LAN7500/LAN7500i

Software User Manual

Revision 1.1 (07-20-10)

0.1 Revision History

Table 0.1 Revision History

REVISION LEVEL AND DATE	SECTION/FIGURE/ENTRY	CORRECTION
Rev. 1.1 (07-20-10)	Section 6.1, "Linux Driver Installation," on page 57	Updated installation procedures and Figure 6.1, Figure 6.2, and Figure 6.3
	Section 6.2, "Linux Driver Uninstallation," on page 62	Updated uninstallation procedures and Figure 6.9.
	Section 9.3.1, "insmod," on page 81	Updated device driver loading procedure
	Chapter 1, "Introduction," on page 10	Updated Mac OS X driver support for 10.5.x and 10.6.x.
	All	Updated all references to “NDIS 6.2” to “NDIS 6.20”.
	Section 11.3.2, "Test Time," on page 104	Updated test time and link mode values. Added note to end of section.
	Section 6.2, "Linux Driver Uninstallation," on page 62	Rephrased note.
	Section 5.2, "MAC OS X Driver Uninstallation," on page 54	Rephrased note.
	Section 8.1, "Windows Driver Customization and Localization," on page 69	Updated description and tables to reflect new merged Windows XP/Vista/7 installer package.
	Section 2.2, "Windows XP 32/64-Bit Driver Installation via INF," on page 15	Updated note to include reference to the Driver Customization chapter. Updated Figure 2.11, Figure 2.12, Figure 2.13, and supporting text to reflect the new file structure of the Windows XP/Vista/7 installer package.
	Section 3.2, "Windows Vista 32/64-Bit Driver Installation via INF," on page 26	Updated note to include reference to the Driver Customization chapter. Updated Figure 3.11, Figure 3.12, Figure 3.13, and supporting text to reflect the new file structure of the Windows XP/Vista/7 installer package.
	Section 4.2, "Windows 7 32/64-Bit Driver Installation via INF," on page 38	Updated note to include reference to the Driver Customization chapter. Updated Figure 4.14, Figure 4.15, and supporting text to reflect the new file structure of the Windows XP/Vista/7 installer package.
	Section 8.1.1, "GUI Visual Elements and Strings Customization," on page 71	Updated note to reflect the new file structure of the Windows XP/Vista/7 installer package.
Rev. 1.0 (03-09-10)	All	Initial Release

Table of Contents

List of Tables

List of Figures

Chapter 1 Introduction

This manual provides detailed instructions on the installation and uninstallation of LAN7500/LAN7500i software drivers under various operating systems:

- Windows XP 32/64-Bit Driver
- Windows Vista 32/64-Bit Driver
- Windows 7 32/64-Bit Driver
- MAC OS X Driver (10.5.x/10.6.x)
- 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: 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.

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 8, "Driver Customization," on page 69 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 19.

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.

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.

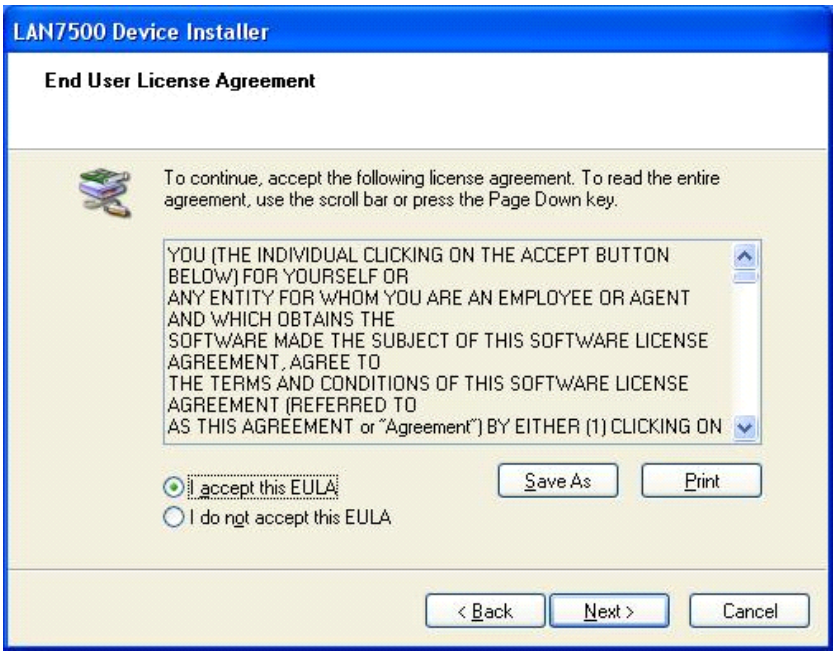

Figure 2.2 End User License Agreement

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.

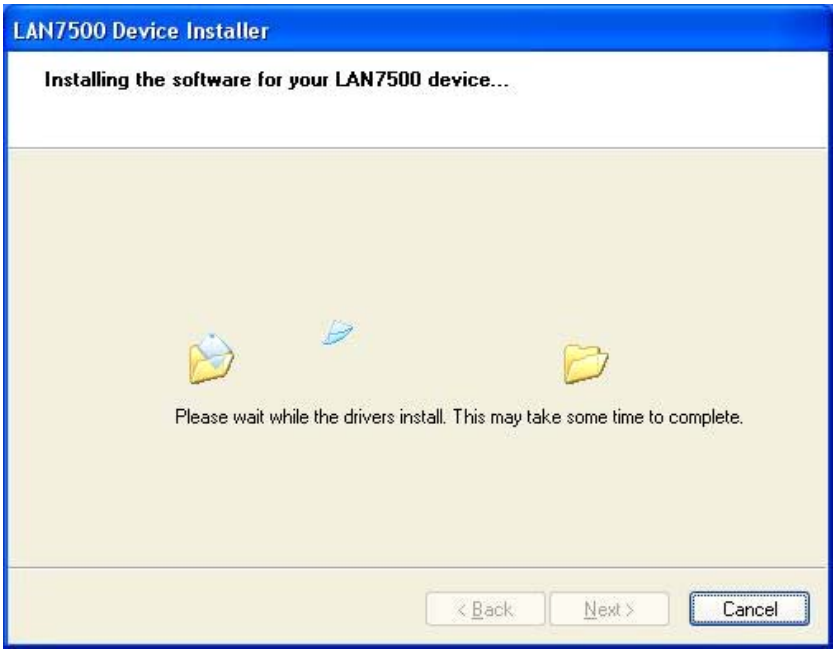

Figure 2.3 Installation Progress Window

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, depending 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

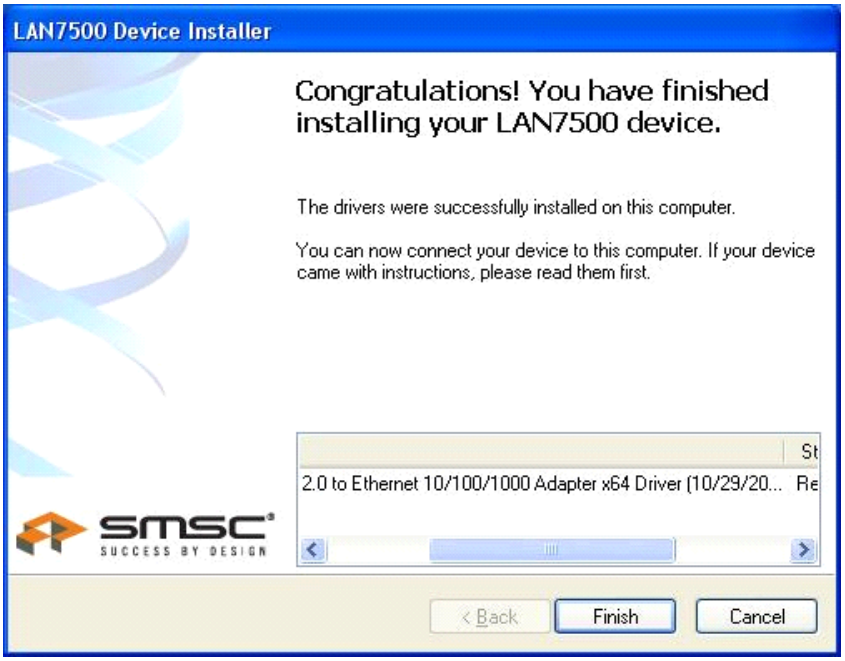

Figure 2.5 Device Driver Installation Complete Screen - Windows XP 64-bit

The device must now be plugged into an available USB port on the computer. Once accomplished, the balloon notification 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. Refer to Chapter 8, "Driver Customization," on page 69 for details on how to locate the files that allow INF-only 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 an available USB port on the computer. Once accomplished, the balloon notification message 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”.

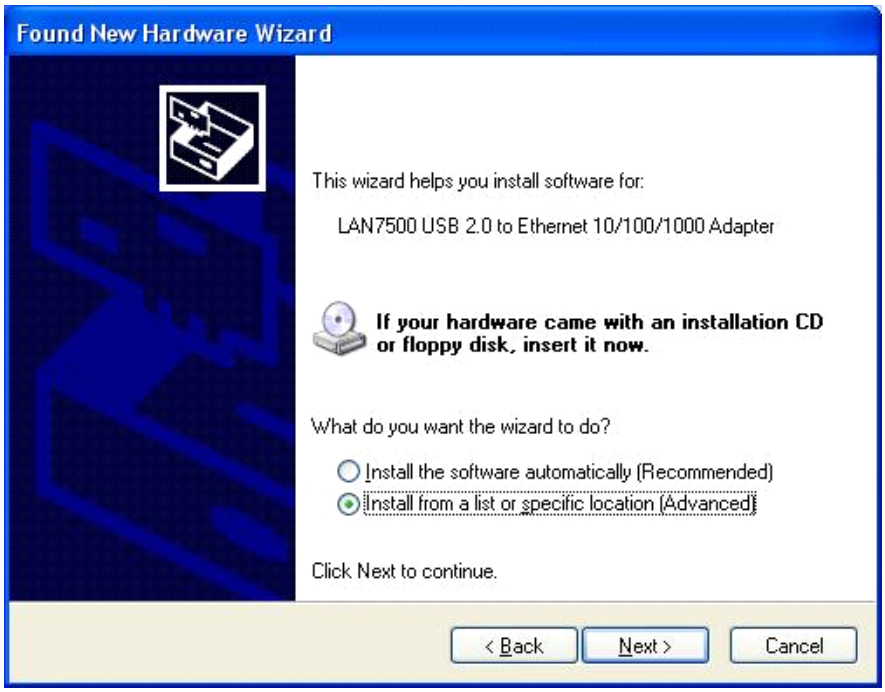

Figure 2.10 Found New Hardware Wizard 2

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.

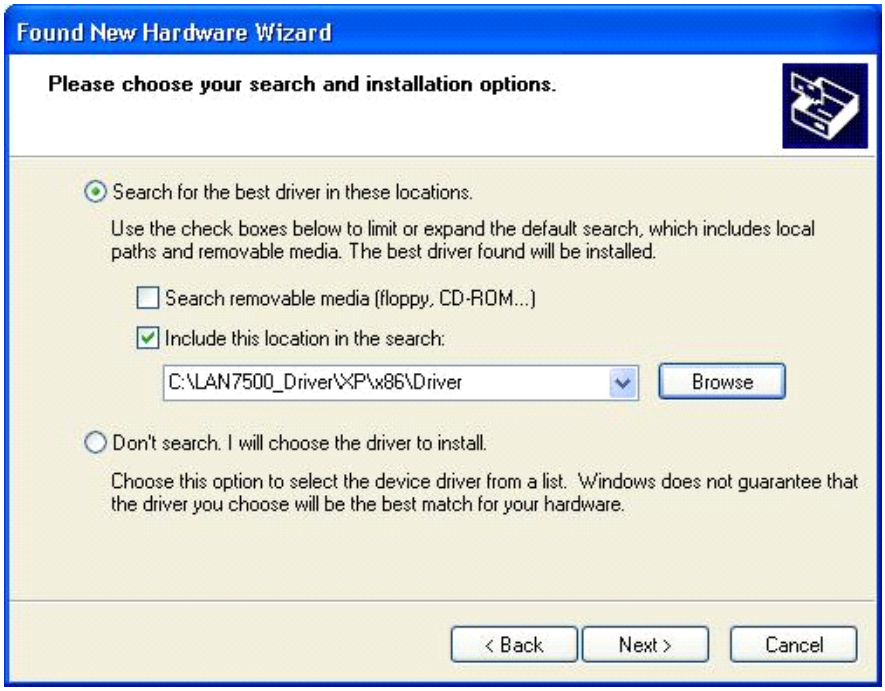

Figure 2.11 Search and Installation Options Window

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: Windows XP installations must use the driver under the “XP” folder. For Windows 32-bit installations, browse to the “x86” folder within the “XP” folder of the SMSC INF distribution files, as shown in Figure 2.12. For Windows 64-bit installations, browse to the “x64” folder within the “XP” folder of the 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

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.

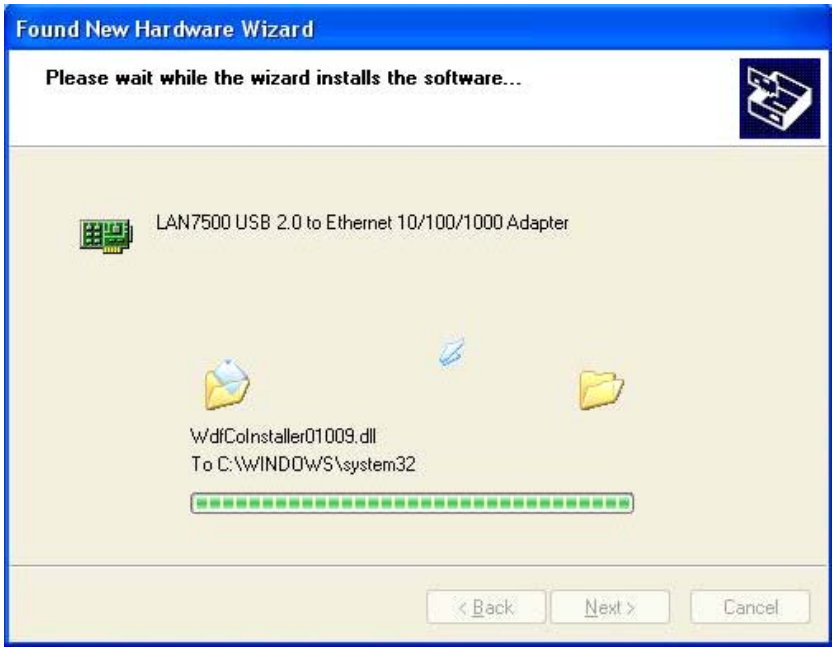

Figure 2.14 Installation Progress Window

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.

