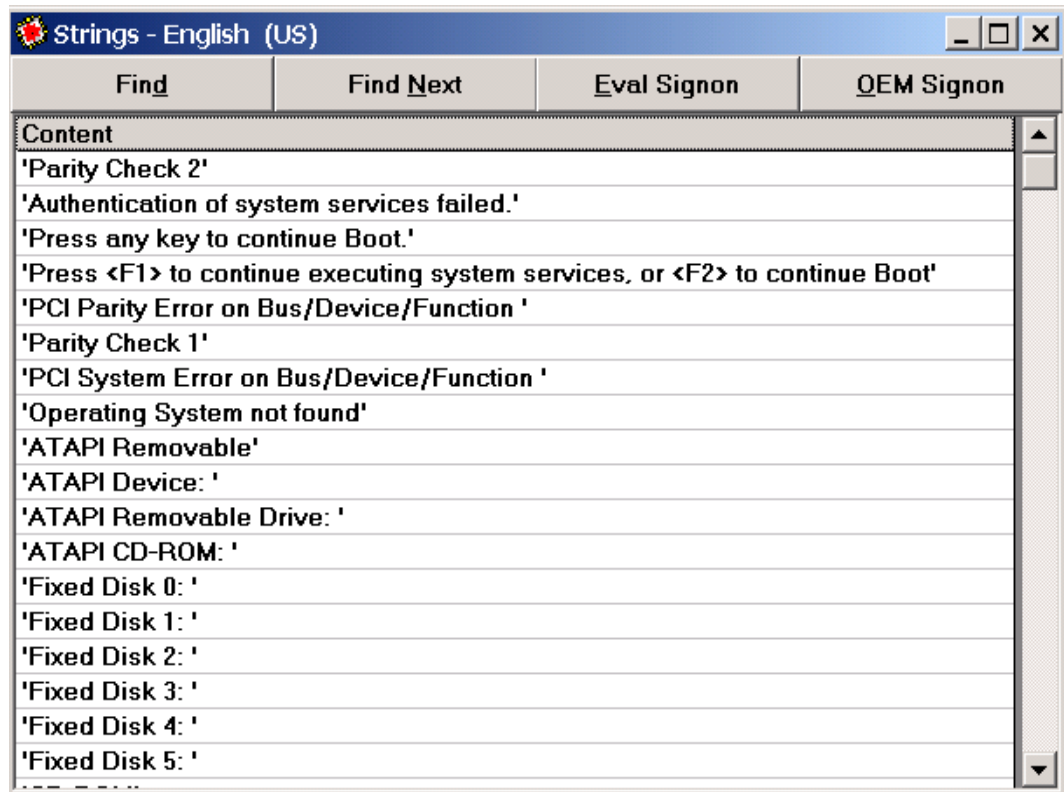


The Setup Table Editor window in the user interface is linked to the String Editor window. Changes in one window affect the other window.

## The String Editor

---

The String Editor allows modifying both setup strings and POST messages.



#### To modify a string:

1. Double-click an item to open a data entry box.
2. Edit the string.
3. Click OK to close the box and save your edit.

To modify the Evaluation Sign-on or OEM Sign-on string, click Eval Sign-on or OEM Sign-on, and edit the string.

