



**LAN9500/LAN9500i
LAN9500A/LAN9500Ai
LAN9512/LAN9514**

Software User Manual

Revision 1.2 (05-01-09)

Copyright © 2009 SMSC or its subsidiaries. All rights reserved.

The information contained herein is confidential and proprietary to SMSC, shall be used solely in accordance with the agreement pursuant to which it is provided, and shall not be reproduced or disclosed to others without the prior written consent of SMSC. Although the information is believed to be accurate, no responsibility is assumed for inaccuracies. SMSC reserves the right to make changes to this document and to specifications and product descriptions at any time without notice. Neither the provision of this information nor the sale of the described semiconductor devices conveys any licenses under any patent rights or other intellectual property rights of SMSC or others. The product may contain design defects or errors known as anomalies, including but not necessarily limited to any which may be identified in this document, which may cause the product to deviate from published specifications. SMSC products are not designed, intended, authorized or warranted for use in any life support or other application where product failure could cause or contribute to personal injury or severe property damage. Any and all such uses without prior written approval of an officer of SMSC will be fully at the risk of the customer. SMSC is a registered trademark of Standard Microsystems Corporation ("SMSC").

SMSC DISCLAIMS AND EXCLUDES ANY AND ALL WARRANTIES, INCLUDING WITHOUT LIMITATION ANY AND ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE, AND AGAINST INFRINGEMENT AND THE LIKE, AND ANY AND ALL WARRANTIES ARISING FROM ANY COURSE OF DEALING OR USAGE OF TRADE. IN NO EVENT SHALL SMSC BE LIABLE FOR ANY DIRECT, INCIDENTAL, INDIRECT, SPECIAL, PUNITIVE, OR CONSEQUENTIAL DAMAGES; OR FOR LOST DATA, PROFITS, SAVINGS OR REVENUES OF ANY KIND; REGARDLESS OF THE FORM OF ACTION, WHETHER BASED ON CONTRACT; TORT; NEGLIGENCE OF SMSC OR OTHERS; STRICT LIABILITY; BREACH OF WARRANTY; OR OTHERWISE; WHETHER OR NOT ANY REMEDY OF BUYER IS HELD TO HAVE FAILED OF ITS ESSENTIAL PURPOSE, AND WHETHER OR NOT SMSC HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

0.1 Revision History

Table 0.1 Revision History

REVISION LEVEL AND DATE	SECTION/FIGURE/ENTRY	CORRECTION
1.2 (05-01-09)	All	Initial Revision

Table of Contents

0.1	Revision History	2
Chapter 1 Introduction		9
Chapter 2 Windows XP 32/64-Bit Driver		10
2.1	Windows XP 32/64-Bit Driver Installation via EXE	10
2.2	Windows XP 32/64-Bit Driver Installation via INF	14
2.3	Windows XP 32/64-Bit Driver Uninstallation	18
Chapter 3 Windows Vista 32/64-Bit Driver		21
3.1	Windows Vista 32/64-Bit Driver Installation via EXE	21
3.2	Windows Vista 32/64-Bit Driver Installation via INF	25
3.3	Windows Vista 32/64 Bit Driver Uninstallation	30
Chapter 4 MAC OSX Driver		33
4.1	MAC OSX Driver Installation	33
4.2	MAC OSX Driver Uninstallation	41
Chapter 5 Linux Driver		44
5.1	Linux Driver Installation	44
5.2	Linux Driver Uninstallation	50
Chapter 6 Windows CE Driver		51
6.1	Building a CE Image with LAN9500/LAN9500i Source Files	51
6.2	Building a CE Image with LAN9500/LAN9500i DLL & REG Files	55
6.3	USB Driver Update for WinCE 5.0 & 6.0	56
6.4	EHCI/OHCI Driver Physical Memory	56
Chapter 7 Driver Customization		57
7.1	Windows Driver Customization and Localization	57
7.1.1	GUI Visual Elements and Strings Customization	58
7.1.2	Localization	58
7.1.3	Additional Information	58
7.2	Mac OS X Driver Customization	59
7.2.1	Building a Driver Installation Package	59
Chapter 8 Advanced Driver Parameters		65
8.1	Windows Parameters	65
8.2	Mac OS X Parameters	66
8.3	Linux Parameters	67
8.3.1	insmod	67
8.3.2	cmd9500	67
8.3.3	ethtool	68
8.4	Windows CE Parameters	68
Chapter 9 Pre-Execution Environment (PXE) Support		69
9.1	PXE Overview*	69
Chapter 10 Windows Manufacturing Utility		70
10.1	Installation	70
10.1.1	Setup and System Requirements	70
10.1.2	Decompressing the Windows Manufacturing Utility Contents	71

10.1.3	Installing the Manufacturing Utility Device Driver	72
10.2	Operation	77
10.2.1	Starting the Utility	77
10.2.2	Using the Utility	78
10.2.2.1	EEPROM Contents Editor Tab.....	78
10.2.2.2	EEPROM Programmer Tab	81
10.2.2.3	Device Diagnostics Tab	83
10.2.3	Exiting the Utility	85
10.3	Production Testing Methodology and Coverage.	86
10.3.1	Test Methodology	86
10.3.2	Test Time.	86
10.4	Examples	87
10.4.1	Setting Up an Autoburn Session	87
10.4.2	Sharing the Ping Partner Across Several Assembly Lines	88

Chapter 11 DOS Utility Suite 89

11.1	Environment Requirements and Limitations.	89
11.1.1	OS, Processor Mode and Memory	89
11.1.2	USB	89
11.1.3	Networking Partner	89
11.2	9500EEP.EXE	90
11.2.1	Bar Code Scanner Support	91
11.2.2	9500EEP ini Format	91
11.3	9500TEST.EXE	92

APPENDIX A:EEPROM. 94

A.1	EEPROM Format	94
A.1.1	LAN9500/LAN9500i EEPROM Format	94
A.1.2	LAN9500A/LAN9500Ai EEPROM Format.	96
A.1.3	LAN9512 EEPROM Format.	99
A.1.4	LAN9514 EEPROM Format.	101
A.2	EEPROM Programming	104
A.2.1	EEPROM Programming in Windows.	104
A.2.2	EEPROM Programming in DOS	104
A.2.3	EEPROM Programming in Linux.	104
A.2.3.1	ethtool	104
A.2.3.2	cmd9500	104
A.2.4	EEPROM Programming in Windows CE	105
A.2.4.1	Including CE9500eep.exe in a CE Image	106
A.3	EEPROM-less Design	107
A.3.1	EEPROM Item Defaults and Changeability.	107
A.3.2	Determining EEPROM Necessity	108
A.3.3	Utilizing Unchangeable Defaults	108
A.3.3.1	Example 1 - Missing Manufacturer and Product Strings.....	108
A.3.3.2	Example 2 - Missing Serial Number.....	108
A.3.3.3	Example 3 - Maximum Power	108
A.3.4	Modification of Changeable Defaults.	109
A.3.4.1	Overriding the MAC Address	109
A.3.4.2	Overriding USB Behavioral Parameters.....	111
A.3.5	Typical Cases	112

APPENDIX B:Customer Requirements. 113

B.1	MAC Address	113
B.2	USB Vendor ID and Logo	113



Software User Manual

B.3 Serial Number. 114

B.4 WHQL Logo 114

List of Figures

Figure 2.1	Device Installer Invocation	10
Figure 2.2	End User License Agreement	11
Figure 2.3	Installation Progress Window.	11
Figure 2.4	Device Driver Installation Complete Screen - Windows XP 32-bit.	12
Figure 2.6	Found New Hardware Task Bar Notification 1.	13
Figure 2.7	Found New Hardware Task Bar Notification 2.	13
Figure 2.5	Device Driver Installation Complete Screen - Windows XP 64-bit.	13
Figure 2.8	Found New Hardware Task Bar Notification	14
Figure 2.9	Found New Hardware Wizard	14
Figure 2.10	Found New Hardware Wizard 2.	15
Figure 2.11	Search and Installation Options Window	15
Figure 2.12	Browse Window - Windows XP 32-Bit	16
Figure 2.13	Browse Window - Windows XP 64-Bit	16
Figure 2.14	Installation Progress Window.	17
Figure 2.15	Device Driver Installation Complete Screen.	17
Figure 2.16	Start Menu - Run	18
Figure 2.17	Run Window.	18
Figure 2.18	Device Manager Window - Uninstall	19
Figure 2.19	Confirm Device Removal Window	19
Figure 2.20	Device Removed from Device Manager Window.	20
Figure 3.1	Device Installer Invocation	21
Figure 3.2	End User License Agreement	22
Figure 3.3	Installation Progress Window.	22
Figure 3.4	Device Driver Installation Complete Screen - Windows Vista 32-Bit	23
Figure 3.5	Device Driver Installation Complete Screen - Windows Vista 64-Bit	23
Figure 3.6	Installing Device Driver Software Task Bar Notification.	24
Figure 3.7	Device Driver Software Installed Successfully Task Bar Notification.	24
Figure 3.8	Found New Hardware Window	25
Figure 3.9	Search Online Option Window.	26
Figure 3.10	Insert Disk Window	26
Figure 3.11	Driver Install Options Window	27
Figure 3.12	Browse Window - Windows Vista 32-Bit	27
Figure 3.13	Browse Window - Windows Vista 64-Bit	28
Figure 3.14	Installation Progress Window.	28
Figure 3.15	Device Driver Installation Complete Screen.	29
Figure 3.16	Device Driver Installation Success Task Bar Notification.	29
Figure 3.17	Start Menu Search.	30
Figure 3.18	Run Window.	30
Figure 3.19	Device Manager Window	31
Figure 3.20	Confirm Device Removal Window	31
Figure 3.21	Device Uninstall Progress Window	32
Figure 3.22	Device Removed from Device Manager	32
Figure 4.1	Introduction Screen	33
Figure 4.2	License Window.	34
Figure 4.3	Agree/Disagree Drop Down	35
Figure 4.4	Installation Type Window	36
Figure 4.5	Installer Name/Password Drop Down	36
Figure 4.6	Summary Window Indicating Successful Installation.	37
Figure 4.7	Cancel This Pop-Up If It Appears.	37
Figure 4.8	Open Network Preferences	38
Figure 4.9	Network Window with Pop-Up	38
Figure 4.10	New Service Insertion in Network Table	39

Software User Manual

Figure 4.11 Device Connected with IP Address	40
Figure 4.12 Driver Removal	41
Figure 4.13 Open Network Preferences	41
Figure 4.14 Network Preferences	42
Figure 4.15 Confirm Changes	43
Figure 5.1 Determining Presence of gcc	44
Figure 5.2 Install gcc	44
Figure 5.3 Tar of Distribution File	45
Figure 5.4 Execution of Build Script	45
Figure 5.5 Installing Driver Binaries	46
Figure 5.6 Network Configuration Window	46
Figure 5.7 Add New Ethernet Device Type	47
Figure 5.8 Device Selection	47
Figure 5.9 IP Address Generation	48
Figure 5.10 Device Creation	49
Figure 5.11 Uninstallation Steps	50
Figure 6.1 Example Unzip to DRIVERS Directory	52
Figure 6.2 smsc9500 Folder Contents	52
Figure 6.3 DIRS List of Drivers to be Built	53
Figure 6.4 Adding smsc9500.dll into platform.bib	53
Figure 6.5 Including smsc9500.reg in platform.reg	54
Figure 6.6 Including smsc9500.reg in platform.reg	55
Figure 6.7 Adding smsc9500.dll into platform.bib	55
Figure 7.1 Prescript Creation	59
Figure 7.2 Postscript Creation	59
Figure 7.3 Adding LAN9500.kext	60
Figure 7.4 Selecting Background Image	60
Figure 7.5 Selecting Introduction File	61
Figure 7.6 Selecting Read Me File	61
Figure 7.7 Selecting License File	62
Figure 7.8 Selecting Conclusion File	62
Figure 7.9 Configuration Information	63
Figure 7.10 Content Information	63
Figure 7.11 Scripts Information	64
Figure 8.1 Windows XP Advanced Parameters	65
Figure 8.2 Windows Vista Advanced Parameters	66
Figure 8.3 Mac OS X info.plist Advanced Parameters	67
Figure 10.1 Self-Extraction Window	71
Figure 10.2 Decompressed Contents	72
Figure 10.3 Found New Hardware Wizard Welcome Window	73
Figure 10.4 Install from a List or Specific Location Option	73
Figure 10.5 Don't Search Option	74
Figure 10.6 Install From Disk Option	74
Figure 10.7 Select Network Adapter Window	75
Figure 10.8 Hardware Installation Warning Window	75
Figure 10.9 Hardware Installation Complete Window	76
Figure 10.10 Launching the Windows Manufacturing Utility	77
Figure 10.11 EEPROM Contents Editor Tab	78
Figure 10.12 EEPROM Programmer Tab	81
Figure 10.13 Device Diagnostics Tab	83
Figure A.1 Inserting CE9500eep.exe into Platform.bib	106

List of Tables

Table 0.1	Revision History	2
Table 6.1	Windows CE LAN9500/LAN9500i Device Driver Source Files	51
Table 7.1	Automated Installer Package Contents	57
Table 10.1	Setup and System Requirements	70
Table 10.2	Decompressed File Descriptions	71
Table 10.3	EEPROM Contents Editor Tab - Fields	79
Table 10.4	EEPROM Contents Editor Tab - Actions	80
Table 10.5	EEPROM Programmer Tab - Actions	82
Table 10.6	EEPROM Programmer Tab - Checkboxes	82
Table 10.7	EEPROM Programmer Tab - Status	82
Table 10.8	Device Diagnostics Tab - Tests	83
Table 10.9	Device Diagnostics Tab - Loops	84
Table 10.10	Device Diagnostics Tab - Links	85
Table 10.11	Device Diagnostics Tab - Status Information	85
Table 10.12	Device Diagnostics Tab - Actions	85
Table 10.13	Network Address Assignment Example	88
Table 11.1	9500EEP.EXE Command Switch Definitions	90
Table 11.2	9500TEST.EXE Command Switch Definitions	92
Table A.1	LAN9500/LAN9500i EEPROM Format	94
Table A.2	LAN9500/LAN9500i Configuration Flags Description	95
Table A.3	LAN9500A/LAN9500Ai EEPROM Format	96
Table A.4	LAN9500A/LAN9500Ai Configuration Flags	97
Table A.5	LAN9500A/LAN9500Ai GPIO PME Flags	98
Table A.6	LAN9512 EEPROM Format	99
Table A.7	LAN9512 Configuration Flags Description	101
Table A.8	LAN9514 EEPROM Format	101
Table A.9	LAN9514 Configuration Flags Description	103
Table A.10	cmd9500 Option Descriptions	105
Table A.11	EEPROM Items Defaults and Changeability	107
Table A.12	Descriptor RAM Contents	110
Table A.13	Descriptor RAM Definitions	110

Chapter 1 Introduction

This manual provides detailed instructions on the installation and uninstallation of LAN9500/LAN9500i, LAN9500A/LAN9500Ai, LAN9512, and LAN9514 software drivers under various operating systems:

- [Windows XP 32/64-Bit Driver](#)
- [Windows Vista 32/64-Bit Driver](#)
- [MAC OSX Driver](#) (10.5)
- [Linux Driver](#) (Kernel 2.6.12 or greater)
- [Windows CE Driver](#)

Additional software related information is provided, including appendices for EEPROM and customer requirement related information:

- [Driver Customization](#)
- [Advanced Driver Parameters](#)
- [Pre-Execution Environment \(PXE\) Support](#)
- [Windows Manufacturing Utility](#)
- [DOS Utility Suite](#)
- [EEPROM](#)
- [Customer Requirements](#)

The latest drivers and supporting documentation may be obtained by visiting the SMSC website:

<http://www.smsc.com>

Additional advanced information can be obtained through a free E-Services account. To create a free E-Services account, visit the SMSC E-Services webpage:

<https://www2.smsc.com/main.nsf>

Note: For the purpose of this documentation, the term “device” is used interchangeably with the following SMSC products:

- LAN9500/LAN9500i
- LAN9500A/LAN9500Ai
- LAN9512
- LAN9514

Note: The screen shots contained in this document are for illustration purposes only. The text contained therein may be different from what the user observes on his screen, due to driver and/or OS customization. Unless otherwise noted, all screen shots are shown in relation to the LAN9500/LAN9500i. With the exception of the part number, which may change or remain LAN9500/LAN9500i, the screens and procedures will be identical for the LAN9512, LAN9514, and LAN9500A/LAN9500Ai cases.

Note: Please refer to the respective software release notes for the latest information.

Chapter 2 Windows XP 32/64-Bit Driver

This chapter details the installation and uninstallation of the Windows XP 32/64-bit driver.

The Windows XP 32/64-Bit driver may be installed in two ways:

- [Windows XP 32/64-Bit Driver Installation via EXE](#) (preferred method)
- [Windows XP 32/64-Bit Driver Installation via INF](#)

Note: Official driver releases are provided by SMSC in EXE format only. Refer to [Chapter 7, "Driver Customization,"](#) on page 57 for details on how to locate the files that allow INF-only installation.

Windows XP 32/64-Bit driver uninstallation is detailed in [Section 2.3, "Windows XP 32/64-Bit Driver Uninstallation,"](#) on page 18.

2.1 Windows XP 32/64-Bit Driver Installation via EXE

The folder containing the distribution files may be copied to the desktop or any other convenient, known, place within the directory structure. Clicking the folder will result in the installer EXE file and the release notes TXT file being displayed.

Note: The device must not be plugged into the computer prior to installing the driver software.

To begin installation, click on the installer icon. The following windows will then appear:

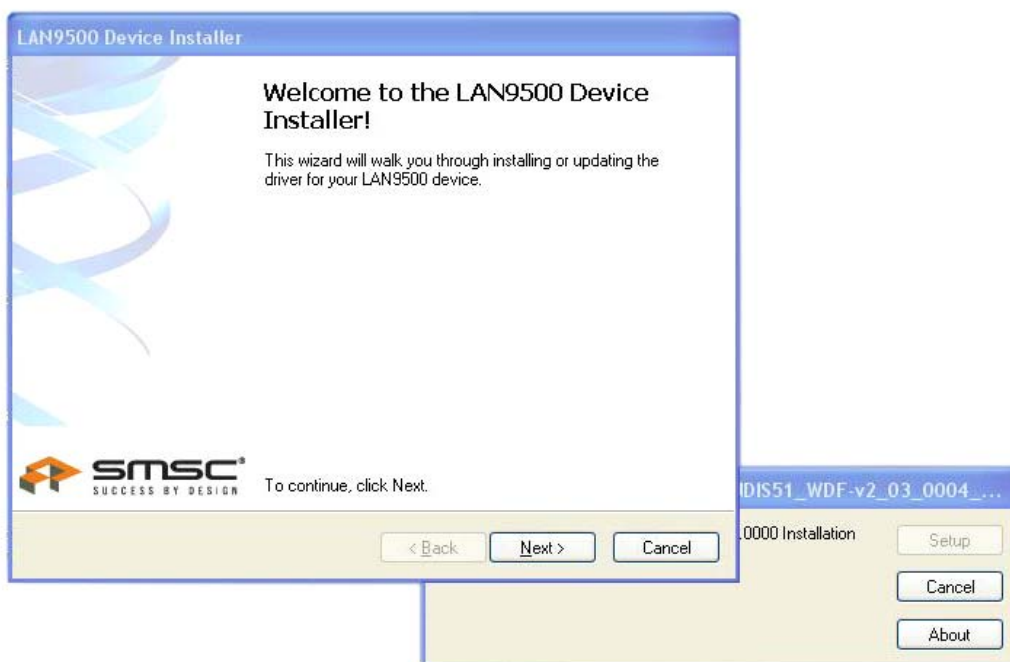


Figure 2.1 Device Installer Invocation

Note: Two windows are launched when the device installer is invoked: the Device Installer window and the Winzip Self Extractor window (hidden behind the Device Installer window).

Continue the installation process by clicking the "Next" button in the Device Installer window.

Software User Manual

The End User License Agreement (EULA) will appear within the Device Installer window. Click the “I accept this EULA” radio button to accept the End User License Agreement. The window will then display the “Next” button, as illustrated in [Figure 2.2](#), which will permit the installation process to continue.

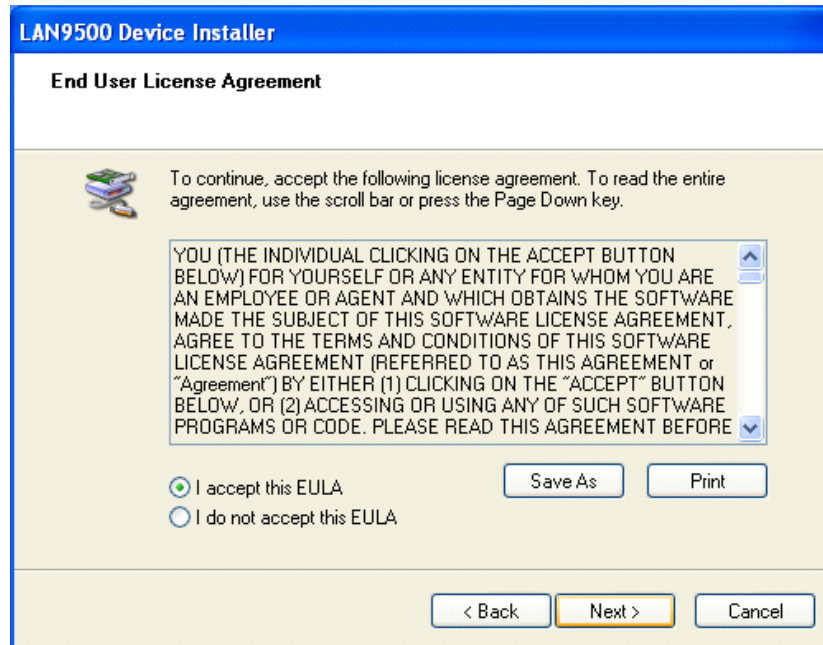


Figure 2.2 End User License Agreement

Click the “Next” button to continue the installation process. The Device Installer window will change to a progress window, as shown in [Figure 2.3](#), indicating some time may be necessary for the installation.

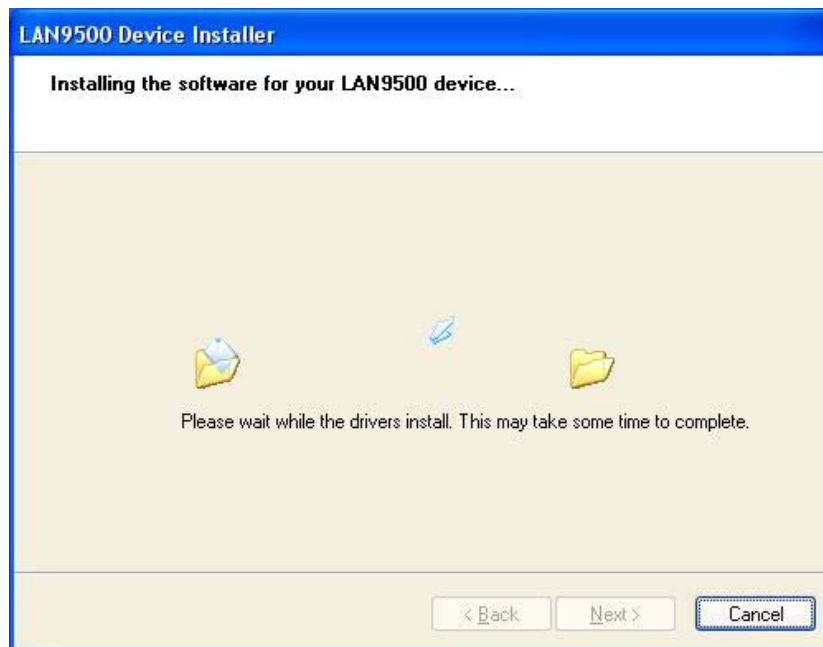


Figure 2.3 Installation Progress Window

When finished, the installation progress window will change to indicate the driver has been installed. Click the “Finish” button to complete the driver installation.

Note: Only the appropriate 32- or 64-bit driver will be installed, dependent on the version of Windows XP being used. [Figure 2.4](#) shows the installation of the 32-bit driver in a Windows XP 32-bit environment. [Figure 2.5](#) shows the installation of the 64-bit driver in a Windows XP 64-bit environment.

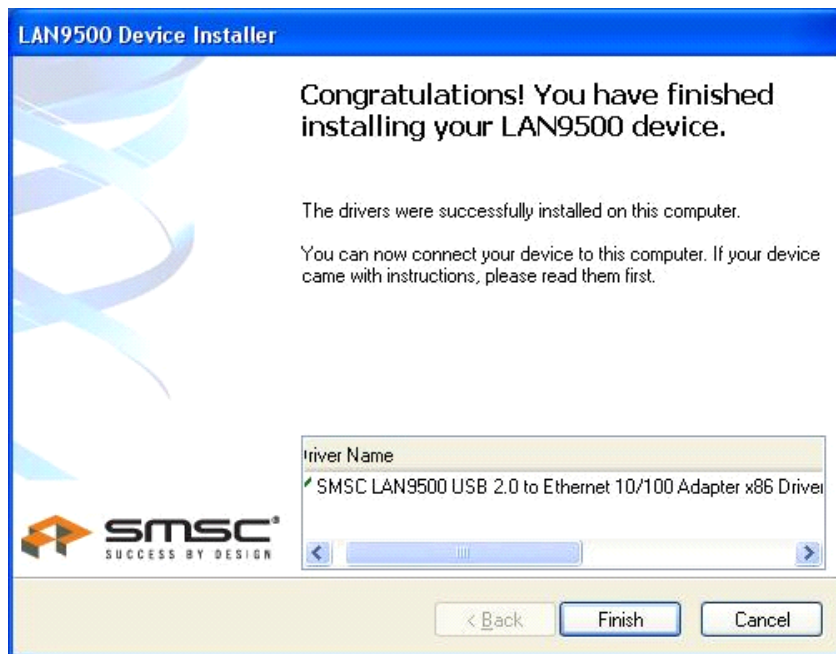


Figure 2.4 Device Driver Installation Complete Screen - Windows XP 32-bit

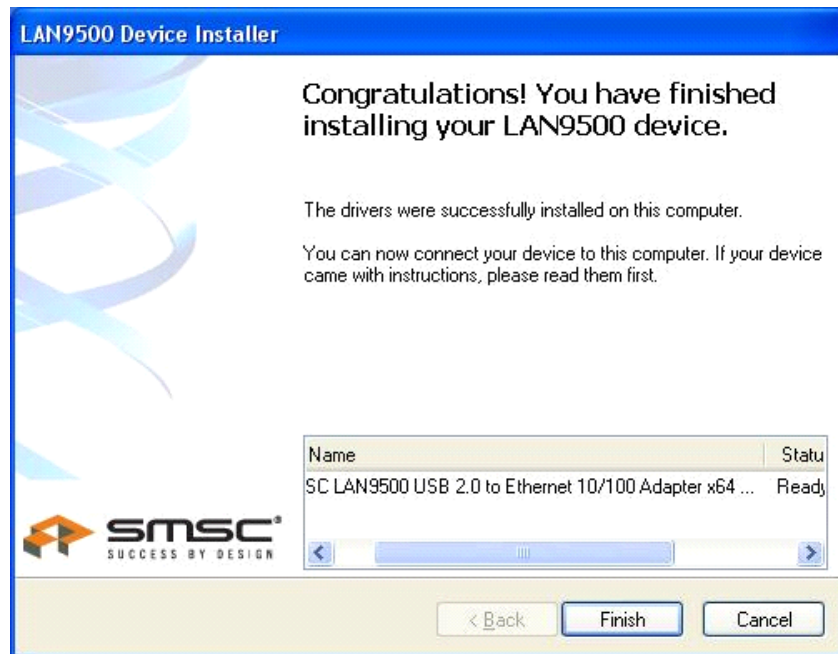


Figure 2.5 Device Driver Installation Complete Screen - Windows XP 64-bit

The device must now be plugged into a free USB port on the computer. Once this is accomplished, the balloon notifications messages shown in [Figure 2.6](#) and [Figure 2.7](#) will appear in the task bar.

At this point, the network device installation is complete. The device will be setup to have its IP address assigned by a DHCP server. If desired, this configuration can be changed to use a manually assigned IP address. This can be achieved from the properties of the internet protocol for the device, which is accessible through the "Network Connections" control panel. Details are outside of the scope of this document.



Figure 2.6 Found New Hardware Task Bar Notification 1



Figure 2.7 Found New Hardware Task Bar Notification 2

2.2 Windows XP 32/64-Bit Driver Installation via INF

The device driver may be alternatively installed via an INF file. This section details the INF installation method.

Note: The EXE method described in [Section 2.1](#) is the preferred method of installation.

Before beginning installation, the folder containing the SMSC INF distribution files may be copied to the desktop or any other convenient, known, place within the directory structure.

To begin installation, connect the device to a free USB port on the computer. Once this is accomplished, the balloon notifications messages shown in [Figure 2.8](#) will appear in the task bar.



Figure 2.8 Found New Hardware Task Bar Notification

The Found New Hardware Wizard window will then pop-up as shown in [Figure 2.9](#). Click the “No, not at this time” radio button and click “Next”.



Figure 2.9 Found New Hardware Wizard

Select the “Install from a list or specific location (Advanced)” radio button, as shown in [Figure 2.10](#), and click “Next”.

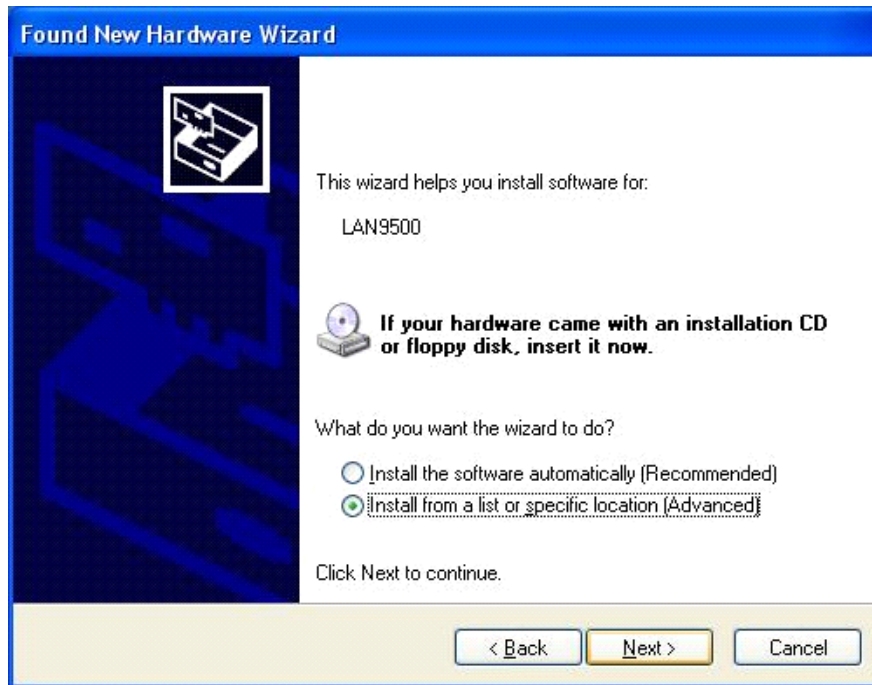


Figure 2.10 Found New Hardware Wizard 2

Select the “Search for the best driver in these locations” radio button. Check the “Include this location in the search” checkbox and click the “Browse” button.

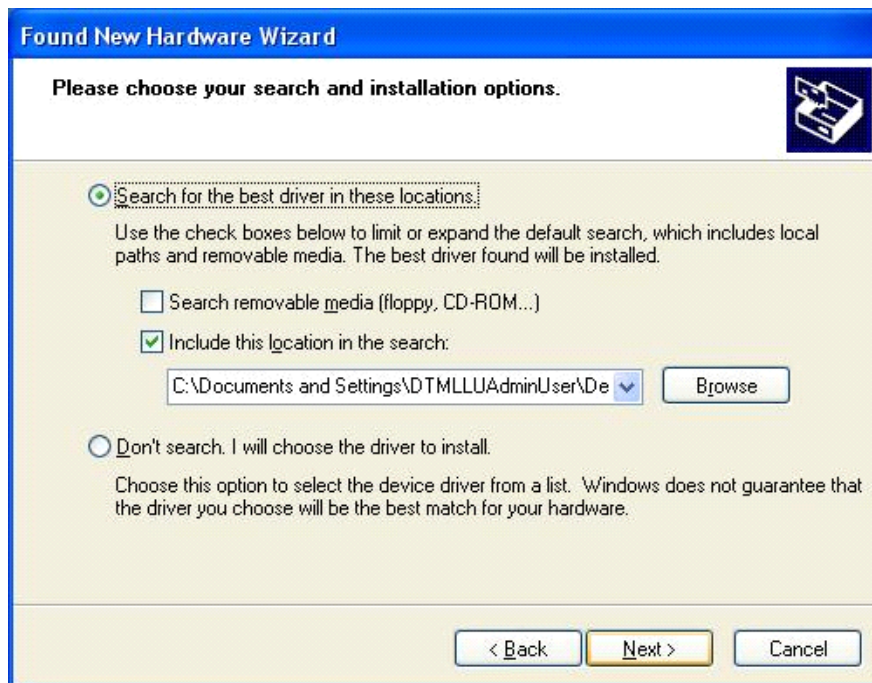


Figure 2.11 Search and Installation Options Window

Via the “Browse for Folder” window, browse to the location of the copied SMSC INF distribution files. Click “OK”, and then “Next” to install.

Note: For Windows 32-bit installations, browse to the “x86” folder within the copied SMSC INF distribution files, as shown in [Figure 2.12](#). For Windows 64-bit installations, browse to the “x64” folder within the copied SMSC INF distribution files, as shown in [Figure 2.13](#).



Figure 2.12 Browse Window - Windows XP 32-Bit



Figure 2.13 Browse Window - Windows XP 64-Bit

Software User Manual

The Device Installer window will change to a progress window, as shown in [Figure 2.14](#), indicating some time may be necessary for the installation.

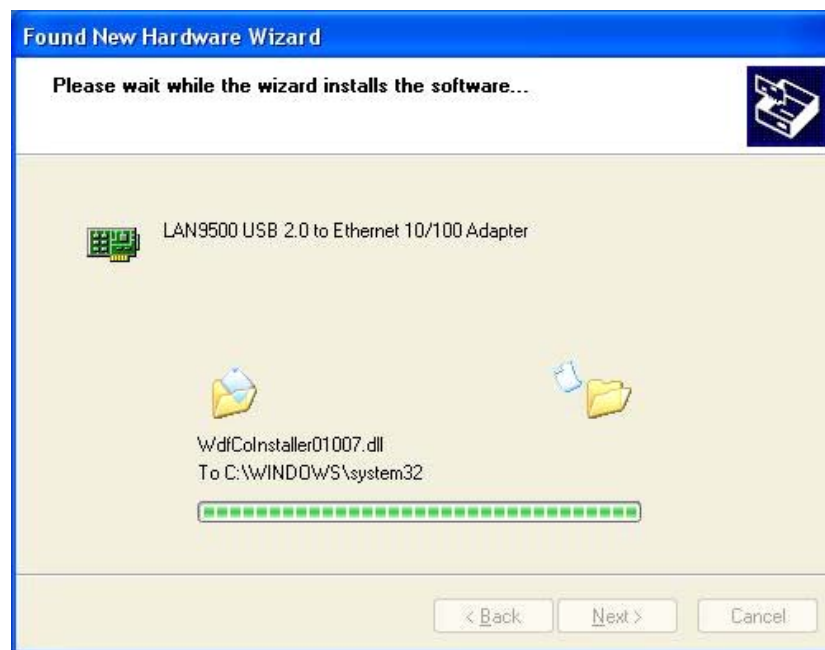


Figure 2.14 Installation Progress Window

When finished, the installation progress window will change to indicate the driver has been installed. Click the "Finish" button to complete the driver installation.



Figure 2.15 Device Driver Installation Complete Screen

2.3 Windows XP 32/64-Bit Driver Uninstallation

Note: Manual uninstallation is the only supported form of uninstallation at this time. An automated uninstaller via the “Add/Remove Programs” dialog may be included in future revisions.

To uninstall the Windows XP 32/64-bit software, select “Run...” from the start menu, as shown in [Figure 2.16](#).