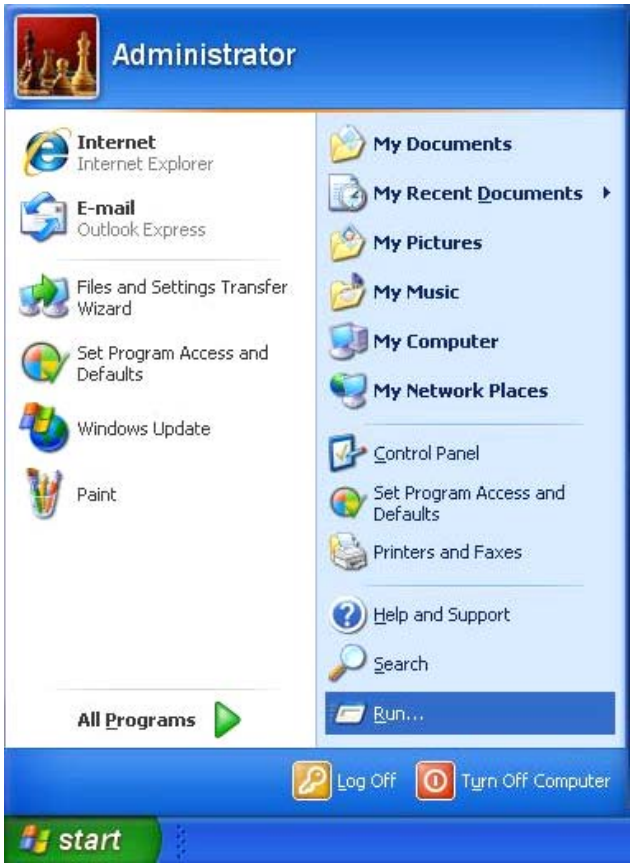

Figure 2.16 Start Menu - Run

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

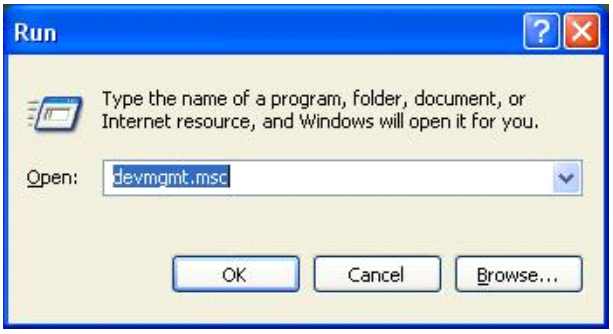

Figure 2.17 Run Window

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.

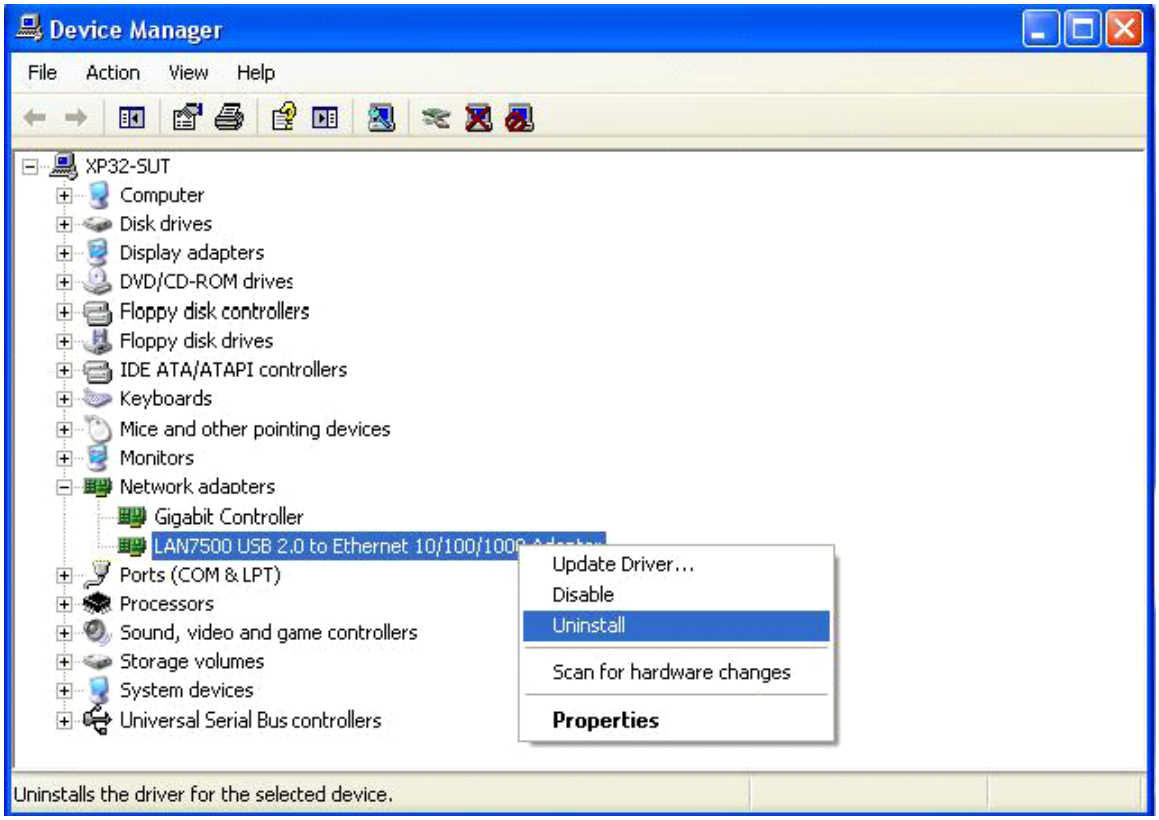

Figure 2.18 Device Manager Window - Uninstall

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 `net7500-<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 `net7500-<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 Manger window may now be closed.

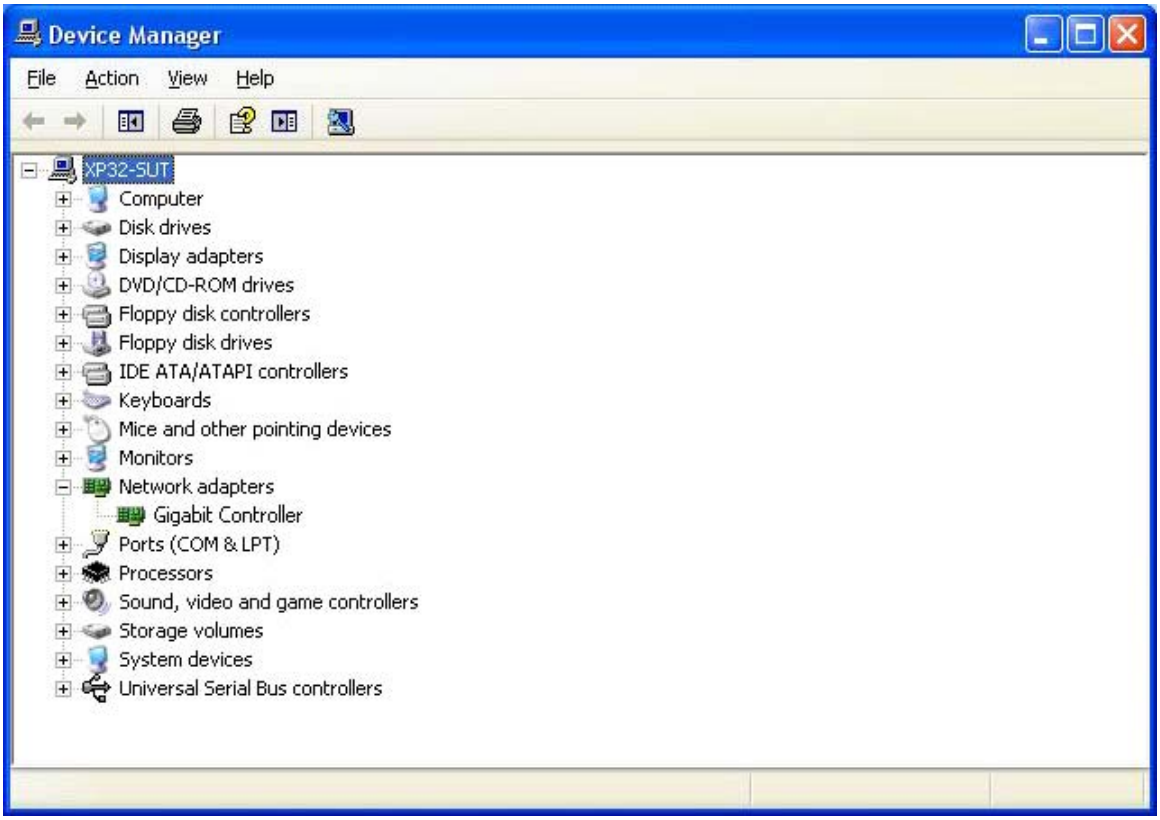

Figure 2.20 Device Removed from Device Manager Window

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 8, "Driver Customization," on page 69 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 31.

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.

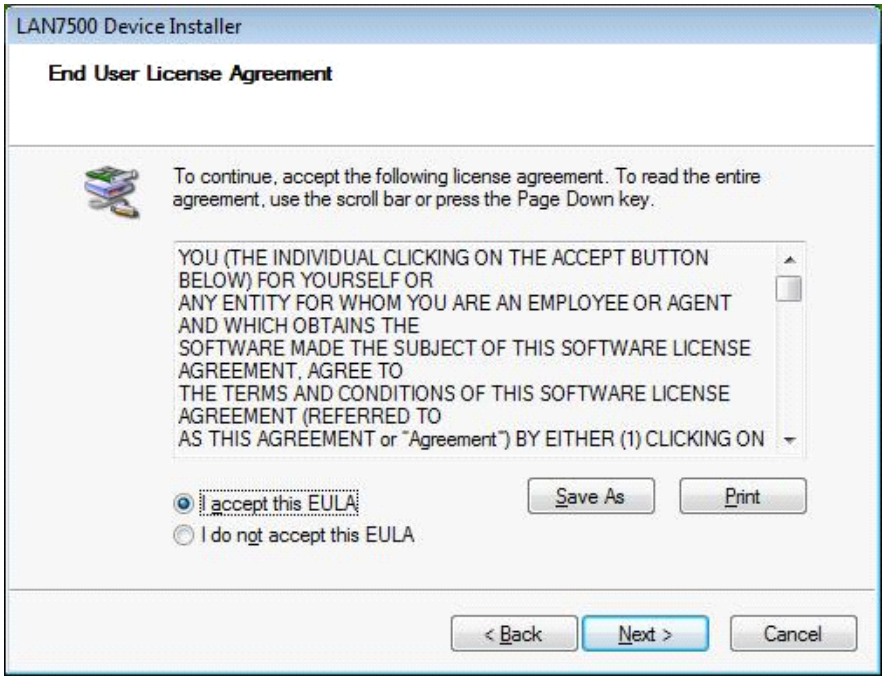

Figure 3.2 End User License Agreement

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

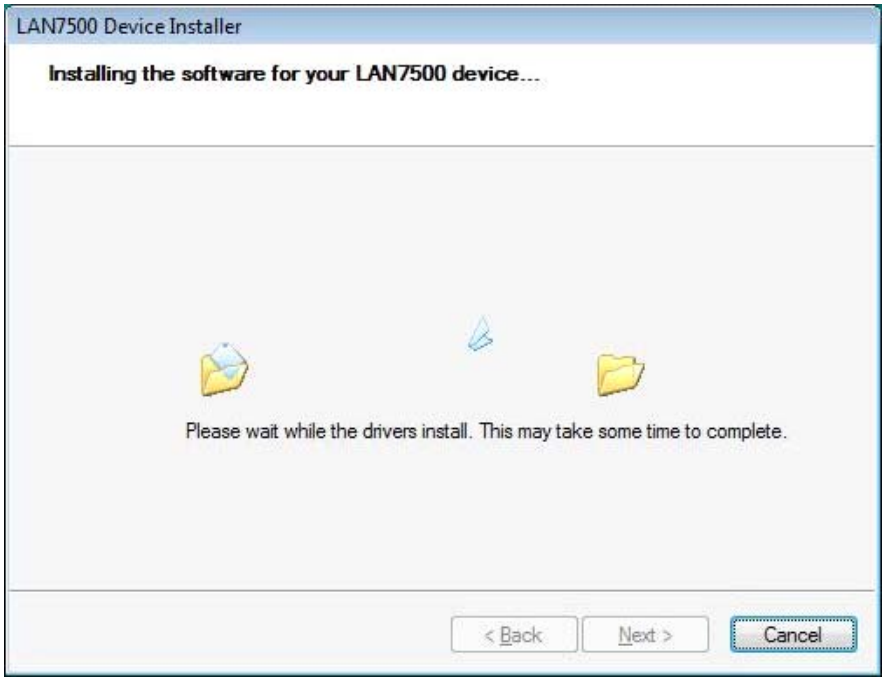

Figure 3.3 Installation Progress Window

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 an available USB port on the computer. Once accomplished, the balloon notification message shown in Figure 3.6 will appear in the task bar.

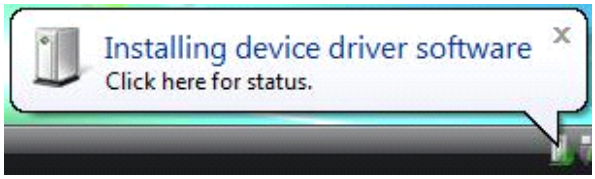

Figure 3.6 Installing Device Driver Software Task Bar Notification

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

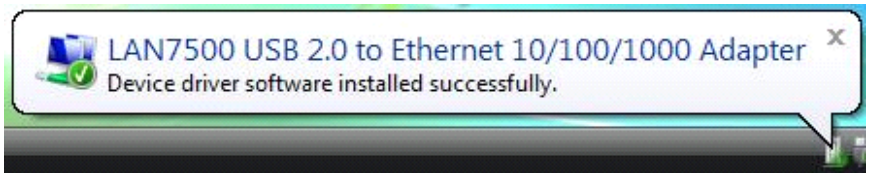

Figure 3.7 Device Driver Software Installed Successfully Task Bar Notification

At this point, the network device installation is complete. The device will be setup to have it's 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. Refer to Chapter 8, "Driver Customization," on page 69 for details on how to locate the files that allow INF-only 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 an available USB port on the computer. Once 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.