#### To find a string:

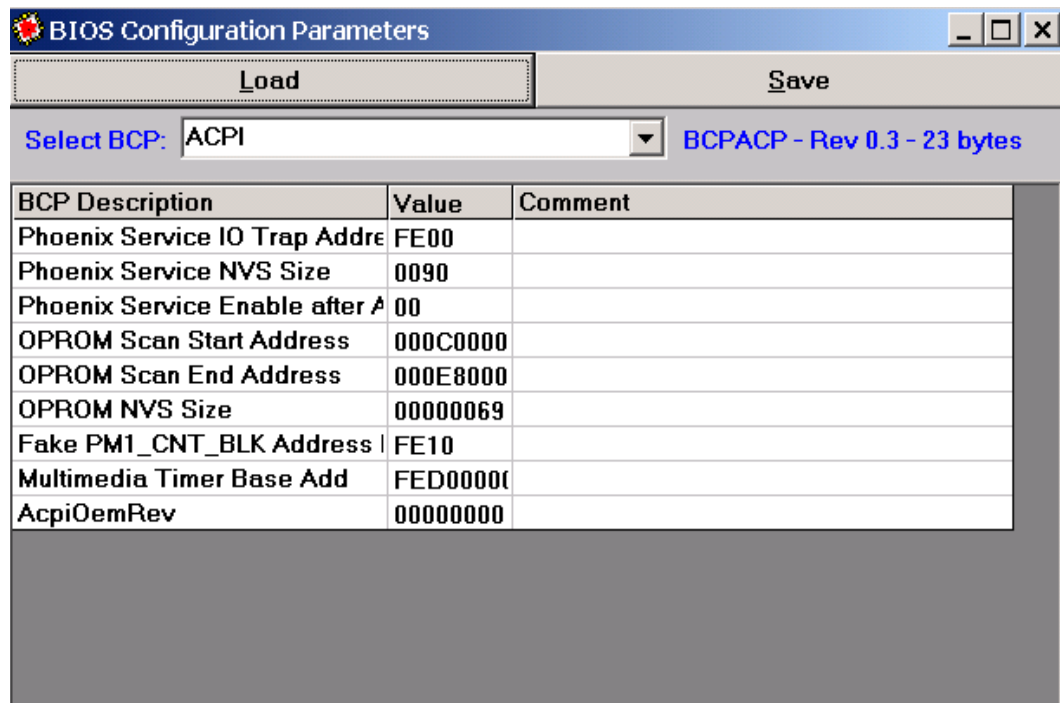
1. Click the Find button.
2. Type the string you want to find and click OK.
3. Click the Find Next button to locate the next occurrence of the string.
4. Repeat step 3 until you find each occurrence.



The String Editor window in the user interface is linked to the Setup Table window. Changes in one window affect the other window. The String Editor *cannot* edit the DBCS string.

## The BCP Editor

The BIOS Configuration Parameters window allows you to view the current values of the BCP structures and change them. Settings can be loaded and saved.



Selecting the BCP Editor displays a window for editing the BIOS Configuration Parameters (BCP), which are data values stored in the BIOS code that control the behavior of BIOS routines during POST, Setup, and Runtime. The TrustedCore or BIOS build uses special macros to create and format these parameters in a set of special BCP structures.

For example, BIOS read-and-write routines that access the chipset use data stored in the chipset BCP structure to determine the index and data-port addresses for the chipset and the width of the registers.

Use the Save button at the top to create and name a new .bcp file, which you can use in other projects. The Save button stores the current selections.

Use the Load button to load a .bcp file with the selections specified in another project.

When you select a BCP structure from the drop-down menu in the top pane, the BCP Editor displays the related BCPs and the current value and a short comment describing the function of the BCPs. You can change the values in this window but not the descriptions or comments.

An asterisk (\*) indicates the field is turned on.

There are two methods of changing BCP values:

- Turn an item on (\*) or off by double-clicking the value or by pressing <Enter>. See Selection Fields.
- Change the value by double-clicking the value and modifying it. See Value Fields.

## The Register Editor

The Register Tables window allows changing the initialization values used for setting up chipsets. These values can be loaded from and saved to \*.REG files.