Figure 2.16 Start Menu - Run

In the Run window, type “devmgmt.msc” and click “OK”, as shown in [Figure 2.17](#). This will open the Device Manager window.

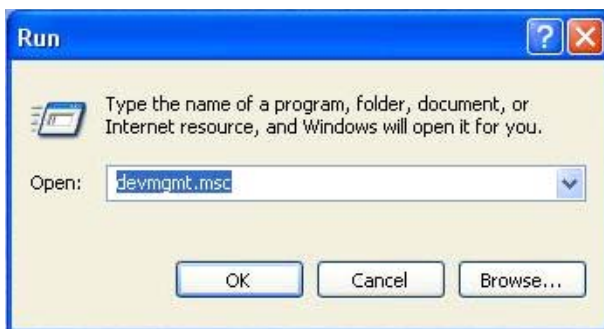


Figure 2.17 Run Window

Software User Manual

In the Device Manager window, browse to the “Network Adapters” section and select the device. Right click on the device and select “Uninstall”, as shown in [Figure 2.18](#).

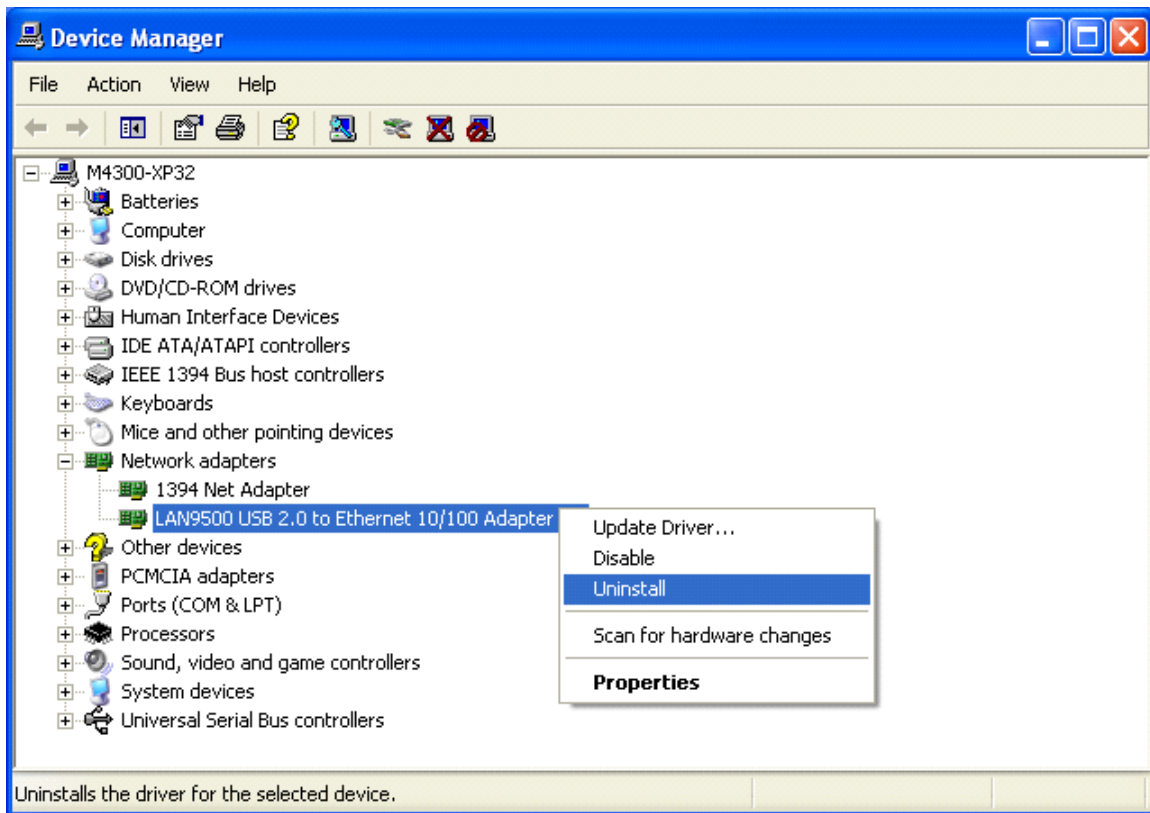


Figure 2.18 Device Manager Window - Uninstall

Click “OK” in the “Confirm Device Removal” window, as shown in [Figure 2.19](#).



Figure 2.19 Confirm Device Removal Window

For a full uninstall, the `oem<n>.inf` files that are a copy of the `net9500-<arch>-n51f.inf` file (which Windows created at installation time) must be removed from the `<WINDIR>\inf` folder. Do NOT remove `oem<n>.inf` files that are not a copy of the `net9500-<arch>-n51f.inf` files, since this could affect operation of other devices.

The uninstallation process is now complete. As illustrated in [Figure 2.20](#), the device driver will no longer be listed in the Device Manager window. The Device Manager window may now be closed.

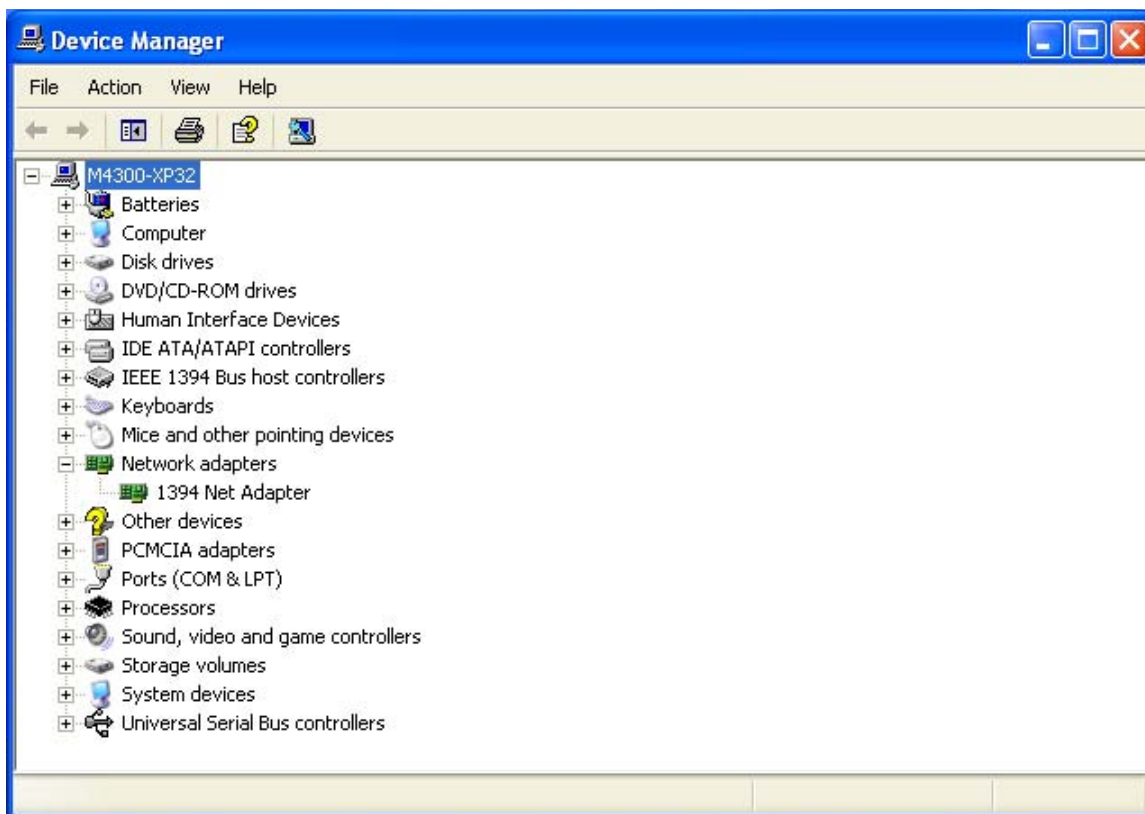


Figure 2.20 Device Removed from Device Manager Window

Chapter 3 Windows Vista 32/64-Bit Driver

This chapter details the installation and uninstallation of the Windows Vista 32/64-bit driver.

The Windows Vista 32/64-bit driver may be installed in two ways:

- [Windows Vista 32/64-Bit Driver Installation via EXE](#) (preferred method)
- [Windows Vista 32/64-Bit Driver Installation via INF](#)

Note: Official driver releases are provided by SMSC in EXE format only. Refer to [Chapter 7, "Driver Customization," on page 57](#) for details on how to locate the files that allow INF-only installation.

Windows Vista 32/64-bit driver uninstallation is detailed in [Section 3.3, "Windows Vista 32/64 Bit Driver Uninstallation," on page 30](#).

3.1 Windows Vista 32/64-Bit Driver Installation via EXE

The folder containing the distribution files should be copied to the desktop or any other convenient, known, place within the directory structure. Clicking the folder will result in the installer EXE file and the release notes TXT file being displayed.

To begin installation, click on the installer icon. The following windows will then appear:

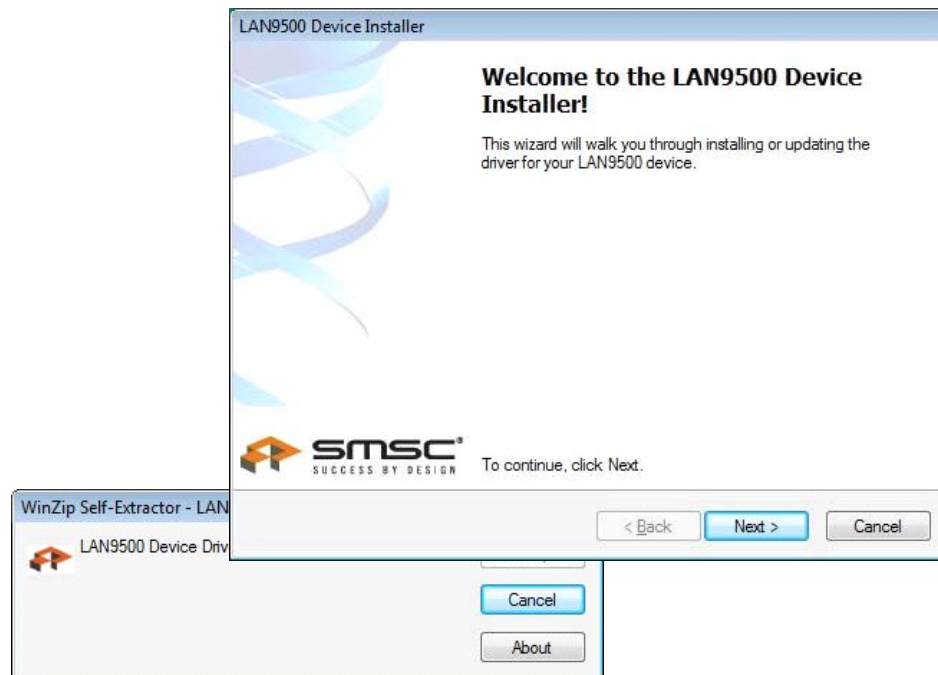


Figure 3.1 Device Installer Invocation

Note: Two windows are launched when the device installer is invoked: the Device Installer window and the Winzip Self Extractor window (hidden behind the Device Installer window).

Continue the installation process by clicking the "Next" button in the Device Installer window.

The End User License Agreement (EULA) will appear within the Device Installer window. Click the "I accept this EULA" radio button to accept the End User License Agreement. The window will then

display the “Next” button, as illustrated in [Figure 3.2](#), which will permit the installation process to continue.

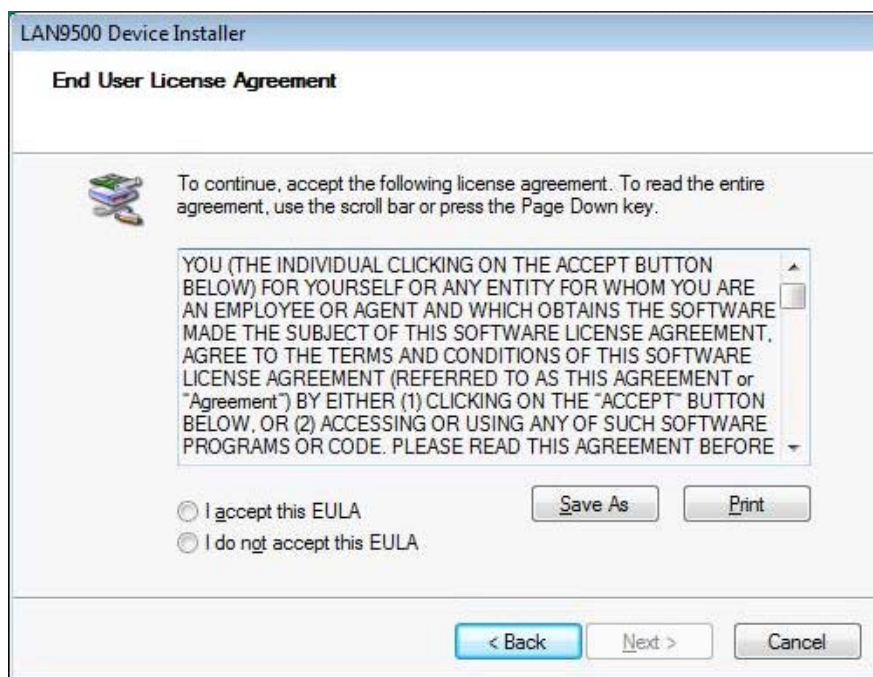


Figure 3.2 End User License Agreement

Click the “Next” button to continue the installation process. The Device Installer window will change to a progress window, as shown in [Figure 3.3](#), indicating some time may be necessary for the installation

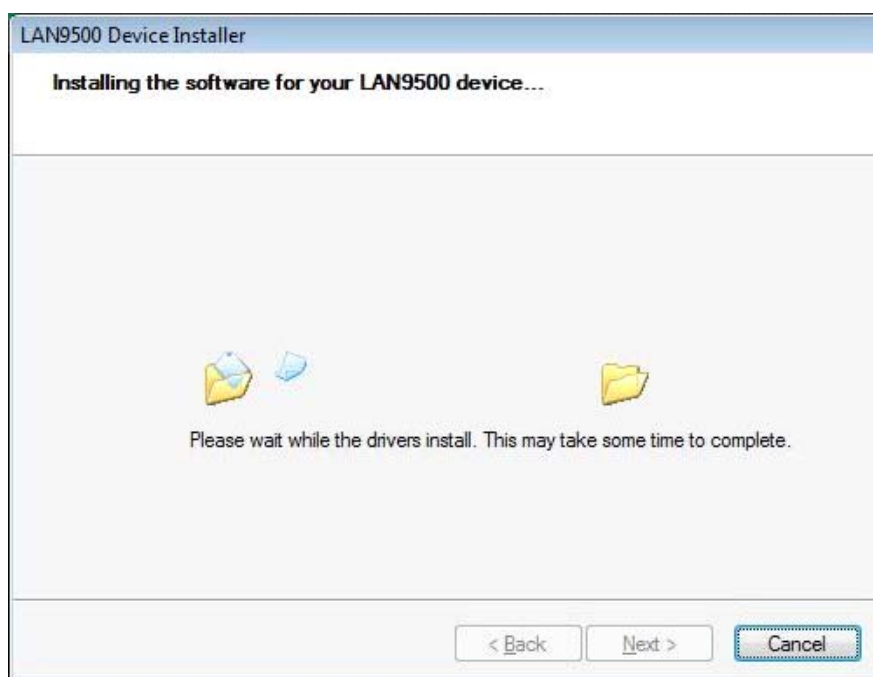


Figure 3.3 Installation Progress Window

Software User Manual

When finished, the installation progress window will change to indicate the driver has been installed. Click the “Finish” button to complete the driver installation.

Only the appropriate 32- or 64-bit driver will be installed, dependent on the version of Windows Vista being used. [Figure 3.4](#) shows the installation of the 32-bit driver in a Windows Vista 32-bit environment. [Figure 3.5](#) shows the installation of the 64-bit driver in a Windows Vista 64-bit environment



Figure 3.4 Device Driver Installation Complete Screen - Windows Vista 32-Bit



Figure 3.5 Device Driver Installation Complete Screen - Windows Vista 64-Bit

The device must now be plugged into a free USB port on the computer. Once this is accomplished, the balloon notification message shown in [Figure 3.6](#) will appear in the task bar.



Figure 3.6 Installing Device Driver Software Task Bar Notification

When the software installation is complete, the balloon notification message shown in [Figure 3.7](#) will appear, signaling completion of the installation.



Figure 3.7 Device Driver Software Installed Successfully Task Bar Notification

At this point, the network device installation is complete. The device will be setup to have its IP address assigned by a DHCP server. If desired, this configuration can be changed to use a manually assigned IP address. This can be achieved from the properties of the internet protocol for the device, which is accessible through the "Network Connections" control panel. Details are outside of the scope of this document

3.2 Windows Vista 32/64-Bit Driver Installation via INF

The device driver may be alternatively installed via an INF file. This section details the INF installation method.

Note: The EXE method described in [Section 3.1](#) is the preferred method of installation.

Before beginning installation, the folder containing the SMSC INF distribution files may be copied to the desktop or any other convenient, known, place within the directory structure.

To begin installation, connect the device to a free USB port on the computer. Once this is accomplished, the “Found New Hardware” window will appear, as shown in [Figure 3.8](#). Click the “Locate and install driver software (recommended)” option.



Figure 3.8 Found New Hardware Window

Click “Don’t search online”, as shown in [Figure 3.9](#).

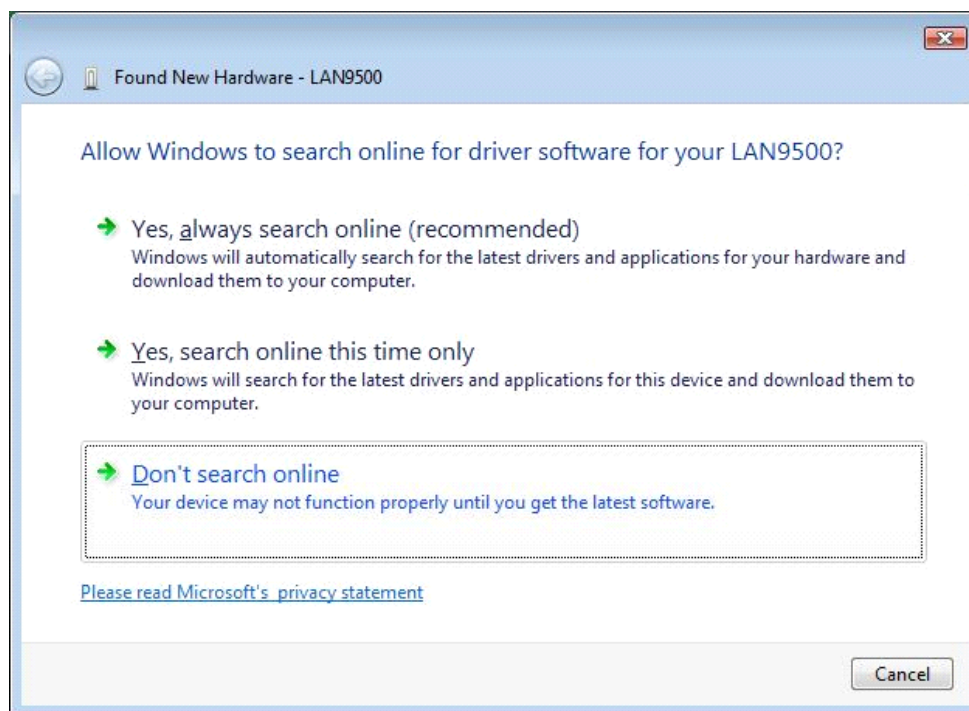


Figure 3.9 Search Online Option Window

Click “I don’t have the disc. Show me other options”, as shown in [Figure 3.10](#), and click “Next”.

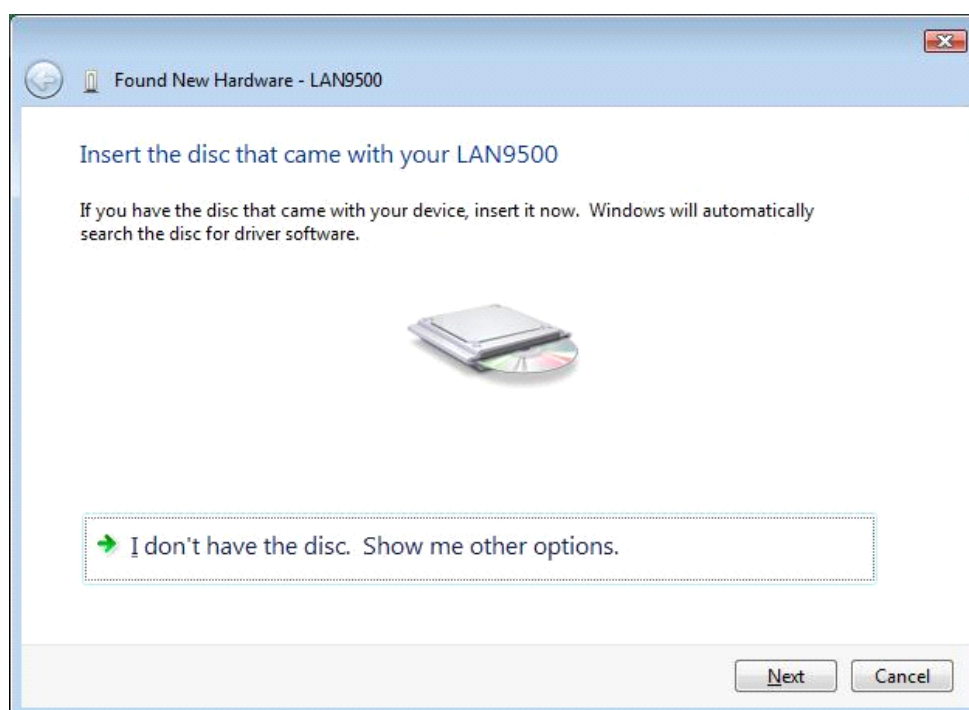


Figure 3.10 Insert Disk Window

Software User Manual

Click “Browse my computer for driver software (advanced)”, as shown in [Figure 3.11](#).

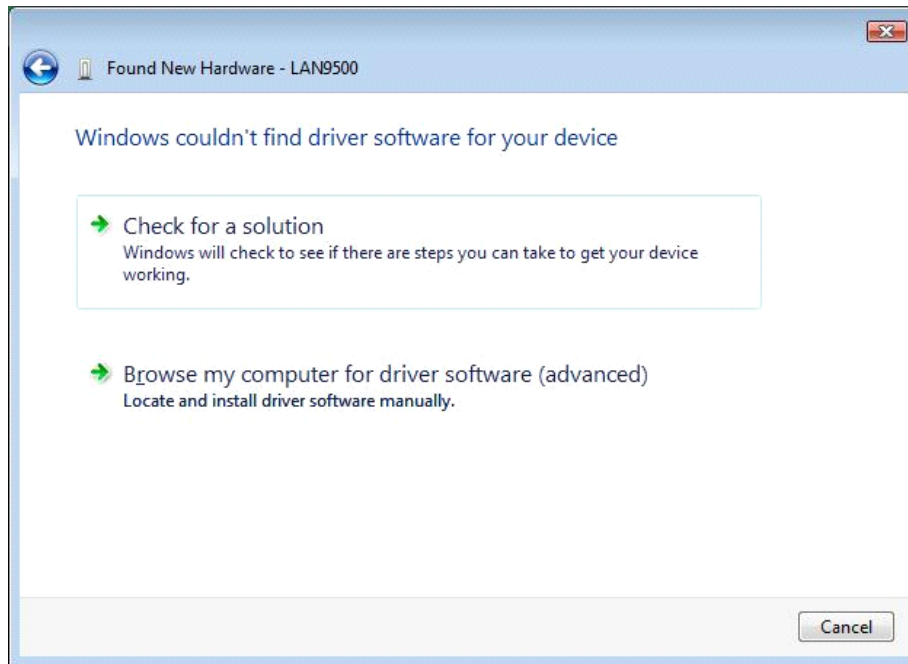


Figure 3.11 Driver Install Options Window

Click the “Browse...” button and browse to the location of the copied SMSC INF distribution files. Click “OK”, and then “Next” to install.

Note: For Windows 32-bit installations, browse to the “x86” folder within the copied SMSC INF distribution files, as shown in [Figure 3.12](#). For Windows 64-bit installations, browse to the “x64” folder within the copied SMSC INF distribution files, as shown in [Figure 3.13](#).

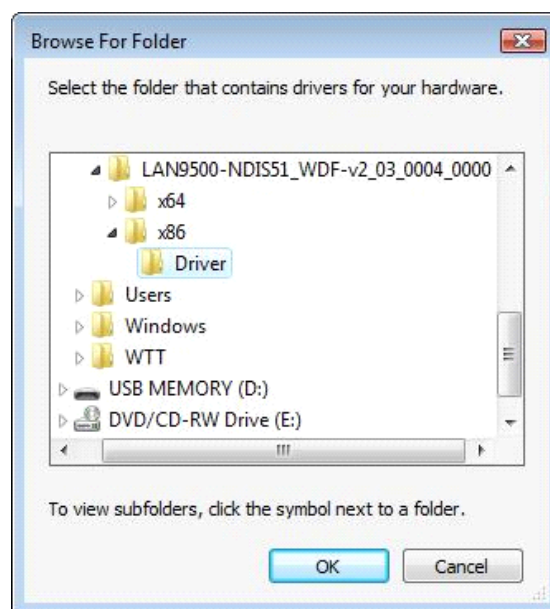


Figure 3.12 Browse Window - Windows Vista 32-Bit

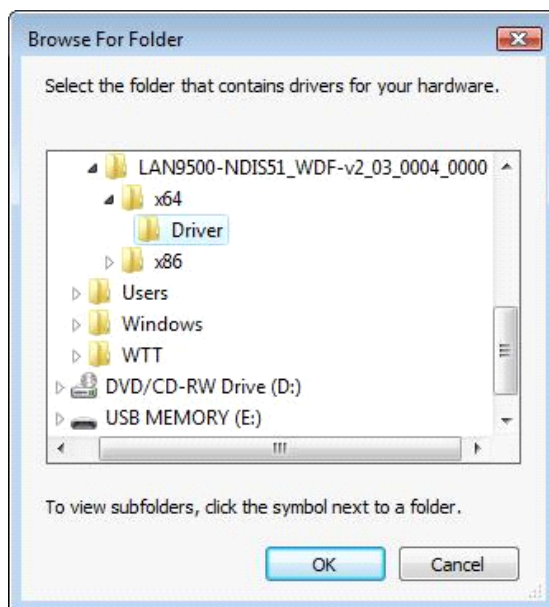


Figure 3.13 Browse Window - Windows Vista 64-Bit

The Device Installer window will change to a progress window, as shown in [Figure 3.14](#), indicating some time may be necessary for the installation.

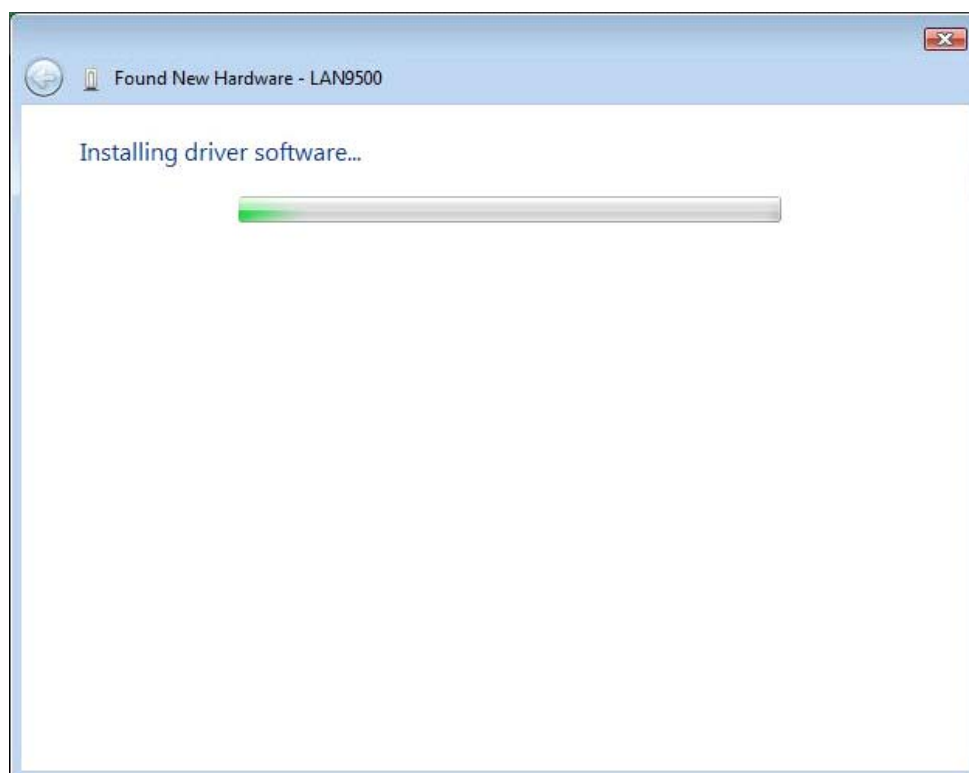


Figure 3.14 Installation Progress Window

Software User Manual

When finished, the installation progress window will change to indicate the driver has been installed. Click the “Close” button to complete the driver installation.

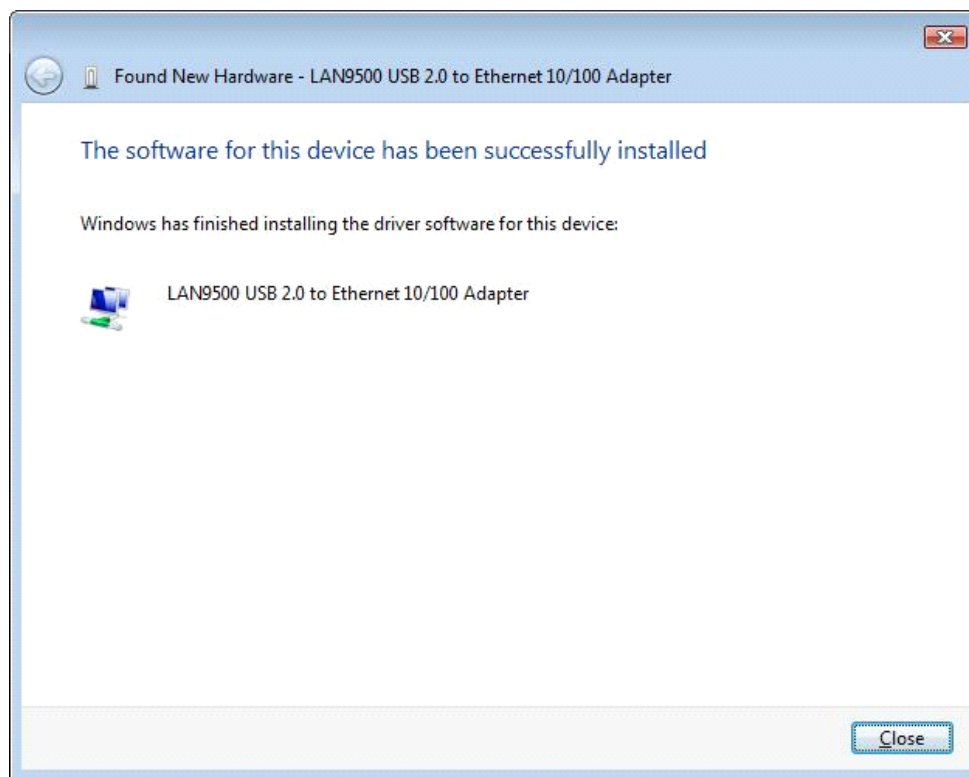


Figure 3.15 Device Driver Installation Complete Screen

A device driver installation notification will pop-up as shown in [Figure 3.16](#), indicating the driver installation is complete.

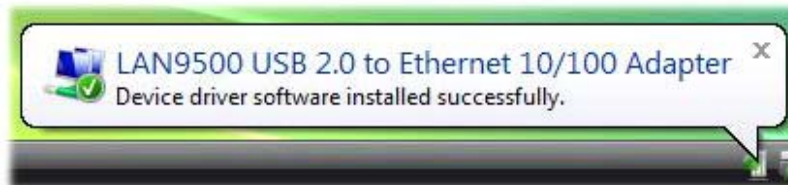


Figure 3.16 Device Driver Installation Success Task Bar Notification

3.3 Windows Vista 32/64 Bit Driver Uninstallation

Note: Manual uninstallation is the only supported form of uninstallation at this time. An automated uninstaller via Vista's "Programs and Features" may be included in future revisions.

To uninstall the Windows Vista 32/64-bit software, type "Run" in the Start menu search field and click on "Run" in the programs list, as shown in [Figure 3.17](#).

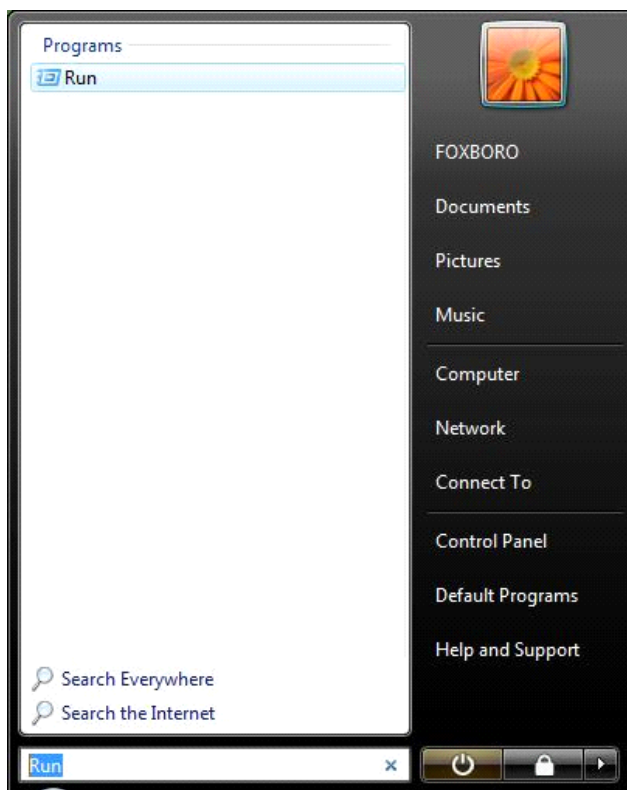


Figure 3.17 Start Menu Search

In the Run window, type "devmgmt.msc" and click "OK", as shown in [Figure 3.18](#). This will open the Device Manager window.

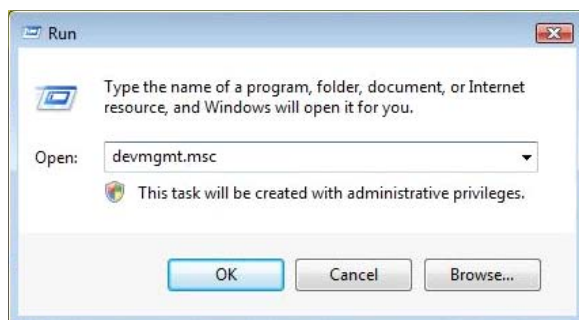


Figure 3.18 Run Window

Software User Manual

In the Device Manager window, browse to the “Network adapters” section and select the device. Right click on the device and select “Uninstall”, as shown in [Figure 3.19](#).