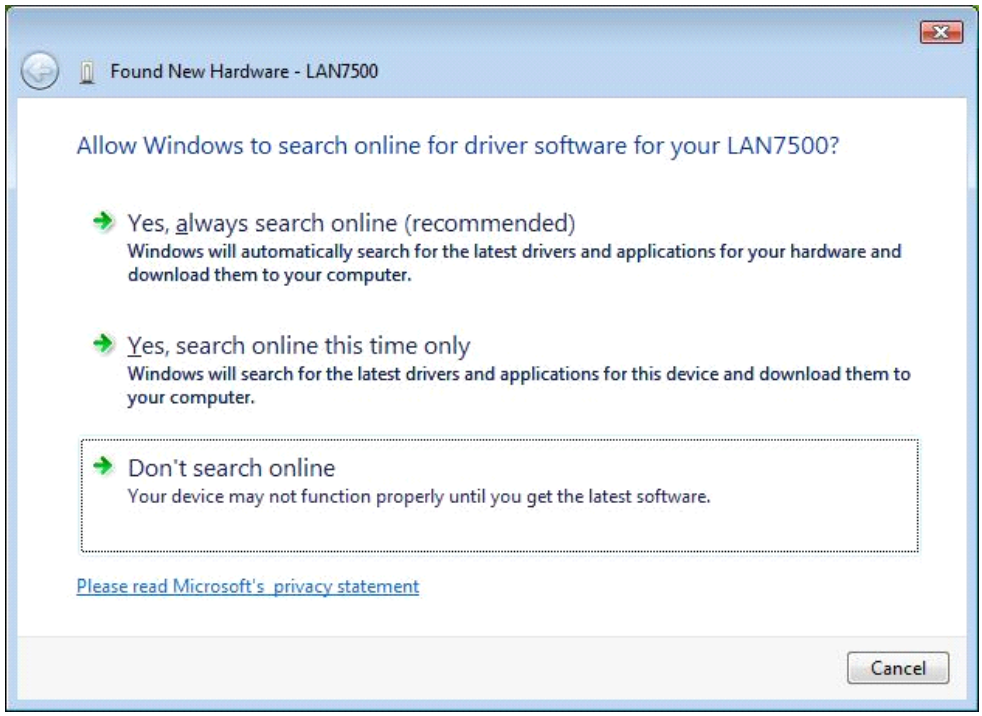

Figure 3.9 Search Online Option Window

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

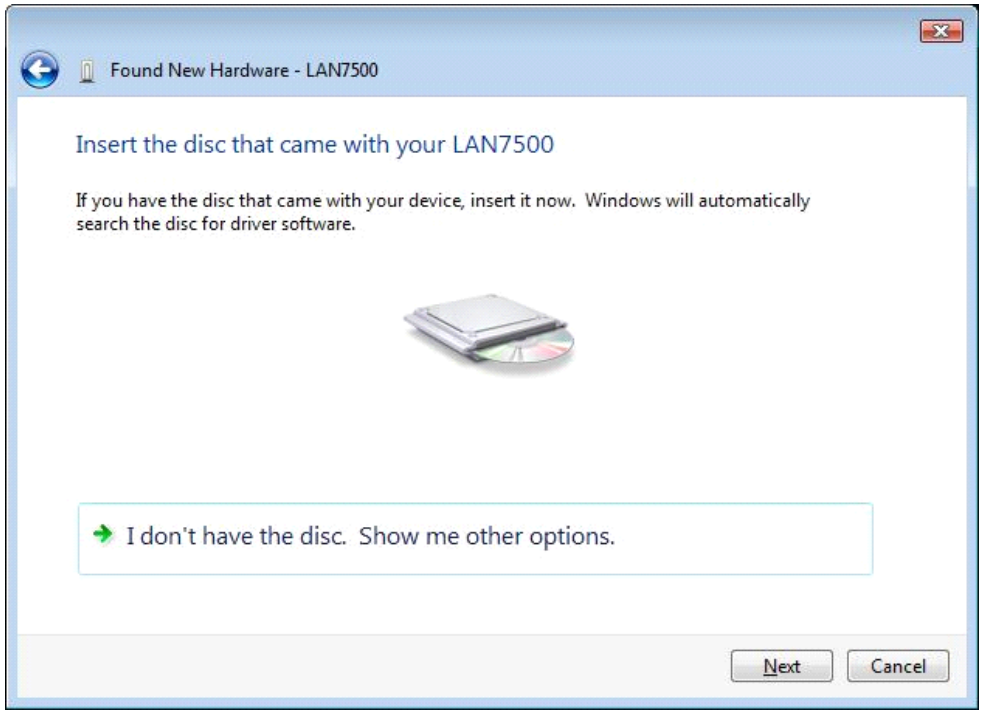

Figure 3.10 Insert Disk Window

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

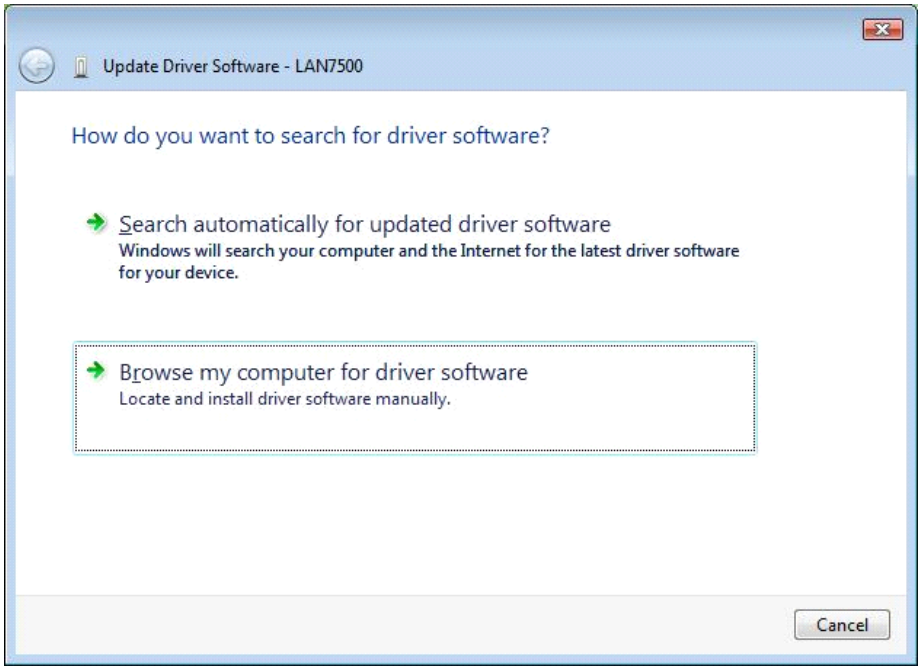

Figure 3.11 Driver Install Options Window

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

Note: Windows Vista installations must use the driver under the “Vista” folder. For Windows 32-bit installations, browse to the “x86” folder within the “Vista” folder of the SMSC INF distribution files, as shown in Figure 3.12. For Windows 64-bit installations, browse to the “x64” folder within the “Vista” folder of the SMSC INF distribution files, as shown in Figure 3.13.

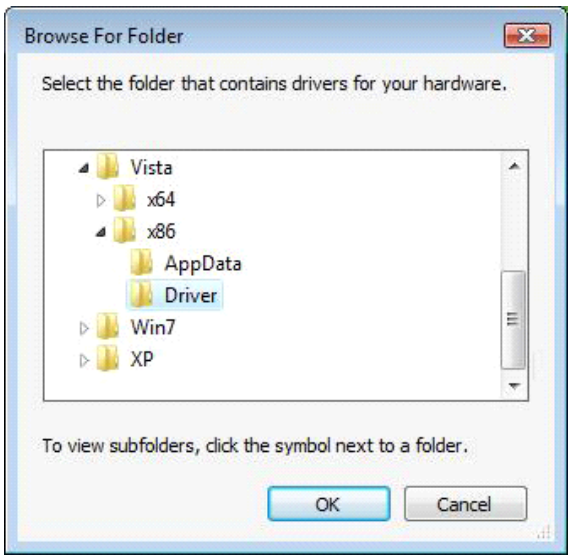

Figure 3.12 Browse Window - Windows Vista 32-Bit

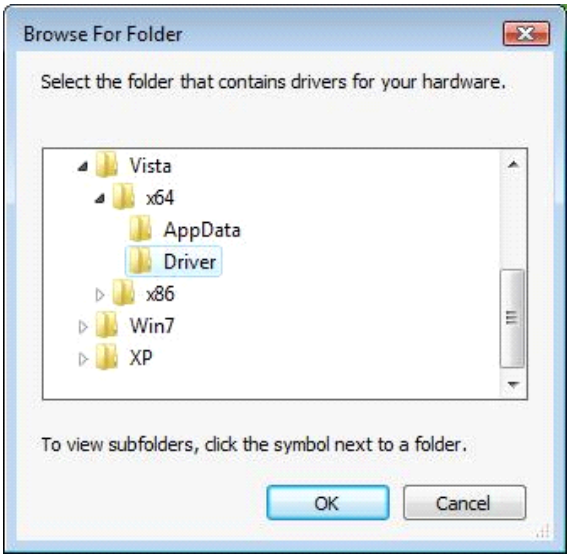

Figure 3.13 Browse Window - Windows Vista 64-Bit

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.

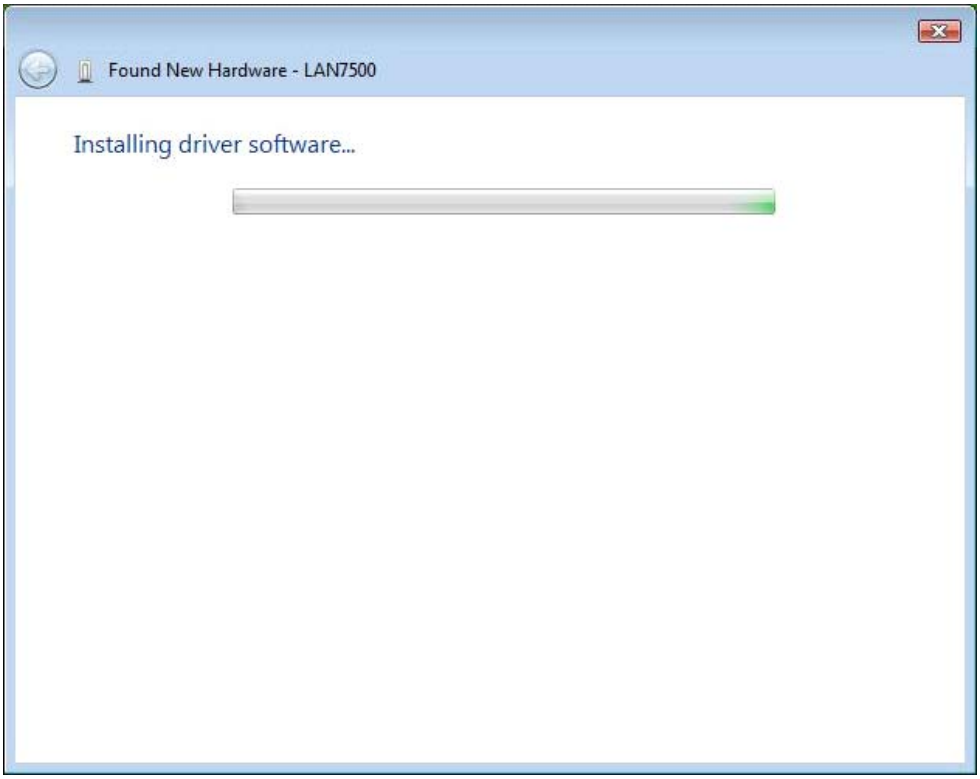

Figure 3.14 Installation Progress Window

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

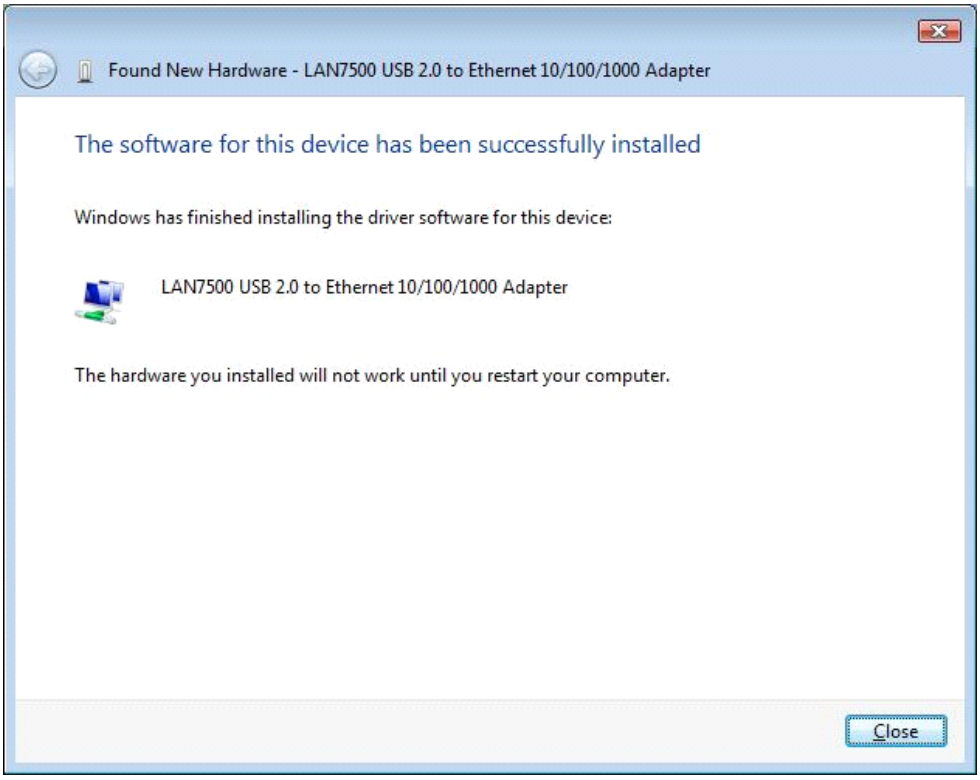

Figure 3.15 Device Driver Installation Complete Screen

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

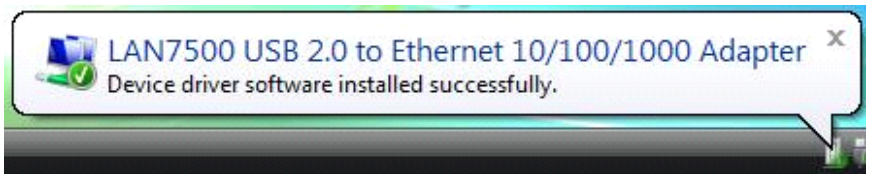

Figure 3.16 Device Driver Installation Success Task Bar Notification

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.