Type	Index	Mask	Value	Comment
Initial POST Value	0000025	FFFFFFF	0000000	
Custom	0000025	FFFFFFF	0000000	
Initial POST Value	0000025	FFFFFFF	0000000	
Custom	0000025	FFFFFFF	0000000	
Initial POST Value	0000025	FFFFFFF	0000000	
Custom	0000025	FFFFFFF	0000000	
Initial POST Value	0000025	0000000	0000000	
Custom	0000025	0000000	0000000	
Initial POST Value	0000026	FFFFFFF	0000000	
Custom	0000026	FFFFFFF	0000000	
Initial POST Value	0000026	FFFFFFF	0000000	
Custom	0000026	FFFFFFF	0000000	
Initial POST Value	0000026	0000000	0000000	

To edit the registry:

1. Select a register-configurable device in the top panel by clicking a radio button. The editor displays the related register values in the bottom panel.
2. Double-click an item in the Mask or Values column. You cannot edit the Types, the Index, or the Comments.

The register values are of two types:

- Initial POST Values, loaded early in POST
- Setup Defaults, the register values loaded in Setup or late in POST if the CMOS checksum is bad or if AutoConfiguration is installed.

3. Edit the value.
4. Click OK.
5. To save the edited registry as a new file, click the Save button and name the file.

## The DMI Editor

The DMI Editor window allows you to change the values of the DMI strings.

DMI Strings	
Description	String
Motherboard Manufacturer	'Intel Corporation'
Motherboard Model	'GrantsDale CRB Board'
Motherboard Version	'Revision A '
Serial Number	'400'
System Manufacturer Name	'Intel Corporation'
System Product Name	'GrantsDale CRB Board'
System Version	'Revision A '
System Serial Number	'0123456789'
Chassis Manufacturer Name	'No Enclosure'
Chassis Version	'N/A'
Chassis Serial Number	'None'
Chassis Model	'No Asset Tag'
UUID	00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h

## ROM Image DMI Editor

If you are editing a ROM image, selecting the DMI Editor displays a window like this:

Description	String
Motherboard Manufacturer Name	'Intel Corporation'
Motherboard Model	'GrantsDale CRB Board'
Motherboard Version	'Revision A '
Serial Number	'400'
System Manufacturer Name	'Intel Corporation'
System Product Name	'GrantsDale CRB Board'
System Version	'Revision A '
System Serial Number	'0123456789'

You can change only the String values in this window.

## BIOS Project DMI Editor

The DMI String Table displays the Desktop Management Interface information. To edit the information, double-click the content in the **String** column and type the new value.

## The PCI Routing Table

The PCI Routing window allows changing the routing for PCI devices, changing device numbers, changing slot labels and setting bits in the PCI IRQ mask, with the exception of embedded devices that only support setting device numbers and the IRQ mask.

PCI Routing Table										
Load					Save					
Bus N	Device	Slot_Li	INT_A	IRQ	INT_B	IRQ	INT_C	IRQ	INT_D	IRQ
00	1F	00	—	1CF0	—	1CF0	—	DEF8	—	1CF0
04	00	00	—	1CF0	—	DEF8	—	DEF8	—	DEF8
0A	03	05	03	1CF0	02	1CF0	05	1CF0	06	1CF0
0A	02	04	02	1CF0	03	1CF0	01	1CF0	00	1CF0
0A	01	03	06	1CF0	05	1CF0	04	1CF0	07	1CF0
0A	00	02	05	1CF0	06	1CF0	07	1CF0	04	1CF0
00	1E	00	—	1CF0	—	1CF0	—	DEF8	—	DEF8
00	1D	00	—	1CF0	—	1CF0	—	1CF0	—	1CF0
08	02	09	03	1CF0	00	1CF0	01	1CF0	02	1CF0
0A	02	08	02	1CF0	03	1CF0	00	1CF0	01	1CF0
04	02	07	01	1CF0	02	1CF0	03	1CF0	00	1CF0
02	02	06	00	1CF0	01	1CF0	02	1CF0	03	1CF0
00	1C	00	—	1CF0	—	1CF0	—	1CF0	—	1CF0
00	1B	00	—	1CF0	—	DEF8	—	DEF8	—	DEF8
00	02	00	—	1CF0	—	DEF8	—	DEF8	—	DEF8
01	00	01	00	1CF0	01	1CF0	02	1CF0	03	1CF0
00	01	00	—	1CF0	—	DEF8	—	DEF8	—	DEF8
00	00	00	—	DEF8	—	DEF8	—	DEF8	—	DEF8

PCI interrupts are routed through the chipset to the Peripheral Interrupt Controller (PIC) and are level triggered. Most PCI chipsets provide for PCI-IRQ routing, a programmable mechanism using chipset registers for routing interrupts from PCI devices (embedded devices or slots) to IRQ lines (0 to 15) on the PIC.