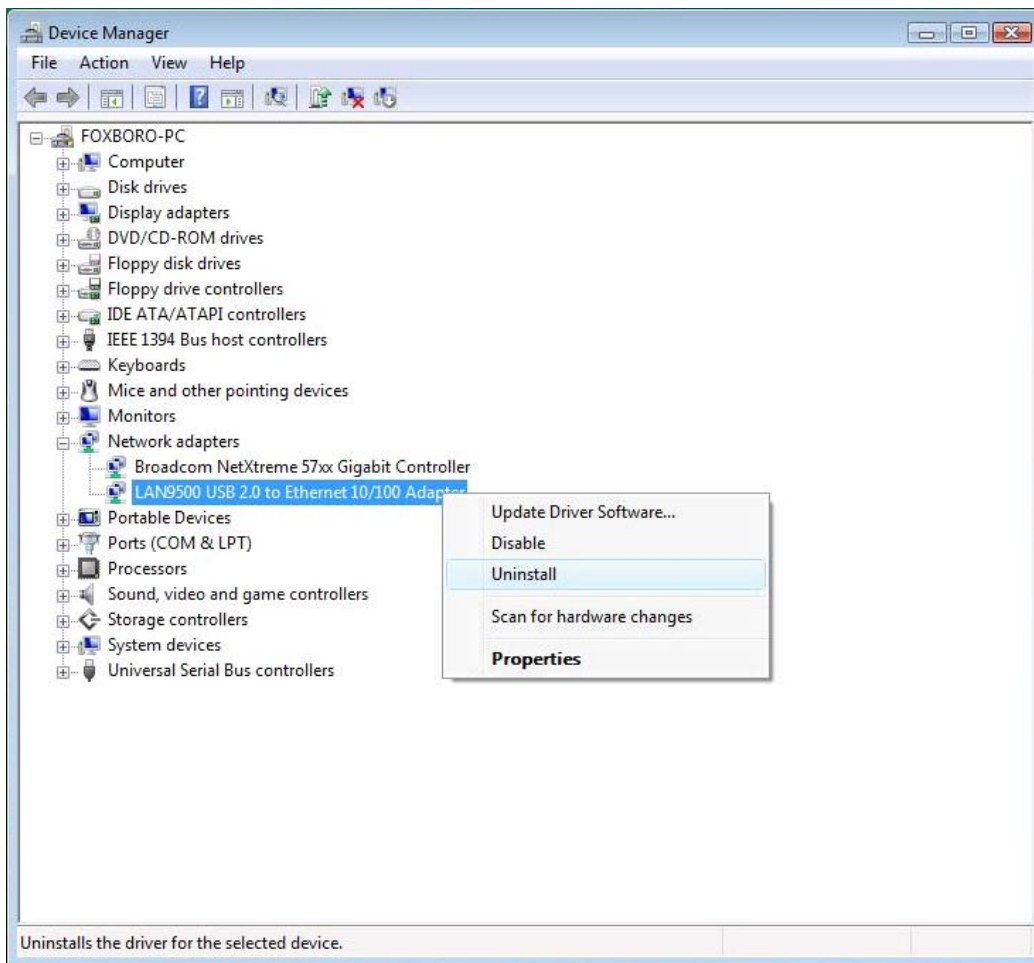


Figure 3.19 Device Manager Window

In the “Confirm Device Uninstall” window, select the “Delete the driver software for this device” checkbox and click “OK”, as shown in [Figure 3.20](#).



Figure 3.20 Confirm Device Removal Window

The Confirm Device Uninstall window will display a progress indicator while the device drivers are being removed, as shown in [Figure 3.21](#).



Figure 3.21 Device Uninstall Progress Window

The uninstallation process is now complete. As illustrated in [Figure 3.22](#), the device driver will no longer be listed in the Device Manager window. The Device Manager window may now be closed.

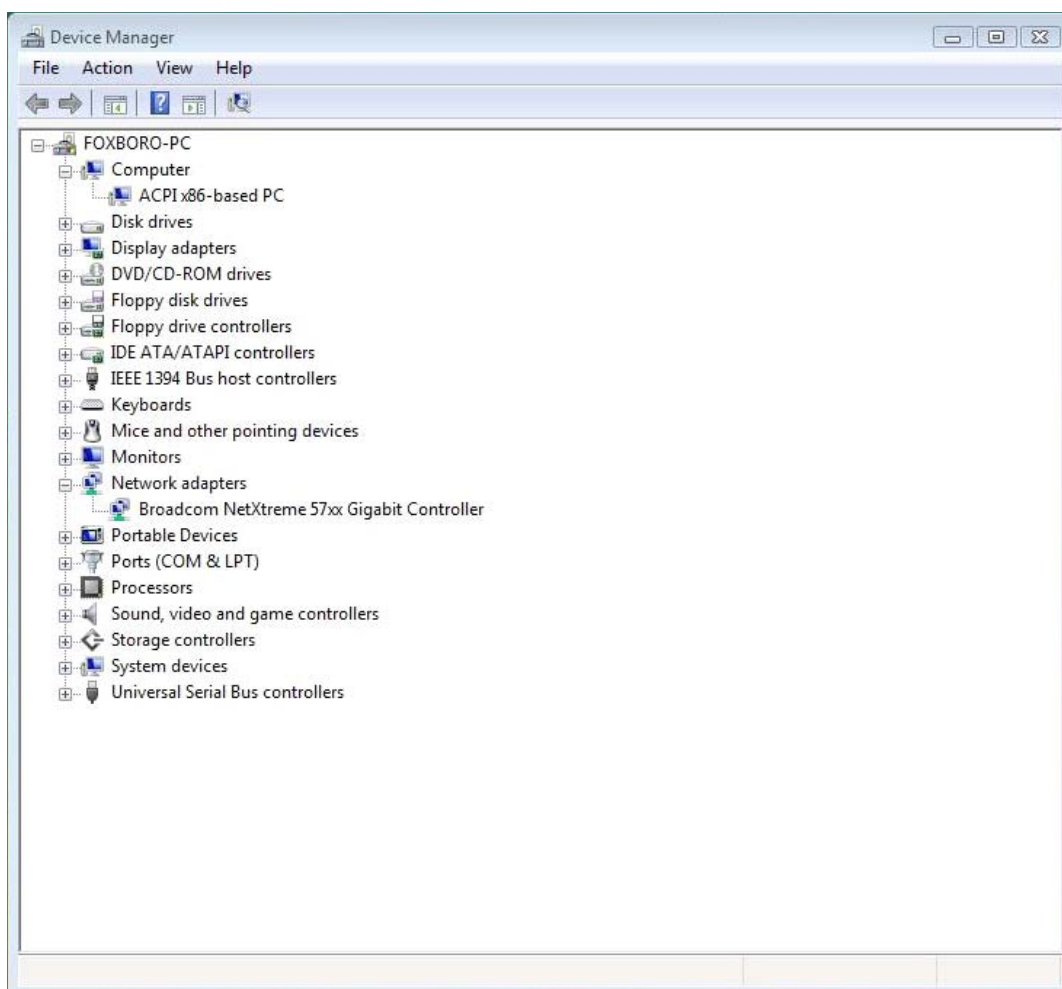


Figure 3.22 Device Removed from Device Manager

Chapter 4 MAC OSX Driver

4.1 MAC OSX Driver Installation

The folder containing the distribution files may be copied to the desktop or any other convenient, known place within the directory structure. Clicking the folder will result in the installer package file and the release notes file being displayed.

Note: The device should not be plugged into the computer prior to installing the driver software.

To begin installation, double click on the installer icon. The following introduction window will then appear:

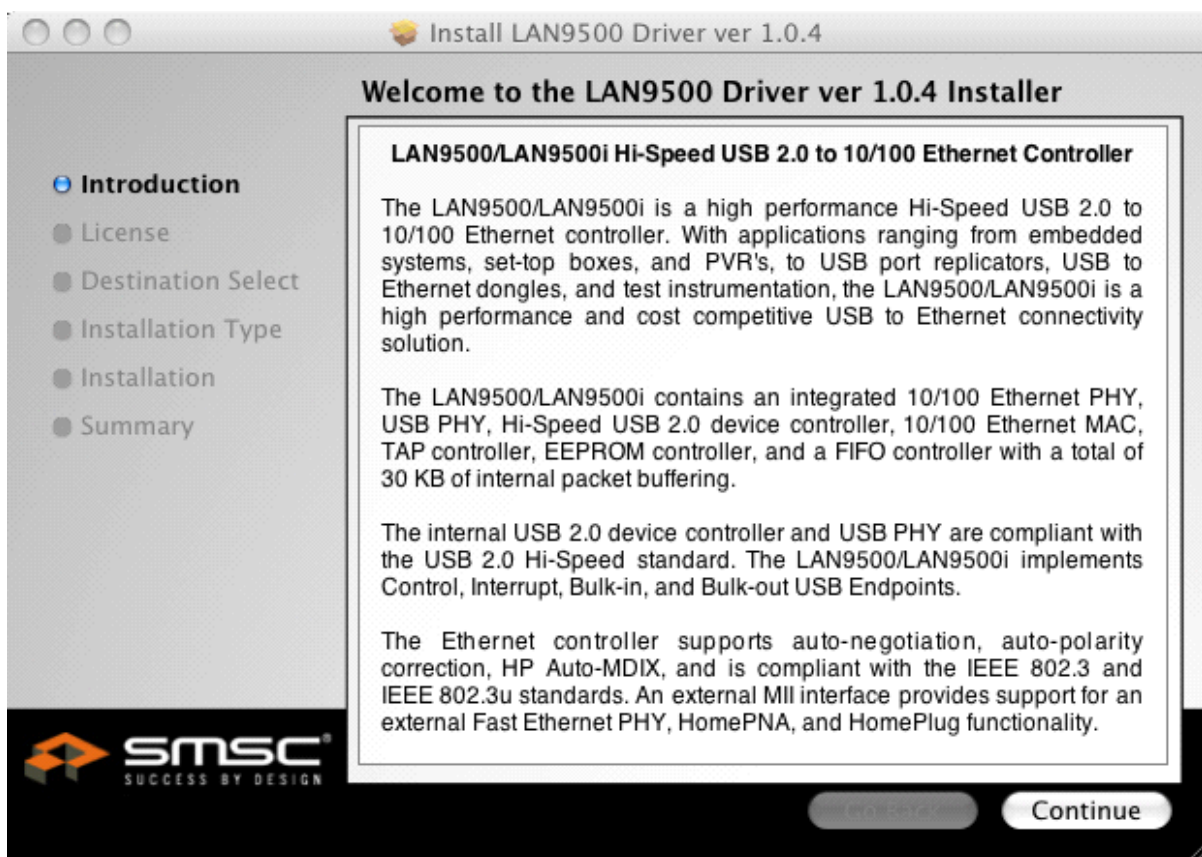


Figure 4.1 Introduction Screen

Click Continue. The “License” window, illustrated in [Figure 4.2](#), will then appear.

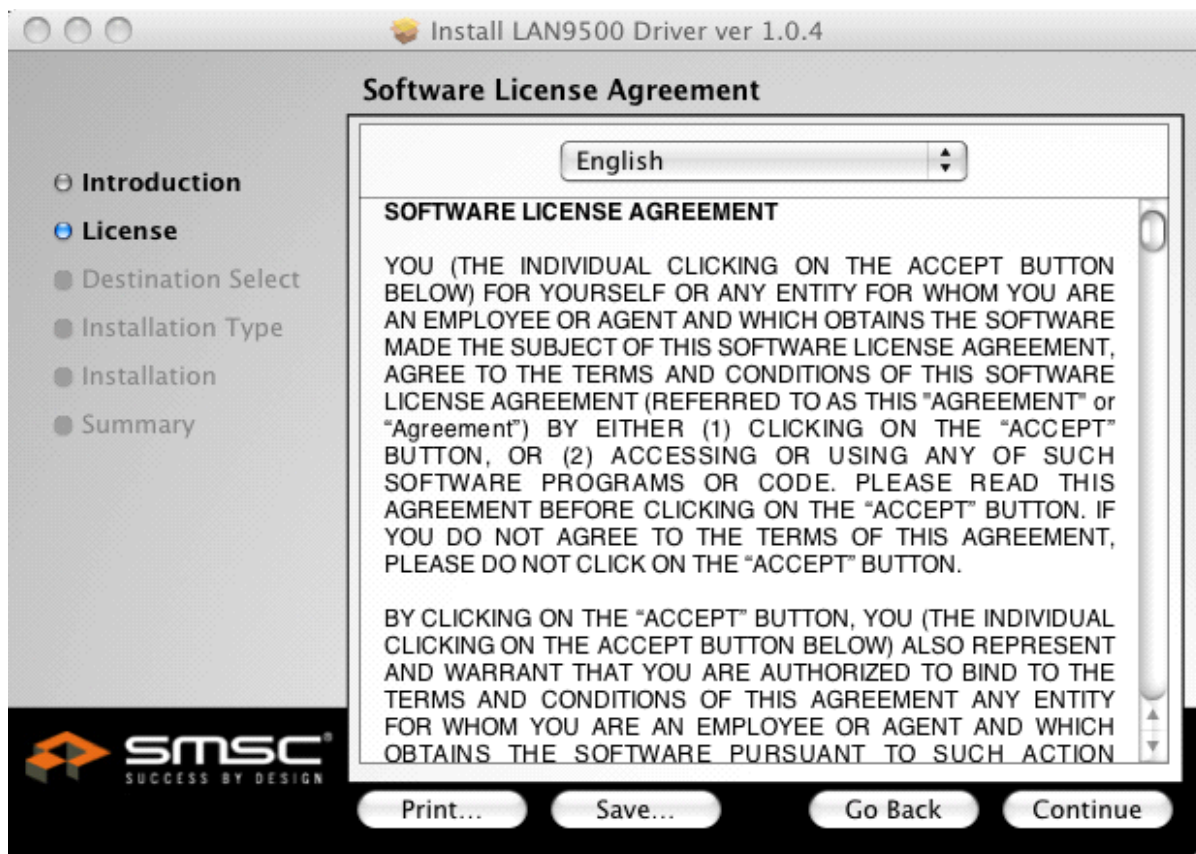


Figure 4.2 License Window

Click Continue. The drop down menu, illustrated in [Figure 4.3](#), will then appear.

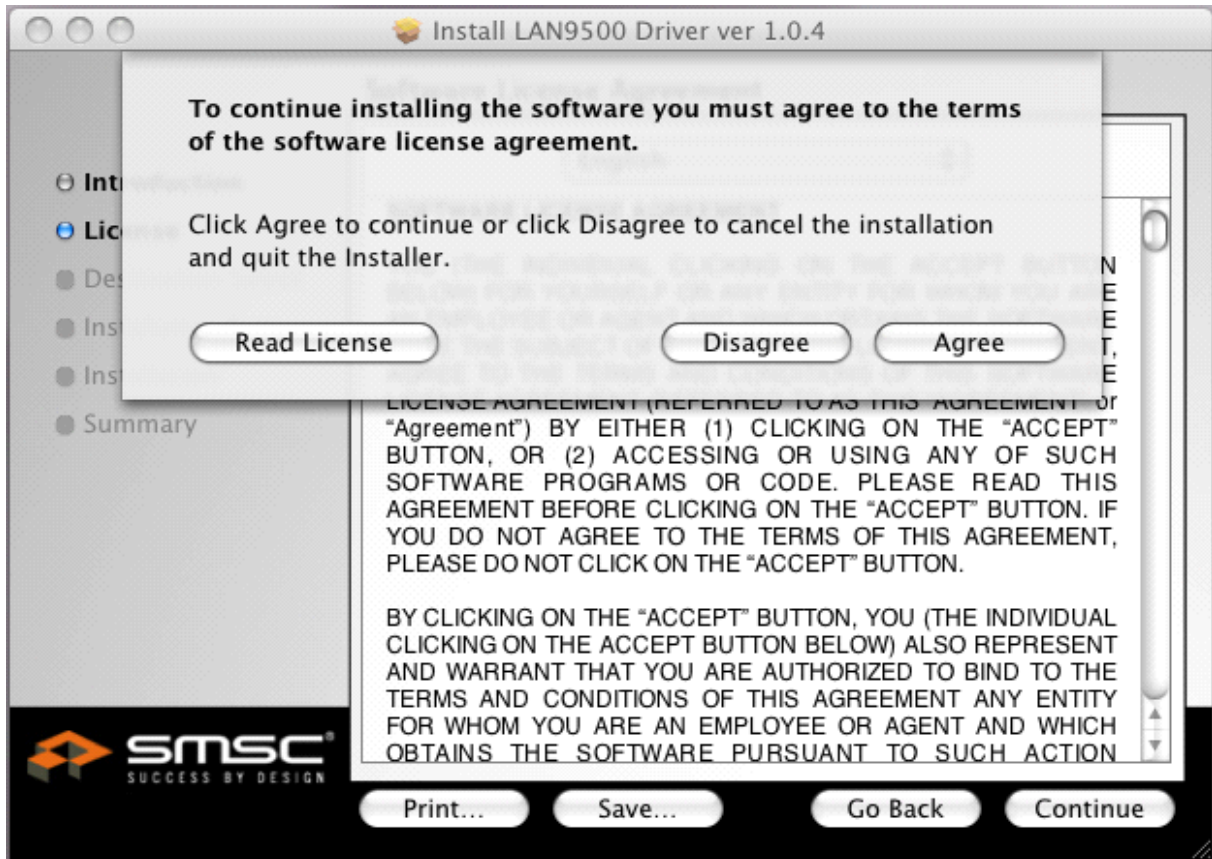


Figure 4.3 Agree/Disagree Drop Down

Click the "Agree" button. The "Installation Type" window, illustrated in [Figure 4.4](#), will then appear.

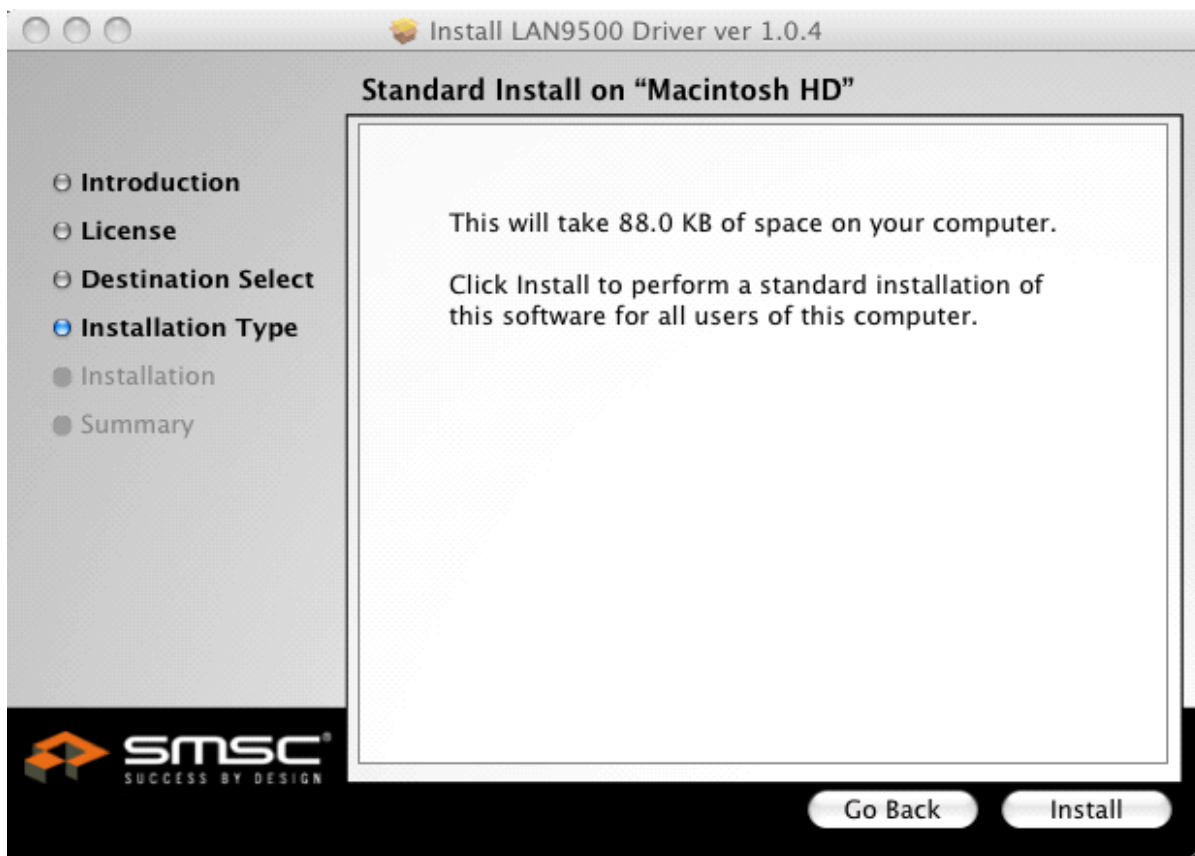


Figure 4.4 Installation Type Window

Click the “Install” button. A drop down requesting the Name and Password of a user with system administration rights will then appear, as illustrated in [Figure 4.5](#).



Figure 4.5 Installer Name/Password Drop Down

Software User Manual

Type in the appropriate Name and Password entries, then click "OK". The Summary window, indicating successful software installation, illustrated in [Figure 4.6](#), will then appear

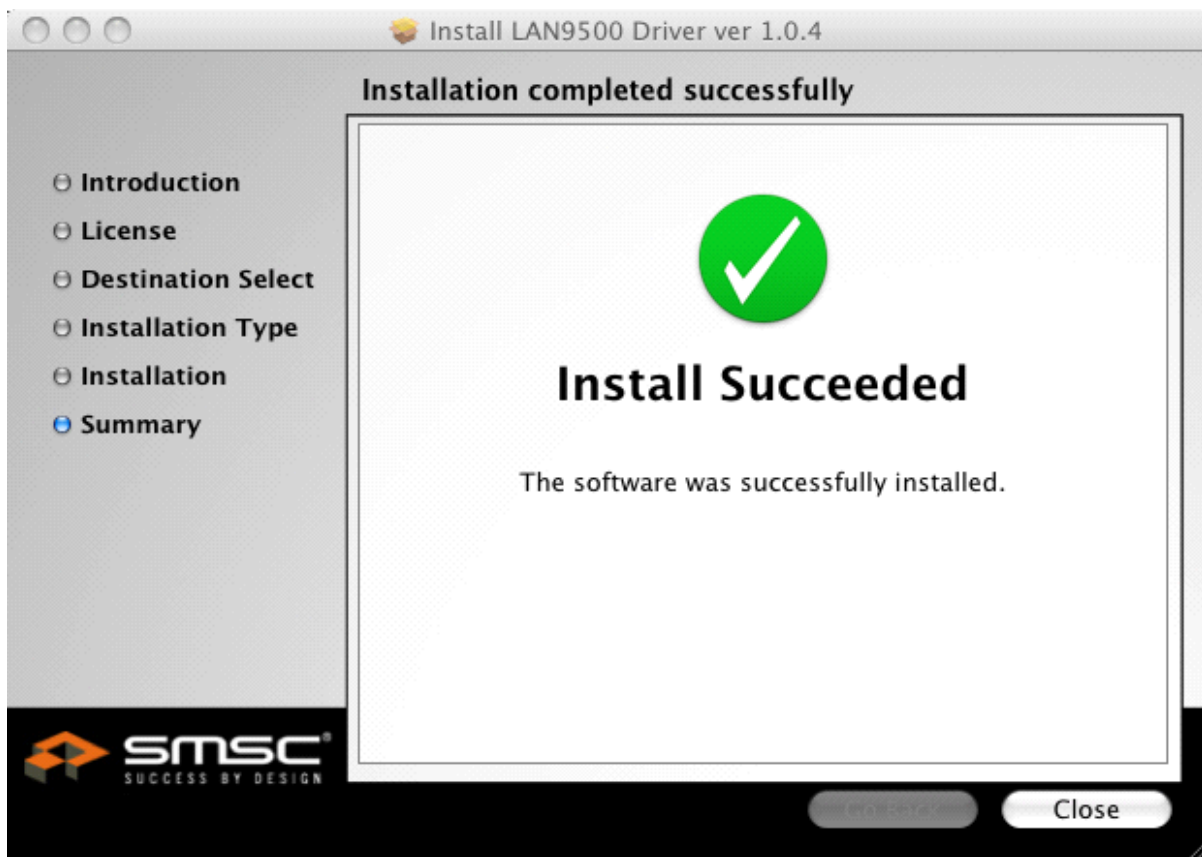


Figure 4.6 Summary Window Indicating Successful Installation

Click "Close" to complete the driver software installation.

The device must now be plugged into a free USB port on the computer. Upon device insertion, the pop up illustrated in [Figure 4.7](#) may or may not be observed. In the case where it is observed, click "Cancel" and proceed to the next step.



Figure 4.7 Cancel This Pop-Up If It Appears

Continue the configuration process by clicking the systray network icon (top right corner of the screen) and selecting “Open Network Preferences...”, as illustrated in [Figure 4.8](#).

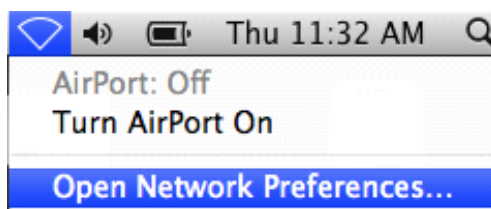


Figure 4.8 Open Network Preferences

The Network window will appear, as illustrated in [Figure 4.9](#).

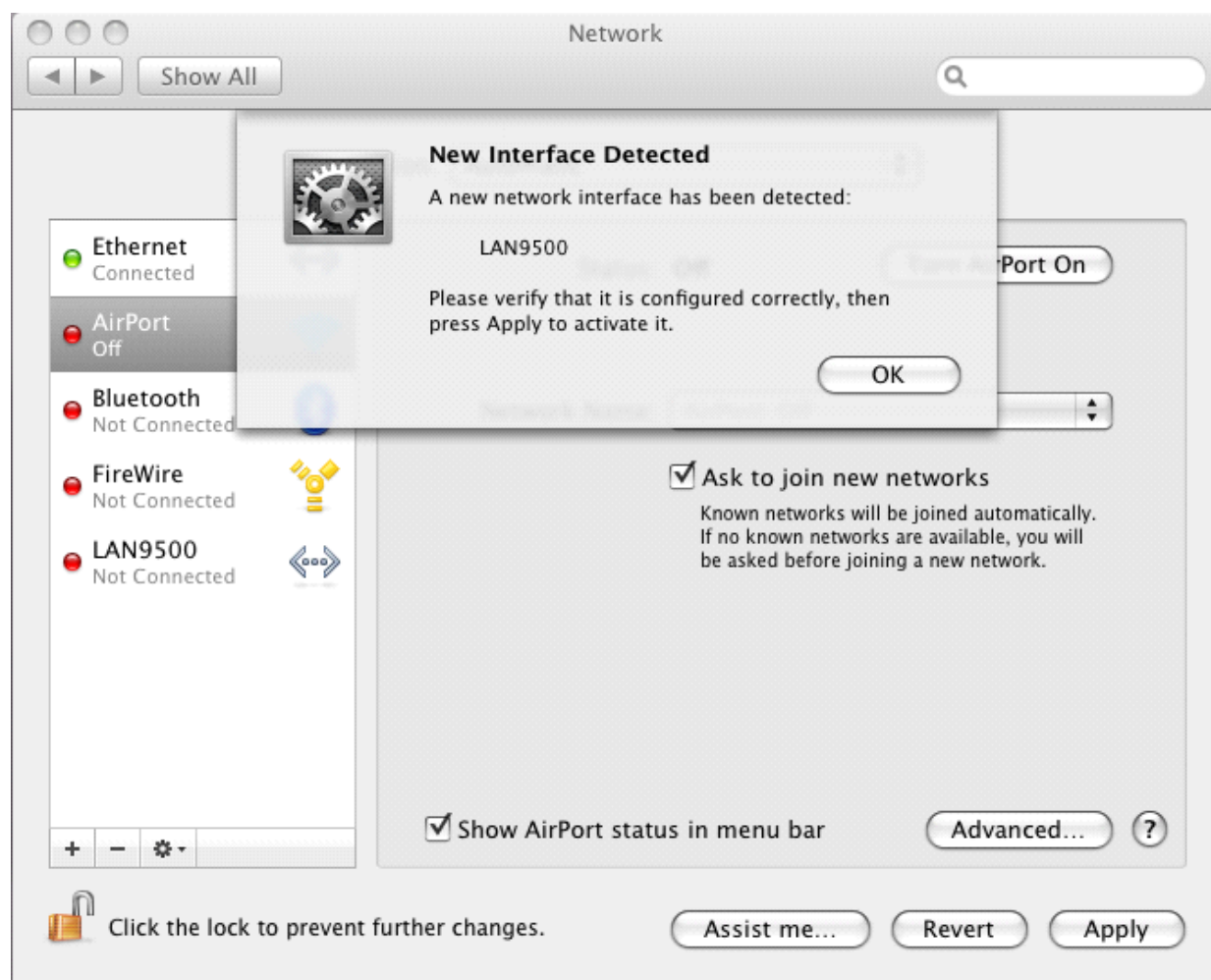


Figure 4.9 Network Window with Pop-Up

Click “OK” to dismiss the pop-up window. An entry for the device will now be added to the Network window, as illustrated in [Figure 4.10](#).

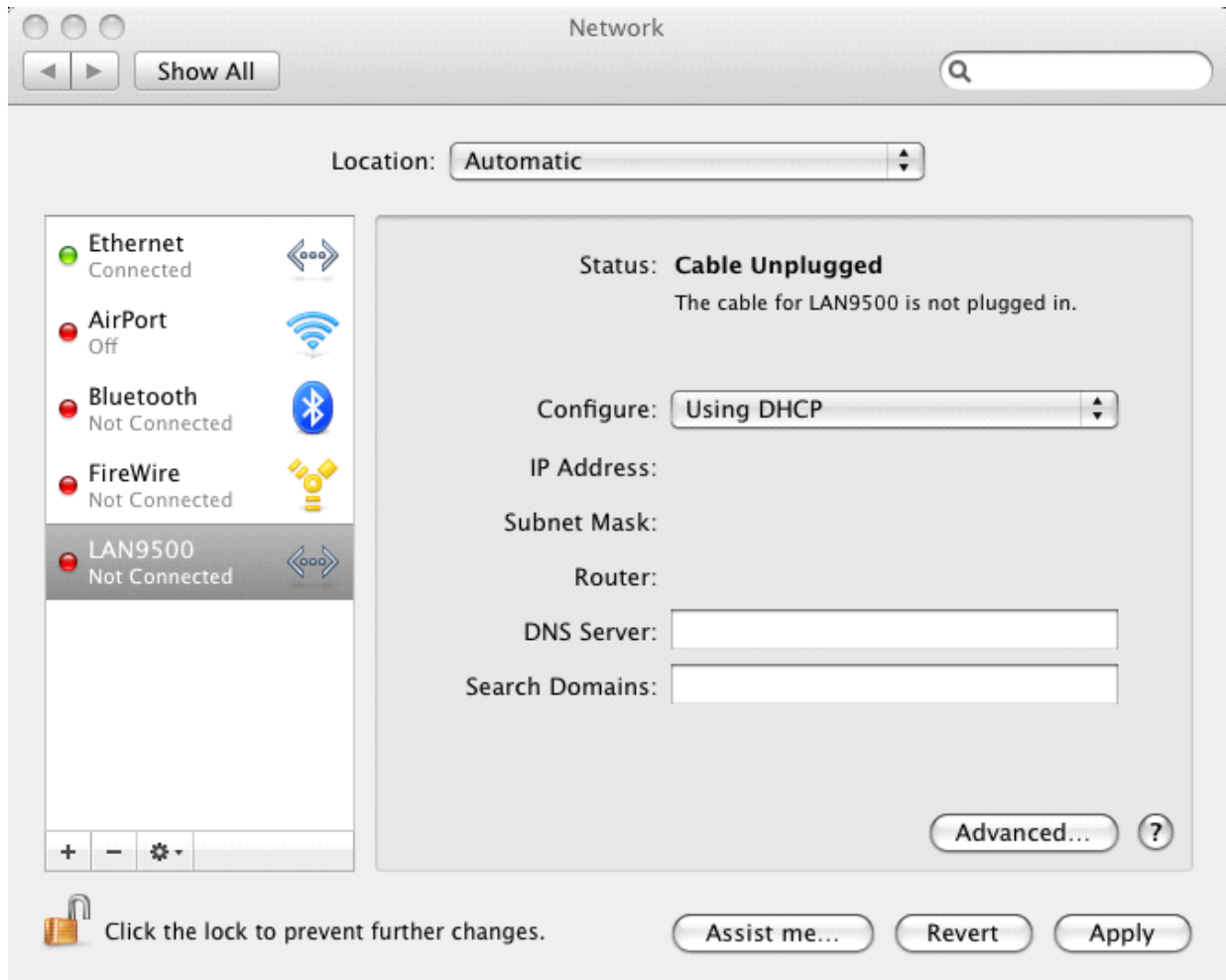


Figure 4.10 New Service Insertion in Network Table

The new service must be configured to either use DHCP or a fixed IP address, depending on the network configuration. Specification of this portion of the configuration is outside the scope of this document. Upon completion of the configuration, click the “Apply” button to complete the installation.

After obtaining an IP address from the DHCP server, the Network window will appear similar to [Figure 4.11](#).

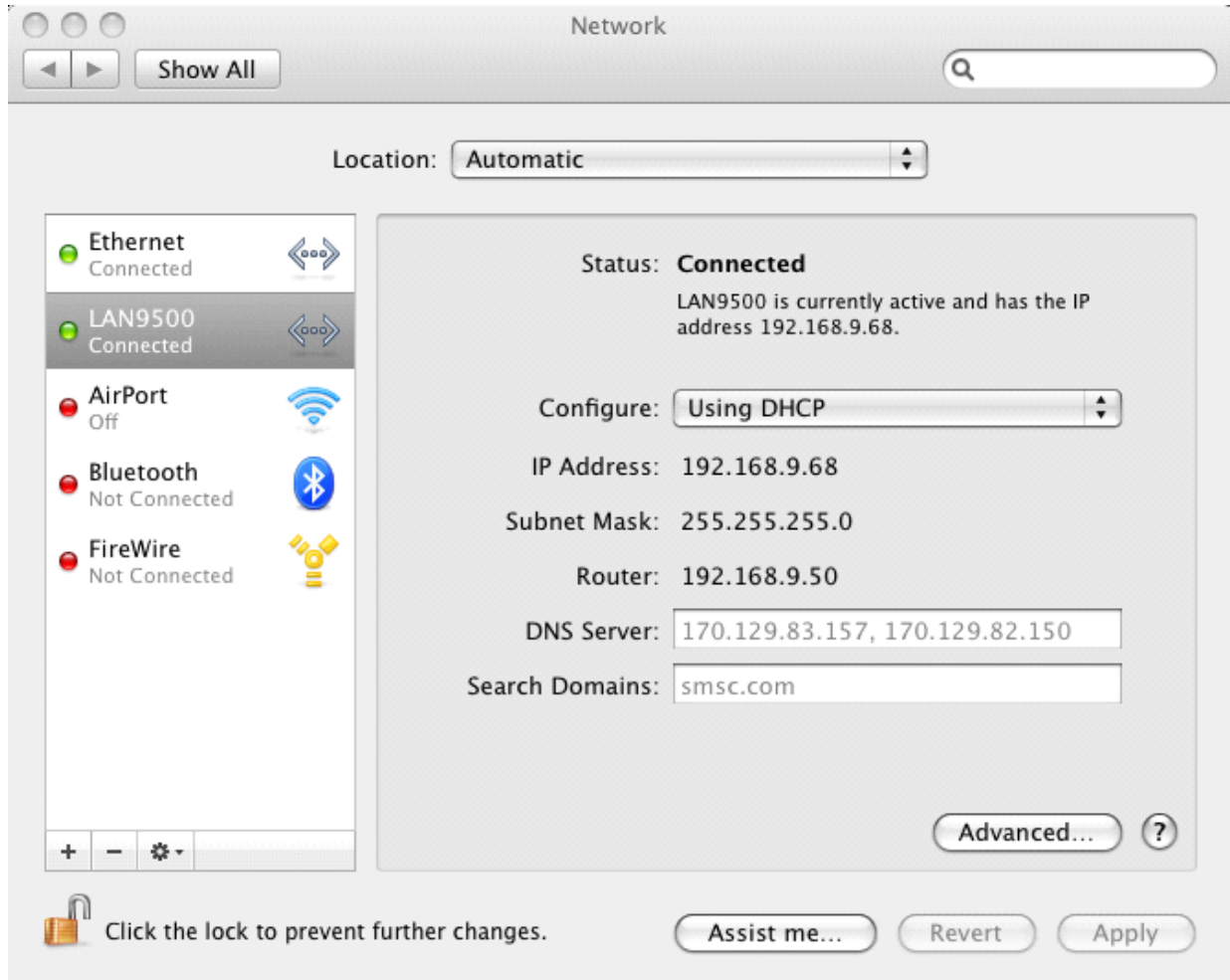


Figure 4.11 Device Connected with IP Address

4.2 MAC OSX Driver Uninstallation

Note: The device must be safely removed from the computer before beginning the uninstallation.

To begin the uninstallation process, open up a command terminal and enter the removal command “`sudo rm -rf /System/Library/Extensions/LAN9500.kext/`”, as indicated in [Figure 4.12](#). This causes the device driver to be removed from the operating system.