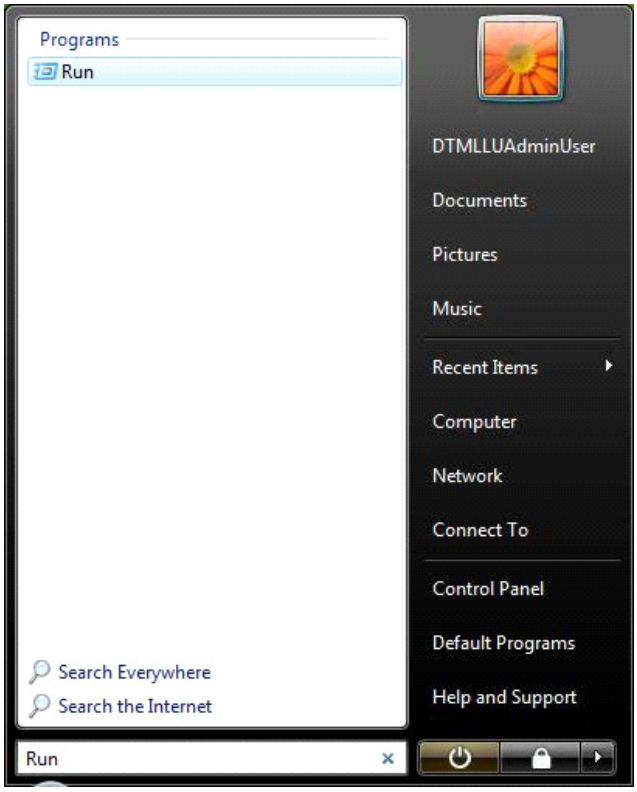

Figure 3.17 Start Menu Search

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

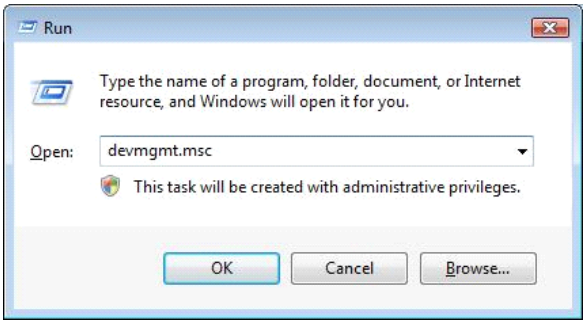

Figure 3.18 Run Window

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.

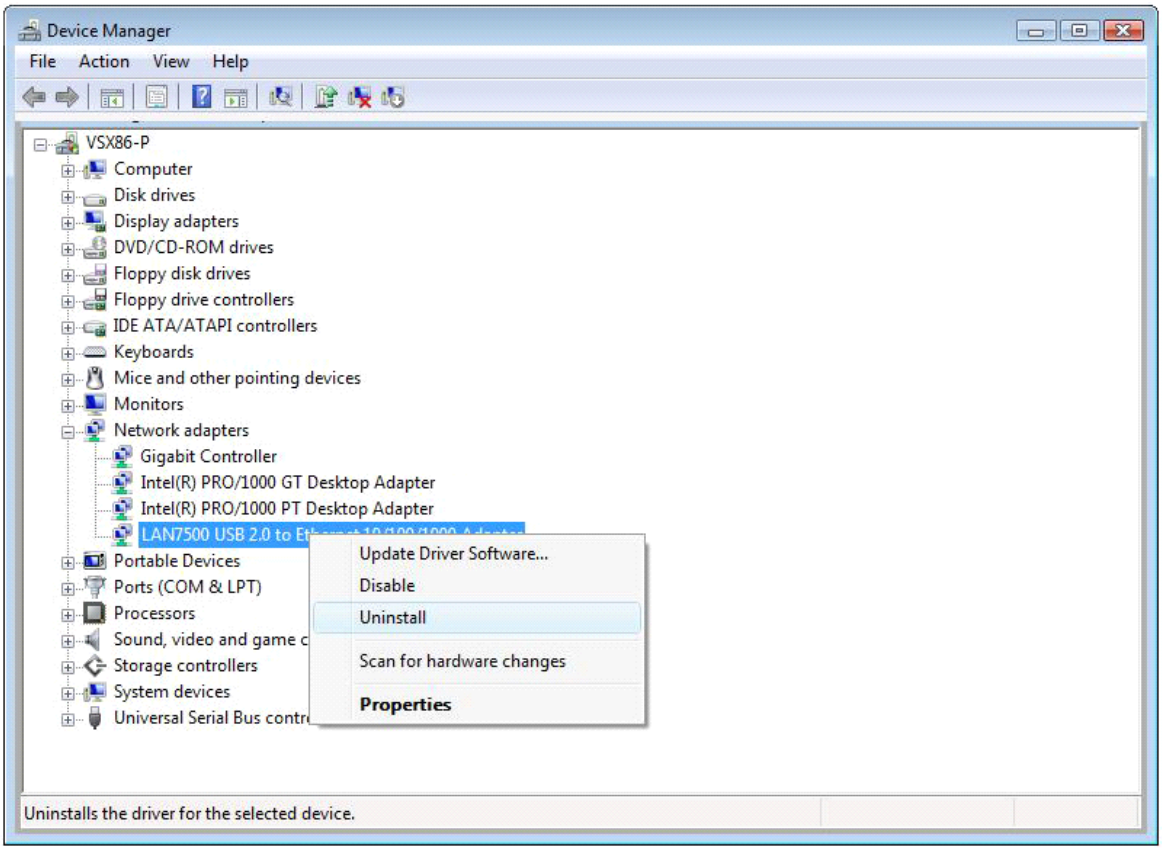

Figure 3.19 Device Manager Window

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.

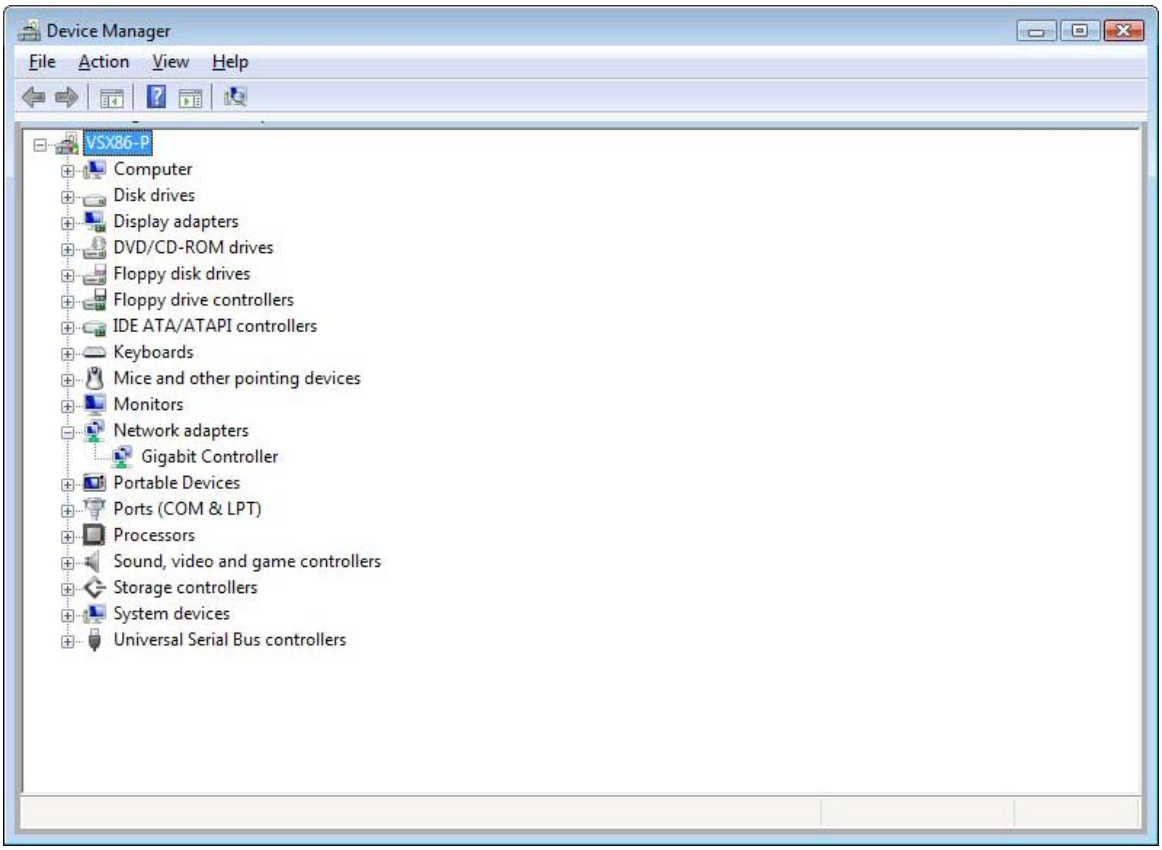

Figure 3.22 Device Removed from Device Manager

Chapter 4 Windows 7 32/64-Bit Driver

This chapter details the installation and uninstallation of the Windows 7 32/64-bit driver.

The Windows 7 32/64-bit driver may be installed in two ways:

- Windows 7 32/64-Bit Driver Installation via EXE (preferred method)
- Windows 7 32/64-Bit Driver Installation via INF

Note: Official driver releases are provided by SMSC in EXE format only. Refer to Chapter 8, "Driver Customization," on page 69 for details on how to locate the files that allow INF-only installation.

Windows 7 32/64-bit driver uninstallation is detailed in Section 4.3, "Windows 7 32/64 Bit Driver Uninstallation," on page 44.

4.1 Windows 7 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:

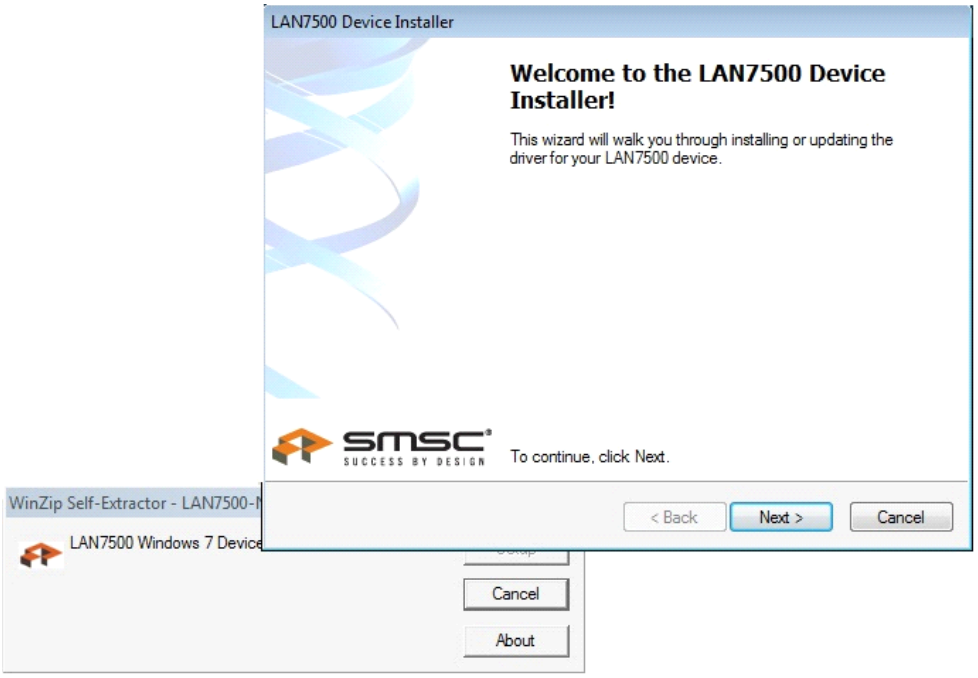

Figure 4.1 Device Installer Invocation

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 4.2, which will permit the installation process to continue.