Most PCI devices don't care which IRQ they use, and the PnP software assigns them automatically. Exceptions such as PCI IDE devices, however, require the use of a particular IRQ.

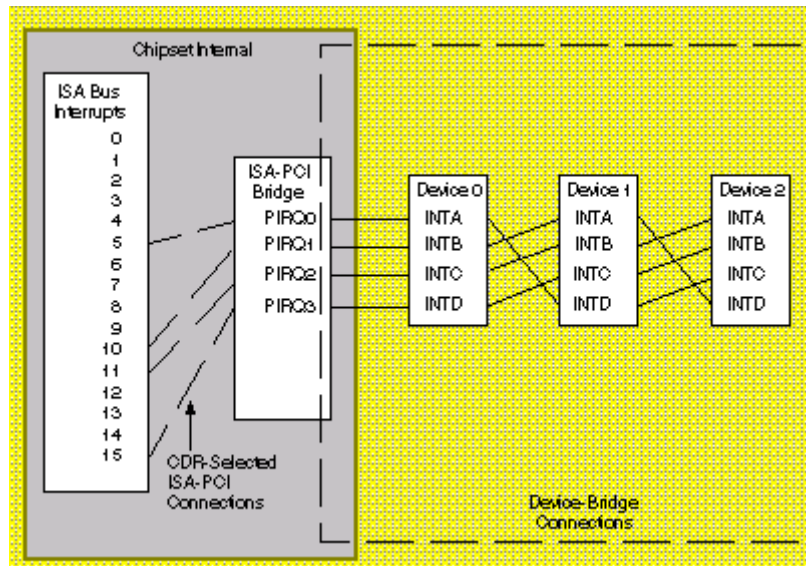
PCI devices can have up to four physical interrupt lines, INTA to INTD. Single function devices always use INTA. On multi-function devices, these four lines can be assigned to different functions of the device. More than one function can be assigned to the same interrupt line.

Furthermore, since the PCI devices can share the same interrupt lines, different functions from different devices can share the same interrupt line, as shown in the following illustration. Different interrupt lines can share IRQs on the Programmable Interrupt Controller (PIC).

When more than one slot is tied to the same PIRQ, a cascading query of the device drivers determines which function on which PCI device initiated the interrupt.

## PCI interrupt routing

The following illustration shows the hardware connections in a typical PCI platform. Different chipsets/platforms support different configurations and a different number of slots.



The Conflict Detection and Resolution (CDR) algorithm in POST uses two PCI Routing tables to determine which IRQ to assign to which PCI device:

- The Device-Bridge Hardware Connections. On the device-bridge (right) side, there is a table in the BIOS that describes which PIRQ is hardware-connected to which INT# line on each device. For example, INTA of Device 2 is connected to PIRQ2 on the bridge, INTB to PIRQ3, INTC to PIRQ0, and INTD to PIRQ1.
- The Possible ISAInterrupt-PIRQ Connections. On the ISA-bridge (left) side, there another table for each device with IRQ bitmaps that describe which ISA interrupts can be selected and connected to each PIRQ. Each bit set represents the IRQ available for that PIRQ for a specified device.

Use the following chart for modifying fields on the PCI Routing Window.