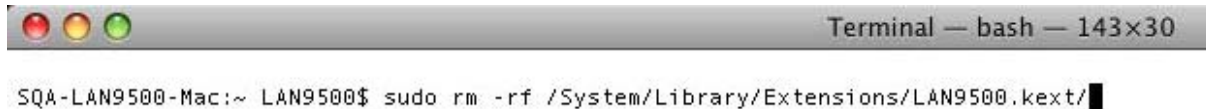


Figure 4.12 Driver Removal

The final step is to remove the network entry that was created during the installation process. Click the systray network icon (top right corner of the screen) and select “Open Network Preferences...”, as illustrated in [Figure 4.13](#).

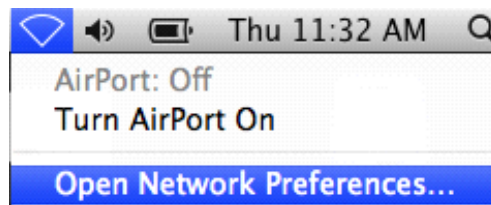


Figure 4.13 Open Network Preferences

The Network window will appear, as illustrated in [Figure 4.14](#).

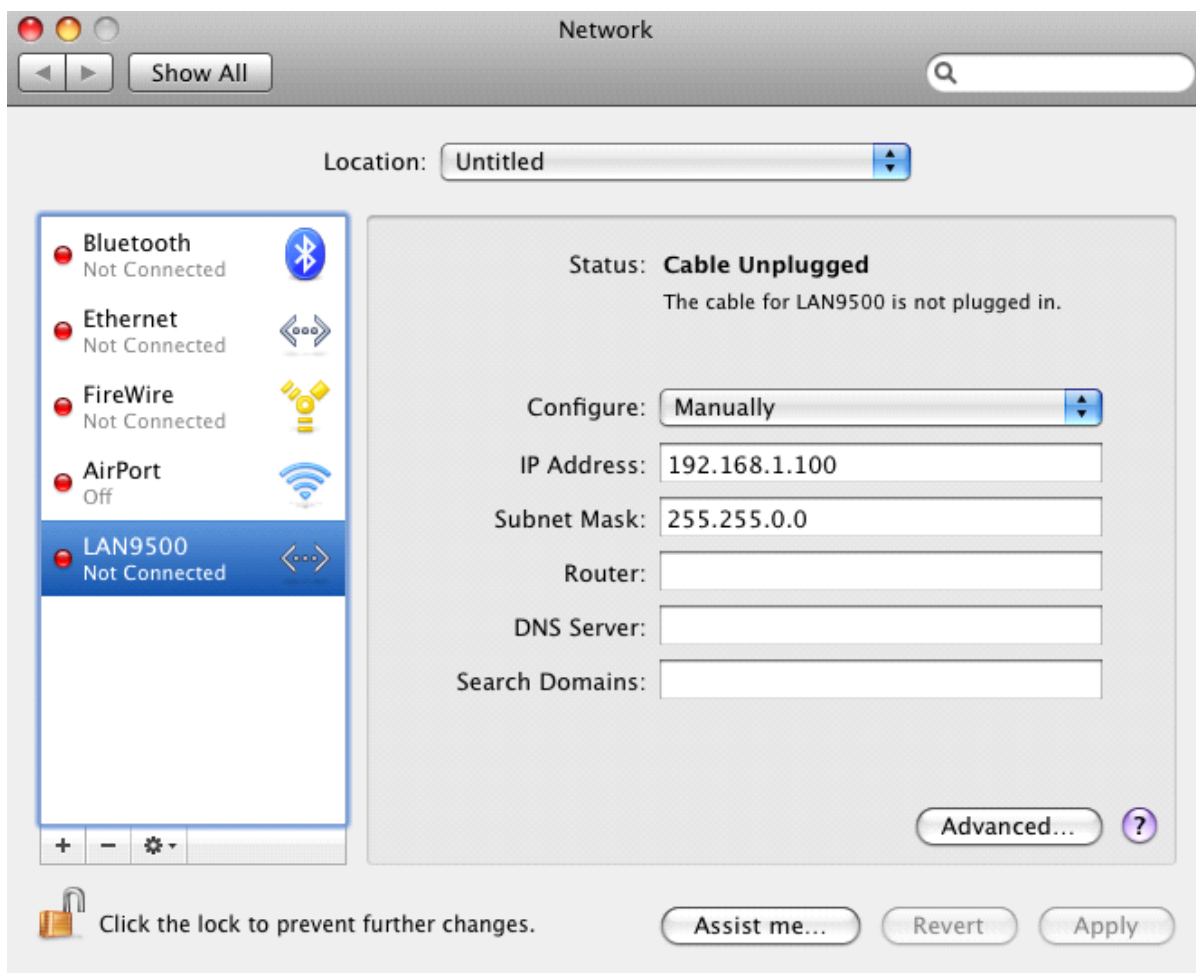


Figure 4.14 Network Preferences

Click on the LAN9500 network entry to select and highlight it. Then click the “-” button to remove the LAN9500 network entry from the operating system.

When the window is exited, a prompt to confirm and apply the changes will appear, as in [Figure 4.15](#).

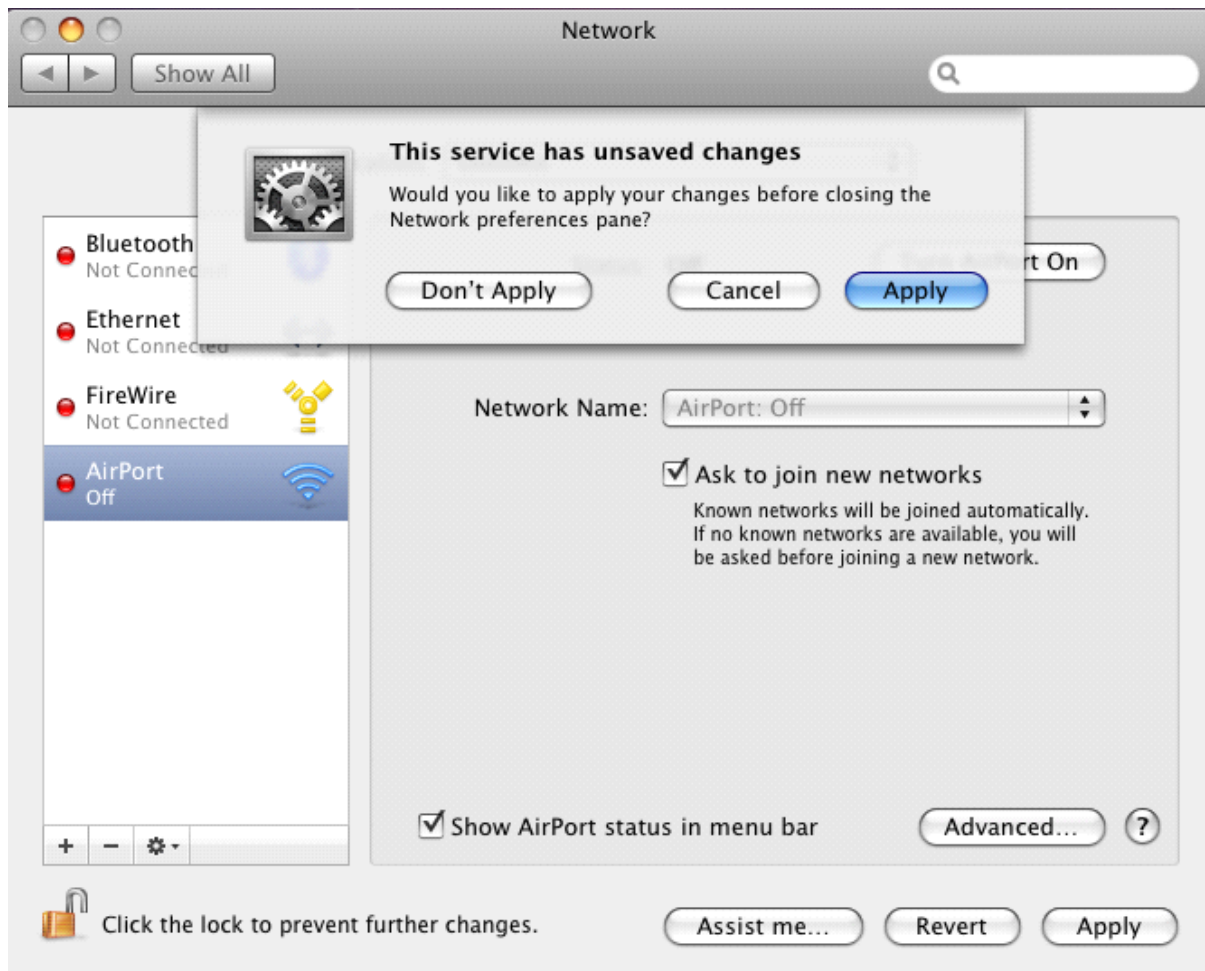


Figure 4.15 Confirm Changes

Click the "Apply" button to confirm and accept the changes.

The device uninstallation is now complete.

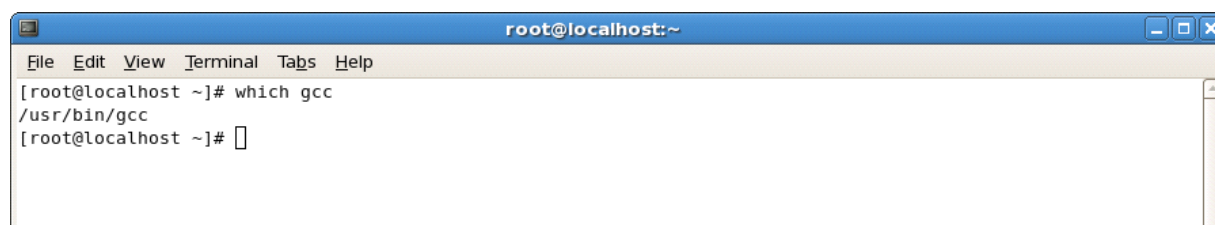
Chapter 5 Linux Driver

5.1 Linux Driver Installation

The software is distributed as a “tar” file containing all source files and release notes. The source files must be compiled and linked in order to generate a binary image for loading. The distribution file may be copied to the desktop or any other convenient, known, place within the directory structure.

Note: The device should not be plugged into the computer prior to installing the driver software.

Since the files must be compiled as part of the software build process, it is necessary to determine whether or not the compiler (gcc) is installed. To determine this, type “which gcc” at the command line. If gcc is installed, the directory path of gcc will be displayed. If not, “no gcc” will be displayed and a list of searched paths will be displayed. [Figure 5.1](#) illustrates the case where gcc is installed and its path is returned.



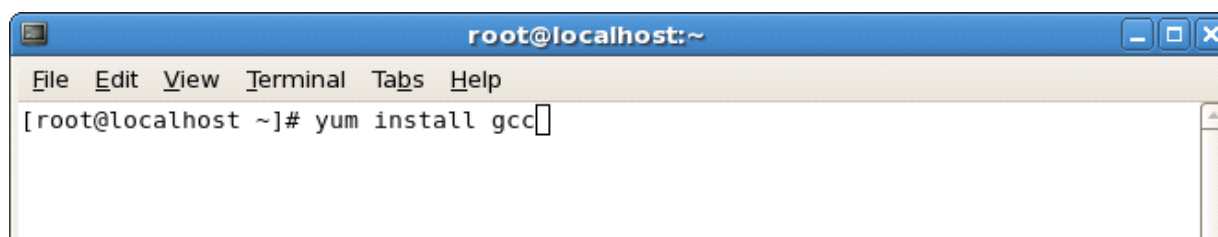
```

root@localhost:~
File Edit View Terminal Tabs Help
[root@localhost ~]# which gcc
/usr/bin/gcc
[root@localhost ~]#

```

Figure 5.1 Determining Presence of gcc

If gcc is not installed, it must be installed before the driver build process can continue. The computer must be connected to the internet and “yum install gcc” is entered on the command line to install gcc. [Figure 5.2](#) illustrates this.



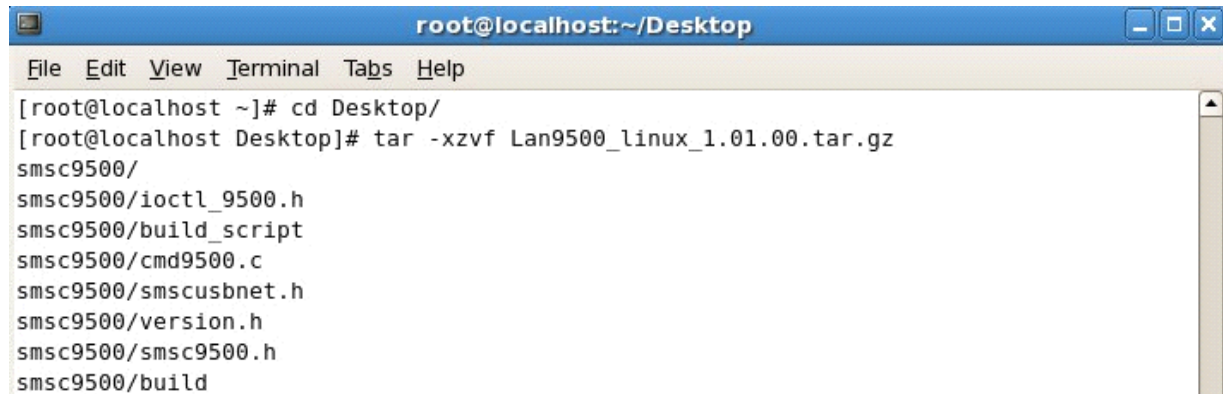
```

root@localhost:~
File Edit View Terminal Tabs Help
[root@localhost ~]# yum install gcc

```

Figure 5.2 Install gcc

When the presence of gcc has been established, the build process can proceed. The first step is to extract the source files, build files, and release notes from the “tar” distribution file. This is accomplished by changing to the directory the “tar” distribution file resides in and by typing “tar -xzf filename” on the command line, where filename is the name of the “tar” distribution file. As a result of executing the command, the directory SMSC9500 is created and 8 files are extracted from the “tar” file and inserted into it. [Figure 5.3](#) illustrates this.



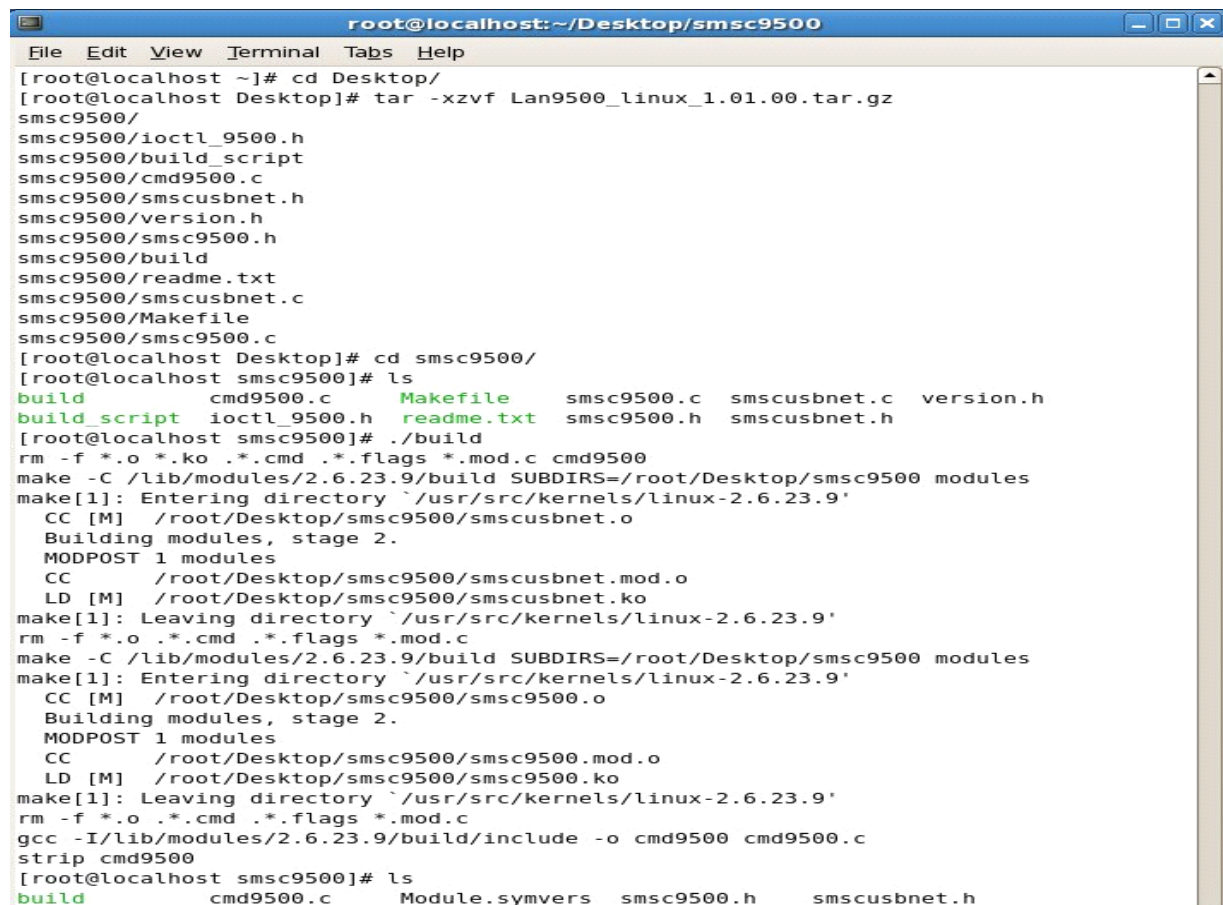
```

root@localhost:~/Desktop
File Edit View Terminal Tabs Help
[root@localhost ~]# cd Desktop/
[root@localhost Desktop]# tar -xvzf Lan9500_linux_1.01.00.tar.gz
smc9500/
smc9500/ioctl_9500.h
smc9500/build_script
smc9500/cmd9500.c
smc9500/smcusbnet.h
smc9500/version.h
smc9500/smc9500.h
smc9500/build

```

Figure 5.3 Tar of Distribution File

In order to continue the build process, change to the SMSC9500 directory by entering the `cd` command and type `./build` to invoke the build script. Figure 5.4 illustrates change of the directory and invocation of the build script. It also illustrates use of the `ls` command to list the contents of the SMSC9500 directory before and after invocation of the build script.



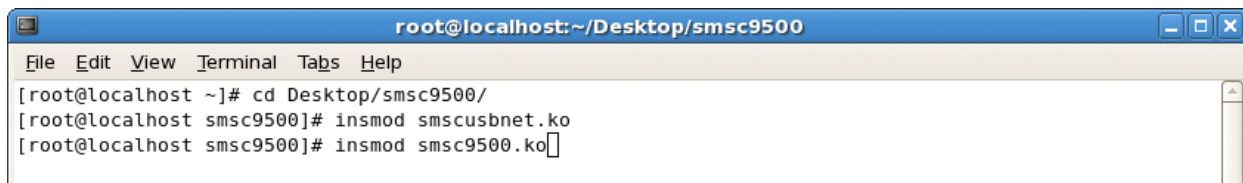
```

root@localhost:~/Desktop/smc9500
File Edit View Terminal Tabs Help
[root@localhost ~]# cd Desktop/
[root@localhost Desktop]# tar -xvzf Lan9500_linux_1.01.00.tar.gz
smc9500/
smc9500/ioctl_9500.h
smc9500/build_script
smc9500/cmd9500.c
smc9500/smcusbnet.h
smc9500/version.h
smc9500/smc9500.h
smc9500/build
smc9500/readme.txt
smc9500/smcusbnet.c
smc9500/Makefile
smc9500/smc9500.c
[root@localhost Desktop]# cd smc9500/
[root@localhost smc9500]# ls
build      cmd9500.c      Makefile      smc9500.c      smcusbnet.c      version.h
build_script  ioctl_9500.h  readme.txt    smc9500.h      smcusbnet.h
[root@localhost smc9500]# ./build
rm -f *.o *.ko *.cmd *.flags *.mod.c cmd9500
make -C /lib/modules/2.6.23.9/build SUBDIRS=/root/Desktop/smc9500 modules
make[1]: Entering directory `/usr/src/kernels/linux-2.6.23.9'
CC [M] /root/Desktop/smc9500/smcusbnet.o
Building modules, stage 2.
MODPOST 1 modules
CC /root/Desktop/smc9500/smcusbnet.mod.o
LD [M] /root/Desktop/smc9500/smcusbnet.ko
make[1]: Leaving directory `/usr/src/kernels/linux-2.6.23.9'
rm -f *.o *.cmd *.flags *.mod.c
make -C /lib/modules/2.6.23.9/build SUBDIRS=/root/Desktop/smc9500 modules
make[1]: Entering directory `/usr/src/kernels/linux-2.6.23.9'
CC [M] /root/Desktop/smc9500/smc9500.o
Building modules, stage 2.
MODPOST 1 modules
CC /root/Desktop/smc9500/smc9500.mod.o
LD [M] /root/Desktop/smc9500/smc9500.ko
make[1]: Leaving directory `/usr/src/kernels/linux-2.6.23.9'
rm -f *.o *.cmd *.flags *.mod.c
gcc -I/lib/modules/2.6.23.9/build/include -o cmd9500 cmd9500.c
strip cmd9500
[root@localhost smc9500]# ls
build      cmd9500.c      Module.symvers  smc9500.h      smcusbnet.h

```

Figure 5.4 Execution of Build Script

At the completion of the build script, two binary module, `smcusbnet.ko` and `smc9500.ko` are generated, as indicated in [Figure 5.4](#). These modules are installed via use of the “`insmod`” command, as illustrated in [Figure 5.5](#).



```

root@localhost:~/Desktop/smc9500
File Edit View Terminal Tabs Help
[root@localhost ~]# cd Desktop/smc9500/
[root@localhost smc9500]# insmod smcusbnet.ko
[root@localhost smc9500]# insmod smc9500.ko

```

Figure 5.5 Installing Driver Binaries

At this point, software installation is complete. The LAN9500/LAN9500i device must now be plugged into a free USB port on the computer, in order to be configured. After the device is plugged in, click on System->Administrator->Network on the task bar to bring up the Network Configuration window, as illustrated in [Figure 5.6](#).



Figure 5.6 Network Configuration Window

Software User Manual

Click “New”, to add a new device type. [Figure 5.7](#) illustrates the resulting window.



Figure 5.7 Add New Ethernet Device Type

Select “Ethernet connection” device type and click the “Forward” button. The window will now appear as illustrated in [Figure 5.8](#).

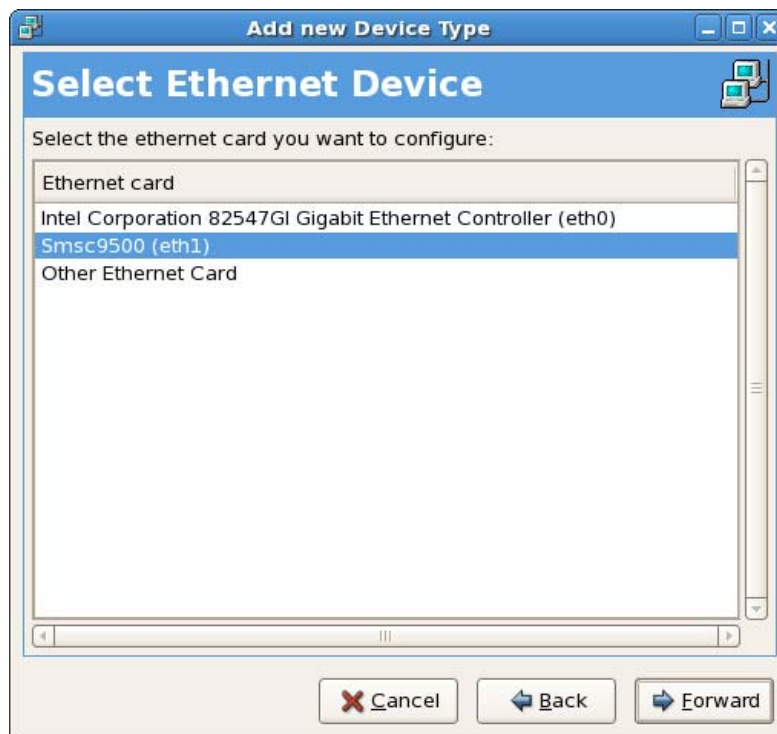


Figure 5.8 Device Selection

Select “Smsc9500” and click the “Forward” button. The window will now appear as illustrated in [Figure 5.9](#).

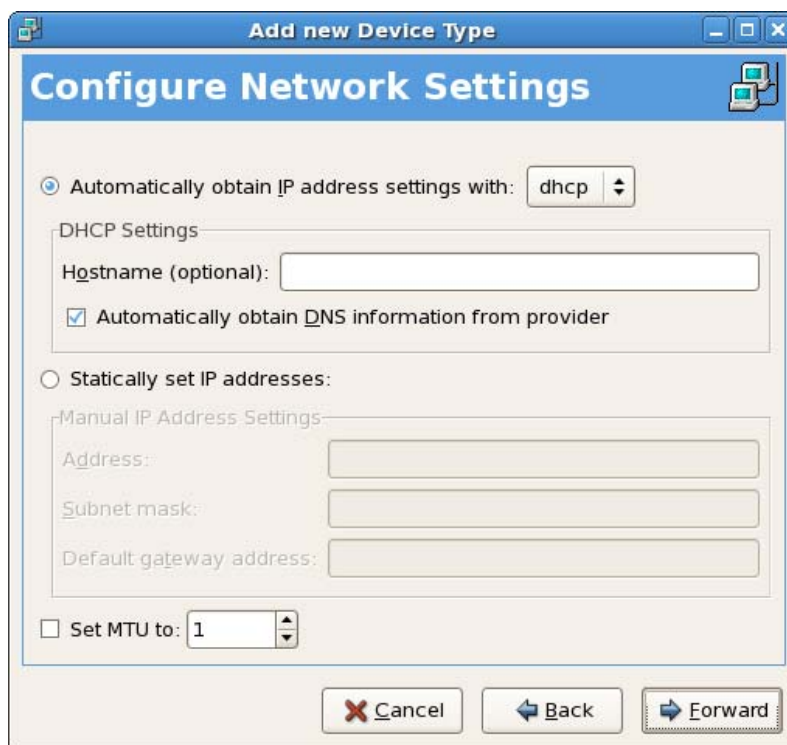


Figure 5.9 IP Address Generation

The new device must be configured to either use DHCP or a fixed IP address, depending on the network configuration. Specification of this portion of the configuration is outside the scope of this document. For illustration purposes, automatic IP address generation is selected in [Figure 5.9](#). Upon completion of the configuration, click the “Forward” button. The window will now appear as illustrated in [Figure 5.10](#).

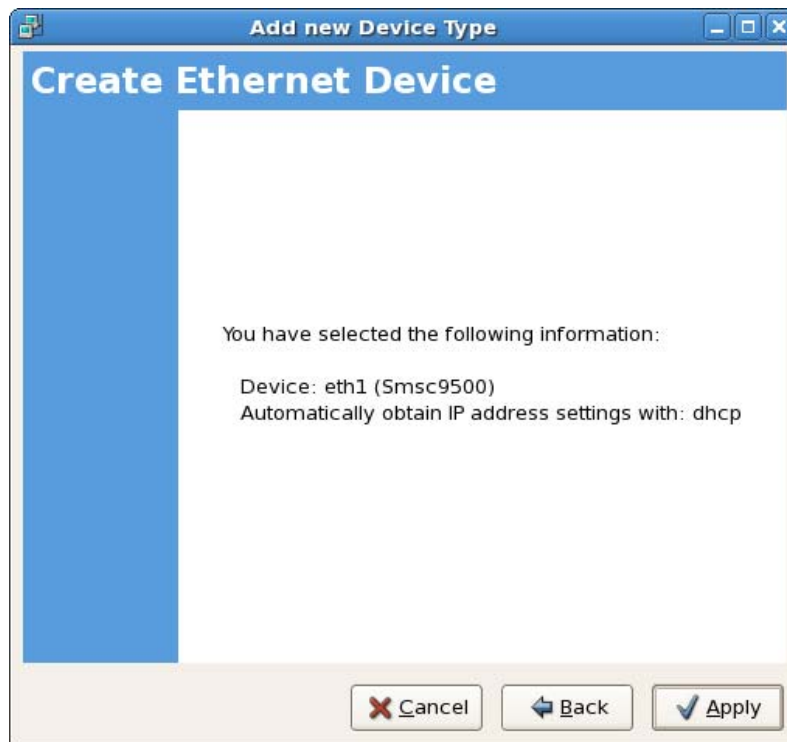


Figure 5.10 Device Creation

Click the "Apply" button to create the device. The Network Configuration screen, illustrated in [Figure 5.6 on page 46](#), will again appear. Select the newly created device (in this case eth1), and click the "Activate" button to activate the ethernet device. Click "File->Save" to retain the network configuration information and complete the installation.

5.2 Linux Driver Uninstallation

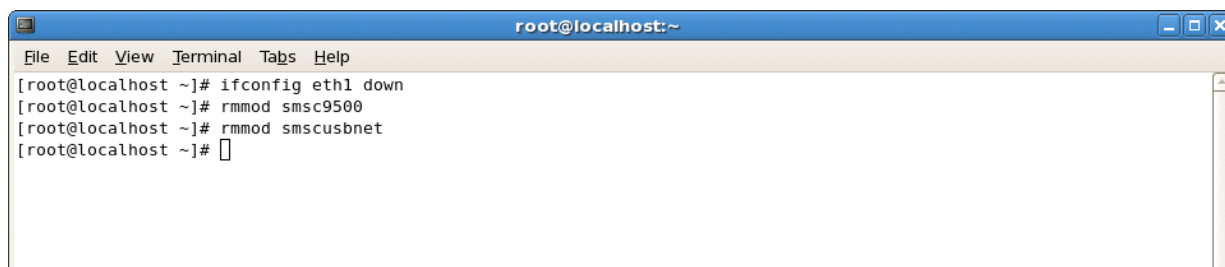
Note: The device must be safely removed from the computer before beginning the uninstallation.

Open a command terminal and bring the LAN9500 network interface down by typing “ifconfig <interface> down”, where <interface> is the interface associated with the LAN9500.

Remove the LAN9500 ethernet module by typing “rmmod smsc9500”

Remove the LAN9500 USB module by typing “rmmod smscusbnet”

These steps are illustrated in [Figure 5.11](#).



```
root@localhost:~  
File Edit View Terminal Tabs Help  
[root@localhost ~]# ifconfig eth1 down  
[root@localhost ~]# rmmod smsc9500  
[root@localhost ~]# rmmod smscusbnet  
[root@localhost ~]#
```

Figure 5.11 Uninstallation Steps

The device uninstallation is now complete.

Chapter 6 Windows CE Driver

This chapter details the methods for building a LAN9500/LAN9500i driver into a Windows CE image. This may be accomplished in two ways:

- Building a CE Image with LAN9500/LAN9500i Source Files
- Building a CE Image with LAN9500/LAN9500i DLL & REG Files

Note: Please ensure all current Microsoft Quick Fix Engineering (QFE) patches for Platform Builder and Windows Embedded CE have been applied before working with the device driver. The latest QFE patches can be downloaded from:
<http://msdn.microsoft.com/en-us/embedded/aa731256.aspx>

6.1 Building a CE Image with LAN9500/LAN9500i Source Files

A Windows CE image containing the device driver may be built directly from the device driver source files. All device driver source files are compressed under the directory "SMSC9500". A list of the device driver source files is provided in [Table 6.1](#).

Table 6.1 Windows CE LAN9500/LAN9500i Device Driver Source Files

FILE NAME	DESCRIPTION
init.c	This module contains initialization helper routines called during miniport initialization
interrupt.c	This module contains miniport functions handling interrupt transactions and other helper routines called by these miniport functions.
lan9500.h	This module contains LAN9500/LAN9500i register and structure definitions.
receive.c	This module contains miniport functions for handling receive packets and other help routines called by these miniport functions.
request.c	This module contains miniport functions for handling Set & Query Information requests.
send.c	This module contains miniport functions for handling Send packets and other helper routines call by the miniport function.
sm9500.c	USB ethernet adapter control/bulk/interrupt transport.
sm9500.h	This module contains structure definitions and function prototypes.
sm9500.reg	This module contains the LAN9500/LAN9500i registry keys.
OS.c, os.h	OS related files
version.h	This module contains version information.
Ioctl.c	This module contains all the ioctl functions.
Ioctl.h	This module contains the Ioctl.c structure definitions and function prototypes.

Note: The latest source code may be downloaded from the E-Services portion of the SMSC website. Refer to <https://www2.smc.com/main.nsf> for additional information.

To build the LAN9500/LAN9500i device driver from the source files into a CE image, the following procedure must be followed:

1. Most BSPs use the "DRIVERS" directory as the root folder for all device drivers. Unzip the LAN9500/LAN9500i driver to the "DRIVERS" folder, for example, %_TARGETPLATROOT%\src\drivers. The folder name "smmc9500" will then be listed under the "DRIVERS" directory.

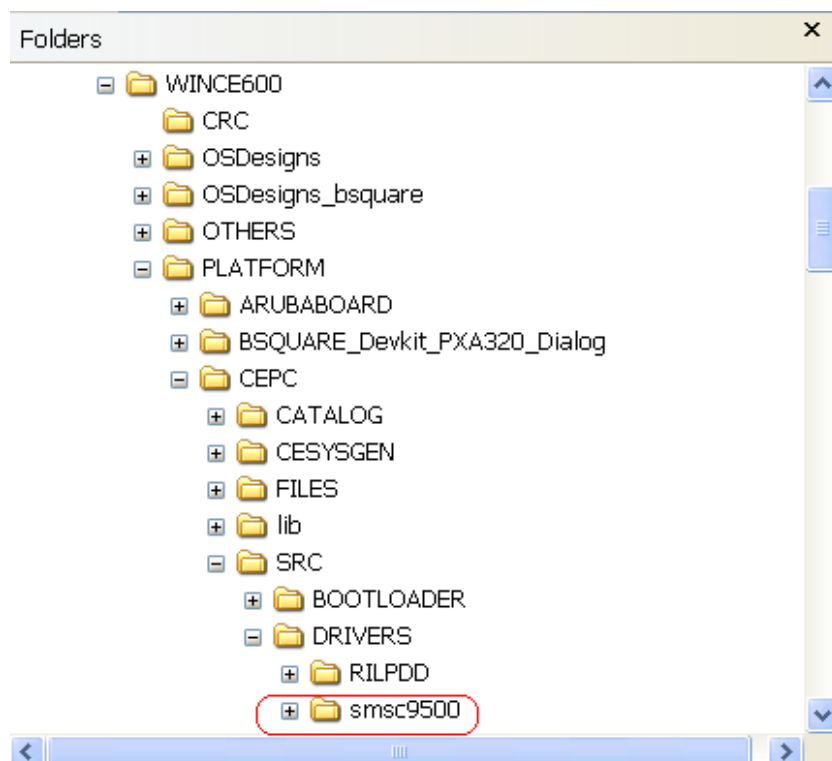


Figure 6.1 Example Unzip to DRIVERS Directory

Name	Size	Type	Date Modified
init.c	142 KB	C Source file	3/25/2009 4:51 ...
interrupt.c	48 KB	C Source file	3/25/2009 3:49 ...
Ioctl.c	7 KB	C Source file	3/30/2009 5:37 ...
Ioctl.h	3 KB	C Header file	3/30/2009 5:37 ...
lan9500.h	33 KB	C Header file	3/25/2009 3:49 ...
MAKEFILE	1 KB	File	5/13/2004 10:47...
OS.c	1 KB	C Source file	3/26/2009 1:31 ...
os.h	8 KB	C Header file	12/11/2008 3:32...
receive.c	85 KB	C Source file	3/25/2009 4:51 ...
request.c	84 KB	C Source file	3/26/2009 11:48...
send.c	63 KB	C Source file	3/25/2009 4:15 ...
smmc9500.c	74 KB	C Source file	3/31/2009 10:21...
smmc9500.def	1 KB	DEF File	12/11/2008 11:3...
smmc9500.h	28 KB	C Header file	3/25/2009 3:49 ...
smmc9500.reg	5 KB	Registration Entr...	3/26/2009 3:30 ...
sources	2 KB	File	2/17/2009 2:59 ...
version.h	2 KB	C Header file	3/31/2009 10:18...
LAN9500 Driver for CE 5.0&6.0 User Manual r.1.10.pdf	139 KB	Adobe Acrobat D...	3/30/2009 6:19 ...