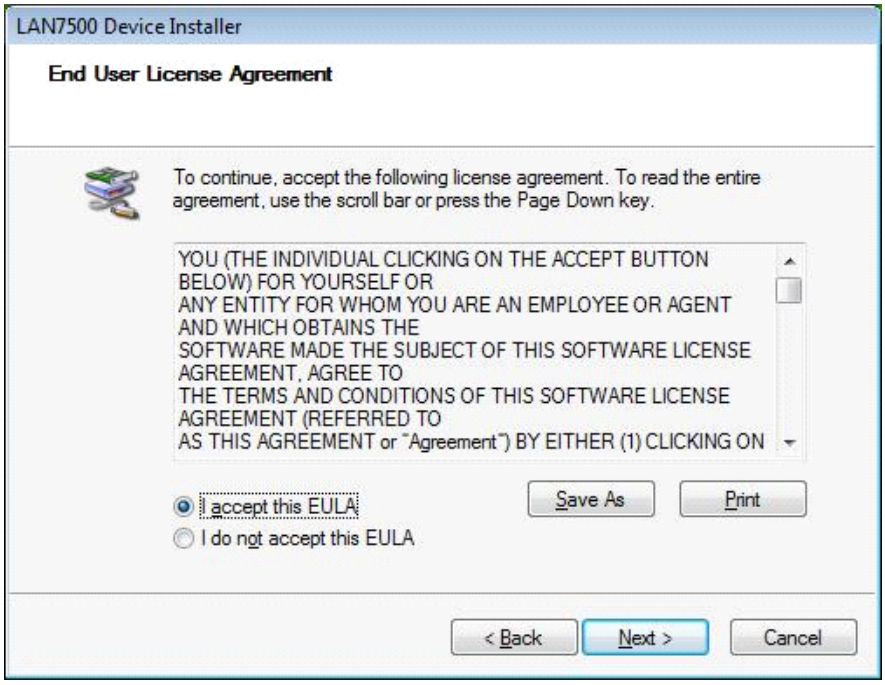

Figure 4.2 End User License Agreement

Click the “Next” button to continue the installation process. The Device Installer window will change to a progress window, as shown in Figure 4.3, indicating some time may be necessary for the installation

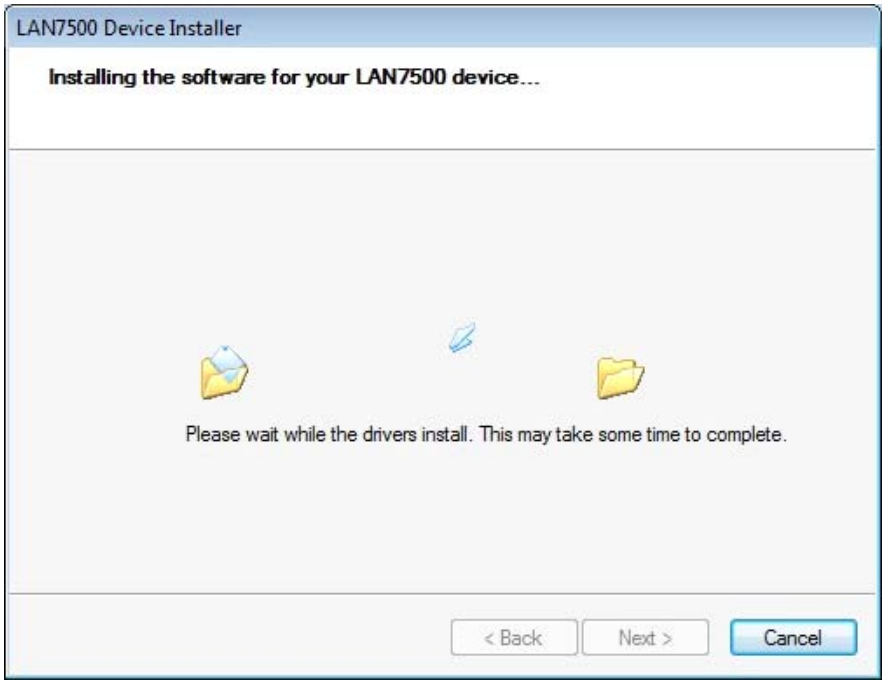

Figure 4.3 Installation Progress Window

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

Only the appropriate 32- or 64-bit driver will be installed, dependent on the version of Windows 7 being used. Figure 4.4 shows the installation of the 32-bit driver in a Windows 7 32-bit environment. Figure 4.5 shows the installation of the 64-bit driver in a Windows 7 64-bit environment

Figure 4.4 Device Driver Installation Complete Screen - Windows 7 32-Bit

Figure 4.5 Device Driver Installation Complete Screen - Windows 7 64-Bit

The device must now be plugged into an available USB port on the computer. Once accomplished, the balloon notification message shown in Figure 4.6 will appear in the task bar.

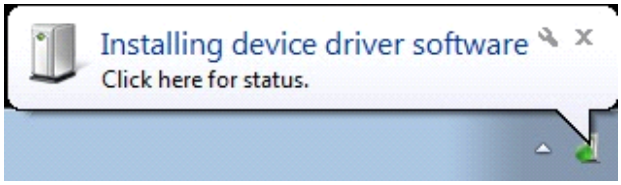

Figure 4.6 Installing Device Driver Software Task Bar Notification

When the software installation is complete, the balloon notification message shown in Figure 4.7 will appear, signaling completion of the installation.

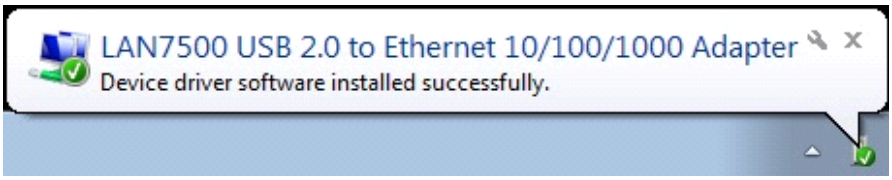

Figure 4.7 Device Driver Software Installed Successfully Task Bar Notification

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

4.2 Windows 7 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 4.1 is the preferred method of installation. Refer to Chapter 8, "Driver Customization," on page 69 for details on how to locate the files that allow INF-only 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 an available USB port on the computer. Once accomplished, the balloon notification shown in Figure 4.8 will appear in the task bar, indicating that no LAN7500 driver is currently installed. Dismiss the balloon notification.

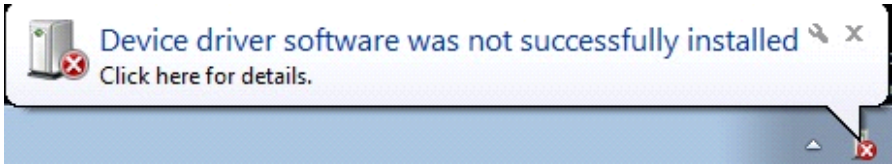

Figure 4.8 Device Driver Not Installed Task Bar Notification

To install the LAN7500 driver, type “Run” in the Start menu search field and click on “Run” in the programs list, as shown in Figure 4.9.

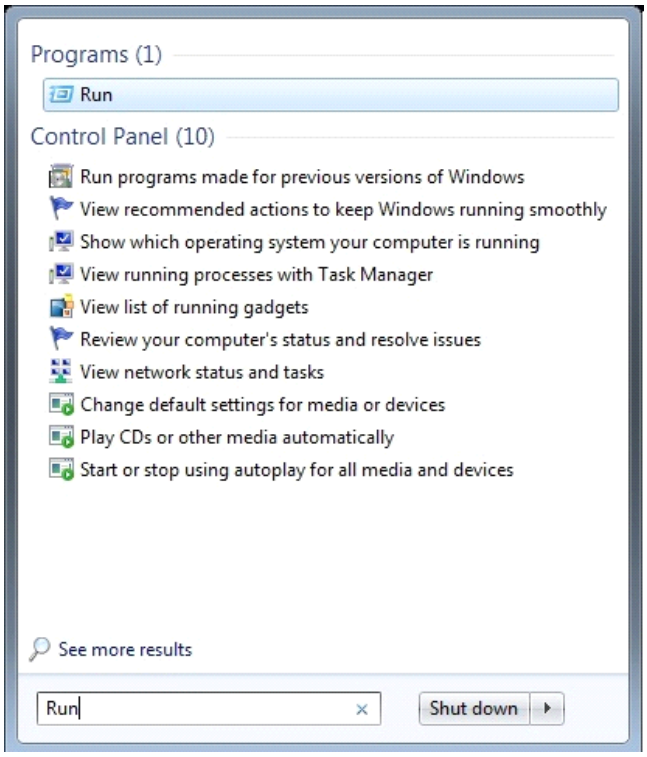

Figure 4.9 Start Menu Search

In the Run window, type “devmgmt.msc” and click “OK”, as shown in Figure 4.10. This will open the Device Manager window.

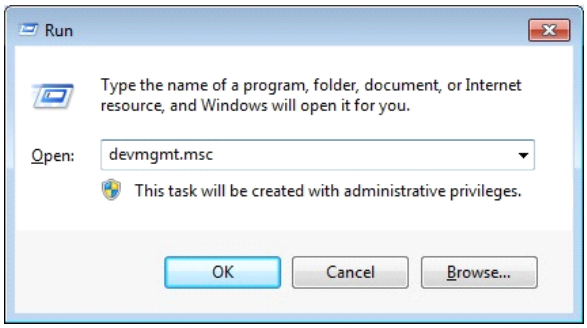

Figure 4.10 Run Window

In the Device Manager window, browse to the “Other devices” section and select the device, as shown in Figure 4.11.

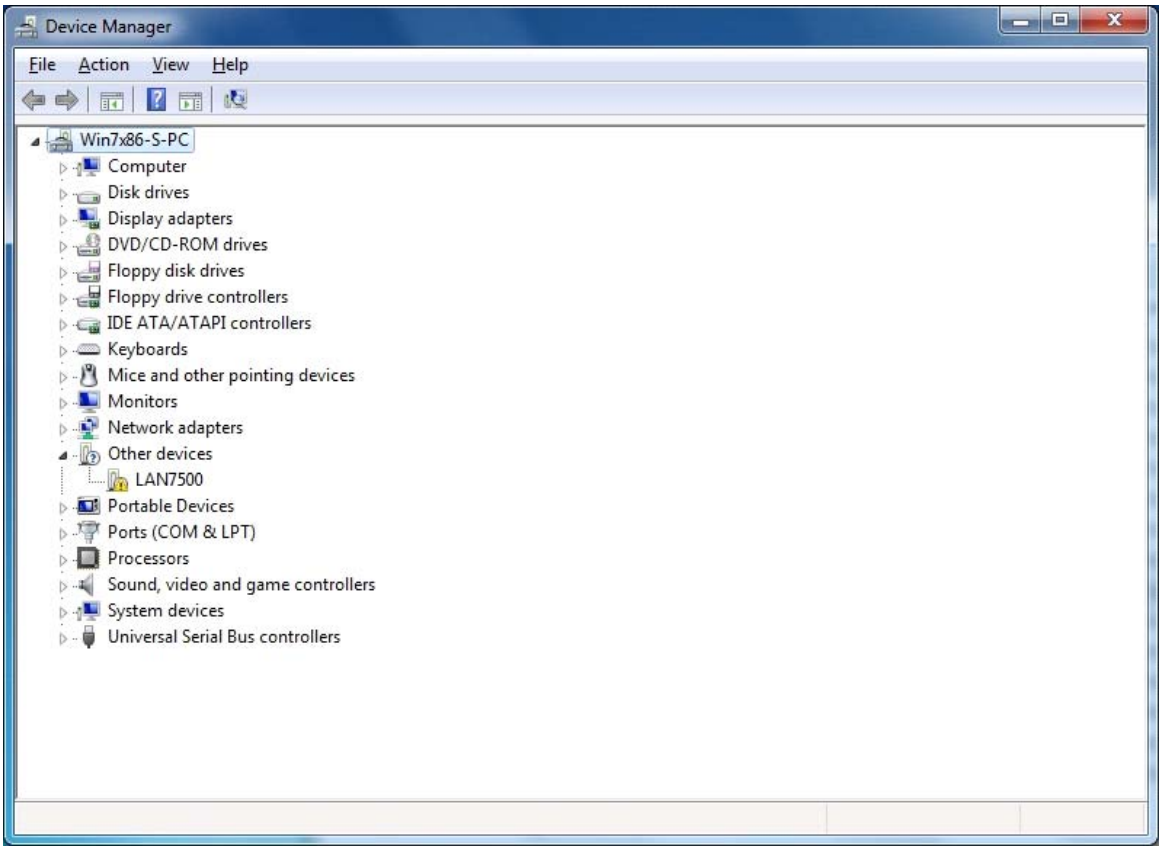

Figure 4.11 Device Manager Window - Other Devices

Right click on the device and select “Update Driver Software”, as shown in Figure 4.12.

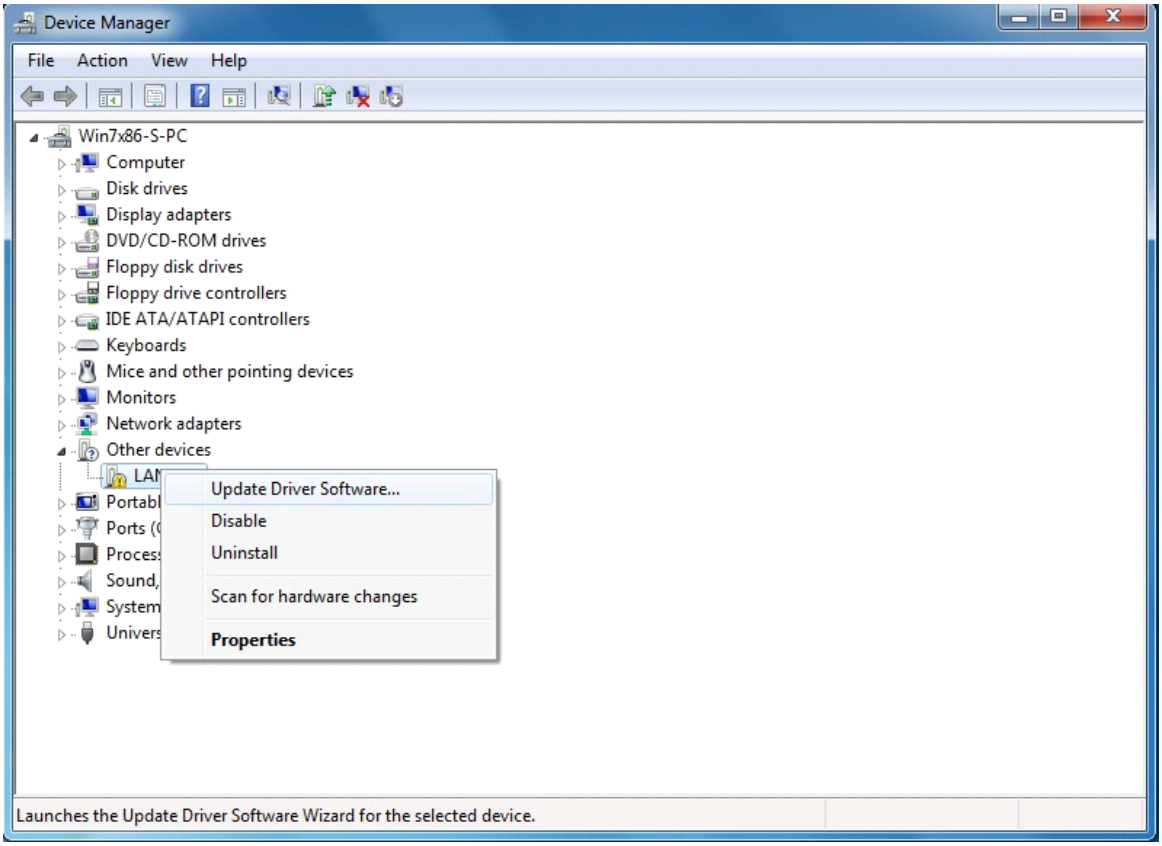

Figure 4.12 Device Manager Window - Update Driver Software

Click “Browse my computer for driver software”, as shown in Figure 4.13.