Bus Number	The host bus number of the slot or embedded device. Identify a device as embedded with the "0" bus number.
DeviceNumber	The device number assigned to the slot by the IDSEL connected. This is a number from 0 to 20.
Slot Number	The slot number silk screened on the board for the slot. This should be zero for embedded devices and nonzero otherwise. If there is no number visible on the board, the numbers should be 1 based, incremented for each slot
INTA INT B INT C INT D	These entries describe the INT-PIRQ connections for each device as determined by the hardware connections. Valid entries are 0 to 3, not necessarily in that order. These entries represent which PIRQ (PIRQ0-PIRQ3) is tied to which interrupt line (INTA-INTD) on the slot. The PIRQ is a zero based number, often 0-3 or in some chipsets 0-15. An empty field indicates that the INT line is not tied to a PIRQ.
IRQ Bitmap A IRQ Bitmap B IRQ Bitmap C IRQ Bitmap D	These bitmaps describe the mappable IRQ lines for the four INT lines (INTA-INTB) for each device, where each bit set represents the IRQ available for that PIRQ. For instance, indicate IRQ 10 by setting the 10th bit. These assignments can be different for each slot.

Select the INTA...INTD columns of the PCI Routing Table window to specify which PIRQ is tied to which slot interrupt line. For instance, under INTA for a certain device, enter 01 if INTA is connected to PIRQ1, 02 for PIRQ2, etc.



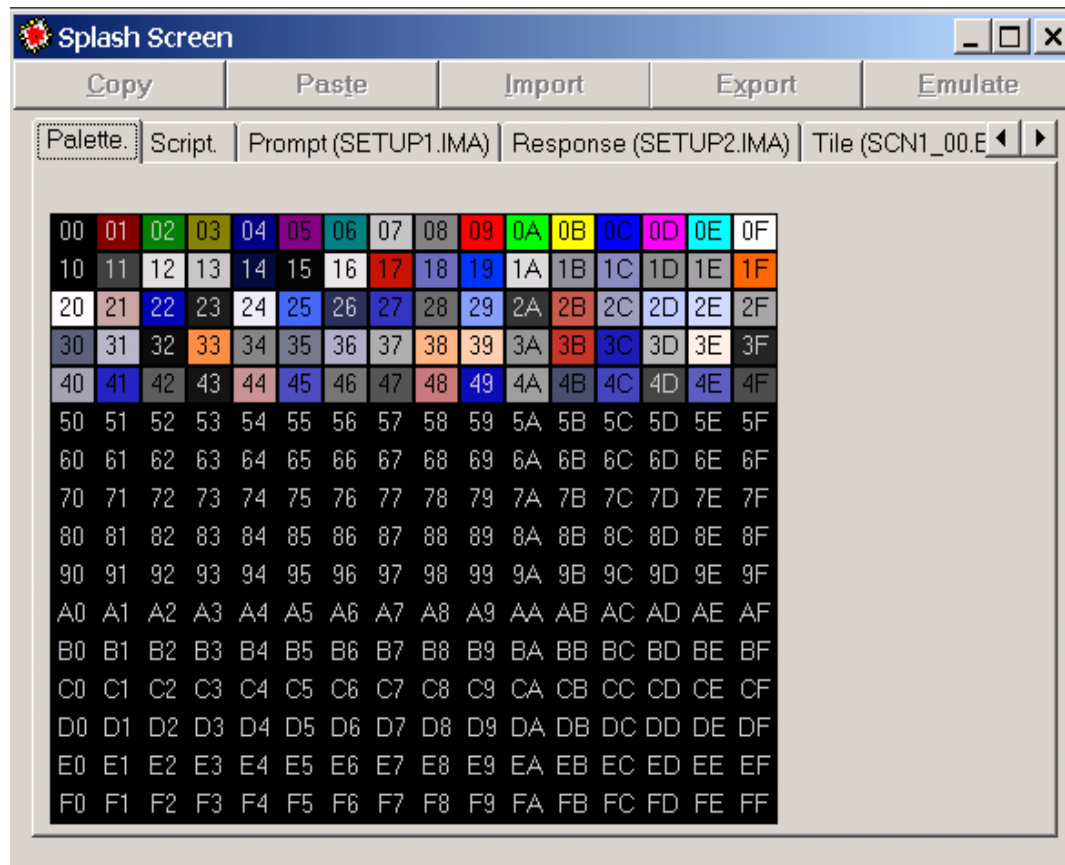
Select the IRQ column to the right of the related INTA...INTD columns to change the bitmaps specifying which IRQs are available for a specific device interrupt line. Selecting the IRQ field brings up a dialog box like this:



Click on the numbers of the IRQs available for this particular device interrupt line. Then select OK when finished.