Figure 6.2 smmc9500 Folder Contents

Software User Manual

- The DIRS file, under the %_TARGETPLATROOT%\src\drivers folder, has a list of drivers to be compiled during the image build procedure. The smsc9500 directory must be added to this list to allow Platform Builder to build the driver automatically.

```

!if 0
Copyright (c) Microsoft Corporation. All rights reserved.
!endif
!if 0
Use of this sample source code is subject to the terms of the Microsoft
license agreement under which you licensed this sample source code. If
you did not accept the terms of the license agreement, you are not
authorized to use this sample source code. For the terms of the license,
please see the license agreement between you and Microsoft or, if applicable,
see the LICENSE.RTF on your install media or the root of your tools installation.
THE SAMPLE SOURCE CODE IS PROVIDED "AS IS", WITH NO WARRANTIES.
!endif

DIRS= \
# @CESYSGEN IF CELLCORE_MODULES_RIL
    rilpdd \
# @CESYSGEN ENDIF CELLCORE_MODULES_RIL
#
# bug - excluding kbdmouse because it presently relies on toolhelp.lib
#       which will not work because kbdmouse.dll loads in the kernel
#       and toolhelp.dll cannot load in the kernel
#
#       kbdmouse \
#       smc9500

```

Figure 6.3 DIRS List of Drivers to be Built

- The driver file must then be included into the target image. The smsc9500.dll file must be added to Platform.bib under the %_TARGETPLATROOT%\FILES folder. This file lists the platform-related files to be included in the final CE image.

```

ENDIF IMGFAKERIL
; @CESYSGEN ENDIF CELLCORE_MODULES_RIL

smc9500.dll                ${_FLATRELEASEDIR}\smc9500.dll      NK      SHK

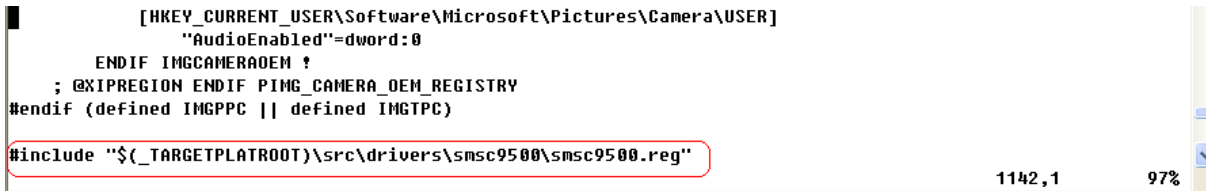
FILES

    cetk.lnk                ${_FLATRELEASEDIR}\cetk.lnk          NK
    clientside.exe          ${_FLATRELEASEDIR}\clientside.exe    NK
; Name                      Path                                Memory Type

```

Figure 6.4 Adding smsc9500.dll into platform.bib

4. The `smc9500` folder contains the LAN9500/LAN9500i registry keys file `smc9500.reg`. This file must be included into `%_TARGETPLATROOT%\FILES\Platform.reg` or alternatively, the contents of the `smc9500.reg` file copied into `%_TARGETPLATROOT%\FILES\Platform.reg`.



```

[HKEY_CURRENT_USER\Software\Microsoft\Pictures\Camera\USER]
    "AudioEnabled"=dword:0
ENDIF IMGCAMERADEM ?
; @XIPREGION ENDIF PIMG_CAMERA_OEM_REGISTRY
#endif (defined IMGPPC || defined IMGTPC)

#include "$(_TARGETPLATROOT)\src\drivers\smc9500\smc9500.reg"

```

1142,1 97%

Figure 6.5 Including smc9500.reg in platform.reg

5. The CE image may now be built. This procedure varies, dependant on the CE version being used.

For CE 5.0, Select "Build OS" -> "Sysgen" from Platform Builder 5.0. This will generate the CE Image, `NK.BIN`, which will include the device driver.

For CE 6.0, Platform Builder is a plug-in under Visual Studio 2005. Select "Build" -> "Build Solution". This will generate the CE Image, `NK.BIN`, which will include the LAN9500/LAN9500i driver.

6.2 Building a CE Image with LAN9500/LAN9500i DLL & REG Files

To build the LAN9500/LAN9500i device driver from the `smc9500.dll` and `smc9500.reg` files into a CE image, the following procedure must be followed:

1. Copy the `smc9500.dll` and `smc9500.reg` files into the `%_TARGETPLATROOT%\FILES` folder.
2. Include `smc9500.reg` into `%_TARGETPLATROOT%\FILES\Platform.reg` or alternatively, the contents of the `smc9500.reg` file can be copied into `%_TARGETPLATROOT%\FILES\Platform.reg`.



```

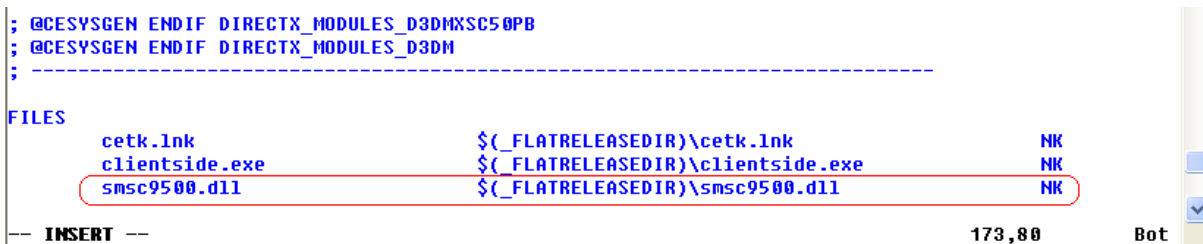
; @CESYSGEN ENDIF DIRECTX_MODULES_D3DMXSC50PB
; @CESYSGEN ENDIF DIRECTX_MODULES_D3DM
;
-----
#include "$(_TARGETPLATROOT)\files\smc9500.reg"

```

-- INSERT -- 240,49 99%

Figure 6.6 Including `smc9500.reg` in `platform.reg`

3. Add `smc9500.dll` to the `%_TARGETPLATROOT%\files\platform.bib` file



```

; @CESYSGEN ENDIF DIRECTX_MODULES_D3DMXSC50PB
; @CESYSGEN ENDIF DIRECTX_MODULES_D3DM
;
-----
FILES
  cetk.lnk                $(_FLATRELEASEDIR)\cetk.lnk                NK
  clientside.exe          $(_FLATRELEASEDIR)\clientside.exe        NK
  smc9500.dll             $(_FLATRELEASEDIR)\smc9500.dll           NK

```

-- INSERT -- 173,80 Bot

Figure 6.7 Adding `smc9500.dll` into `platform.bib`

4. The CE image may now be built. This procedure varies, dependant on the CE version being used.

For CE 5.0, Select "Build OS" -> "Sysgen" from Platform Builder 5.0. This will generate the CE Image, `NK.BIN`, which will include the LAN9500/LAN9500i driver.

For CE 6.0, Platform Builder is a plug-in under Visual Studio 2005. Select "Build" -> "Build Solution". This will generate the CE Image, `NK.BIN`, which will include the LAN9500/LAN9500i driver.

6.3 USB Driver Update for WinCE 5.0 & 6.0

As confirmed by Microsoft, the default USB HCD driver contains a bug. It returns `ERROR_SUCCESS` instead of an `ERROR` code when the `IssueBulkTransfer()` fails. This bug needs to be fixed from Microsoft. Since the device utilizes the USB interface, a USB driver patch has been devised to provide a work around.

To bypass this bug, modify the `IssueBulkTransfer()` function in the `usbclient.c` file located in the `$( _WINCEROOT ) \PUBLIC \COMMON \OAK \DRIVERS \USB \CLASS \COMMON` subdirectory to add the following three bold red lines. Once added, rebuild the image (perform a "Build and Sysgen").

```
} else {  
    //  
    // Synch call completed.  
    // Get transfer status & number of bytes transferred  
    //  
    // ASSERT( pUsbFuncs->lpIsTransferComplete(hTransfer) );  
  
    GetTransferStatus(pUsbFuncs, hTransfer, pBytesTransferred,  
        pUsbRc);  
}  
  
CloseTransferHandle(pUsbFuncs, hTransfer);  
  
} else {  
    dwErr = GetLastError();  
    if (ERROR_SUCCESS == dwErr) {  
        dwErr = ERROR_GEN_FAILURE;  
    }  
  
    DEBUGMSG( ZONE_USBCLIENT, (TEXT("*** IssueBulkTransfer ERROR(3, %d) ***\n"),  
        dwErr ));  
}  
  
} else {  
    dwErr = ERROR_INVALID_PARAMETER;  
}
```

6.4 EHCI/OHCI Driver Physical Memory

By default, the device driver can submit 4 Bulk Out transfers, 4 Bulk In transfers (16k for each high speed USB and 8k for each full speed USB) and 4 interrupt transfers pending. The customer can increase the value of physical memory for the EHCI/OHCI by adding the registry key (PhysicalPageSize) and setting it to another value, allowing support for multiple USB devices, each with several transfers pending. The default value is 128k.

Chapter 7 Driver Customization

The GUI based Windows (XP/Vista) and Mac OS X automated driver installation packages include SMSC product names and artwork that SMSC customers may desire to replace with their own. To facilitate this, these installation packages provide various customization options which are detailed in this chapter.

7.1 Windows Driver Customization and Localization

The Windows driver package is a self extracting executable created with Winzip Self Extractor. When decompressed, the file/folder structure shown in [Table 7.1](#) will be created.

Table 7.1 Automated Installer Package Contents

FILE NAME	DESCRIPTION
\Readme.txt	Contains the latest release information
\install.exe	Executable that checks the platform and launches the appropriate DpInst executable for the architecture (see below)
\x86\DpInstx86.exe	Microsoft redistributable driver package installer utility for the x86 architecture
\x86\DpInst.xml	DpInst configuration/customization file
\x86\AppData\Eula409.txt	End User License Agreement (EULA) text
\x86\AppData\watermark.bmp	Left margin watermark bitmap
\x86\AppData\logo.bmp	Company or device logo bitmap
\x86\Driver\net9500-x86-n51f.inf	NDIS 5.1 driver installation file for x86
\x86\Driver\lan9500-x86-n51f.cat	NDIS 5.1 driver WHQL-signed catalog file for x86
\x86\Driver\lan9500-x86-n51f.sys	NDIS 5.1 driver file for x86
\x86\Driver\WdfCoInstaller01007.dll	Microsoft Wdf Coinstaller v1.7 for x86
\x64\DpInstx64.exe	Microsoft redistributable driver package installer utility for the x64 architecture
\x64\DpInst.xml	DpInst configuration/customization file
\x64\AppData\Eula409.txt	End User License Agreement (EULA) text
\x64\AppData\watermark.bmp	Left margin watermark bitmap
\x64\AppData\logo.bmp	Company or device logo bitmap
\x64\Driver\net9500-x64-n51f.inf	NDIS 5.1 driver installation file for x64
\x64\Driver\lan9500-x64-n51f.cat	NDIS 5.1 driver WHQL-signed catalog file for x64
\x64\Driver\lan9500-x64-n51f.sys	NDIS 5.1 driver file for x64
\x64\Driver\WdfCoInstaller01007.dll	Microsoft Wdf Coinstaller v1.7 for x64

Once the package is customized, as detailed in the following sections, it should be recompressed into an EXE using the Winzip Self Extractor or similar tool.

7.1.1 GUI Visual Elements and Strings Customization

Visual elements used by the installer (logo, watermark and EULA) are located in the `AppData` folder. Other custom textual elements used by the installer wizard are located in the `DpInst.xml` file. These files can be replaced or edited to customize the Installer application's look and feel as desired.

Note: The aforementioned files appear twice in the package, one for each architecture. However, the two sets should be identical.

7.1.2 Localization

The installer package is released with the EULA and custom strings in english only. It can be easily modified to support alternative or additional languages by performing the following:

- replace or add a "language code" section in the `DpInst.xml` file for the desired language
- replace or add EULA files for the desired language

7.1.3 Additional Information

SMSC's automated installer is based on Microsoft DpInst technology. Please refer to the following links to obtain additional information regarding DpInst customization and localization.

Customizing DpInst Wizard Pages:

<http://msdn.microsoft.com/en-us/library/ms790314.aspx>

DpInst Multi-language Customization Instructions:

<http://msdn.microsoft.com/en-us/library/ms791054.aspx>

DpInst Supported Languages:

<http://msdn.microsoft.com/en-us/library/ms790447.aspx>

7.2 Mac OS X Driver Customization

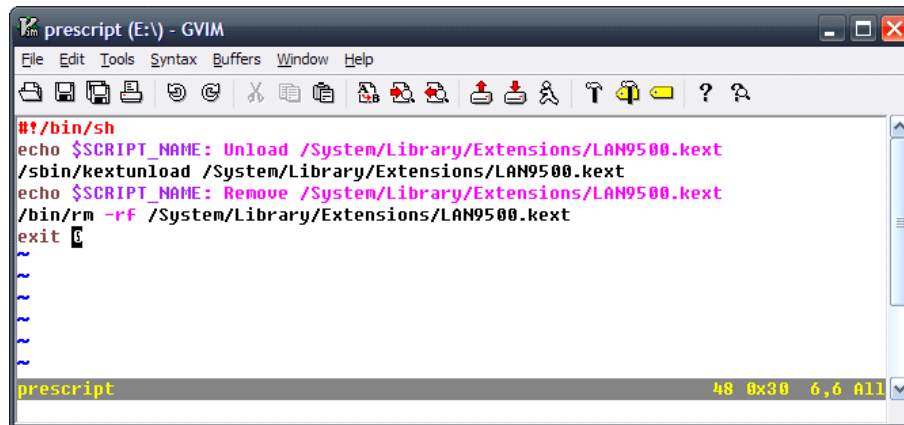
Upon request, SMSC may provide the `LAN9500.kext` kernel extension file to assist the customer in building a customized installation package. This section provides a brief overview of how to create a driver installation package using Apple PackageMaker. For detailed instructions on using Apple PackageMaker, refer to the following links:

<http://developer.apple.com/documentation/DeveloperTools/Conceptual/SoftwareDistribution/Introduction/Introduction.html>

http://s.sudre.free.fr/Stuff/PackageMaker_Howto.html

7.2.1 Building a Driver Installation Package

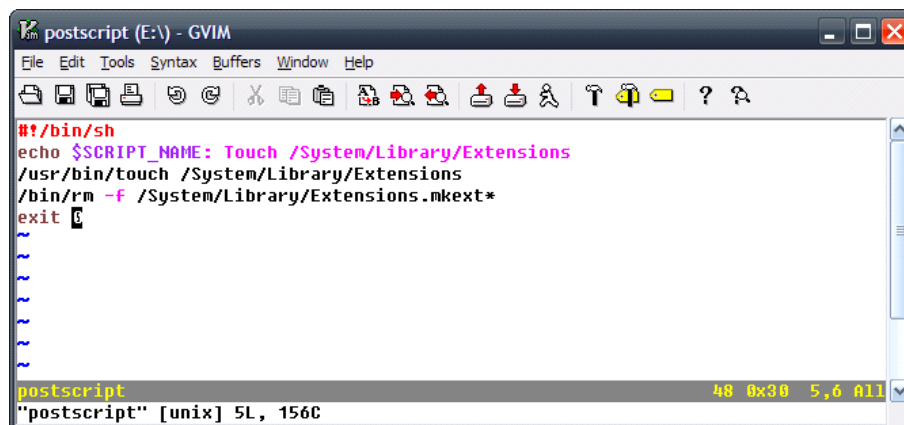
1. Create three rich text format files (optional) for use in the installer
 - `Welcome.rtf` - Welcome message
 - `Readme.rtf` - Readme message
 - `License.rtf` - Software license agreement
2. Create a background image for the installer (optional)
3. Create a prescript and postscript text file as illustrated in [Figure 7.1](#) and [Figure 7.2](#), respectively.



```

#!/bin/sh
echo $SCRIPT_NAME: Unload /System/Library/Extensions/LAN9500.kext
/sbin/kextunload /System/Library/Extensions/LAN9500.kext
echo $SCRIPT_NAME: Remove /System/Library/Extensions/LAN9500.kext
/bin/rm -rf /System/Library/Extensions/LAN9500.kext
exit
  
```

Figure 7.1 Prescript Creation



```

#!/bin/sh
echo $SCRIPT_NAME: Touch /System/Library/Extensions
/usr/bin/touch /System/Library/Extensions
/bin/rm -f /System/Library/Extensions.mkext*
exit
  
```

Figure 7.2 Postscript Creation

4. Execute PackageMaker and add the LAN9500.kext file by selecting the "+" and browsing to its location, as shown in Figure 7.3. Fill in the "Title" and "Description" fields as desired.

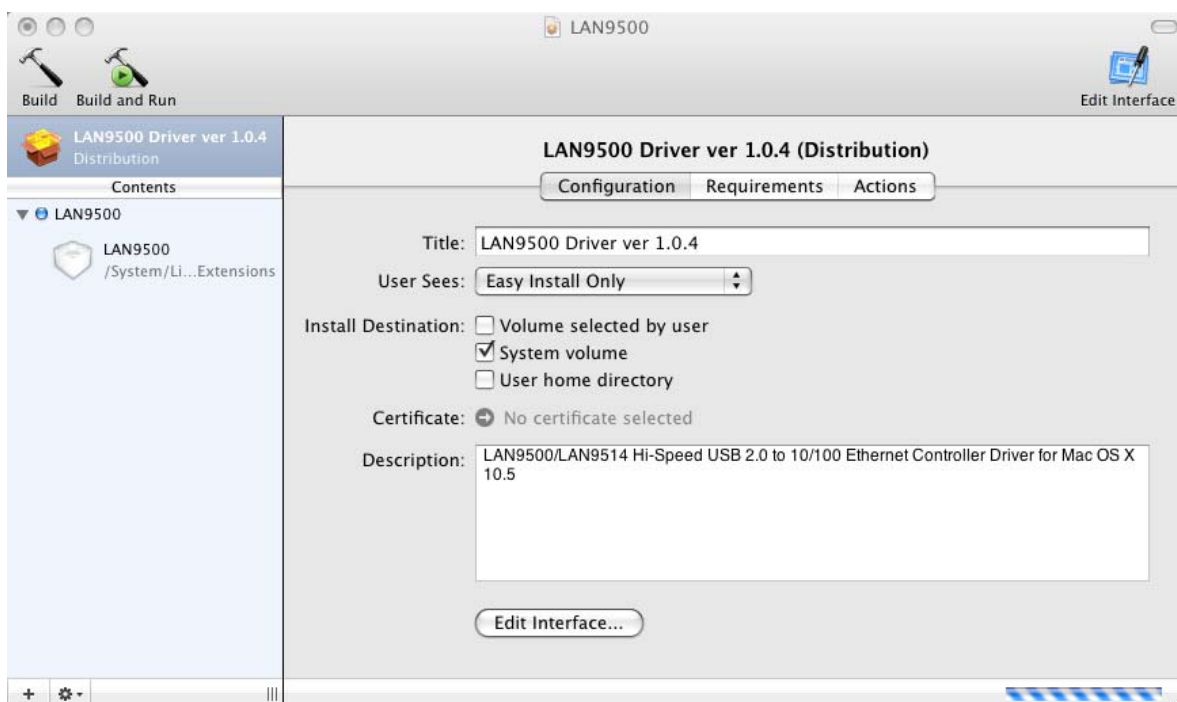


Figure 7.3 Adding LAN9500.kext

5. Open Interface Editor by clicking "Edit Interface"
6. Follow the on-screen prompts to configure the background image, introduction (welcome message), read me, license, and conclusion, as shown in Figure 7.4 through Figure 7.8.

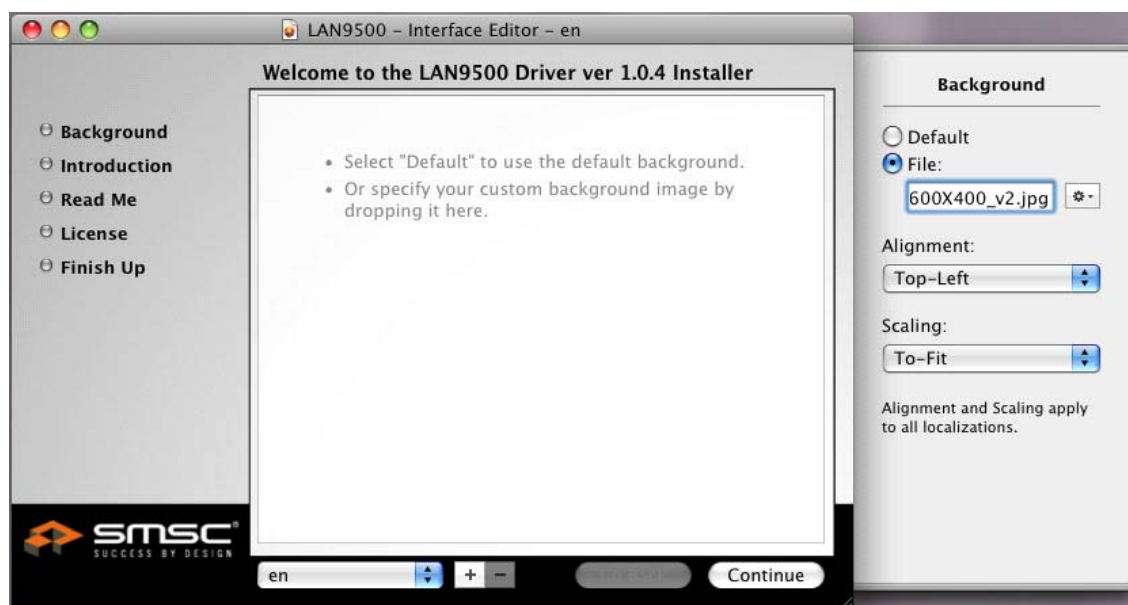


Figure 7.4 Selecting Background Image

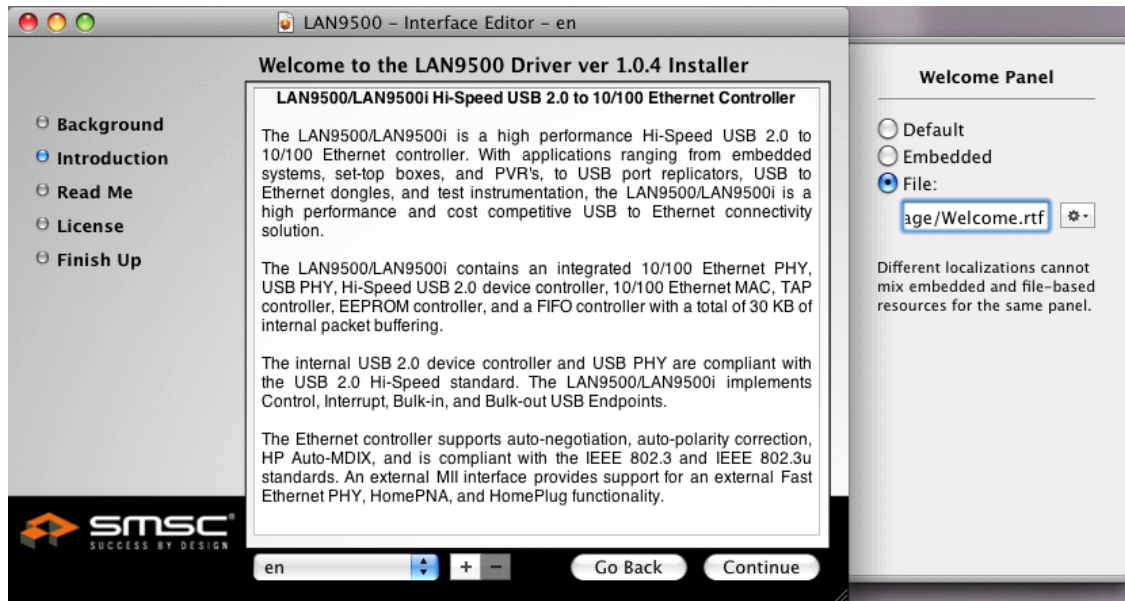


Figure 7.5 Selecting Introduction File

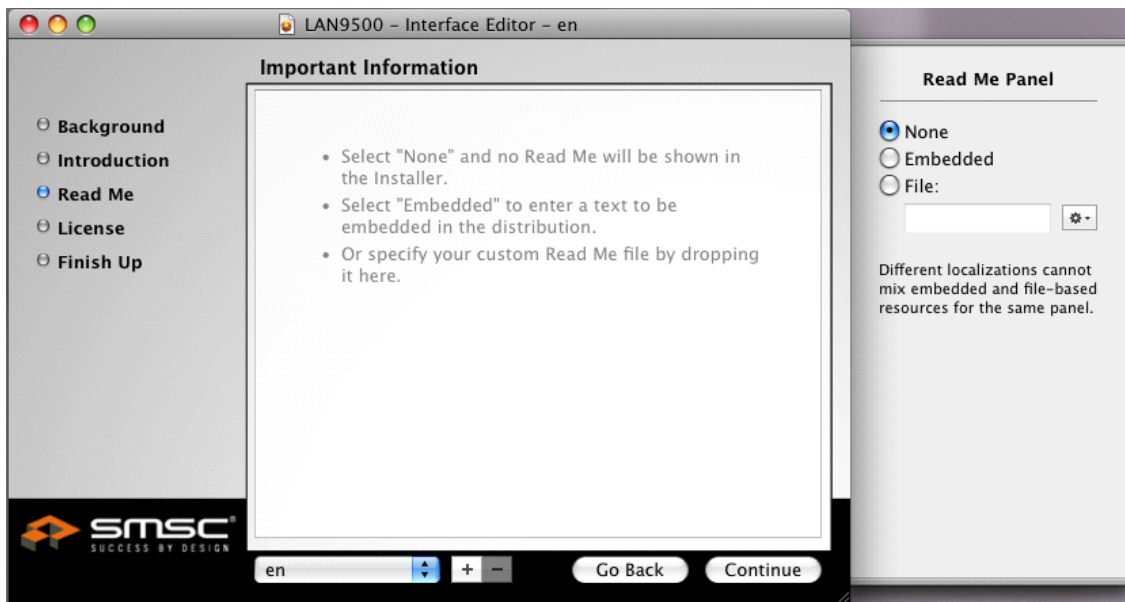


Figure 7.6 Selecting Read Me File

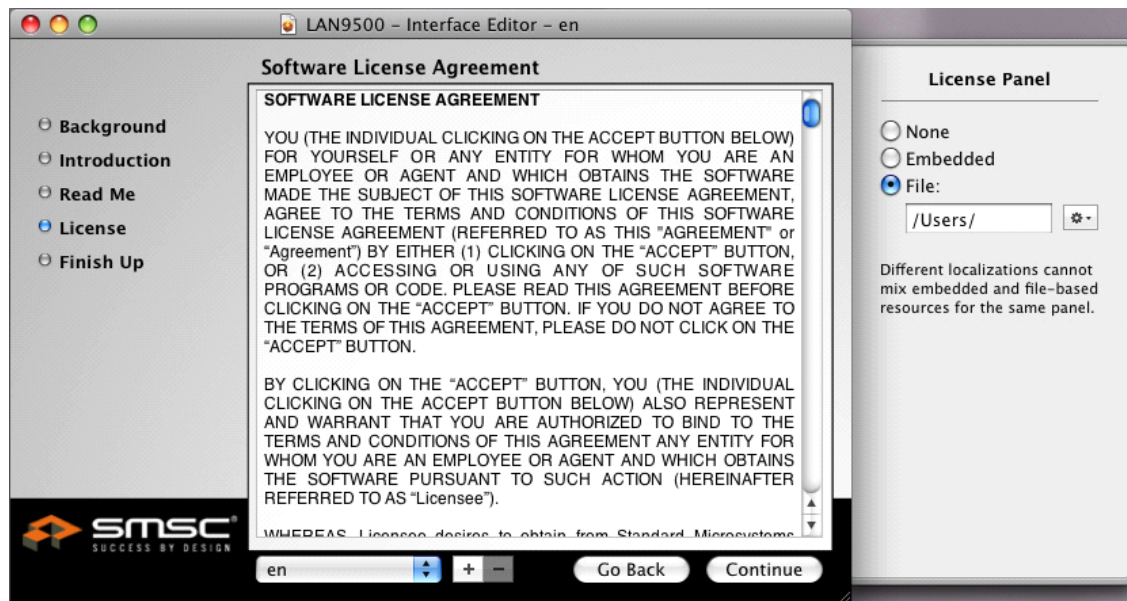


Figure 7.7 Selecting License File

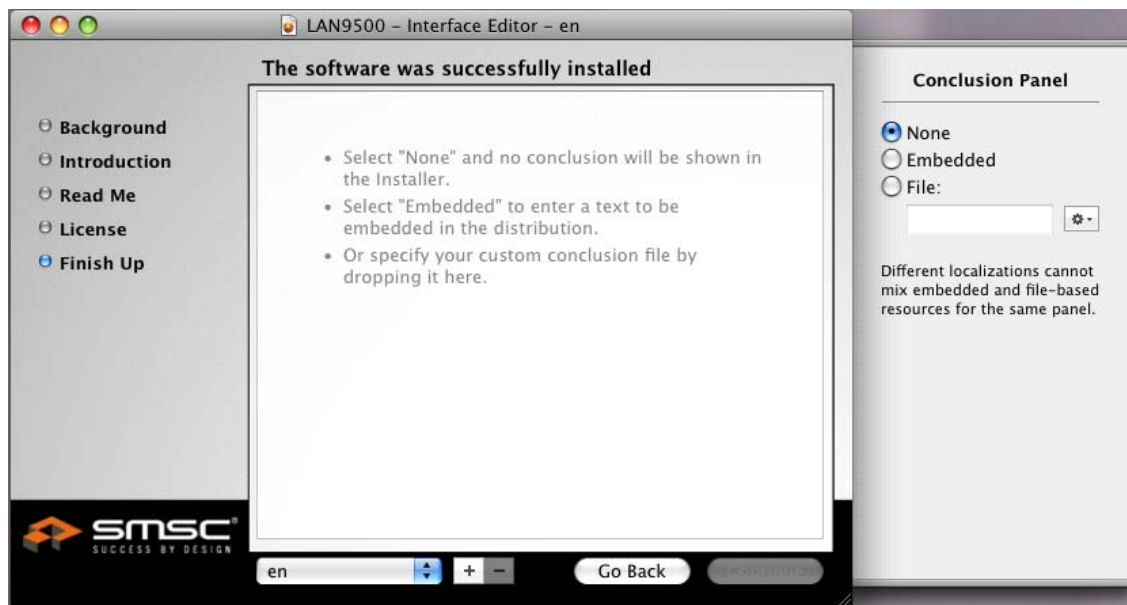


Figure 7.8 Selecting Conclusion File

Software User Manual

- Complete the information under the “Configuration” heading as suggested in [Figure 7.9](#), modifying as necessary for the user’s application.

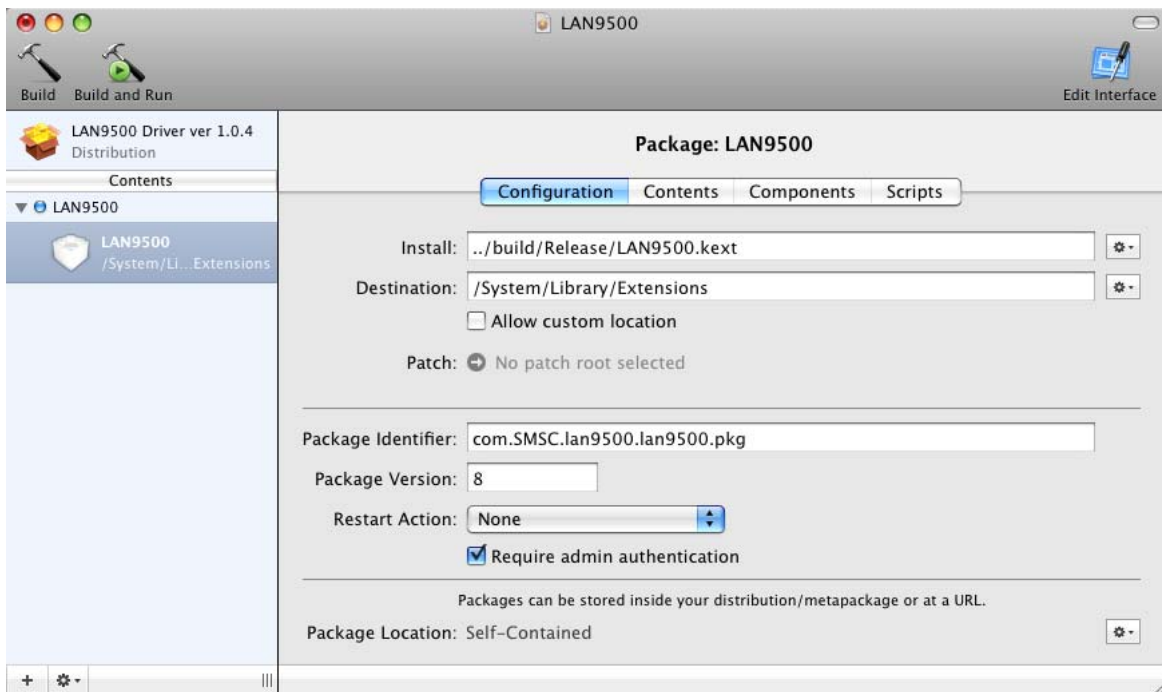


Figure 7.9 Configuration Information

- Modify the file attributes under “Contents” heading as shown in [Figure 7.10](#).

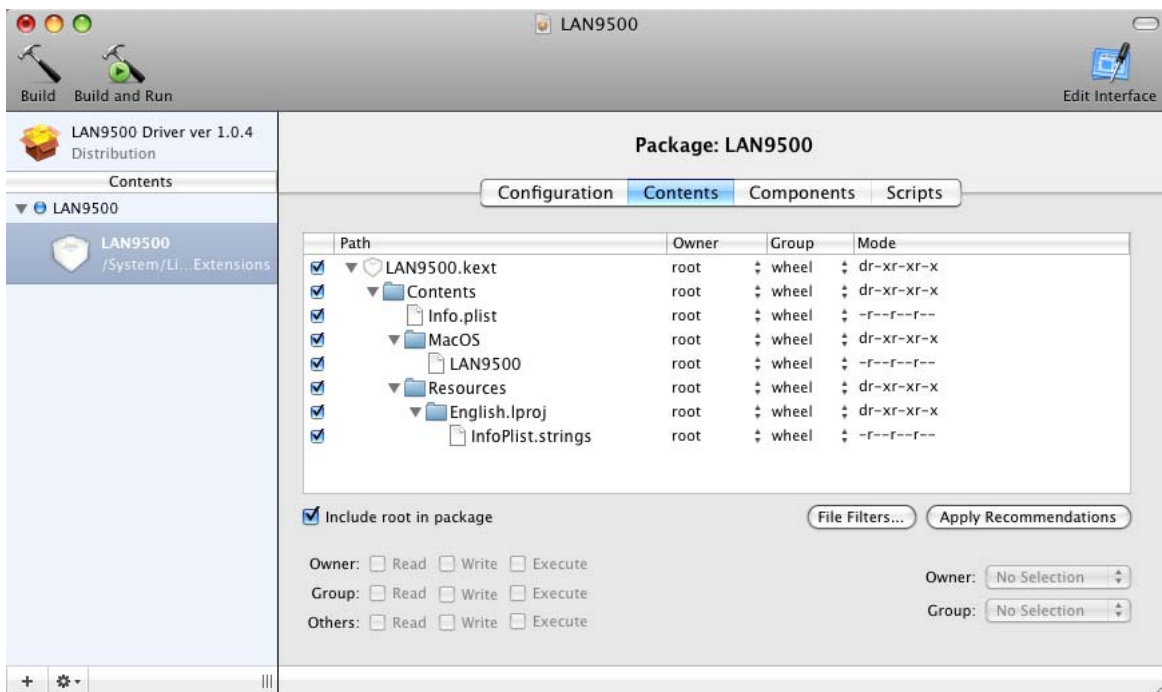


Figure 7.10 Content Information

9. Under the “Scripts” heading, add the prescript and postscript files created earlier to the Preinstall and Postinstall fields as shown in [Figure 7.11](#).