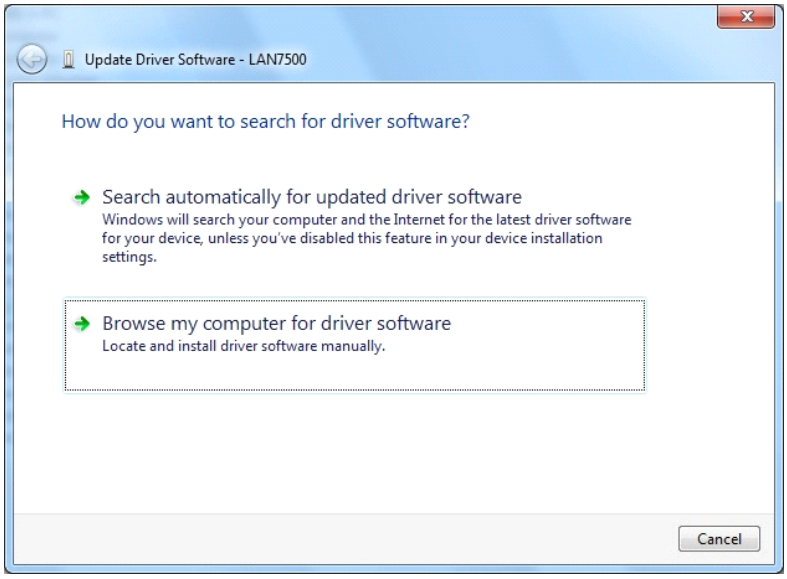

Figure 4.13 Driver Install Options Window

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

Note: Windows 7 installations must use the driver under the “Win7” folder. For Windows 32-bit installations, browse to the “x86” folder within the “Win7” folder of the SMSC INF distribution files, as shown in Figure 4.14. For Windows 64-bit installations, browse to the “x64” folder within the “Win7” folder of the SMSC INF distribution files, as shown in Figure 4.15.

Figure 4.14 Browse Window - Windows 7 32-Bit

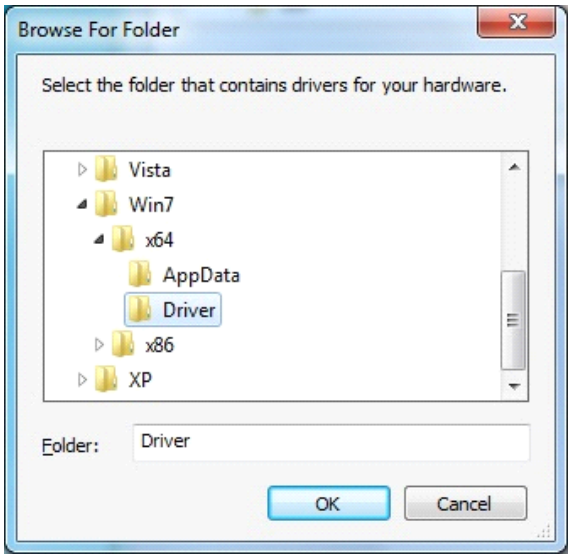

Figure 4.15 Browse Window - Windows 7 64-Bit

The Update Driver Software window will change to a progress window, as shown in Figure 4.16, indicating some time may be necessary for the installation.

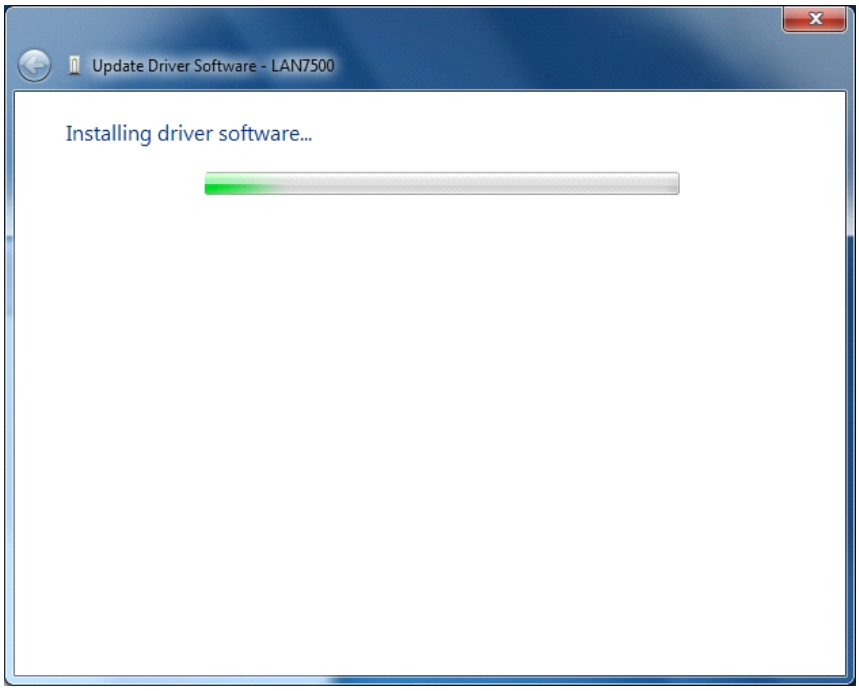

Figure 4.16 Installation Progress Window

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

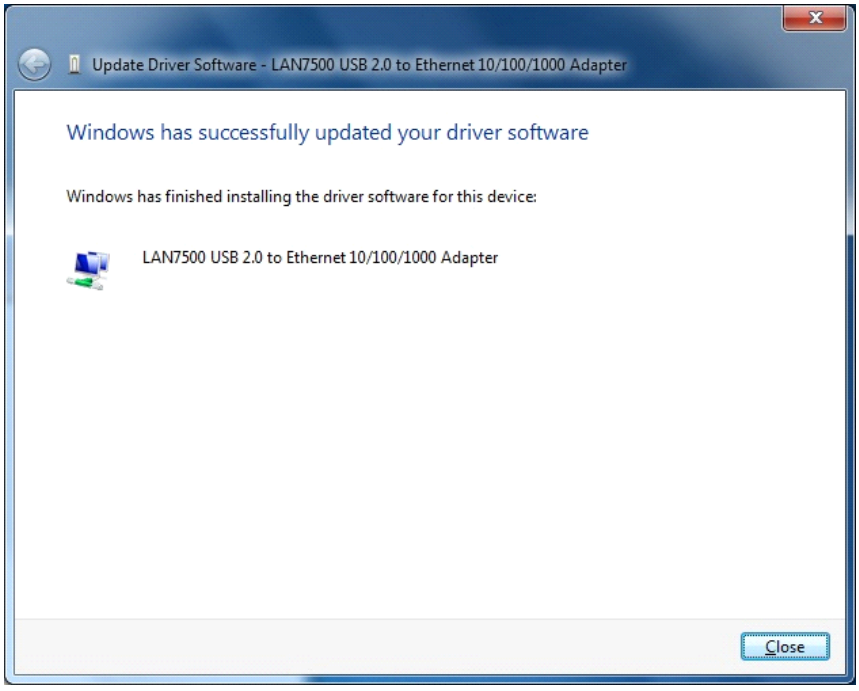

Figure 4.17 Device Driver Update Complete Screen

A device driver installation notification will pop-up as shown in Figure 4.18, indicating the driver installation is complete.

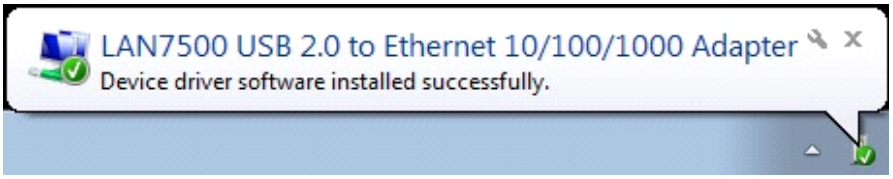

Figure 4.18 Device Driver Installation Success Task Bar Notification

4.3 Windows 7 32/64 Bit Driver Uninstallation

Note: Manual uninstallation is the only supported form of uninstallation at this time. An automated uninstaller via Windows 7's "Programs and Features" may be included in future revisions.

To uninstall the Windows 7 32/64-bit software, type "Run" in the Start menu search field and click on "Run" in the programs list, as shown in Figure 4.19.

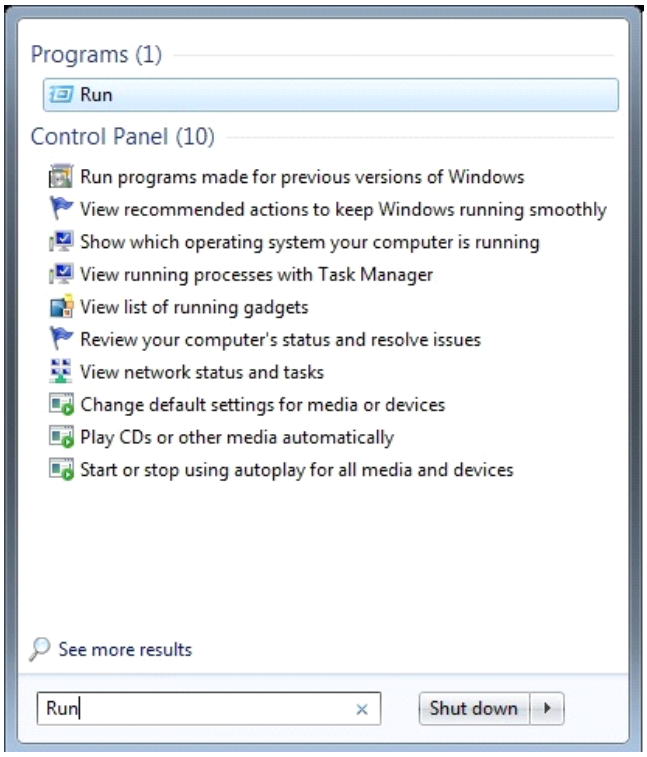

Figure 4.19 Start Menu Search

In the Run window, type "devmgmt.msc" and click "OK", as shown in Figure 4.20. This will open the Device Manager window.

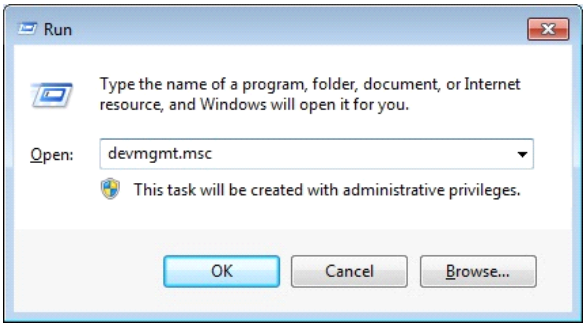

Figure 4.20 Run Window

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 4.21.

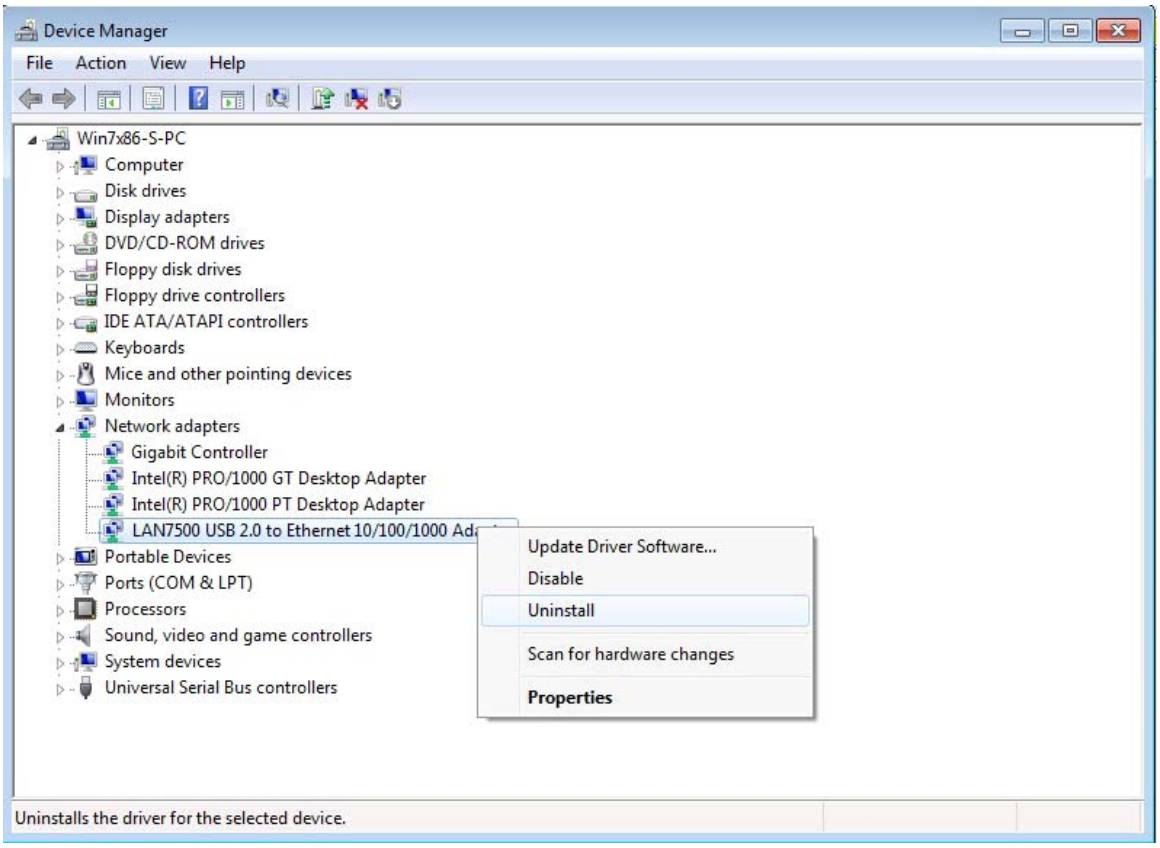

Figure 4.21 Device Manager Window

In the “Confirm Device Uninstall” window, select the “Delete the driver software for this device” checkbox and click “OK”, as shown in Figure 4.22.

Figure 4.22 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 4.23.

Figure 4.23 Device Uninstall Progress Window

The uninstallation process is now complete. As illustrated in Figure 4.24, the device driver will no longer be listed in the Device Manager window. The Device Manager window may now be closed.