## Splash Screen

The Splash screen editor allows you to import a new image, export the existing image, edit the Splash screen script file and emulate the logo screen.





## Tab Descriptions

- **Palette(read-only)**—This tab shows the global palette that will be used for all images. It is for reference only and is not editable. This palette of up to 256 colors varies and is derived from the colors that comprise each splash screen image. This palette is optimized on each build by the utility BOOTSCRN.EXE. During the palette optimization process, colors on some images may be adjusted slightly, as all image colors must conform to the global palette that has a maximum of 256 colors.
- **Script**—This tab displays the unassembled script file. The script informs the splash screen engine how the splash screen was constructed.



Editing script requires technical knowledge, otherwise the splash screen might be corrupted. Changing script is deprecated.

- **Rounded Arc (read-only)**—This image is one of two images in the lower-right corner of the splash screen. This graphic is not editable. No other graphic can replace it.
- **Small Logo(read-only)**—This image is one of two images in the lower-right corner of the splash screen. This graphic is not editable. No other graphic can replace it.
- **Prompt Logo**—This image shows the setup prompt that appears in the lower-left corner of the splash screen. You can customize it by importing a new graphic that you create by clicking the Import button. Refer to the limitations described in the FirstBIOS Splash Screen Customization Guide.
- **Response Logo**—This image shows the message that displays immediately after the user presses the setup key or Firststar launch key. You can customize the text by importing a new graphic that you create by clicking the Import button. You can also copy the image and modify it; click the Copy button. Refer to the \*limitations described in the FirstBIOS Splash Screen Customization Guide.
- **Tile Logo**—This image shows the graphic that fills the background of the splash screen. You can import a new image. You can also export the current image, modify it and then import it.
- **Main Logo #n**—This image shows the main logo(s) to be displayed on the splash screen. nnn is the index number of the logo and must match the index number specified in the script. You can import a new image.
- **Transparent color**—You can create transparent areas in the main logo by designating one transparent color. The current transparent color is indicated above the image. To assign a different color to be the transparent color, click that color within the image. If you do not want to have any color be transparent, right-click anywhere in the image.