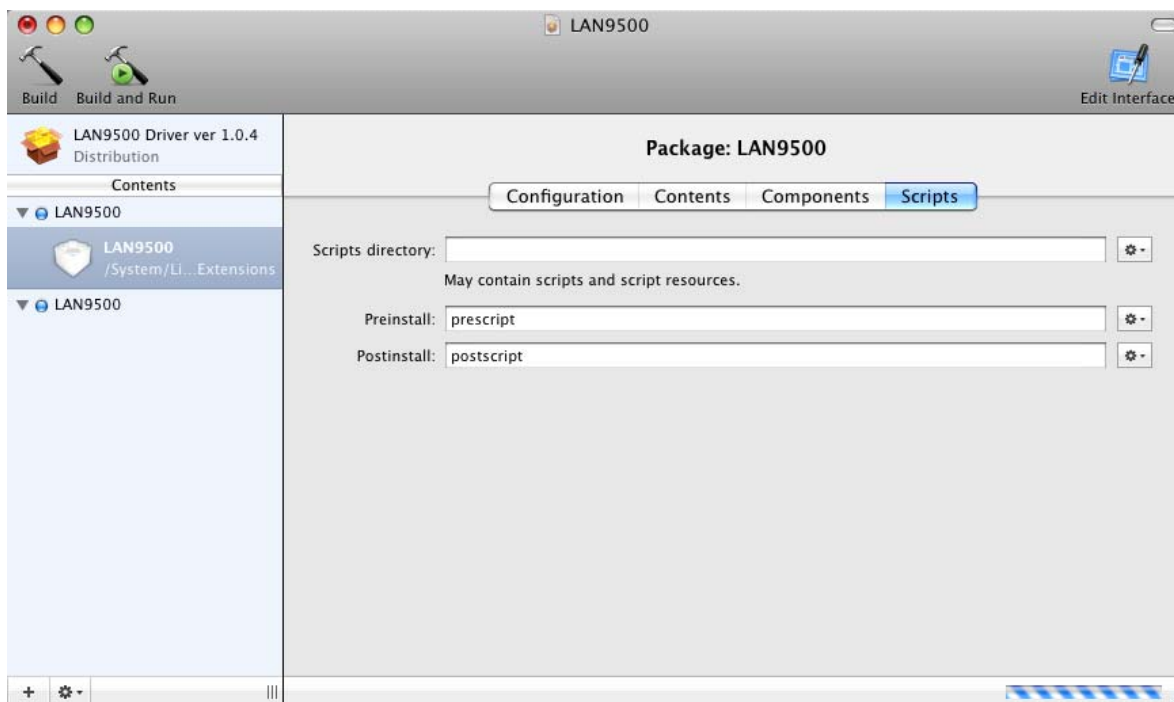


Figure 7.11 Scripts Information

10. Click on the “Build” button to complete the installer package creation.

Chapter 8 Advanced Driver Parameters

The device drivers provide a set of advanced parameters that allow operation of the device to be tailored to a specific application. The methods and parameters vary dependant on the operating system. This chapter provides an overview of how to access these parameters under various operating systems.

8.1 Windows Parameters

Windows parameters are accessible through the “Advanced” tab of the “Device Properties” window. An example of these parameters in Windows XP and Vista can be seen in [Figure 8.1](#) and [Figure 8.2](#), respectively. These parameters may change between driver releases. For additional details, please refer to the release notes (readme.txt) of the particular release being used.

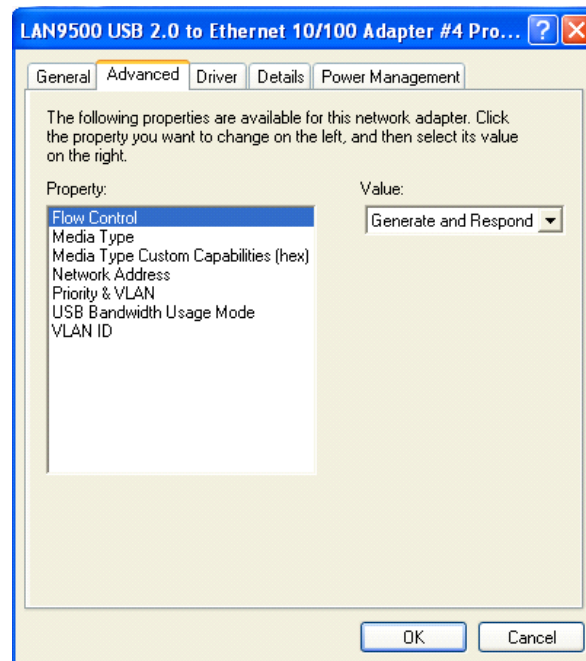


Figure 8.1 Windows XP Advanced Parameters

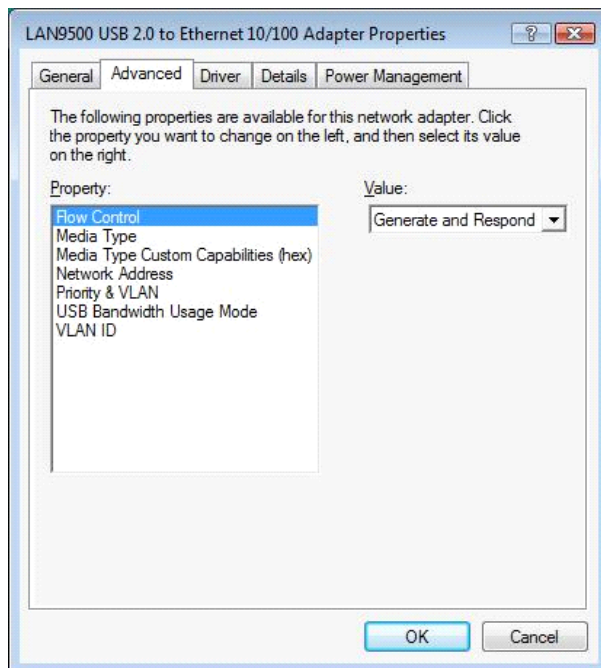


Figure 8.2 Windows Vista Advanced Parameters

8.2 Mac OS X Parameters

Mac OS X device driver parameters are accessible by customizing the `info.plist` file. After installation of the LAN9500/LAN9500i device driver, the following kernal extension (kext) will be created: `/System/Library/Extensions/LAN9500.kext`. The `info.plist` file, which resides within the kext package, contains the device driver parameters. Open the `info.plist` file using the PropertyListEditor application to show the parameters and their default values. Figure 8.3 shows an example `info.plist` file open in the PropertyListEditor.

Note: Changing the `info.plist` contents to invalid values may prevent the loading and execution of the device driver. The keys in the `info.plist` file may be changed between revisions without notice. For detailed key information, refer to the release notes (`readme.txt`) of the particular release being used.

For details regarding kernal extensions (kext) files, refer to the following link:

http://developer.apple.com/DOCUMENTATION/Darwin/Conceptual/KEXTConcept/KEXTConceptPackaging/packaging_kext.html.

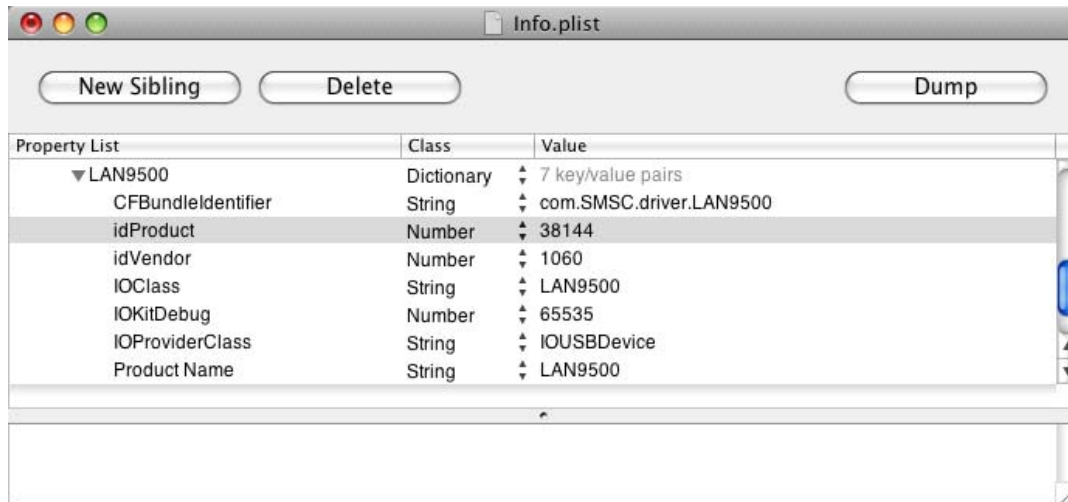


Figure 8.3 Mac OS X info.plist Advanced Parameters

8.3 Linux Parameters

Linux parameters may be modified via the following methods:

- Load Time ([insmod](#))
- Runtime Proprietary ([cmd9500](#))
- Runtime Standard ([ethtool](#))

8.3.1 insmod

The device driver can be loaded with parameters as follows:

```
insmod smscusbnet.ko [parameter1 = value1 parameter2 = value2 ... parameterN = valueN]
```

```
insmod smsc9500.ko [parameter1 = value1 parameter2 = value2 ... parameterN = valueN]
```

For example, to enable the operational mode, type the following commands:

```
insmod smscusbnet.ko operational_mode = 1
insmod smsc9500.ko
```

Note: The supported device driver parameters may change between driver releases. For additional details, please refer to the release notes (readme.txt) of the particular release being used.

8.3.2 cmd9500

Using the SMSC utility "cmd9500", the device parameters can be read or written. The cmd9500 utility is used as follows:

```
./cmd9500 [-h] [-e adaptername] [-c command] [-a address] [-d data] [-f filename]
```

The following example details the cmd9500 method for reading the device configuration:

```
./cmd9500 -e eth0 -c GET_CONFIG
```

Note: The supported device driver parameters may change between driver releases. For additional details, please refer to the release notes (readme.txt) of the particular release being used.

Note: Refer to [Section A.2.3.2, "cmd9500," on page 104](#) for additional information regarding the cmd9500 utility.

8.3.3 ethtool

The device driver supports the standard Linux "ethtool" API. Refer to the Linux documentation for additional information on ethtool.

Note: The supported device driver parameters may change between driver releases. For additional details, please refer to the release notes (readme.txt) of the particular release being used.

Note: Refer to [Section A.2.3.1, "ethtool," on page 104](#) for information on EEPROM programming using ethtool.

8.4 Windows CE Parameters

Windows CE device driver parameters are accessible via the smsc9500.reg file. A sample of a typical smsc9500.reg file is shown below:

```
[HKEY_LOCAL_MACHINE\Comm\SMSC95001\Parms]

"MediaType"=dword:0

;MEDIA_TYPE_AUTO_FULL 0x00
;MEDIA_TYPE_10HD_AUTO 0x01
;MEDIA_TYPE_10FD_AUTO 0x02
;MEDIA_TYPE_100HD_AUTO 0x03
;MEDIA_TYPE_100FD_AUTO 0x04
;MEDIA_TYPE_10HD_FORCED 0x06
;MEDIA_TYPE_10FD_FORCED 0x07
;MEDIA_TYPE_100HD_FORCED 0x08
;MEDIA_TYPE_100FD_FORCED 0x09

.....
```

Note: The supported device driver parameters may change between driver releases. For additional details, please refer to the release notes (readme.txt) of the particular release being used.

Chapter 9 Pre-Execution Environment (PXE) Support

SMSC provides LAN9500/LAN9500i customized PXE-compliant firmware which can be embedded into a PC BIOS to boot pre-OS. This solution is advantageous when utilizing pre-OS environments such as Windows PE over Ethernet.

For additional information on LAN9500/LAN9500i PXE-compliant firmware, contact your SMSC representative, or visit www.SMSC.com.

9.1 PXE Overview*

The Pre-Execution Environment (PXE) is an environment to boot computers using a network interface independently of available data storage devices or installed operating systems. PXE was introduced as part of the Wired for Management framework by Intel and is described in the specification (v2.1) published by Intel and Systemsoft. PXE makes use of several network protocols like IP, UDP, DHCP and TFTP and of concepts like GUID/UUID and Universal Network Device Interface and extends the firmware of the PXE client (the computer to be bootstrapped via PXE) with a set of predefined APIs.

The term PXE client only refers to the role that the machine takes in the PXE boot process. A PXE client can be a server, desktop, laptop or any other machine that is equipped with PXE boot code. The firmware on the client tries to locate a PXE redirection service on the network (Proxy DHCP) in order to receive information about available PXE boot servers. After parsing the answer, the firmware will ask an appropriate boot server for the file path of a network bootstrap program (NBP), download it into the computer's RAM using TFTP, possibly verify it, and finally execute it. If only one NBP is used among all PXE clients it could be specified using BOOTP without any need of a proxy DHCP, but a TFTP boot server is still required.

**Source: http://en.wikipedia.org/wiki/Preboot_Execution_Environment*

Chapter 10 Windows Manufacturing Utility

The Windows Manufacturing Utility application provides a graphical user interface for programming the device EEPROM and performing device tests. This chapter details the installation and operation of the Windows Manufacturing Utility.

Note: The Windows Manufacturing Utility currently supports only the LAN9500/LAN9500i and LAN9500A/LAN9500Ai. A future LAN9512/LAN9514 version of the Windows Manufacturing Utility will be available at a later date.

Note: For the latest release information, please refer to the readme file located at the root of the Windows Manufacturing Utility installation.

10.1 Installation

This section details the installation of the Windows Manufacturing Utility, the Utility Device Driver, and includes setup and system requirements.

10.1.1 Setup and System Requirements

[Table 10.1](#) details the Windows Manufacturing Utility setup and system requirements.

Table 10.1 Setup and System Requirements

CATEGORY	REQUIREMENTS
PCs	■ A 2GHz Pentium 4 or higher PC is recommended for both the EEPROM programmer host system and the test partner system. ■ The test partner should support an IP/ICMP stack capable of replying to 65000 byte ICMP echo requests ("pings of 65000"). ■ Firewalls and any other existing security software must be set to enable "pings of 65000" in both EEPROM programmer host and test partner systems (if unsure on how to do the latter, simply turn firewalls off).
Supported OSs	■ Windows XP x86 (32-bit) MUST be the OS running in the EEPROM programmer host system.
Environment	■ The utility is designed to be used standalone with NO additional applications running on either the EEPROM programmer host or test partner.
Network	■ End-to-end link between the device and the test partner should be on a closed network with no other connected hosts (An exception is granted for sharing a test partner among several assembly lines. See Section 10.4.2, "Sharing the Ping Partner Across Several Assembly Lines," on page 88). ■ The test partner system (or switch) must be capable of (and configured to) autonegotiate advertising of all four 10/100 half/full duplex combinations. ■ For ping tests, it is recommended to place an Ethernet Switch between the device and the test partner for the most consistent results. ■ Also, the device's interface must be set to a static IP address that is on the same subnet as the test partner.
Data	■ Manual editing of the LAN9500Utility.ini file is not supported, and may lead to utility malfunction and/or corrupted EEPROM.
Device Instances	■ Only one device may be plugged into the PC at a time.

10.1.2 Decompressing the Windows Manufacturing Utility Contents

The Windows Manufacturing Utility is distributed in a self extracting EXE file. To begin the installation, double-click the provided LAN9500Utility_vx.x.x.x.exe file, where “x.x.x.x” indicates the release version. This will open the self-extractor window as shown in [Figure 10.1](#).



Figure 10.1 Self-Extraction Window

Indicate the preferred directory for decompression in the “Unzip to folder” field and click “Unzip”.

Note: The default location is: C:\Program Files\LAN9500Utility

Click “OK” and the “Close” to complete the decompression. After the decompression is complete, the following five files should be located in the chosen directory.

Table 10.2 Decompressed File Descriptions

FILE	DESCRIPTION
LAN9500Utility_vx_x_x_x_Readme.txt	A readme file detailing release notes and important information. (the “x_x_x_x” indicates the version number.)
LAN9500Utility.exe	The utility executable
LAN9500Utility.ini	The INI file where settings for the utility are saved
Driver\LAN9500Utility_x86_n51m.inf	Special driver installation script used with the LAN9500Utility only.
Driver\lan9500-x86-n51m.sys	LAN9500 32-bit windows device driver

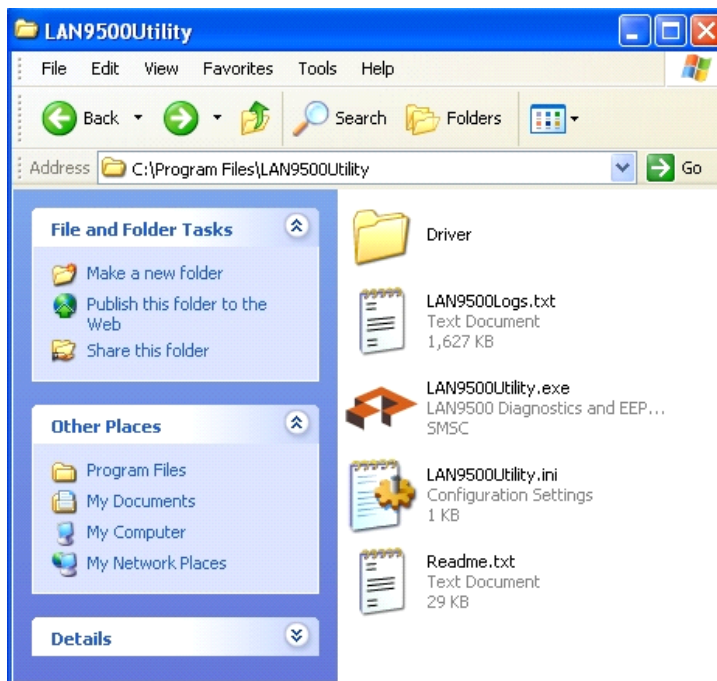


Figure 10.2 Decompressed Contents

10.1.3 Installing the Manufacturing Utility Device Driver

In order to use the Windows Manufacturing Utility, the installation of a special manufacturing utility device driver is required. This special driver is designed exclusively for use with the Windows Manufacturing Utility. Before installing this special manufacturing utility device driver, please note the following:

Note: The Windows Manufacturing Utility will NOT be able to write to the EEPROM with the standard retail device driver package. The special driver package provided with the Windows Manufacturing Utility MUST be used instead. If the standard retail driver has been previously installed on the host system, it MUST be uninstalled before performing any of the instructions provided in this section. Please ensure that NO remnants of a previous installation, such as oem<n>.inf copies or net9500*.inf files, remain in the c:\windows\inf directory.

Note: This special driver enables writing to the EEPROM. If not performed properly, EEPROM write operations can render the device inaccessible. This driver and tool are meant for a production environment only and therefore, MUST NEVER be released to end users.

Note: If the EEPROM programming host machine has internet connectivity, it is recommended to disable automatic updates. This will ensure the special manufacturing version of the device driver included with the utility is NOT automatically updated to the retail version via Windows update (which, as previously stated, does not support EEPROM writing).

Software User Manual

To begin the manufacturing utility device driver installation, plug in a device that has a NON programmed EEPROM. This will trigger a "Found New Hardware Wizard" window. Select the "No, not this time" option to indicate you do not want to search for drivers in via Windows Update and click "Next".



Figure 10.3 Found New Hardware Wizard Welcome Window

Select "Install from a list or specific location (Advanced)" option and click "Next".

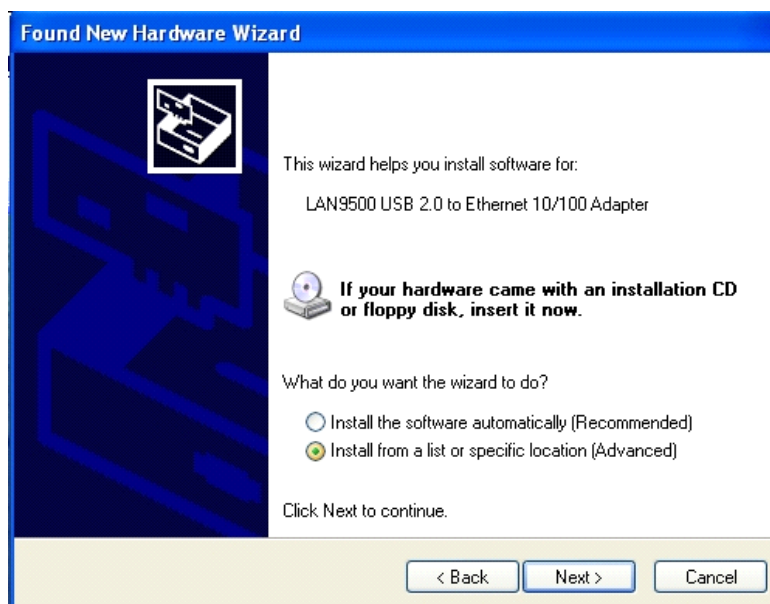


Figure 10.4 Install from a List or Specific Location Option

Select "Don't search. I will choose the driver to install." option and click "Next".

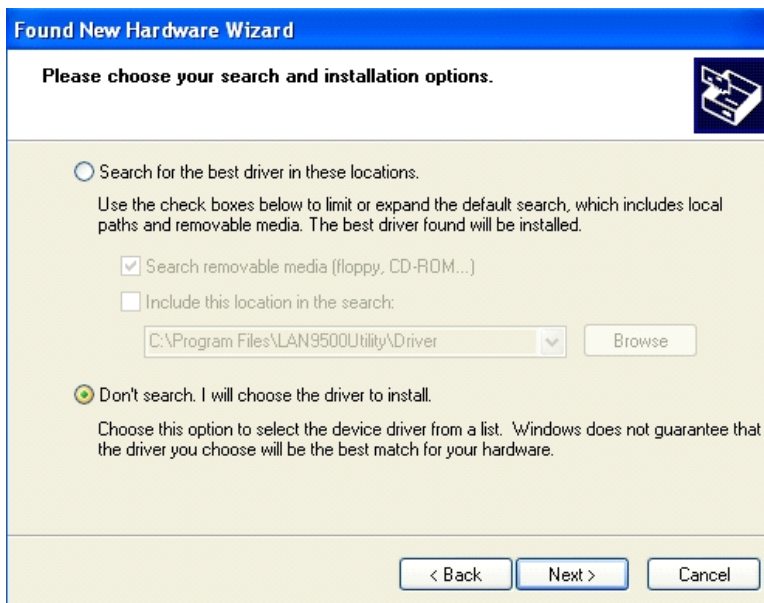


Figure 10.5 Don't Search Option

Select the "Have Disk..." option in the Select Network Adapter window. Browse to the location where the driver has been extracted (i.e., C:\Program Files\LAN9500Utility\Driver) and select "OK".



Figure 10.6 Install From Disk Option

Software User Manual

Click "Next" in Select Network Adapter window. This will begin the driver installation.

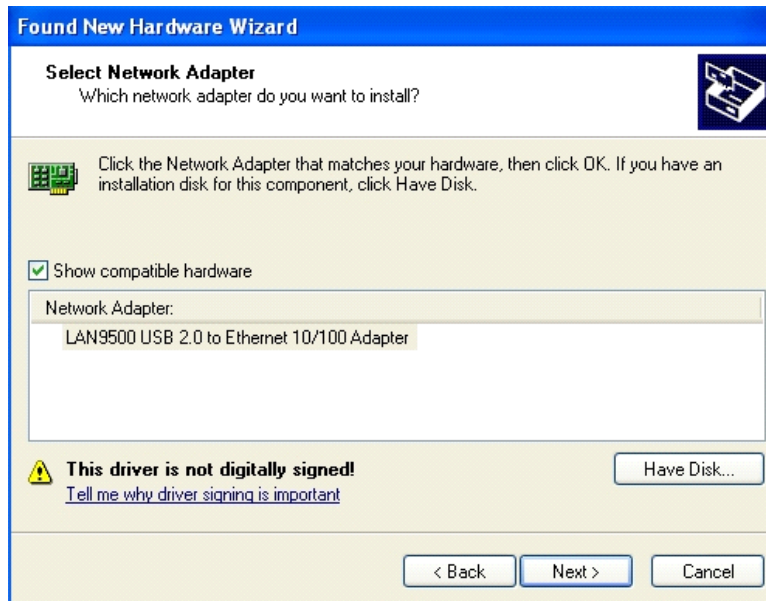


Figure 10.7 Select Network Adapter Window

A Hardware Installation window will appear, notifying the user that the driver is not WHQL certified. Press "Continue anyway" to continue the installation.

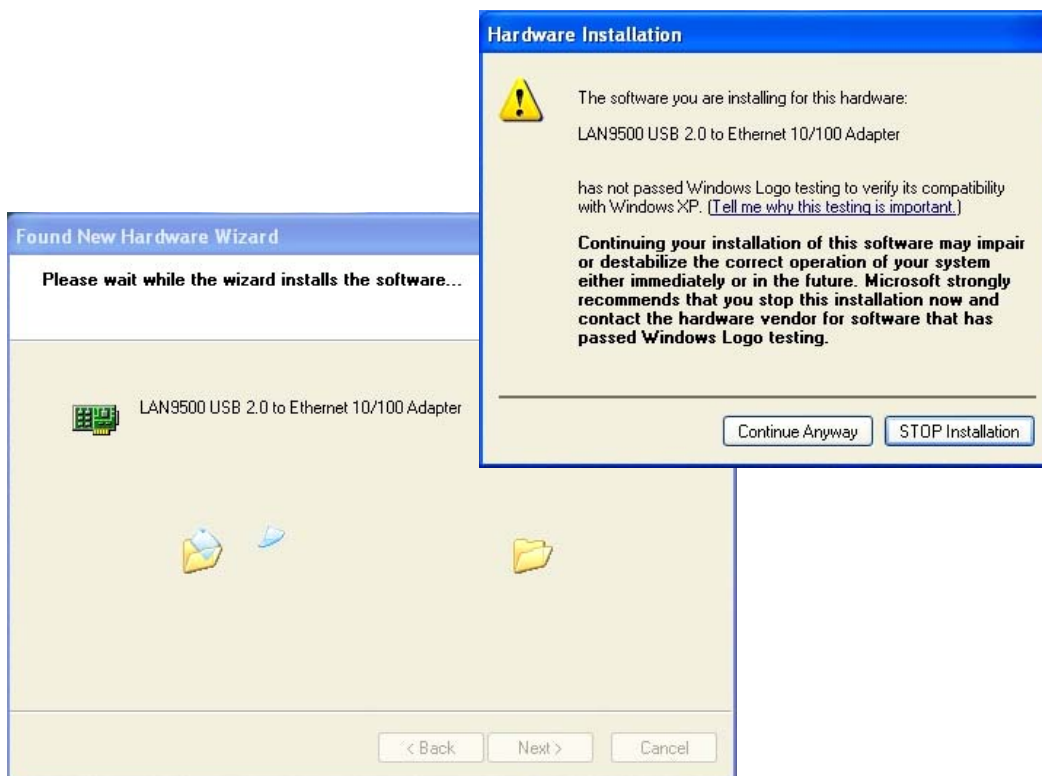


Figure 10.8 Hardware Installation Warning Window



Figure 10.9 Hardware Installation Complete Window

After the driver initialization is completed, it is recommend to setup the device with a manually assigned IP address (NOT DHCP). As mentioned previously, this IP address and the test partner system's network interface should be in the same IP subnet.

10.2 Operation

This section details the operation of the Windows Manufacturing Utility, including the following:

- [Starting the Utility](#)
- [Using the Utility](#)
- [Exiting the Utility](#)

10.2.1 Starting the Utility

After [Decompressing the Windows Manufacturing Utility Contents](#) and [Installing the Manufacturing Utility Device Driver](#), the Windows Manufacturing Utility can be launched by double-clicking the LAN9500Utility.exe file.

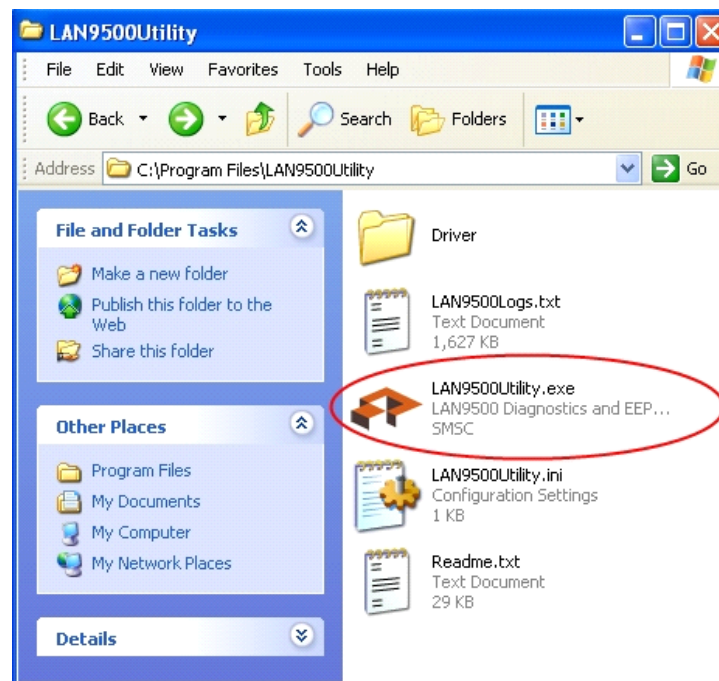


Figure 10.10 Launching the Windows Manufacturing Utility

Note: A desktop shortcut to the LAN9500Utility.exe can be created on the desktop if desired.

10.2.2 Using the Utility

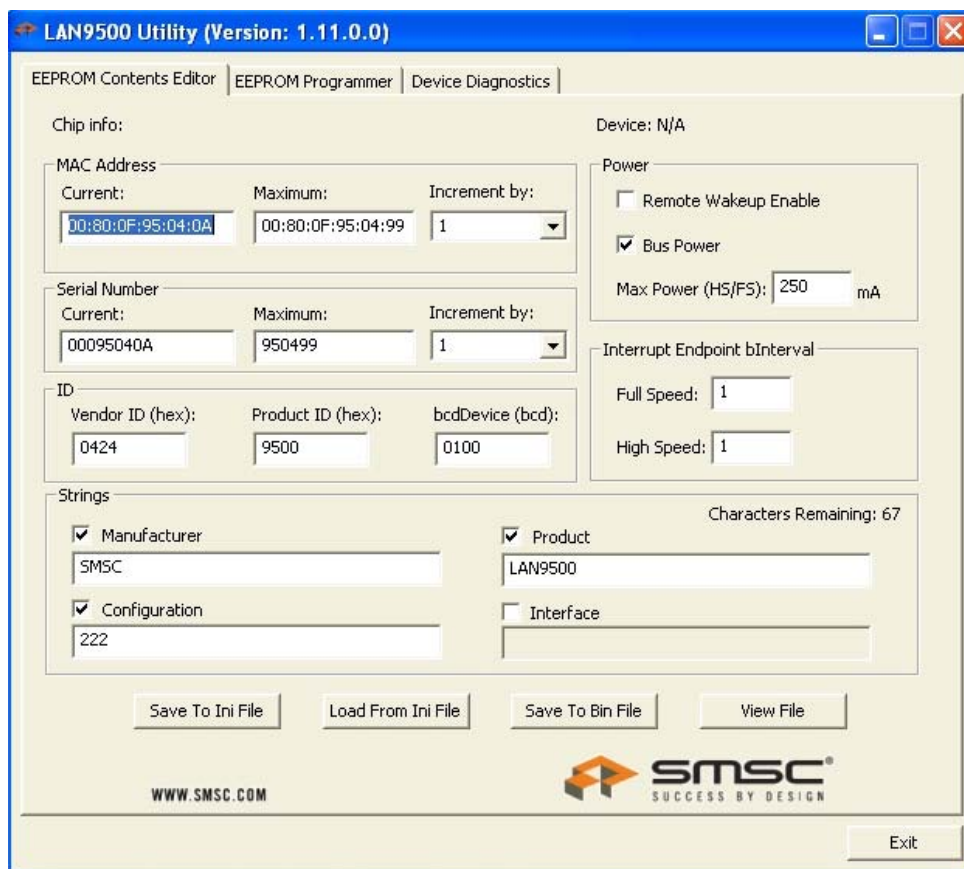
The Windows Manufacturing Utility consists of three main screens which are accessible through the following tabs:

- [EEPROM Contents Editor Tab](#)
- [EEPROM Programmer Tab](#)
- [Device Diagnostics Tab](#)

10.2.2.1 EEPROM Contents Editor Tab

The EEPROM Contents Editor tab allows the user to set the device EEPROM parameters off-line and save them to the LAN9500Utility.ini file. This file will be used to provide the initial parameters to program the device's EEPROM from the [EEPROM Programmer Tab](#). The EEPROM Contents Editor tab can be seen in [Figure 10.11](#).

Note: The EEPROM Contents Editor tab is the default tab shown when the utility is initially started.



LAN9500 Utility (Version: 1.11.0.0)

EEPROM Contents Editor | EEPROM Programmer | Device Diagnostics

Chip info: Device: N/A

MAC Address

Current: 00:80:0F:95:04:0A Maximum: 00:80:0F:95:04:99 Increment by: 1

Serial Number

Current: 00095040A Maximum: 950499 Increment by: 1

ID

Vendor ID (hex): 0424 Product ID (hex): 9500 bcdDevice (bcd): 0100

Power

☐ Remote Wakeup Enable

☒ Bus Power

Max Power (HS/FS): 250 mA

Interrupt Endpoint bInterval

Full Speed: 1

High Speed: 1

Strings

☒ Manufacturer SMSC

☒ Product LAN9500

☒ Configuration 222

☐ Interface

Characters Remaining: 67

Save To Ini File Load From Ini File Save To Bin File View File

WWW.SMSC.COM

SMSC[®] SUCCESS BY DESIGN

Exit

Figure 10.11 EEPROM Contents Editor Tab

Software User Manual

10.2.2.1.1 EEPROM CONTENTS EDITOR FIELDS

The following fields are available for editing in this tab:

Table 10.3 EEPROM Contents Editor Tab - Fields

CATEGORY	FIELD DESCRIPTIONS
MAC Address	These fields define the 6-byte universally unique MAC address parameters. Bytes are separated by a colon.
	Current: Defines the first MAC address (will be incremented after Manual/Auto burn for the next device)
	Maximum: Defines the last MAC address to be used before rolling over to 0
	Increment By: Will be added to Current MAC Address after successful EEPROM burn cycle (max is 255)
Serial Number	These fields define the unique USB serial number (up to 16-digit hexadecimal number).
	Current: Defines the first serial number (will be incremented after Manual/Auto burn for the next device)
	Maximum: Defines the last serial number to be used before rolling over
	Increment By: Will be added to Current serial number after successful EEPROM burn cycle (max is 255)
ID	These fields define the various USB IDs.
	Vendor ID: This is the company USB Vendor ID. It is a 4-digit hexadecimal number.
	Product ID: This is the device USB Product ID. It is a 4-digit hexadecimal number.
	bcdDevice: This is the release number assigned to the device. It is a 4-digit binary-coded decimal (BCD) number. BCD is a decimal number format (only digits 0-9 allowed). Each digit is allocated four bits in the data field's binary representation.
Power/ Remote Wake	These parameters affect values of the USB configuration descriptor. Improper setups which violate the USB 2.0 specification will not be allowed. An error message will be displayed if an illegal value has been entered.
	Remote Wakeup Enable: Enables/disables remote USB wakeup capability (Enabled when checked)
	Bus Power: Selects device configuration between Bus Power and Self Power (Bus-Power when checked)
	Max Power: Maximum Power Consumption in mA (between 2mA and 500mA for BusPower; between 0mA and 100mA for SelfPower) Note: This value must always be an even number.
Interrupt Endpoint Interval	These fields define the interval for polling the interrupt endpoint as defined in the bInterval field of the endpoint descriptor by the USB 2.0 specification. Illegal values will not be allowed; an error message will be displayed if one has been entered.
	Full Speed: Interrupt endpoint bInterval for full speed operation (Valid range of 1-255)
	High Speed: Interrupt endpoint bInterval for high speed operation (Valid range of 1-16)

Table 10.3 EEPROM Contents Editor Tab - Fields (continued)

CATEGORY	FIELD DESCRIPTIONS
Strings	These fields define the user customizable device strings. Each string can be individually enabled/disabled via a check box. The "Characters remaining" counter helps the user adjust the strings' lengths so that all content fits within the EEPROM size.
	Manufacturer: A user definable string used for manufacturer information
	Product: A user definable string used for product information
	Configuration: A user definable string used for configuration information
	Interface: A user definable string used for interface information

Note: It is strongly recommended that a device with the same size EEPROM as the final target be plugged into the system and recognized by the utility. This will allow the utility to identify the size of the EEPROM being edited and properly indicate the "characters remaining count". If no device is plugged in, the EEPROM contents editor will use a default EEPROM size of 256 bytes to calculate the "characters remaining count".