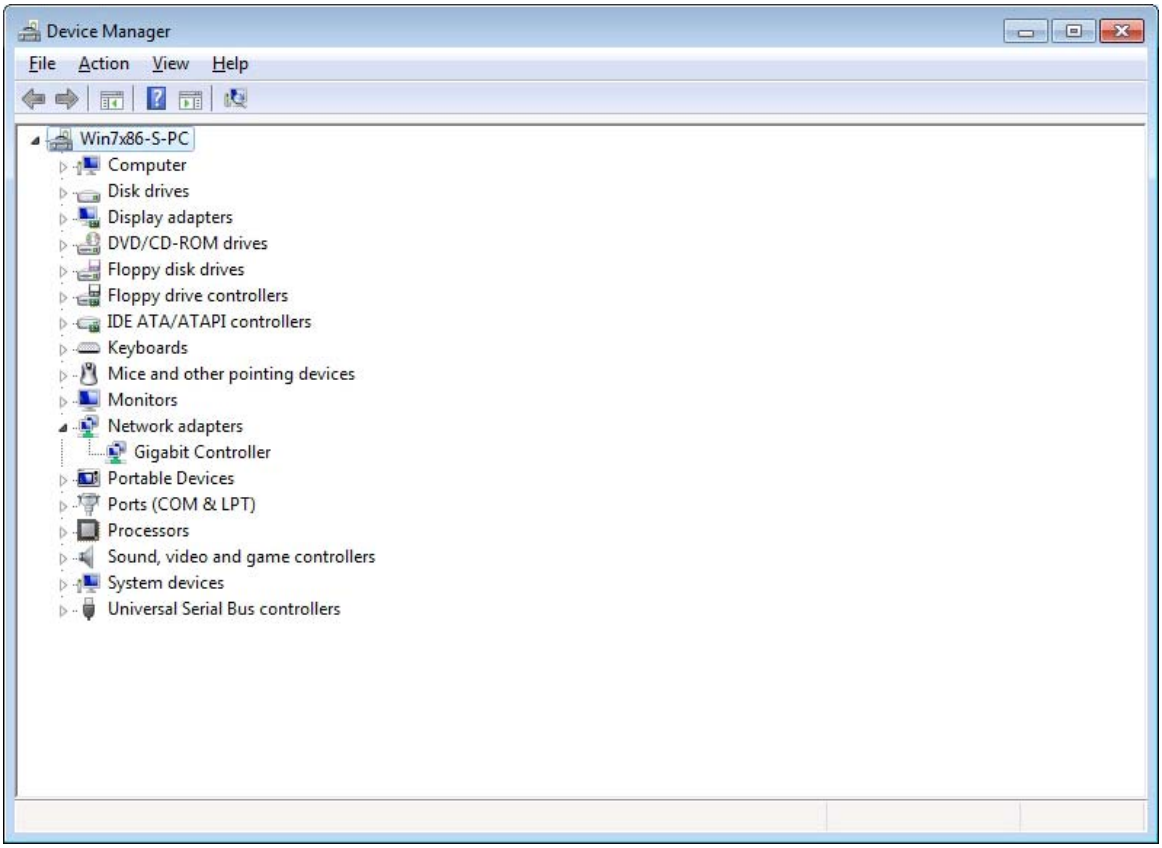

Figure 4.24 Device Removed from Device Manager

Chapter 5 MAC OS X Driver

5.1 MAC OS X 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 5.1 Introduction Screen

Click Continue. The “License” window, illustrated in Figure 5.2, will then appear.

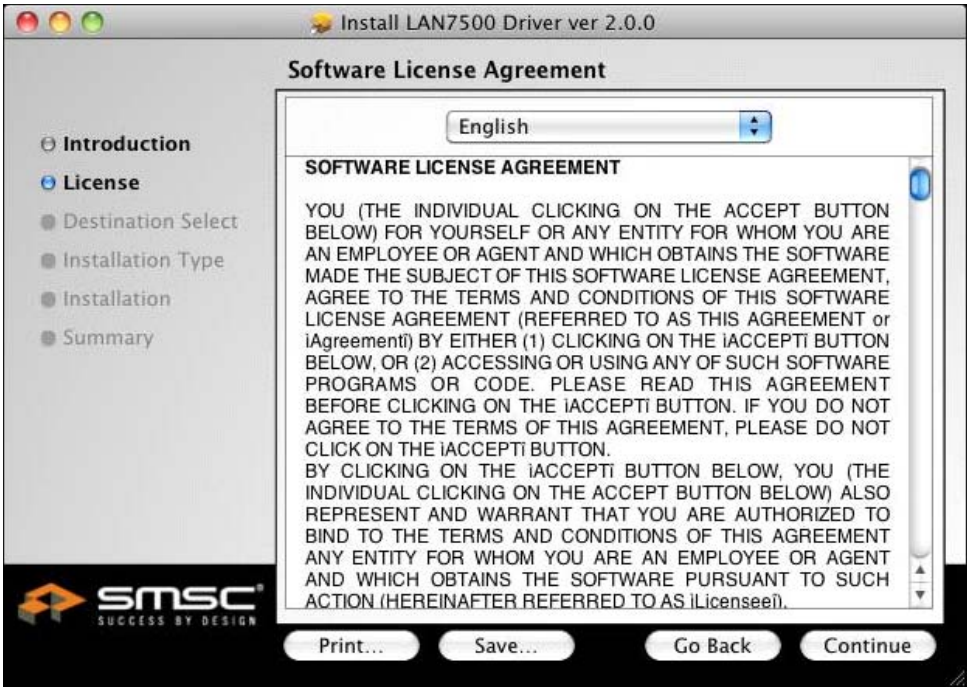

Figure 5.2 License Window

Click Continue. The drop down menu, illustrated in Figure 5.3, will then appear.

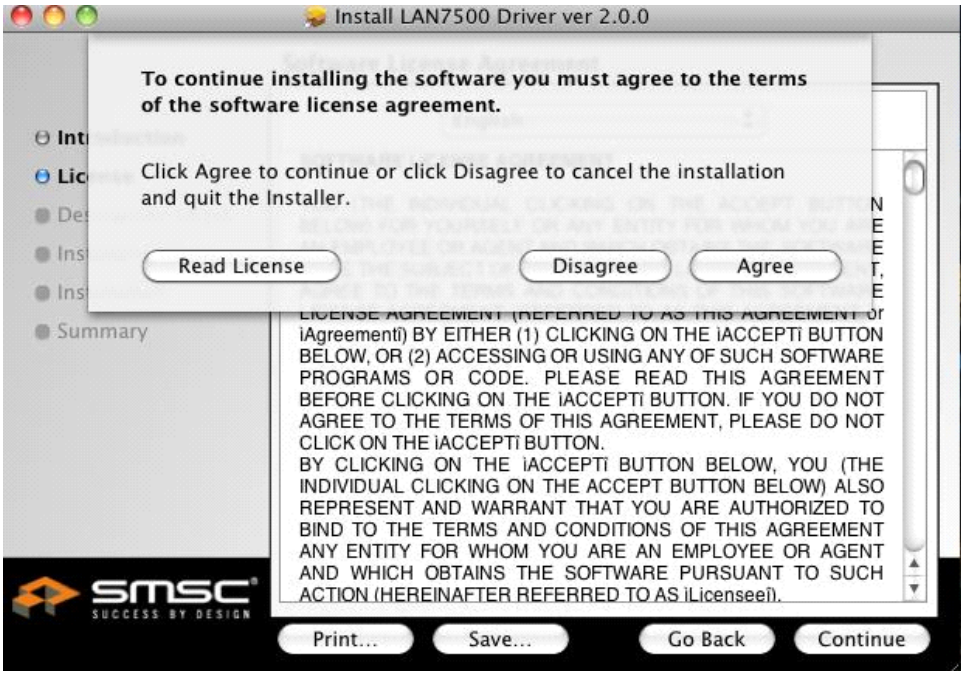

Figure 5.3 Agree/Disagree Drop Down

Click the “Agree” button. The “Installation Type” window, illustrated in Figure 5.4, will then appear.

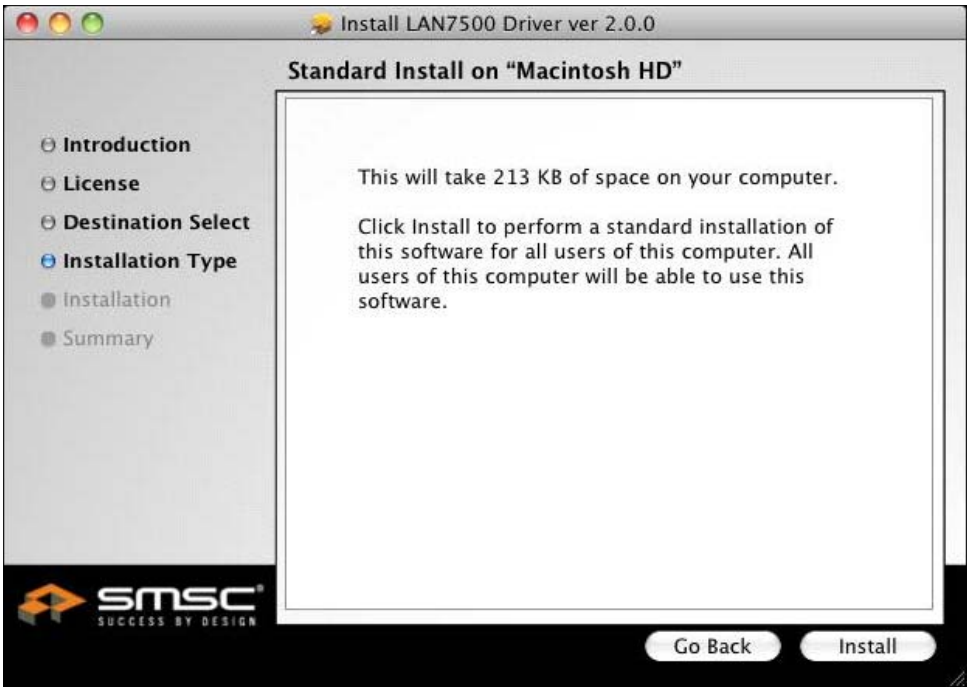

Figure 5.4 Installation Type Window

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 5.5.

Figure 5.5 Installer Name/Password Drop Down

Type in the appropriate Name and Password entries, then click “OK”. The Summary window, indicating successful software installation, illustrated in Figure 5.6, will then appear.

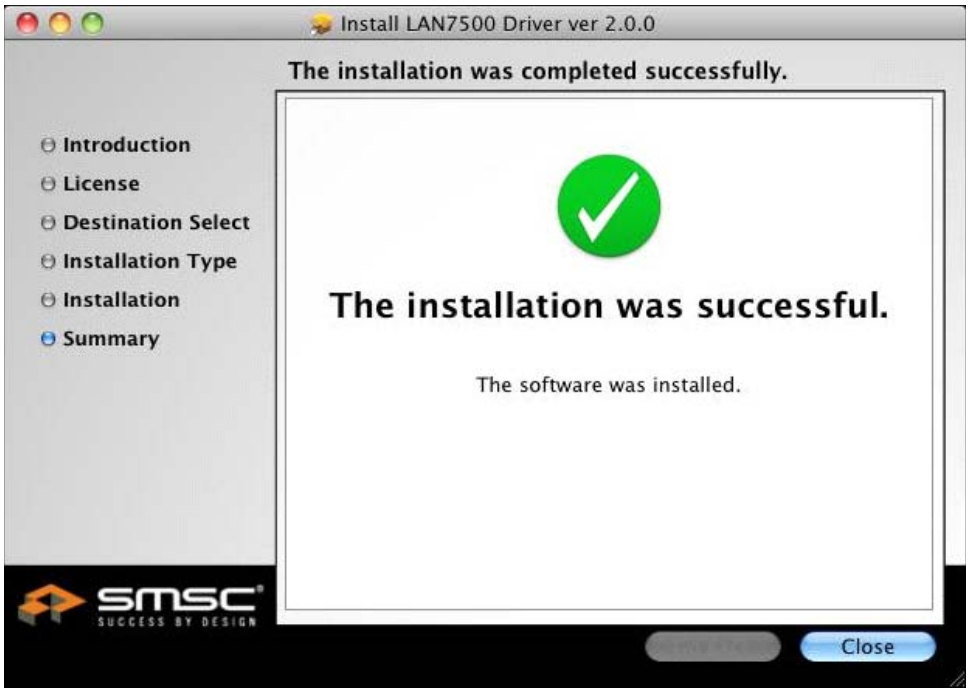

Figure 5.6 Summary Window Indicating Successful Installation

Click “Close” to complete the driver software installation.

The device must now be plugged into an available USB port on the computer. Upon device insertion, the pop up illustrated in Figure 5.7 may or may not be observed. In the case where it is observed, click “Cancel” and proceed to the next step.

Figure 5.7 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 5.8.

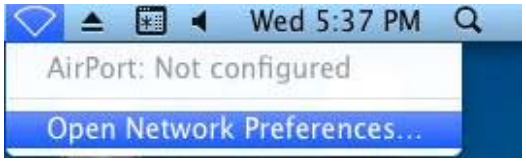

Figure 5.8 Open Network Preferences

The Network window will appear, as illustrated in Figure 5.9.

Figure 5.9 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 5.10.