## Button descriptions (enabled only on tabs with editable content)

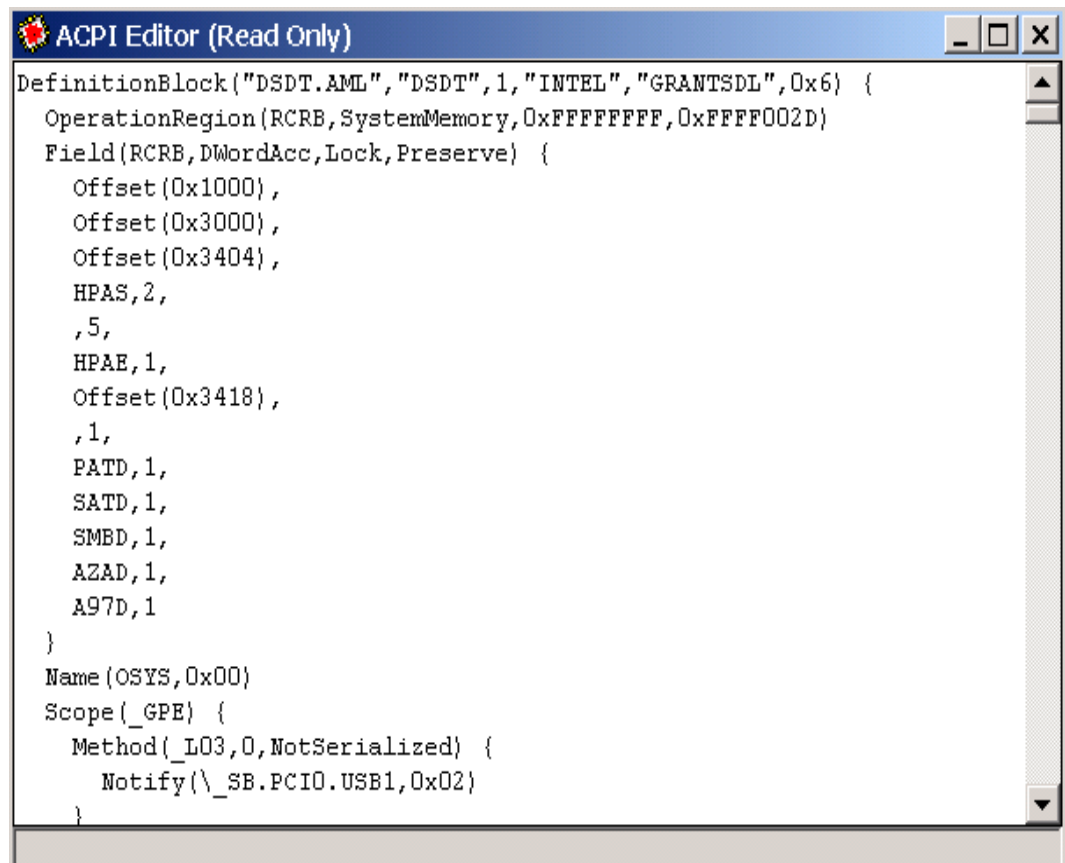
- **Copy**—Copies the currently displayed image to the Windows clipboard. You can then modify it using in the graphics application of your choice and paste or import the new image.
- **Paste**—Pastes an image from the Windows clipboard to this tab, replacing the current image.
- **Import**—Importing a image so you can replace the current image with a selected image file.
- **Export**—Displays a Windows dialog so you can save the current image and modify it using the graphics application of your choice. Once you modify it, you can import the new image.
- **Emulate**—Generate a demo picture so you can view an emulation of a screen.



New image is not always available and availability depends on the size and complexity of the image.

## ACPI Editor

The ACPI Editor disassembles AML (ACPI Machine Language) code back to ASL (ACPI Source Language) code. This allows user to modify any portion of the ASL code, re-compile the modified ASL to AML, and then put it back to BIOS ROM.

The image shows a screenshot of a software window titled "ACPI Editor (Read Only)". The window has a standard Windows-style title bar with minimize, maximize, and close buttons. The main area of the window contains a text editor with ASL (ACPI Source Language) code. The code is as follows:

```
DefinitionBlock("DSDT.AML", "DSDT", 1, "INTEL", "GRANTSDDL", 0x6) {  
    OperationRegion(RCRB, SystemMemory, 0xFFFFFFFF, 0xFFFF002D)  
    Field(RCRB, DWordAcc, Lock, Preserve) {  
        Offset(0x1000),  
        Offset(0x3000),  
        Offset(0x3404),  
        HPAS, 2,  
        , 5,  
        HPAE, 1,  
        Offset(0x3418),  
        , 1,  
        PATD, 1,  
        SATD, 1,  
        SMBD, 1,  
        AZAD, 1,  
        A97D, 1  
    }  
    Name(OSYS, 0x00)  
    Scope(_GPE) {  
        Method(_L03, 0, NotSerialized) {  
            Notify(\_SB.PCI0.USB1, 0x02)  
        }  
    }  
}
```

If the customer has supplied a compliant ASL compiler (currently only Microsoft's ASL.EXE works), the AML code can be edited. If the ASL compiler is not available, it will be a read-only code.