10.2.2.1.2 EEPROM CONTENTS EDITOR ACTIONS

The following action buttons are available in this tab:

Table 10.4 EEPROM Contents Editor Tab - Actions

BUTTON	ACTION DESCRIPTION
Save To Ini File	Saves the information currently displayed in the fields detailed in Section 10.2.2.1.1, "EEPROM Contents Editor Fields" to the LAN9500Utility.ini file.
Load From Ini File	Reads the contents of LAN9500Utility.ini file and refreshes the fields detailed in Section 10.2.2.1.1, "EEPROM Contents Editor Fields" with the file contents.
Save To Bin File	Saves the information currently displayed in the fields detailed in Section 10.2.2.1.1, "EEPROM Contents Editor Fields" to a binary data file to be burned to the EEPROM outside of the Windows Manufacturing Utility
View File	Opens and displays the LAN9500Utility.ini file

10.2.2.2 EEPROM Programmer Tab

The EEPROM Programmer tab allows the user to perform operations such as programming and verifying contents directly on the EEPROM. The EEPROM Programmer tab can be seen in [Figure 10.12](#).

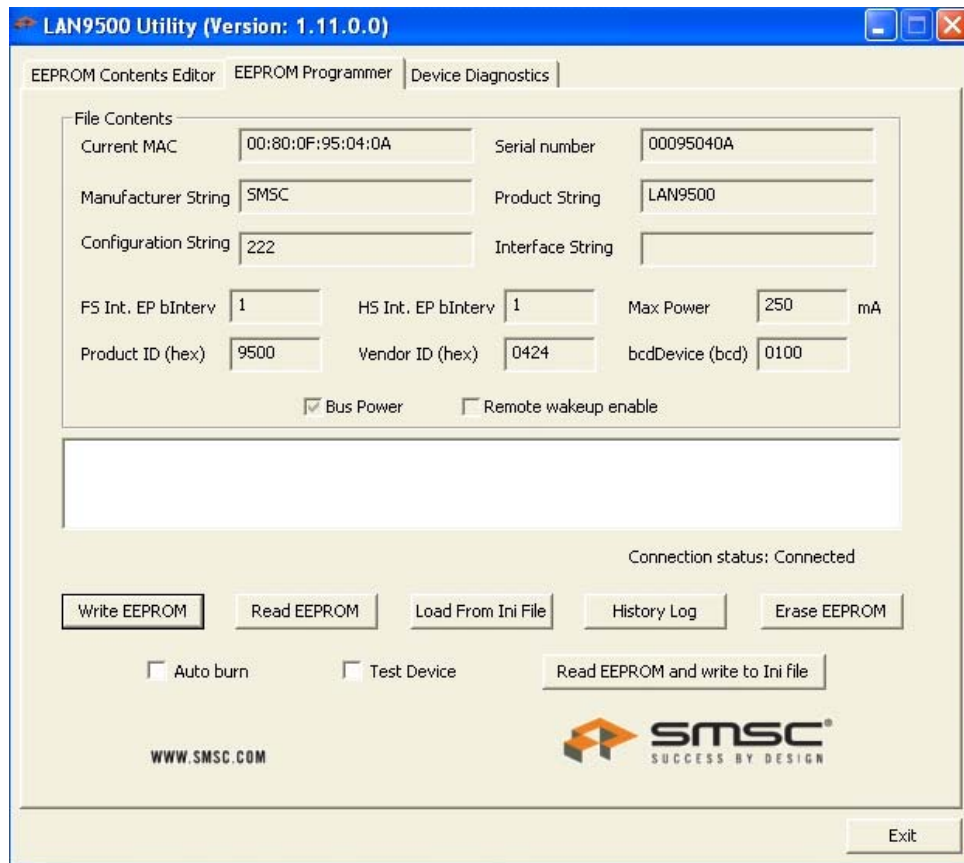


Figure 10.12 EEPROM Programmer Tab

Note: If a device is plugged in a USB port and the driver is properly loaded, the indicator in lower right corner of the tab will read "Connected". Otherwise the indicator will read "Not Connected" and an unprogrammed device should be plugged in.

10.2.2.2.1 EEPROM/FILE CONTENTS FIELDS

These fields show the non-editable information that the utility has read from either the EEPROM or the ini file. For definitions of these fields, refer to [Table 10.3, "EEPROM Contents Editor Tab - Fields," on page 79](#).

10.2.2.2.2 EEPROM PROGRAMMER TAB ACTIONS & STATUS

The following action buttons are available in this tab:

Table 10.5 EEPROM Programmer Tab - Actions

BUTTON	ACTION DESCRIPTION
Read EEPROM and Write to Ini File	This button reads the attached device's EEPROM and saves it to the LAN9500Utility.ini file.
Write EEPROM	This button burns the information from the LAN9500Utility.ini file into the EEPROM of the attached device. The status is reported in the list box and also saved in the LAN9500Logs.txt file.
Read EEPROM	This button reads the attached device's EEPROM and displays it in the EEPROM/File Contents Fields . The status is reported in the list box and also saved in the LAN9500Logs.txt file.
Load From Ini File	This button reads the LAN9500Utility.ini file and displays it's contents in the EEPROM/File Contents Fields .
History Log	This button opens LAN9500Logs.txt file in text format.
Erase EEPROM	This button erases the EEPROM of the attached device.

The following checkboxes are available in this tab:

Table 10.6 EEPROM Programmer Tab - Checkboxes

CHECKBOX	DESCRIPTION
Auto Burn	When this checkbox is selected, the user will be prompted for confirmation. Clicking on "Yes" will start the autoburn procedure. This operating mode allows for automatic programming of boards in a serial manner with a minimal amount of user intervention. The utility will instruct the operator when to unplug a device that has already been programmed and when to plug in a new unprogrammed device, but will perform everything else automatically.
Test Device	When this checkbox is selected, the tests chosen in the Device Diagnostics Tab will be executed before the EEPROM programming begins (except for the EEPROM contents verification which is performed after programming). An error message will prompt the user if any of the selected tests fail. Unchecking the checkbox will disable these tests.

The following status fields are available in this tab:

Table 10.7 EEPROM Programmer Tab - Status

STATUS FIELD	DESCRIPTION
Detailed Status	A scrollable area which records all of the activity initiated from this tab in the current session. Note: This information is also written to the LAN9500Logs.txt file for later review.
Status Overview	A text line showing the current/last state of diagnostic activity. When the tab is first entered the status is not shown. The status will be updated as the programming and tests progress with the last entry in the Detailed Status field.

10.2.2.3 Device Diagnostics Tab

The Device Diagnostics tab allows diagnostic tests to be performed on the device. Tests to be performed are selected using the provided checkboxes. Status information is shown while tests are running and when they are completed. The Device Diagnostics tab can be seen in [Figure 10.13](#).

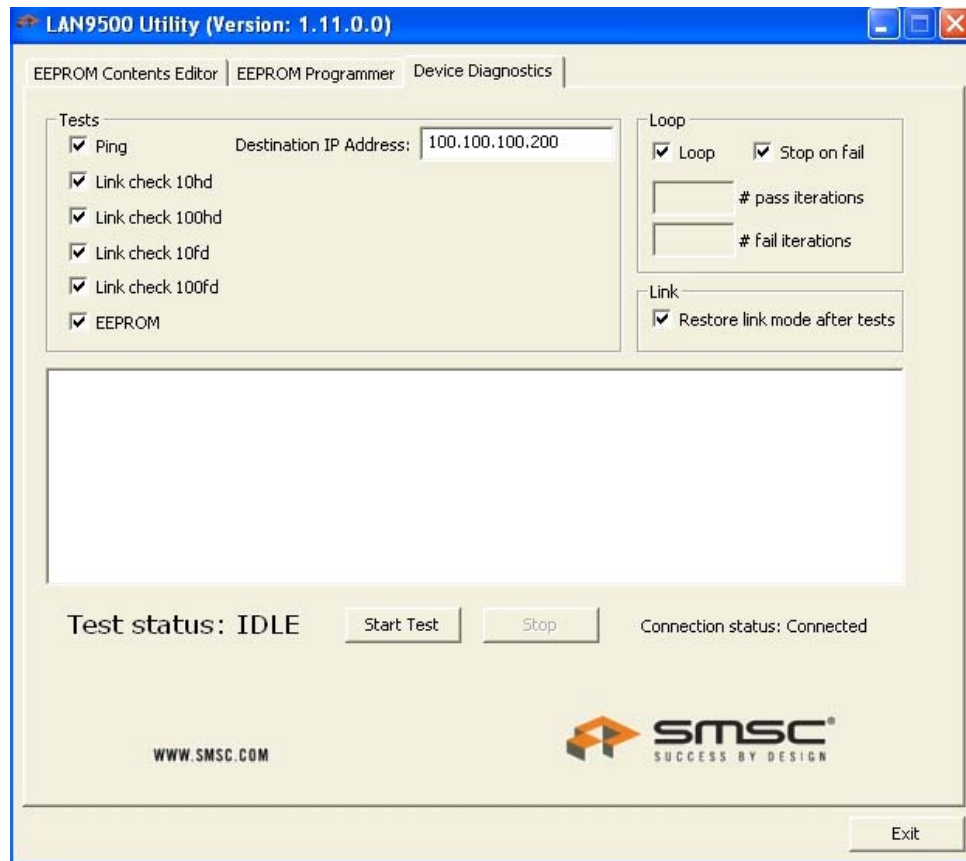


Figure 10.13 Device Diagnostics Tab

10.2.2.3.1 TESTS SECTION

The Tests section of the Device Diagnostics tab is used to enable and configure the tests to be run. The following test options are available in this tab:

Table 10.8 Device Diagnostics Tab - Tests

TESTS	DESCRIPTION
Ping	When checked, the application will transmit a 65KB ICMP echo request datagram (a.k.a. ping) to the connected partner PC through the device. The test passes if the entire datagram is echoed back. The test fails if 5 consecutive ping attempts do not receive a response.
Destination IP Address	In this box, enter the IP address of the test partner PC used for the ping test. This box is greyed out if the "Ping" box is unchecked. Note: THE DESTINATION IP MUST BE ON THE SAME SUBNET AS THE DEVICE. This must be verified prior to testing. Otherwise, the stated results may be invalid.

Table 10.8 Device Diagnostics Tab - Tests (continued)

TESTS	DESCRIPTION
Link Check 10hd	When checked, the utility will attempt to establish a 10Mbps/Half-Duplex link through the device with the partner PC. If the Ping test is enabled, it will be performed immediately after the link is established. If a link is established and the ping is successful, the test passes. Otherwise, the test fails.
Link Check 100hd	When checked, the utility will attempt to establish a 100Mbps/Half-Duplex link through the device with the partner PC. If the Ping test is enabled, it will be performed immediately after the link is established. If a link is established and the ping is successful, the test passes. Otherwise, the test fails.
Link Check 10fd	When checked, the utility will attempt to establish a 10Mbps/Full-Duplex link through the device with the partner PC. If the Ping test is enabled, it will be performed immediately after the link is established. If a link is established and the ping is successful, the test passes. Otherwise, the test fails.
Link Check 100fd	When checked, the utility will attempt to establish a 100Mbps/Full-Duplex link through the device with the partner PC. If the Ping test is enabled, it will be performed immediately after the link is established. If a link is established and the ping is successful, the test passes. Otherwise, the test fails.
EEPROM	When checked, this test verifies that the device EEPROM has been programmed correctly. When this test is run standalone from the Device Diagnostics tab, the MAC address and Serial Number in the EEPROM are checked for consistency with the ranges specified by the LAN9500Utility.ini file. When this test is run from the EEPROM Programmer Tab, the MAC address and Serial Number in the EEPROM are checked to exactly match the Current MAC Address and Current Serial number specified by the LAN9500Utility.ini file.

10.2.2.3.2 LOOP SECTION

The Loop section of the Device Diagnostics tab allows the selected tests in the [Tests Section](#) to run continuously. The following loop options are available in this tab:

Table 10.9 Device Diagnostics Tab - Loops

OPTION	DESCRIPTION
Loop	When checked, the selected tests will run continuously until manually stopped. When unchecked, each selected test will run only once. This setting is only effective when tests are running standalone from the Device Diagnostics Tab.
Stop on fail	If this checkbox is selected in conjunction with the Loop checkbox, test execution will stop when an error is encountered. This option is useful for providing access to the device at the point an error occurred. This option is valid only if the Loop checkbox is selected.
# pass iterations	This non-editable field displays the number of successfully executed test runs since the start of the current loop.
# fail iterations	This non-editable field displays the number of test runs that have produced at least one error since the start of the current loop. Note: If the "Stop on fail" checkbox is selected, this number will never be greater than 1.

Software User Manual

10.2.2.3.3 LINK SECTION

The Link section of the Device Diagnostics tab provides the following option:

Table 10.10 Device Diagnostics Tab - Links

OPTION	DESCRIPTION
Restore link mode after tests	When this checkbox is selected, the link mode will be restored to its previous state before the test started. This option is not valid during autoburn.

10.2.2.3.4 STATUS INFORMATION

The following status information is provided on the Device Diagnostics tab:

Table 10.11 Device Diagnostics Tab - Status Information

OPTION	DESCRIPTION
Detailed Status	A scrollable area which records all of the activity initiated from the Device Diagnostics tab in the current session. Note: This information is also written to the <code>LAN9500Logs.txt</code> file for later review.
Status Overview	A large, boldfaced text line showing the current/last state of diagnostic activity. When the Device Diagnostics tab is first entered, the status is shown as "IDLE". If a test is executing, the status is shown as "TESTING". If a test cycle has finished executing, the status will be shown as "PASSED" or "FAILED", depending on the most recent result.
Connection Status	As in the EEPROM Programmer Tab , this line will read "Connected" if the device is connected and the device driver is properly loaded. Otherwise, it will read "Not connected." In the "Not Connected" state, the device will not function and testing should not be attempted.

10.2.2.3.5 DEVICE DIAGNOSTICS TAB ACTIONS

The following action buttons are available in this tab:

Table 10.12 Device Diagnostics Tab - Actions

BUTTON	ACTION DESCRIPTION
Start Test	Begins execution of a test run with the tests that have been selected. These tests will loop if the "Loop" checkbox is selected, and will run continuously until the "Stop" button is pressed. If "Stop on fail" is checked, then testing will also stop if any failure is encountered. If the "Loop" box is not checked, then each selected test will run once and the loop will end.
Stop	Stops test execution

10.2.3 Exiting the Utility

The Windows Manufacturing Utility can be exited in any of the following way:

- Clicking the "Exit" button
- Clicking the window's close box
- Pressing the ESC key from any tab

10.3 Production Testing Methodology and Coverage

10.3.1 Test Methodology

A Windows driver identifies a device by its serial number. When a device is connected with a different serial number than the previous devices Windows has seen before, Windows installs a new "adapter instance" for the driver. If the driver is WHQL certified, this installation is generally performed silently in the background. If a DHCP server is also available for IP address assignment, the device will be ready to run immediately after. However, if the driver is not certified (and the special driver used by the Windows Manufacturing Utility is purposely not certified), the user is prompted for driver installation. This behavior of Windows recognizing each new programmed serial number and loading a new driver instance causes two problems in a production environment:

1. Installation of a driver takes time. Even if the device driver was WHQL certified and installed automatically/silently, this process would require approximately 15 additional seconds of unproductive time for each autoburn cycle.
2. A machine used for production will generally program thousands of NIC cards, which would lead to thousands of entries in the registry. This leads to a slow performing production machine at risk of exceeding the maximum allowed registry size. This could potentially be prevented by uninstalling the driver for each instance after test completion. However this solution would again increase the critical "assembly time per NIC".

In order to avoid the problems explained above, the Windows Manufacturing Utility tests and programs as much of the device as possible WITHOUT the new EEPROM contents (i.e., serial number) actually taking effect (and requiring Windows to install a new adapter instance). The downside to this method is that the utility is NOT testing the device with its final EEPROM contents being utilized. The production workflow assumption is that all devices are plugged into the host system with unprogrammed EEPROMs and they are all recognized as the same device by Windows. As soon as an unprogrammed device is connected, the utility performs the selected tests (out of ping, link tests). If these programs pass, the EEPROM is programmed (and verified if selected). If there is no failure detected during the testing or EEPROM verification, the utility instructs the operator to disconnect the device (which has completed its cycle) and connect the next one.

To address the concern of the production line not testing each device using its final EEPROM contents, it is suggested to take random control samples from each lot (i.e., 1 out of 100 or 1 out of 1000). This sample can be tested by connecting it to a PC, installing the production device driver and performing typical end user network activity. Alternatively, the manufacturing device driver may be installed and the device tested using the Device Diagnostics tab of the Windows Manufacturing Utility.

10.3.2 Test Time

Performing all available tests takes approximately 9 seconds. This majority of the time is due to the autonegotiation cycles, which take approximately 2 seconds each with 4 link modes in the default settings (all tests enabled). There is always a balance between test time and coverage. If the manufacturer is comfortable with lowering the test coverage, test options can be disabled from the Device Diagnostics tab to reduce the test time.

In particular, some manufacturers may desire to utilize one of the following reduced sets of test options:

- EEPROM verify + Ping test: approximately 1 second per device
- EEPROM verify + Ping test + 100FD Link test: approximately 3 seconds per device.

10.4 Examples

This section details examples of typical Windows Manufacturing Utility operations.

10.4.1 Setting Up an Autoburn Session

This section details how to set up a typical autoburn session.

One Time Setup:

1. In the [EEPROM Contents Editor Tab](#), edit the EEPROM contents per the application requirements. At a minimum, the current (i.e.: first) and max serial number and MAC addresses must be selected specifically for this session.
2. In the [Device Diagnostics Tab](#), select the tests to run for each programming cycle.
3. In the [EEPROM Programmer Tab](#), select the Test Device box.

Starting the Autoburn Operation:

1. In the [EEPROM Programmer Tab](#), select the autoburn checkbox and click "Yes" in the pop-up window.

Running Loop (fully automatic, except for user plugging and unplugging devices)

1. The utility performs the previously selected tests (except EEPROM verify). If the tests fail, the autoburn loop is stopped.
2. The utility programs the EEPROM of the current device.
3. If previously selected, the utility performs an EEPROM verification. If the test fails, the autoburn loop is stopped.
4. The utility prompts the user to remove the device (this device program / test cycle is completed)
5. The utility prompts the user to connect a new "empty EEPROM" device
6. The running loop starts again from step 1.

Stopping the autoburn operation

1. Press cancel at the device plug or unplug prompts.

10.4.2 Sharing the Ping Partner Across Several Assembly Lines

1. Using an unprogrammed device, install the driver on each of the programming PCs as per [Section 10.1.3, "Installing the Manufacturing Utility Device Driver," on page 72](#).
2. Ensure each PC's connected device has a different IP address from all the other network devices in the same subnet.
3. Assign each PC's unprogrammed device a different MAC address:
 - a. Open Device Manager
 - b. Double click on the device to open it's properties
 - c. Click the "Advanced" tab
 - d. Select "Network Address" and assign a different address to each PC. Please note, the lower two bits of the lower nibble of the first byte must be 10b (address is locally assigned, address is not multicast). Refer to [Table 10.13](#) for an example.

Table 10.13 Network Address Assignment Example

PC	IP ADDRESS	MAC ADDRESS
1 st	192.168.1.1	02-00-00-00-00-01
2 nd	192.168.1.2	02-00-00-00-00-02
3 rd	192.168.1.3	02-00-00-00-00-03
...	...	...
n th	192.168.1.n	02-00-00-00-00-0n
Single Ping Partner: 192.168.1.100		

Chapter 11 DOS Utility Suite

The DOS Utility Suite is a set of EXEs that run from DOS and provide support for programming the EEPROM and testing basic functionality in a production/manufacturing environment. The DOS Utility Suite is comprised of the following executables:

- **9500EEP.EXE** - EEPROM programming and contents verification utility
- **9500TEST.EXE** - Basic functionality test utility

Note: The DOS utility runs under DOS only (Windows 98 version 4.10.2222 and MS-DOS version 6.22). It cannot run in a Windows command prompt.

Note: The DOS utility currently supports only the LAN9500/LAN9500i and LAN9500A/LAN9500Ai. A future version will add support for the LAN9512/LAN9514 at a later date.

These executables along with environment requirements are discussed in the following sections.

11.1 Environment Requirements and Limitations

The following sub-sections detail the requirements and limitations for execution of the DOS Utilities.

11.1.1 OS, Processor Mode and Memory

1. The DOS Utilities run under plain DOS only. They cannot run in a Windows command prompt.
2. The DOS Utilities assume the system is running in the x86 processor's real mode at the time of invocation. The utilities will switch to protected mode, stay in protected mode during execution, and switch back to real mode before exit.
3. Due to #2 above, no expanded memory managers (i.e., EMM386.exe) can be running when using the DOS Utilities. Note: extended memory managers (i.e., HIMEM.SYS) are compatible and may be used.
4. The DOS Utilities are stand alone EXEs which are completely unloaded from memory on exit. After the utilities exit, the device will not provide any service whatsoever (i.e., networking access) to the DOS operating system.

11.1.2 USB

1. The DOS Utilities work with EHCI host controllers only.
2. The device can be located on any port of any EHCI host controller, or on any port behind ONE (and only one) HIGH SPEED hub connected to any port on any EHCI host controller.
3. The DOS Utilities will scan the PCI and USB buses starting from their lowest ordinals until a device is found. Once found, the utility stops scanning and uses the found device. Only one (and the first found) device is supported.
4. It is assumed that during execution, the DOS Utilities can take complete control of the USB EHCI host controller (and high speed hub if behind one) that the device is connected to. In particular, the DOS utilities may reset the host controller/hub during initialization and again before exiting, and will not save or restore any context. Note that this will most likely cause other USB devices under the same host controller to stop operating.

11.1.3 Networking Partner

1. To use the 9500TEST.EXE, a network partner is REQUIRED. While some functionality can be supported with a simple link network partner (i.e., a switch), support of ALL 9500TEST.EXE test features, including data passing tests, requires the network partner be another machine with ping echo capabilities.

11.2 9500EEP.EXE

The 9500EEP.EXE utility is used to program or verify the EEPROM contents. The function parameters are provided by either command line switches, files in the same directory as the application, or a combination of both. Upon exit, the utility displays an appropriate status message. It also returns an errcode=0 if it passes, or !=0 if it fails, which is suitable for scripting/batching.

The 9500EEP.EXE command line switches are used as follows:

```
9500EEP (-b <bin_filename> -i <ini_filename>) (-w -r -v) [-M <mac_address>
-S <serial_num> -m <mac_increment> -s <serial_increment> -q -h]
```

Table 11.1 9500EEP.EXE Command Switch Definitions

COMMAND SWITCH	DESCRIPTION
-b <bin_filename>	Uses the binary file <code>bin_filename</code> as a “base” for the EEPROM contents. This option is mutually exclusive with the -i option, but either -b or -i must be present when any operations other than displaying the current EEPROM content of the device (-r) are selected. The MAC address and/or serial in the binary <code>bin_filename</code> can be overwritten using the -m or -s options.
-i <ini_filename>	Uses the options file <code>ini_filename</code> as the “base” for the EEPROM contents. This option is mutually exclusive with the -b option, but either -b or -i must be present when any operations other than displaying the current EEPROM content of the device (-r) are selected. The MAC address and/or serial in the <code>ini_filename</code> can be overwritten using the -m or -s options.
-w	Writes the EEPROM contents specified in the file (+ overrides) to the device and verifies them. The input file specified with the -b or -i will be updated after the verification is complete according to the (mac and serial) increment values. Either -w, -r or -v must be specified.
-r	Reads the current EEPROM contents in the device and writes to the <code>9500.bin</code> binary file. Either -w, -r or -v must be specified.
-v	Reads the current EEPROM contents in the device and verifies against the EEPROM contents specified in the file + overrides. Either -w, -r or -v must be specified.
-M <mac_address>	Overrides the MAC address included in the -b or -i options with the <code>mac_address</code> provided. The supported <code>mac_address</code> formats are “ab:cd:ef:gh:ij:kl”, “ab-cd-ef-gh-ij-kl” or “abcdefghijkl”, where “a” through “l” are hex digits (e.g., 00:80:0F:95:04:00).
-m <mac_address>	Overrides the increment for the MAC address from the <code>ini_filename</code> if the -i option is used, or the default of 0 (no increment) if the -b option is used.
-S <serial_num>	Overrides the serial number address included in the -b or -i with the <code>serial_num</code> provided. The <code>serial_num</code> format is “abcdefghi”, where “a” through “i” are hex digits (e.g., 000950404).
-s <serial_increment>	Overrides the increment for the serial number from the <code>ini_filename</code> if the -i option is used, or the default of 0 (no increment) if the -b option is used.
-q	Quiet operation.
-h	Displays help menu.

11.2.1 Bar Code Scanner Support

For many production lines, the ability to support MAC address programming from a barcode scanned ASCII file is highly desirable. This functionality is supported using the standard DOS utility command switches and simple file concatenation constructs provided in DOS. This functionality is explained below using an example:

A user desires to run the following command:

```
9500EEP -b filename.bin -w -M <mac_addr>-S <serial_num>
```

If the mac address (e.g., 00:11:22:33:44:55) is scanned into the `mac.txt` file and the serial number (e.g., 012345678) is scanned into the `serial.txt` file, the desired run command can be produced with the following DOS command lines:

```
copy /A cmdpfix.txt+mac.txt+s.txt+serial.txt command.bat
```

...where `cmdpfix.txt` is an ASCII file that contains the characters:

```
9500EEP -b filename.bin -w -M
```

...and `s.txt` a file that contains the characters:

```
-S
```

The following `command.bat` file will be created:

```
9500EEP -b filename.bin -w -M 00:11:22:33:44:55 -S 012345678
```

11.2.2 9500EEP ini Format

The items in the 9500EEP ini are self-explanatory based upon their names. An illustrative example `LAN9500Utility.ini` is shown below for reference:

Note: The 9500EEP ini format is identical to the format used by the LAN9500/LAN9500i Windows Manufacturing Utility. See [Section 10.2.2.1.1, "EEPROM Contents Editor Fields," on page 79](#) for descriptions of each field.

```
[MacAddress]
CurrentMacAddress=00:80:0F:95:04:00
MacAddressIncrement=1
MaximumMacAddress=00:80:0F:95:04:99
[SerialNumber]
CurrentSerialNumber=00950400
SerialNumberIncrement=1
MaximumSerialNumber=00950499
[String]
ManufacturerString=SMSC
ManufactureEnable=1
ProductString=LAN9500
ProductEnable=1
ConfigurationString=CFG
ConfigurationEnable=0
InterfaceString=I/F
InterfaceEnable=0
[ID]
VendorID=0424
ProductID=9500
[PollingInterval]
HighSpeed=1
FullSpeed=1
[Power]
BusPower=1
RemoteWakeUpEnable=0
MaxPower=250
```

11.3 9500TEST.EXE

The 9500TEST.EXE DOS Utility is used to perform basic functionality tests on the device in a production manufacturing environment. Its parameters are provided by command line switches. Upon exit, the utility displays an appropriate status message. It also returns an errcode=0 if it passes, or !=0 if it fails, which is suitable for scripting/batching.

Note: IMPORTANT: A network partner device/system is always required to be in the same network segment as 9500TEST.EXE:

- For the link (-l) test options, the network partner can simply be a switch with support for the 4 combinations of link and duplex.
- If the data passing option (-d master) is used, the network partner should be a station in the same network segment with ping echo capabilities.

The 9500TEST.EXE command line switches are used as follows:

```
9500TEST (-l <link_mode> -d <data_mode> -i <ip_address> -I <ip_address>
-V <Vendor_ID> -P <Product_ID> -M <MAC_address> -r -q -h)
```

Note: At least one of the -l or -d options must be specified.

Table 11.2 9500TEST.EXE Command Switch Definitions

COMMAND SWITCH	DESCRIPTION
-l <link_mode>	The utility will attempt to autonegotiate using one of the following restricted negotiation "link mode" capabilities: "10hd" - for 10BASE-T Half-Duplex "10fd" - for 10BASE-T Full-Duplex "100hd" - for 100BASE-TX Half-Duplex "100fd" - for 100BASE-TX Full-Duplex If the "-d ping" option is present, the -l will be performed first. If the "-d iloopback" or "-d eloopback" option is present, the -l option will be ignored.
-d <data_mode>	The utility will perform data passing tests. Valid data_mode values are "ping", "iloopback", or "eloopback". If the option selected is "ping", the utility will send a few packets and expect to receive them back from the partner. If the option selected is "iloopback", the utility will test with internal loopback mode and no partner is required. If the option selected is "eloopback", the utility will send a few packets, expecting to receive the same packets from the external loopback partner. An external loopback plug is required to use this operation.
-i <ip_address>	This option sets the IP address of the LAN9500/LAN9500i device. It is required when the "-d ping" option is present.
-I <ip_address>	This option sets the partner's IP address. It is required when the "-d ping" option is present.
-V <Vendor_ID>	Specifies the Vendor ID to identify the device. This command switch is optional, but if used, must be used with the -P command switch. If not specified, the internal chip ID register will be used to identify the device.
-P <Product_ID>	Specifies the Product ID to identify the device. This command switch is optional, but if used, must be used with the -V command switch. If not specified, the internal chip ID register will be used to identify the device.

Table 11.2 9500TEST.EXE Command Switch Definitions (continued)

COMMAND SWITCH	DESCRIPTION
-M <MAC_address>	Specifies the MAC address to be written into the MAC address register instead of reading it from EEPROM
-r	Reloads the Ethernet EEPROM contents before beginning the test
-q	Quiet operation
-h	Displays help menu

Note: At least one of the -l or -d options must be specified.

APPENDIX A: EEPROM

A.1 EEPROM Format

The EEPROM format for data storage varies by device. The following sections detail the EEPROM contents for each device:

- [LAN9500/LAN9500i EEPROM Format](#)
- [LAN9500A/LAN9500Ai EEPROM Format](#)
- [LAN9512 EEPROM Format](#)
- [LAN9514 EEPROM Format](#)

All EEPROM offsets are given in units of 16-bit word offsets. A length field with a value of zero indicates that the field does not exist in the EEPROM. The device will use the field's HW default value in this case.

Note: For Device Descriptors, the only valid values for the length are 0 and 18.

Note: For Configuration and Interface Descriptors, the only valid values for the length are 0 and 18.

Note: The EEPROM programmer must ensure that if a String Descriptor does not exist in the EEPROM, the referencing descriptor must contain 00h for the respective string index field.

Note: If all string descriptor lengths are zero, then a Language ID will not be supported.

A.1.1 LAN9500/LAN9500i EEPROM Format

Table A.1 LAN9500/LAN9500i EEPROM Format

EEPROM ADDRESS	EEPROM CONTENTS
00h	0xA5
01h	MAC Address [7:0]
02h	MAC Address [15:8]
03h	MAC Address [23:16]
04h	MAC Address [31:24]
05h	MAC Address [39:32]
06h	MAC Address [47:40]
07h	Full-Speed Polling Interval for Interrupt Endpoint
08h	Hi-Speed Polling Interval for Interrupt Endpoint
09h	Configuration Flags
0Ah	Language ID Descriptor [7:0]
0Bh	Language ID Descriptor [15:8]
0Ch	Manufacturer ID String Descriptor Length (bytes)
0Dh	Manufacturer ID String Descriptor EEPROM Word Offset

Table A.1 LAN9500/LAN9500i EEPROM Format (continued)

0Eh	Product Name String Descriptor Length (bytes)
0Fh	Product Name String Descriptor EEPROM Word Offset
10h	Serial Number String Descriptor Length (bytes)
11h	Serial Number String Descriptor EEPROM Word Offset
12h	Configuration String Descriptor Length (bytes)
13h	Configuration String Descriptor Word Offset
14h	Interface String Descriptor Length (bytes)
15h	Interface String Descriptor Word Offset
16h	Hi-Speed Device Descriptor Length (bytes)
17h	Hi-Speed Device Descriptor Word Offset
18h	Hi-Speed Configuration and Interface Descriptor Length (bytes)
19h	Hi-Speed Configuration and Interface Descriptor Word Offset
1Ah	Full-Speed Device Descriptor Length (bytes)
1Bh	Full-Speed Device Descriptor Word Offset
1Ch	Full-Speed Configuration and Interface Descriptor Length (bytes)
1Dh	Full-Speed Configuration and Interface Descriptor Word Offset

Note: EEPROM byte addresses past 1Dh can be used to store data for any purpose.

[Table A.2](#) describes the LAN9500/LAN9500i Configuration Flags.

Table A.2 LAN9500/LAN9500i Configuration Flags Description

BITS	NAME	DESCRIPTION
7:3	RESERVED	-
2	Remote Wakeup Support	0 = The device does not support remote wakeup. 1 = The device supports remote wakeup.
1	RESERVED	-
0	Power Method	0 = The device is bus powered. 1 = The device is self powered.

A.1.2 LAN9500A/LAN9500Ai EEPROM Format

Table A.3 LAN9500A/LAN9500Ai EEPROM Format

EEPROM ADDRESS	EEPROM CONTENTS
00h	0xA5
01h	MAC Address [7:0]
02h	MAC Address [15:8]
03h	MAC Address [23:16]
04h	MAC Address [31:24]
05h	MAC Address [39:32]
06h	MAC Address [47:40]
07h	Full-Speed Polling Interval for Interrupt Endpoint
08h	Hi-Speed Polling Interval for Interrupt Endpoint
09h	Configuration Flags
0Ah	Language ID Descriptor [7:0]
0Bh	Language ID Descriptor [15:8]
0Ch	Manufacturer ID String Descriptor Length (bytes)
0Dh	Manufacturer ID String Descriptor EEPROM Word Offset
0Eh	Product Name String Descriptor Length (bytes)
0Fh	Product Name String Descriptor EEPROM Word Offset
10h	Serial Number String Descriptor Length (bytes)
11h	Serial Number String Descriptor EEPROM Word Offset
12h	Configuration String Descriptor Length (bytes)
13h	Configuration String Descriptor Word Offset
14h	Interface String Descriptor Length (bytes)
15h	Interface String Descriptor Word Offset
16h	Hi-Speed Device Descriptor Length (bytes)
17h	Hi-Speed Device Descriptor Word Offset
18h	Hi-Speed Configuration and Interface Descriptor Length (bytes)
19h	Hi-Speed Configuration and Interface Descriptor Word Offset
1Ah	Full-Speed Device Descriptor Length (bytes)
1Bh	Full-Speed Device Descriptor Word Offset
1Ch	Full-Speed Configuration and Interface Descriptor Length (bytes)
1Dh	Full-Speed Configuration and Interface Descriptor Word Offset

Table A.3 LAN9500A/LAN9500Ai EEPROM Format (continued)

1Eh	GPIO7:0 Wakeup Enables Bit x = 0 -> GPIOx Pin Disabled for Wakeup Use Bit x = 1 -> GPIOx Pin Enabled for Wakeup Use
1Fh	GPIO10:8 Wakeup Enables Bit x = 0 -> GPIO(x+8) Pin Disabled for Wakeup Use Bit x = 1 -> GPIO(x+8) Pin Enabled for Wakeup Use Note: Bits 7:3 Unused
20h	GPIO PME Flags

Note: EEPROM byte addresses past 20h can be used to store data for any purpose.

Table A.4 describes the Configuration Flags.

Table A.4 LAN9500A/LAN9500Ai Configuration Flags

BITS	NAME	DESCRIPTION																	
7:6	RESERVED	-																	
5:4	PHY Boost	This field provides the ability to boost the electrical drive strength of the HS output current to the upstream port. 00 = Normal electrical drive strength 01 = Elevated electrical drive strength (+4% boost) 10 = Elevated electrical drive strength (+8% boost) 11 = Elevated electrical drive strength (+12% boost)																	
3	RESERVED	-																	
2	Remote Wakeup Support	0 = The device does not support remote wakeup. 1 = The device supports remote wakeup.																	
1	LED Select	This bit determines the functionality of external LED pins. <table border="1"> <thead> <tr> <th>BIT VALUE</th><th>PIN NAME</th><th>FUNCTION</th></tr> </thead> <tbody> <tr> <td rowspan="3">0</td><td>nSPD_LED</td><td>Speed Indicator</td></tr> <tr> <td>nLNKA_LED</td><td>Link and Activity Indicator</td></tr> <tr> <td>nFDX_LED</td><td>Full Duplex Link Indicator</td></tr> <tr> <td rowspan="3">1</td><td>nSPD_LED</td><td>Speed Indicator</td></tr> <tr> <td>nLNKA_LED</td><td>Link Indicator</td></tr> <tr> <td>nFDX_LED</td><td>Activity Indicator</td></tr> </tbody> </table>	BIT VALUE	PIN NAME	FUNCTION	0	nSPD_LED	Speed Indicator	nLNKA_LED	Link and Activity Indicator	nFDX_LED	Full Duplex Link Indicator	1	nSPD_LED	Speed Indicator	nLNKA_LED	Link Indicator	nFDX_LED	Activity Indicator
BIT VALUE	PIN NAME	FUNCTION																	
0	nSPD_LED	Speed Indicator																	
	nLNKA_LED	Link and Activity Indicator																	
	nFDX_LED	Full Duplex Link Indicator																	
1	nSPD_LED	Speed Indicator																	
	nLNKA_LED	Link Indicator																	
	nFDX_LED	Activity Indicator																	
0	Power Method	0 = The device is bus powered. 1 = The device is self powered.																	

Table A.5 describes the GPIO PME flags.

Table A.5 LAN9500A/LAN9500Ai GPIO PME Flags

BIT	NAME	DESCRIPTION
7	GPIO PME Enable	Setting this bit enables the assertion of the GPIO0 or GPIO8 pin, as a result of a Wakeup (GPIO) pin, Magic Packet, or PHY Link Up. The host processor may use the GPIO0/GPIO8 pin to asynchronously wake up, in a manner analogous to a PCI PME pin. GPIO0 signals the event when operating in Internal PHY mode, while GPIO8 signals the event when operating in External PHY mode. Internal or External PHY mode of operation is dictated by the PHY_SEL pin. 0 = The device does not support GPIO PME signaling. 1 = The device supports GPIO PME signaling. Note: When this bit is 0, the remaining GPIO PME parameters in this flag byte are ignored.
6	GPIO PME Configuration	This bit selects whether the GPIO PME is signaled on the GPIO pin as a level or a pulse. If pulse is selected, the duration of the pulse is determined by the setting of the GPIO PME Length bit of this flag byte. The level of the signal or the polarity of the pulse is determined by the GPIO PME Polarity bit of this flag byte. 0 = GPIO PME is signaled via a level. 1 = GPIO PME is signaled via a pulse. Note: If GPIO PME Enable is 0, this bit is ignored.
5	GPIO PME Length	When the GPIO PME Configuration bit of this flag byte indicates that the GPIO PME is signaled by a pulse on the GPIO pin, this bit determines the duration of the pulse. 0 = GPIO PME pulse length is 1.5 mS. 1 = GPIO PME pulse length is 150 mS. Note: If GPIO PME Enable is 0, this bit is ignored.
4	GPIO PME Polarity	Specifies the level of the signal or the polarity of the pulse used for GPIO PME signaling. 0 = GPIO PME signaling polarity is low. 1 = GPIO PME signaling polarity is high. Note: If GPIO PME Enable is 0, this bit is ignored.
3	GPIO PME Buffer Type	This bit selects the output buffer type for GPIO0/GPIO8. 0 = Open drain driver / open source 1 = Push-Pull driver Note: Buffer Type = 0, Polarity = 0 implies Open Drain Buffer Type = 0, Polarity = 1 implies Open Source Note: If GPIO PME Enable is 0, this bit is ignored.
2	GPIO PME WOL Select	Three types of wakeup events are supported; Magic Packet, PHY Link Up, and Wakeup Pin(s) assertion. Wakeup Pin(s) are selected via the GPIO Wakeup Enables specified in bytes 1Eh and 1Fh of the EEPROM. This bit selects whether Magic packet or Link Up wakeup events are supported. 0 = Magic packet wakeup supported. 1 = PHY linkup wakeup supported. (not supported in External PHY mode) Note: If GPIO PME Enable is 0, this bit is ignored.

Table A.5 LAN9500A/LAN9500Ai GPIO PME Flags (continued)

BIT	NAME	DESCRIPTION
1	GPIO10 Detection Select	This bit selects the detection mode for GPIO10 when operating in PME mode. In PME mode, GPIO10 is usable in both Internal and External PHY mode as a wakeup pin. This parameter defines whether the wakeup should occur on an active high or active low signal. 0 = Active-low detection for GPIO10 1 = Active-high detection for GPIO10 Note: If GPIO PME Enable is 0, this bit is ignored.
0	RESERVED	-

A.1.3 LAN9512 EEPROM Format

Table A.6 LAN9512 EEPROM Format

EEPROM ADDRESS	EEPROM CONTENTS
00h	0xA5
01h	MAC Address [7:0]
02h	MAC Address [15:8]
03h	MAC Address [23:16]
04h	MAC Address [31:24]
05h	MAC Address [39:32]
06h	MAC Address [47:40]
07h	Full-Speed Polling Interval for Interrupt Endpoint
08h	Hi-Speed Polling Interval for Interrupt Endpoint
09h	Configuration Flags
0Ah	Language ID Descriptor [7:0]
0Bh	Language ID Descriptor [15:8]
0Ch	Manufacturer ID String Descriptor Length (bytes)
0Dh	Manufacturer ID String Descriptor EEPROM Word Offset
0Eh	Product Name String Descriptor Length (bytes)
0Fh	Product Name String Descriptor EEPROM Word Offset
10h	Serial Number String Descriptor Length (bytes)
11h	Serial Number String Descriptor EEPROM Word Offset
12h	Configuration String Descriptor Length (bytes)
13h	Configuration String Descriptor Word Offset

Table A.6 LAN9512 EEPROM Format (continued)

14h	Interface String Descriptor Length (bytes)
15h	Interface String Descriptor Word Offset
16h	Hi-Speed Device Descriptor Length (bytes)
17h	Hi-Speed Device Descriptor Word Offset
18h	Hi-Speed Configuration and Interface Descriptor Length (bytes)
19h	Hi-Speed Configuration and Interface Descriptor Word Offset
1Ah	Full-Speed Device Descriptor Length (bytes)
1Bh	Full-Speed Device Descriptor Word Offset
1Ch	Full-Speed Configuration and Interface Descriptor Length (bytes)
1Dh	Full-Speed Configuration and Interface Descriptor Word Offset
1Eh-1Fh	RESERVED
20h	Vendor ID LSB Register (VIDL)
21h	Vendor ID MSB Register (VIDM)
22h	Product ID LSB Register (PIDL)
23h	Product ID MSB Register (PIDM)
24h	Device ID LSB Register (DIDL)
25h	Device ID MSB Register (DIDM)
26h	Config Data Byte 1 Register (CFG1)
27h	Config Data Byte 2 Register (CFG2)
28h	Config Data Byte 3 Register (CFG3)
29h	Non-Removable Devices Register (NRD)
2Ah	Port Disable (Self) Register (PDS)
2Bh	Port Disable (Bus) Register (PDB)
2Ch	Max Power (Self) Register (MAXPS)
2Dh	Max Power (Bus) Register (MAXPB)
2Eh	Hub Controller Max Current (Self) Register (HCMCS)
2Fh	Hub Controller Max Current (Bus) Register (HCMCB)
30h	Power-on Time Register (PWRT)
31h	Boost_Up Register (BOOSTUP)
32h	RESERVED
33h	Boost_3:2 Register (BOOST32)
34h	RESERVED
35h	Port Swap Register (PRTSP)

Table A.6 LAN9512 EEPROM Format (continued)

36h	Port Remap 12 Register (PRTR12)
37h	Port Remap 3 Register (PRTR3)
38h	RESERVED
39h	Status/Command Register (STCD)

Note: EEPROM byte addresses past 39h can be used to store data for any purpose.

[Table A.7](#) describes the LAN9512 Configuration Flags.

Table A.7 LAN9512 Configuration Flags Description

BITS	NAME	DESCRIPTION
7:3	RESERVED	00000b
2	Remote Wakeup Support	0 = The device does not support remote wakeup. 1 = The device supports remote wakeup.
1	RESERVED	0b
0	Power Method	0 = The device Controller is bus powered. 1 = The device Controller is self powered.

A.1.4 LAN9514 EEPROM Format

Table A.8 LAN9514 EEPROM Format

EEPROM ADDRESS	EEPROM CONTENTS
00h	0xA5
01h	MAC Address [7:0]
02h	MAC Address [15:8]
03h	MAC Address [23:16]
04h	MAC Address [31:24]
05h	MAC Address [39:32]
06h	MAC Address [47:40]
07h	Full-Speed Polling Interval for Interrupt Endpoint
08h	Hi-Speed Polling Interval for Interrupt Endpoint
09h	Configuration Flags
0Ah	Language ID Descriptor [7:0]
0Bh	Language ID Descriptor [15:8]
0Ch	Manufacturer ID String Descriptor Length (bytes)

Table A.8 LAN9514 EEPROM Format (continued)

0Dh	Manufacturer ID String Descriptor EEPROM Word Offset
0Eh	Product Name String Descriptor Length (bytes)
0Fh	Product Name String Descriptor EEPROM Word Offset
10h	Serial Number String Descriptor Length (bytes)
11h	Serial Number String Descriptor EEPROM Word Offset
12h	Configuration String Descriptor Length (bytes)
13h	Configuration String Descriptor Word Offset
14h	Interface String Descriptor Length (bytes)
15h	Interface String Descriptor Word Offset
16h	Hi-Speed Device Descriptor Length (bytes)
17h	Hi-Speed Device Descriptor Word Offset
18h	Hi-Speed Configuration and Interface Descriptor Length (bytes)
19h	Hi-Speed Configuration and Interface Descriptor Word Offset
1Ah	Full-Speed Device Descriptor Length (bytes)
1Bh	Full-Speed Device Descriptor Word Offset
1Ch	Full-Speed Configuration and Interface Descriptor Length (bytes)
1Dh	Full-Speed Configuration and Interface Descriptor Word Offset
1Eh-1Fh	RESERVED
20h	Vendor ID LSB Register (VIDL)
21h	Vendor ID MSB Register (VIDM)
22h	Product ID LSB Register (PIDL)
23h	Product ID MSB Register (PIDM)
24h	Device ID LSB Register (DIDL)
25h	Device ID MSB Register (DIDM)
26h	Config Data Byte 1 Register (CFG1)
27h	Config Data Byte 2 Register (CFG2)
28h	Config Data Byte 3 Register (CFG3)
29h	Non-Removable Devices Register (NRD)
2Ah	Port Disable (Self) Register (PDS)
2Bh	Port Disable (Bus) Register (PDB)
2Ch	Max Power (Self) Register (MAXPS)
2Dh	Max Power (Bus) Register (MAXPB)
2Eh	Hub Controller Max Current (Self) Register (HCMCS)

Table A.8 LAN9514 EEPROM Format (continued)

2Fh	Hub Controller Max Current (Bus) Register (HCMCB)
30h	Power-on Time Register (PWRT)
31h	Boost_Up Register (BOOSTUP)
32h	Boost_5 Register (BOOST5)
33h	Boost_4:2 Register (BOOST42)
34h	RESERVED
35h	Port Swap Register (PRTSP)
36h	Port Remap 12 Register (PRTR12)
37h	Port Remap 34 Register (PRTR34)
38h	Port Remap 5 Register (PRTR5)
39h	Status/Command Register (STCD)

Note: EEPROM byte addresses past 39h can be used to store data for any purpose.

[Table A.9](#) describes the LAN9514 Configuration Flags.

Table A.9 LAN9514 Configuration Flags Description

BIT	NAME	DESCRIPTION
7:3	RESERVED	00000b
2	Remote Wakeup Support	0 = The device does not support remote wakeup. 1 = The device supports remote wakeup.
1	RESERVED	0b
0	Power Method	0 = The device Controller is bus powered. 1 = The device Controller is self powered.

A.2 EEPROM Programming

EEPROM programming is performed differently depending on the OS used. This chapter details the various methods of programming the device EEPROM:

- [EEPROM Programming in Windows](#)
- [EEPROM Programming in DOS](#)
- [EEPROM Programming in Linux](#)
- [EEPROM Programming in Windows CE](#)

A.2.1 EEPROM Programming in Windows

EEPROM programming in Windows is accomplished via the Windows Manufacturing Utility. Refer to [Chapter 10, "Windows Manufacturing Utility," on page 70](#) for detailed information on the usage of this tool.

A.2.2 EEPROM Programming in DOS

EEPROM programming in DOS is accomplished via the DOS Utility Suite. Refer to [Chapter 11, "DOS Utility Suite," on page 89](#) for detailed information on the usage of these tools.

A.2.3 EEPROM Programming in Linux

In Linux, there are two available methods to read and modify the device's EEPROM:

- [ethtool](#)
- [cmd9500](#)

A.2.3.1 ethtool

Using standard "ethtool" commands, the device EEPROM can be read or written.

To read the EEPROM contents, use ethtool as follows:

```
ethtool -e|--eeprom-dump ethX [raw on|off] [offset N] [length N]
```

The "-e" option retrieves and prints an EEPROM dump for the specified device. When `raw` is enabled, the raw EEPROM data is dumped to stdout. The `length` and `offset` parameters allow dumping of only the specified portions of the EEPROM. By default, the entire EEPROM is dumped.

To write the EEPROM contents, use ethtool as follows:

```
ethtool -E|--change-eeprom ethX [magic N] [offset N] [value N]
```

The "-E" option writes to the EEPROM of the specified device. The `offset` and `value` parameters specify the byte to be written and its new value, respectively. The `magic` parameter is used to provide the device-specific magic key (0x9500 for the LAN9500/LAN9500i), preventing accidental writing to the EEPROM.

A.2.3.2 cmd9500

Using the SMSC utility "cmd9500", the device EEPROM can be read or written. The cmd9500 utility is used as follows:

```
./cmd9500 [-h] [-e adaptername] [-c command] [-a address] [-d data] [-f filename]
```

The cmd9500 options are defined in [Table A.10](#).

Table A.10 cmd9500 Option Descriptions

OPTION	DESCRIPTION
-h	Displays command usage information, other options are ignored.
-e	Specifies the adapter name (eth0,eth1...). If no adapter name is specified, cmd9500 will attempt to auto-detect.
-c	Specifies the command code: DUMP_EEPROM = dumps the EEPROM contents READ_EEPROM = reads a value from the EEPROM at the specified address WRITE_EEPROM = writes a value to the EEPROM at the specified address WRITE_FILE_TO_EEPROM = writes the specified binary file contents (-f) into EEPROM READ_EEPROM_TO_FILE = reads EEPROM and writes contents to the specified binary file (-f) VERIFY_EEPROM_WITH_FILE = compares EEPROM contents with the specified binary file (-f)
-a	The EEPROM address to be read/written
-d	Data to be written in hexadecimal format
-f	Specifies the file name used for WRITE_FILE_TO_EEPROM, READ_EEPROM_TO_FILE, and VERIFY_EEPROM_WITH_FILE commands

A.2.4 EEPROM Programming in Windows CE

In Windows CE, the device EEPROM can be read or written using the CE9500EEP.exe SMSC utility. The CE9500EEP utility can be downloaded from the E-Services section of the SMSC website. To create an E-Services account, visit the SMSC E-Services webpage: <https://www2.smc.com/main.nsf>.

To use the CE9500EEP utility, double click the CE9500EEP icon or run it at the command line. When run, the following options are available:

1. program EEPROM
2. program & Verify EEPROM
3. Read EEPROM
- Q. Exit

Selecting option 1 or 2 will write the EEPROM. For these options, the user must provide the EEPROM contents in a binary file located in the \Windows\ directory. The file name must be 9500.bin.

Selecting option 3, "Read EEPROM", will read the EEPROM contents to a binary file, 9500.bin, located in the \Windows\ directory.

Selecting Q, will exit the application.

For the latest CE9500EEP information, refer to the release readme file.

A.2.4.1 Including CE9500eep.exe in a CE Image

The CE9500eep.exe file can be included in a CE image via two methods. If the CE target supports USB mass storage devices, a USB flash drive can be used to copy the CE9500eep.exe file to the \Windows\ folder of the image while the CE device is actually running. The same method ("runtime") is applicable to any other means by which files can be copied into the running CE device.

For those CE targets that do not support USB mass storage devices or any other means of copying files into the running CE device (and therefore require the utility to be built into the image), the following method should be used:

1. Copy the CE9500eep.exe file to the %_TARGETPLATROOT%\FILES\ folder.
2. Edit the Platform.bib file in the %_TARGETPLATROOT%\FILES\ folder to include the CE9500eep.exe file into the target image, as shown in [Figure A.1](#).



Figure A.1 Inserting CE9500eep.exe into Platform.bib

3. Build the CE Image:

For CE 5.0, Select "Build OS" -> "Sysgen" from Platform Builder 5.0. This will generate the CE image, NK.BIN, which will include the CE9500eep.exe utility.

For CE 6.0, Platform Builder is a plug-in under the Visual Studio 2005. Select "Build" -> "Build solution". This will generate the CE image, NK.BIN, which will include the CE9500eep.exe utility.

A.3 EEPROM-less Design

Reducing BOM costs has prompted many designers to utilize the LAN9500/LAN9500i without EEPROM. The purpose of this appendix section is to:

- Determine if a particular design may utilize the LAN9500/LAN9500i without EEPROM
- Detail the implications of utilizing the LAN9500/LAN9500i without EEPROM
- Explain alternative configuration requirements when utilizing the LAN9500/LAN9500i without EEPROM

Note: Unless otherwise noted, the contents of this section are specifically written for the LAN9500/LAN9500i, LAN9512, and LAN9514 devices. While the LAN9500A device can operate in an EEPROM-less environment in the same way as the previously mentioned devices, the LAN9500A provides enhanced support for EEPROM-less operation. If using a LAN9500A, refer to the LAN9500A Datasheet for additional details on the enhanced EEPROM-less support.

A.3.1 EEPROM Item Defaults and Changeability

[Table A.11](#) details the items stored in the EEPROM. For each individual item, the default contents and the item's changeability when EEPROM is not present are shown. A "Change Required" column provides an overview of whether the default value is suitable when there is no EEPROM.

Table A.11 EEPROM Items Defaults and Changeability

Item	Change Possible	Change Required	Default
Mac Address	Yes	Yes	FF-FF-FF-FF-FF-FF
Full Speed Polling Interval	Note A.1	Note A.3	1ms
High Speed Polling Interval	Note A.1	Note A.3	1ms
Configuration Flags	Note A.2	Note A.3	Bus powered, remote wakeup supported
Maximum Power	No	Note A.3	500mA if bus powered, 2mA if self powered
Vendor ID	No	Note A.3	424h
Product ID	No	Note A.3	9500h
Language ID	No	Note A.3	Not present
Manufacturer String	No	Note A.3	Not present
Product String	No	Note A.3	Not present
Serial Number String	No	Note A.3	Not present
Configuration String	No	Note A.3	Not present
Interface String	No	Note A.3	Not present
Full Speed Device Descriptor	No	Note A.3	No strings, 424/9500 Vendor/Product ID
Full Speed Config Descriptor	No	Note A.3	No string, others per Config Flags & MaxPow
High Speed Device Descriptor	No	Note A.3	No strings, 424/9500 Vendor/Product ID
High Speed Config Descriptor	No	Note A.3	No string, others per Config Flags & MaxPow

Note A.1 The information on the endpoint descriptor itself cannot be changed. However, the polling interval used when submitting interrupt transactions may be changed depending on the OS. Refer to [Section A.3.4.2, "Overriding USB Behavioral Parameters"](#) for additional information.

Note A.2 Configuration Flags can be overwritten by PWR_SEL and RMT_WKP straps. Remote wake up behavior can be modified in some operating systems despite the value returned by the Config descriptor.

Note A.3 Depends on the particular device application.

A.3.2 Determining EEPROM Necessity

When utilizing the device without EEPROM, the following limitations are established:

- Standard USB requests will be answered with the non-changeable default values detailed in [Table A.11](#).
- Items that are listed as changeable in [Table A.11](#) can be changed via software only.

If the above limitations are acceptable to the designer, then the device may be used without EEPROM. Otherwise, the designer must use EEPROM.

Note: Using the device without EEPROM requires additional software development effort by the designer to modify the changeable items in [Table A.11](#) from their default values. Details on how to change these values via software are provided in [Section A.3.4.1, "Overriding the MAC Address"](#) and [Section A.3.4.2, "Overriding USB Behavioral Parameters"](#).

A.3.3 Utilizing Unchangeable Defaults

As detailed in [Table A.11](#), there are many parameters that cannot be changed if the device is used without EEPROM. The following sub-sections detail typical examples of these cases.

Note: These examples are intended to illustrate the main limitations of EEPROM-less design. They are not exhaustive, and do not represent all possible design cases.

A.3.3.1 Example 1 - Missing Manufacturer and Product Strings

An EEPROM-less design will not contain any strings, including Manufacturer and Product strings.

Several implementations of USB System software (i.e. Windows XP/Vista, Mac OSX, etc.) display information about devices based on Manufacturer/Product strings as part of their plug and play discovery and/or bus browsing applications. Without EEPROM, the design will not be able to provide Company or Product specific names in these scenarios (PnP discovery, USB device browsers).

A.3.3.2 Example 2 - Missing Serial Number

An EEPROM-less design will not contain any strings, including the Serial Number string.

In many cases it is desirable to associate software configuration items (i.e.: manually assigned IP addresses, wake up configurations, etc.) with a particular device if the device is moved from one USB port to another, and/or more than one LAN9500/LAN9500i is used simultaneously in the same system (connected to different USB ports). Environments such as Windows use the serial number string in order to uniquely identify a device in these conditions. Using an EEPROM-less configuration will provide an unsuccessful experience in this scenario.

A.3.3.3 Example 3 - Maximum Power

An EEPROM-less design will use a Configuration descriptor that specifies maximum power consumption of 500 mA for a bus powered configuration, or 2 mA for a self powered configuration.

In most cases these values are acceptable since they specify the maximum allowable power for a bus powered device (the device is nevertheless allowed to consume less); or the minimum for a self powered device (value specifies power drawn from the bus, but self powered devices will most likely take all their power from their self-source rather than the bus). If other values are required for a particular application, EEPROM is required.

A.3.4 Modification of Changeable Defaults

As shown in [Table A.11](#), the MAC address MUST always be changed. This is the only item that must be changed in all cases.

A.3.4.1 Overriding the MAC Address

As per Ethernet 802.3 standards, the permanent MAC address for every device worldwide must be unique, and the MAC address for every device in a network (even if locally assigned instead of using its permanent address) must be unique. The permanent MAC Address for the device is provided from EEPROM. However, if a design does not utilize EEPROM, the default value is FF-FF-FF-FF-FF-FF. This value must be changed for two reasons:

- It is not unique
- It is the broadcast address and therefore invalid as a device's MAC address

During runtime operation, the MAC Address resides in two registers: ADDR_HI and ADDR_LO. When an EEPROM is present, these registers are automatically loaded from the EEPROM during a power up or reload operation. If an EEPROM is not present, changing these two registers during the device's software initialization allows the device to properly operate. Therefore, designs which can load the MAC address registers via software at device initialization can specify a unique MAC Address, alleviating conflicts.

Software based MAC address overrides can be accomplished via the following methods:

- [MAC Address Overrides using Custom Software](#)
- [MAC Address Overrides using Off-The-Shelf Software Solutions](#)
- [MAC Address Overrides using Descriptor RAM](#)
- [MAC Address Overrides using Modified LAN9500/LAN9500i Drivers](#)

A.3.4.1.1 MAC ADDRESS OVERRIDES USING CUSTOM SOFTWARE

In cases where the designer is utilizing custom software or has access to modify the source code of the USB/Network software used in the host system, complete control over the device MAC address is possible. This is accomplished by configuring the USB device to allow register writes, overwriting the MAC address registers before any other data is sent or received.

A.3.4.1.2 MAC ADDRESS OVERRIDES USING OFF-THE-SHELF SOFTWARE SOLUTIONS

When using an off-the-shelf software solution, the designer may not have access to modify the USB/Network Host software, the LAN9500/LAN9500i driver, or both. In these cases, careful evaluation is required to determine if the chosen software solution provides hooks to override the MAC address and evaluate how the solution will fit within the application implementation.

The following sub-sections detail examples of utilizing off-the-self software solutions under various operating systems.

Note: Additional details regarding specific parameters (registry, info-plist, insmod, etc.) can be found in the respective LAN9500/LAN9500i driver documentation.

Windows XP/Vista

Windows provides a "Network Address" registry parameter. If this key is present in the registry instance of the LAN9500/LAN9500i device (this can be easily done through the advanced properties of the device), the driver reads the key and uses it to populate the MAC address registers during the driver initialization. This solution requires the designer to generate a unique registry image for each system or alternatively read the information from elsewhere and populate the registry entry before the device driver loads.

Mac OSX

The LAN9500/LAN9500i Mac OSX driver does not currently support MAC address overrides. Support may be added in a future release by use of an info-plist parameter or other existing Mac OSX infrastructure.

Linux

The LAN9500/LAN9500i driver provides a load time option to specify the MAC address of the device. This is achieved by using the `mac_addr_hi16` and `mac_addr_lo32` insmod parameters. Note that these parameters override all instances of the device, so they cannot be used if more than one device is in use simultaneously. Similar to the Windows XP/Vista case, this solution requires the generation of a unique loading script for each system at image burn time, or alternatively, the information can be read from elsewhere and the LAN9500/LAN9500i load script modified before the device driver is loaded. Additionally, because the LAN9500/LAN9500i Linux driver source code is open, a custom solution may be devised. Refer to [Section A.3.4.1.4, "MAC Address Overrides using Modified LAN9500/LAN9500i Drivers"](#) for additional information.

Windows CE

Overrides are supported in the same way as the Windows XP/Vista driver. Alternatively, WinCE source code is provided by SMSC, allowing designers to create their own solution. Refer to [Section A.3.4.1.4, "MAC Address Overrides using Modified LAN9500/LAN9500i Drivers"](#) for additional information.

A.3.4.1.3 MAC ADDRESS OVERRIDES USING DESCRIPTOR RAM

Beginning in April 2009, SMSC added a new standardized method to the regular release of Windows, Windows CE, Linux, and PXE drivers which allows pre-OS boot software (i.e., PC Bios) to pass MAC address information to the driver.

A special area inside of the device (descriptor RAM) may be filled by the pre-OS boot software with the format shown in [Table A.12](#). The definitions of each byte are detailed in [Table A.13](#).

Table A.12 Descriptor RAM Contents

Address	Byte 3	Byte 2	Byte 1	Byte 0
0x0	's' (0x73)	'm' (0x6D)	's' (0x73)	'c' (0x63)
0x4	ADDRL			
0x8	ADDRH			
0xC	~CRC16		CRC16	

Table A.13 Descriptor RAM Definitions

Item	Description
First Byte (Address0x0)	The first byte contains the "smc" signature (0x736D7363).
ADDRL	Contains the low address bytes of the desired Ethernet address. This value will be written to the ADDR register.
ADDRH	Contains the high address bytes of the desired Ethernet address. This value will be written to the ADDRH register.
CRC16	Contains the 16-bit "IP-style" CRC of the array of bytes from address 0x00 to 0x0B.
~CRC16	Contains the bitwise-xor of the CRC16 and 0xFFFF.

Note: For example, if the desired Ethernet address is 00:12:34:56:78:9A, ADDR should be 0x56341200 and ADDRH should be 0x00009A78.



Software User Manual

SMSC standard drivers for Windows, Windows CE, Linux and PXE released after April 15th, 2009 will check for the presence of this information. If found, and the CRCs are validated, the driver will use this information to populate the MAC address registers of the device.

IMPORTANT NOTES:

- In order to program the descriptor RAM, a set of operations with the registers of the LAN9500 device need to be performed. Sample code illustrating these registers operations, as well as the CRC16 generation routine, is available via the E-Services portion of the SMSC website. Refer to <https://www2.smc.com/main.nsf> for additional information.
- Writing to the device registers implies that the preOS-boot environment being used is able to perform USB enumeration (to find the device) and later perform USB vendor command transfers on the device.
- The LAN9500A provides an enhanced variation of EEPROM-less operation for pre-OS boot environments that allows customization of not only the MAC address, but all of the USB descriptors as well. Drivers released after April 15th, 2009 fully support this LAN9500A mode of operation. Refer to the LAN9500A specification for details.

A.3.4.1.4 MAC ADDRESS OVERRIDES USING MODIFIED LAN9500/LAN9500i DRIVERS

As mentioned in [Section A.3.4.1.2, "MAC Address Overrides using Off-The-Shelf Software Solutions"](#) previously, there are several off-the-shelf drivers for which the LAN9500/LAN9500i driver's source code is available. In these cases, if solutions such as generating a unique flash image for each system and/or modifying registry/load scripts before driver initialization are not optimal, it is also possible to modify the drivers directly.

In order to show the possibilities of what can be accomplished via driver customization, an example is provided below. Please note that this example does not represent an exhaustive solution for all applications. Because each particular application/implementation will be unique, this example is intended to present general guidelines only.

Example:

Implementing a function or separate module that reads the MAC Address data on demand

To correctly assign the MAC address to the device, it must be populated not only before the device starts receiving and sending data, but also before upper layers of the networking stack are notified. If the MAC address has not been communicated to the driver via standard registry keys or load time parameters, then it must be communicated to (or procured by) the driver in some other way during early initialization.

One solution to this problem is a custom written function that is called early-on during driver initialization to fetch the MAC address from its specific location in storage (i.e. flash image address). If the designer is not concerned with modularity and does not break any host software rules, the entire function can be completely embedded within the LAN9500/LAN9500i driver. This function can then be called very early-on, during the driver initialization, and have all the MAC address information readily available by the time it is required.

While achieving the same result, a more elegant solution is to create a separate module that is loaded before the LAN9500/LAN9500i driver (or loaded on demand by the LAN9500/LAN9500i driver). This module would retrieve the MAC address information from its specific location in storage (i.e. flash image address) and expose an interface (i.e.: exported function, IOCTL, etc.) for the LAN9500/LAN9500i driver to call when the MAC address information is needed.

A.3.4.2 Overriding USB Behavioral Parameters

While the MAC address value MUST be changed from its default in all cases, there are other parameters that depend on the application. USB descriptor values are fixed once the decision is made to not use an EEPROM and the straps are selected. However, some USB host software behaviors which are set through the USB descriptors can, in some cases, still be modified on-the-fly.

Similar to the case of the MAC address discussed in [Section A.3.4.1, "Overriding the MAC Address"](#), if custom software is created, the designer has the control to modify almost any desired USB behavior.

Similar to what was discussed in [Section A.3.4.1.4, "MAC Address Overrides using Modified LAN9500/LAN9500i Drivers"](#), if the host software allows modification, custom parameters can be stored elsewhere in the system (flash image address) and procured by the driver at initialization time.

When utilizing off-the-shelf OS/driver software, the following items should be noted:

Interrupt Polling Interval Override

Most SMSC drivers do not use the LAN9500/LAN9500i interrupt endpoint in its default configuration, but allow for alternative modes that do. If use of the interrupt endpoint is required and the 1ms default polling interval the device uses for an EEPROM-less configuration is not optimal, it is possible to override it at initialization time in Mac OSX and Linux, but not in Windows.

Other USB Behavioral Parameter Overrides

In addition to the polling interval, some USB Host software environments may allow the designer to disregard information read from the standard descriptors and override them with custom information. Please check with the specific software environment used to verify if this option is available.

A.3.5 Typical Cases

Two typical LAN9500/LAN9500i design cases are detailed below. For each, the ability to function without EEPROM is assessed.

Stand alone USB-to-Ethernet dongle for the PC environment

The device is disassociated (and typically shipped separately) from the system, so a MAC address must be assigned by the device itself with no system interaction. Also, it is desirable to be able to use several devices simultaneously, or maintain per device parameters when moving across different USB ports in the same system. Therefore a serial number string is needed. Finally USB guidelines require the manufacturer to use its own USB Manufacturer ID, not SMSC's. Given all the above, an attached EEPROM is required.

Soldered down LAN9500/LAN9500i with flash space available for LAN9500/LAN9500i Information which can be read at initialization time & can function with unchangeable defaults

Since the device is soldered down there is a one-to-one correspondence between the host and the LAN9500/LAN9500i as well as the specific USB port it is connected to. There is storage in flash for the MAC address and other device unique parameters and the design permits reading them early at initialization time. Finally, the design is able to function properly with unchangeable parameters. Therefore, there are no limitations preventing the usage of an EEPROM-less design.

APPENDIX B: Customer Requirements

This appendix details various customer requirements for device utilization, which include the following:

- [MAC Address](#)
- [USB Vendor ID and Logo](#)
- [Serial Number](#)
- [WHQL Logo](#)

B.1 MAC Address

If an organization manufactures or plans to manufacture products using ISO/IEC 8802 standards, it must obtain an Organizationally Unique Identifier (OUI) from the Institute of Electrical and Electronics Engineers, Inc. (IEEE). The three-octet OUI can be used to generate Universal LAN MAC addresses and Protocol Identifiers, per the ANSI/IEEE 802 standard, for use in Local and Metropolitan Area Network applications.

The IEEE has been designated by the ISO Council to act as the single, world-wide registration authority for the implementation of International Standards in the ISO/IEC 8802 series. For further details contact:

IEEE Registration Authority
IEEE Standards Department
445 Hoes Lane
Piscataway NJ 08854

Phone: (732) 465-6481
Fax: (732) 562-1571
E-mail: IEEE.Registration.Authority@ieee.org
Web: <http://standards.ieee.org/regauth/oui/index.shtml>

B.2 USB Vendor ID and Logo

Obtaining a USB vendor ID is recommended for all applications. For information on obtaining a Vendor ID, refer to the following USB organization website link:

<http://www.usb.org/developers/vendor/>

Note: The USB-IF logos may be used only in conjunction with products that have passed USB-IF compliance testing and are currently on the integrators list. This requires that the company be assigned a USB Vendor ID number.

Although SMSC silicon has passed all compliance tests, final stand alone products must also go through the compliance testing process. The company must have a unique Vendor ID to satisfy the USB-IF requirements. However, if the company decides not to use the USB logo for stand alone devices or if the product is a soldered down device, the company may use the SMSC VID and PID.

B.3 Serial Number

In many cases it is desirable to associate software configuration items (i.e., manually assigned IP addresses, wake up configurations, etc.) with a particular device. This allows the device to be moved from one USB port to another and/or be used simultaneously with more than one SMSC device in the same system (connected to different USB ports). Environments such as Windows use the serial number string in order to uniquely identify a device in these conditions. Thus, a unique serial number string is required for Windows to avoid reinstallation of the driver each time the device is plugged into a different port.

Therefore, if the customer wants to use the SMSC VID and PID, SMSC requires the serial number to be equal to the MAC Address.

B.4 WHQL Logo

Refer to the following frequently asked questions regarding WHQL logo usage:

Q: Some of our customers will have a different product name on their boxes (sales/marketing perspective), but don't care that Windows UI will show SMSC's company and product name. They will use our PnP ID (USB-IF assigned Vendor ID and SMSC assigned Product ID). Therefore, the standard driver and INF files can be used with no further modifications. What is the standard procedure in this case?

A: If these customers will display the WHQL logo on the package or device, they must either obtain WHQL logo compliance themselves, or complete a reseller agreement with SMSC. By completing the reseller agreement, they will have signed all the legal agreements regulating the use of the WHQL logo. No WLK testing or upload of logs will be required.

Note: Reseller agreements can only be made with customers that have a Winqual account.

Q: Some of our customers are willing to use our unmodified driver binary, but want to use their PnP ID (customer's USB-IF assigned Vendor ID and customer's assigned Product ID) and/or reflect their Company and Product name in Windows' UI. Because of this, these customers need to modify the INF. What is the standard procedure in this case?

A: If a change is made to an INF that is already part of a logo-ed submission, the customer can utilize the Microsoft Acceptable Device and Driver Update Policy (Policy-0015). This policy, referred to commonly as Driver Update Acceptable (DUA), allows for certain changes in the INF and driver package without retesting. However, a submission must be made. Once the submission is complete, the submission number with the modified INF may be shipped.

Note: Submissions can only be made by customers that have a Winqual account.