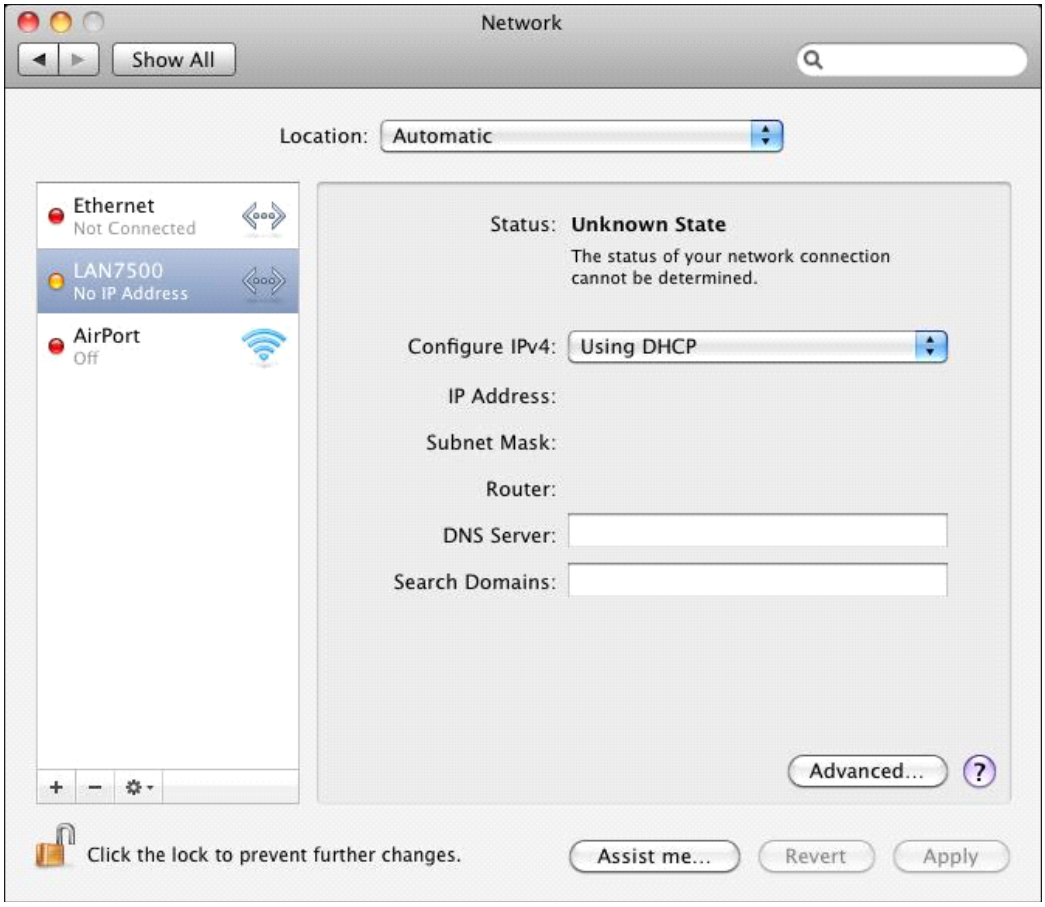

Figure 5.10 New Service Insertion in Network Table

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 5.11.

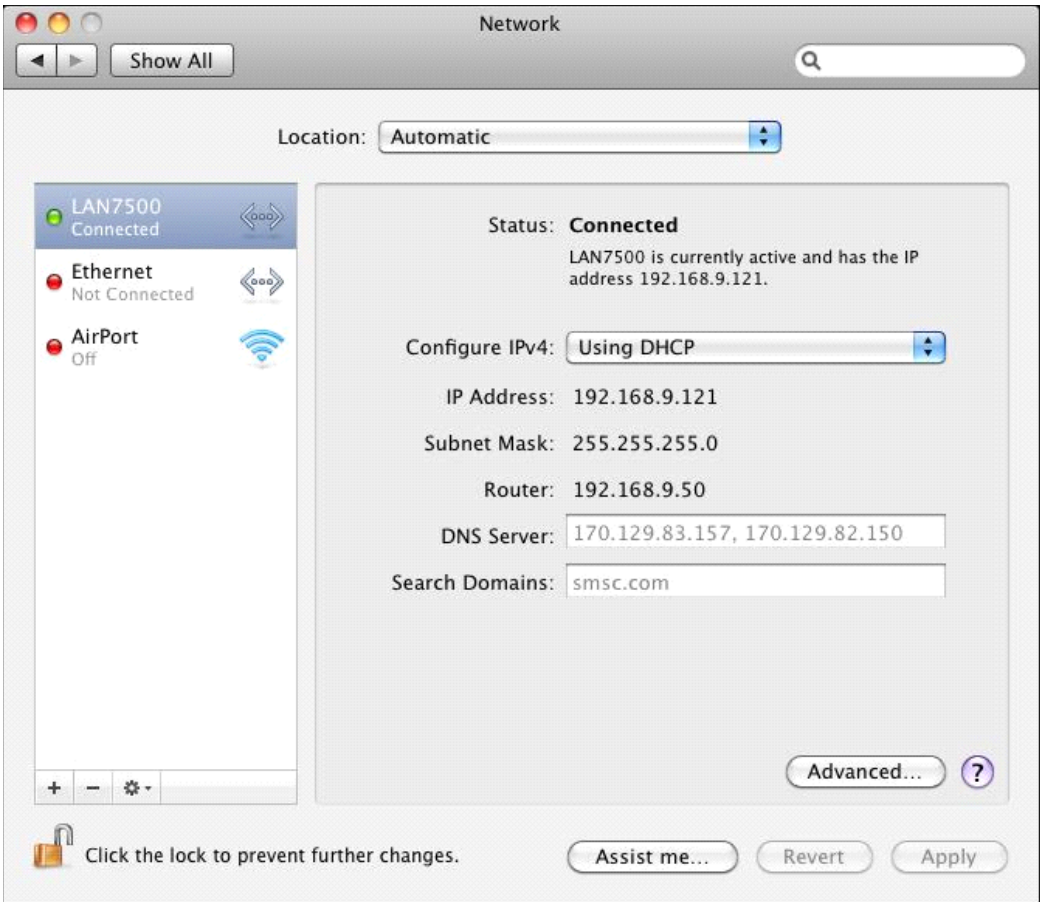

Figure 5.11 Device Connected with IP Address

5.2 MAC OS X Driver Uninstallation

Note: The device must be unplugged from the computer before beginning the uninstallation. If the device cannot be removed from the system, use "`sudo kextunload /System/Library/Extensions/LAN7500.kext`" command to forcefully unload the driver.

To begin the uninstallation process, open up a command terminal and enter the removal command "`sudo rm -rf /System/Library/Extensions/LAN7500.kext/`", as indicated in Figure 5.12. This causes the device driver to be removed from the operating system.

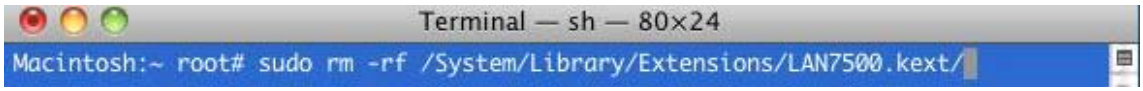

Figure 5.12 Driver Removal

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 5.13.

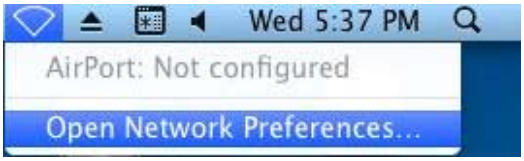

Figure 5.13 Open Network Preferences

The Network window will appear, as illustrated in Figure 5.14.

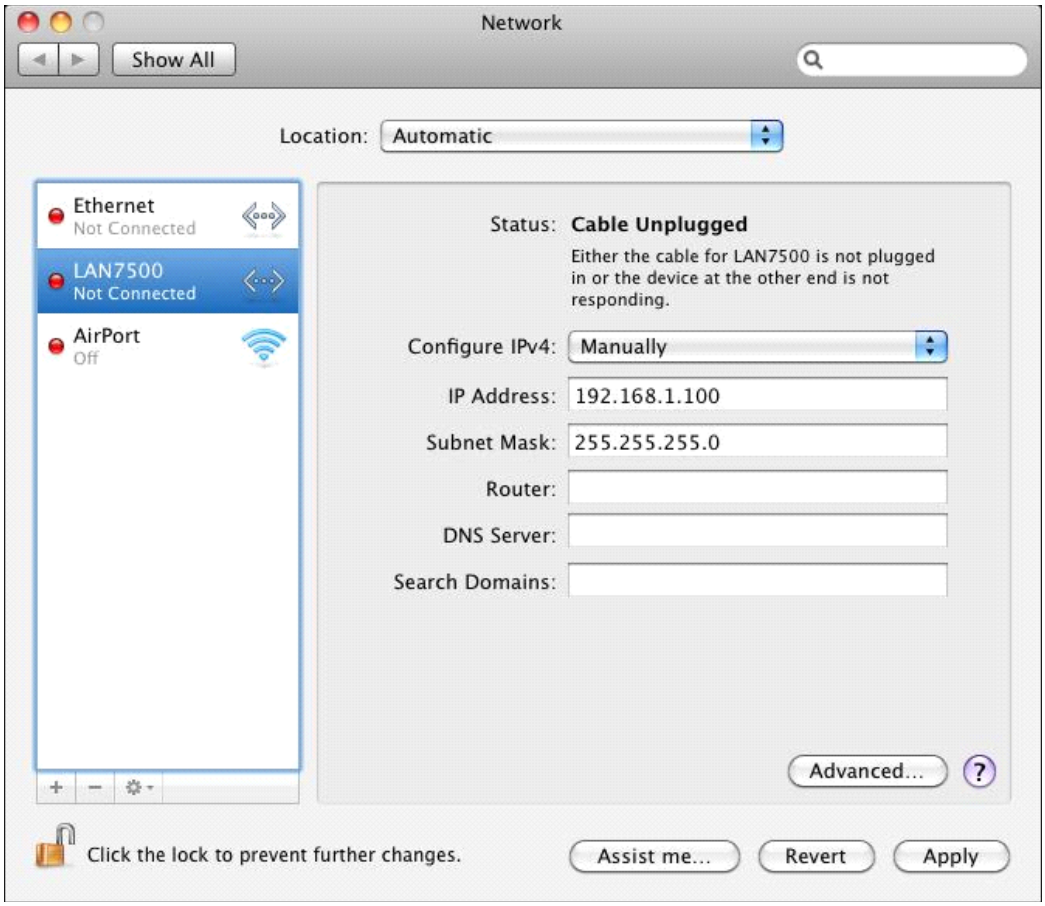

Figure 5.14 Network Preferences

Click on the LAN7500 network entry to select and highlight it. Then click the “-” button to remove the LAN7500 network entry from the operating system.

When the window is exited, a prompt to confirm and apply the changes will appear, as in Figure 5.15.

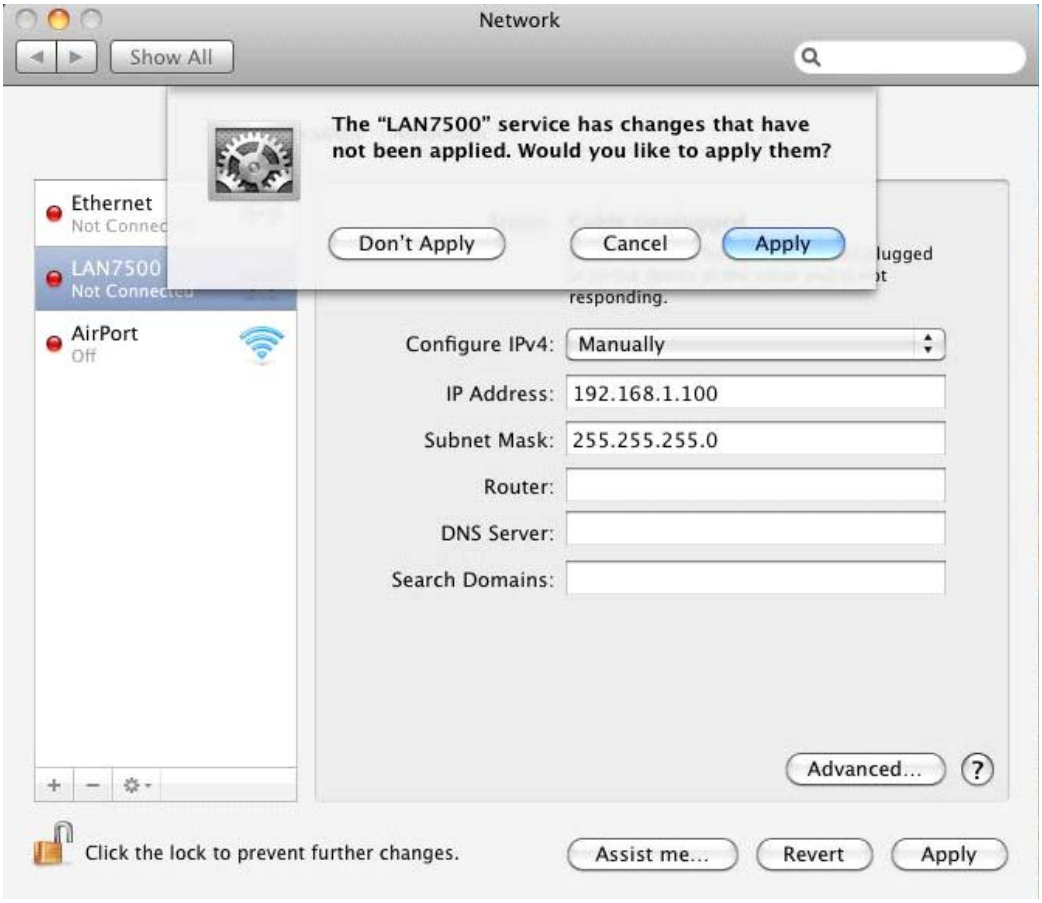

Figure 5.15 Confirm Changes

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

The device uninstallation is now complete.

Chapter 6 Linux Driver

SMSC offers two versions of the Linux drivers:

- The **official** LAN7500 Linux driver
  (available via e-services)
- The **public** LAN7500 Linux driver
  (available March/April 2010 with the Linux kernal source at kernel.org)

Both versions of the Linux drivers are distributed in source form, covered under GPLv2, and are supported by SMSC. The two versions of the Linux driver differ in that the public driver is designed to satisfy kernel LoC metrics by maximizing reuse of the existing in-kernel frameworks, despite whether such a decision results in the best performance, feature availability, etc.. Alternatively, the official driver is designed to fully utilize the LAN7500 features.

This chapter describes the installation and uninstallation of the official LAN7500 software device driver only. Refer to the standard kernel documentation distributed within the kernel for information on the installation and uninstallation of the public version of the LAN7500 software device driver.

6.1 Linux Driver Installation

The Linux LAN7500 software driver 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.

The files must be compiled as part of the software build process. The compiler gcc is required to build the software. If gcc is not installed, refer to the Linux distribution documentation for gcc installation instructions.

When the presence of gcc has been established, the build process can proceed by extracting the source files, build files, and release notes from the “tar” distribution file. Changing to the directory that the “tar” distribution file resides in and type “`tar -xzvf filename`” on the command line, where filename is the name of the “tar” distribution file. This command will create the “smsc7500” directory and populate it with the files extracted from the “tar” distribution file. An example of this can be seen in Figure 6.1.

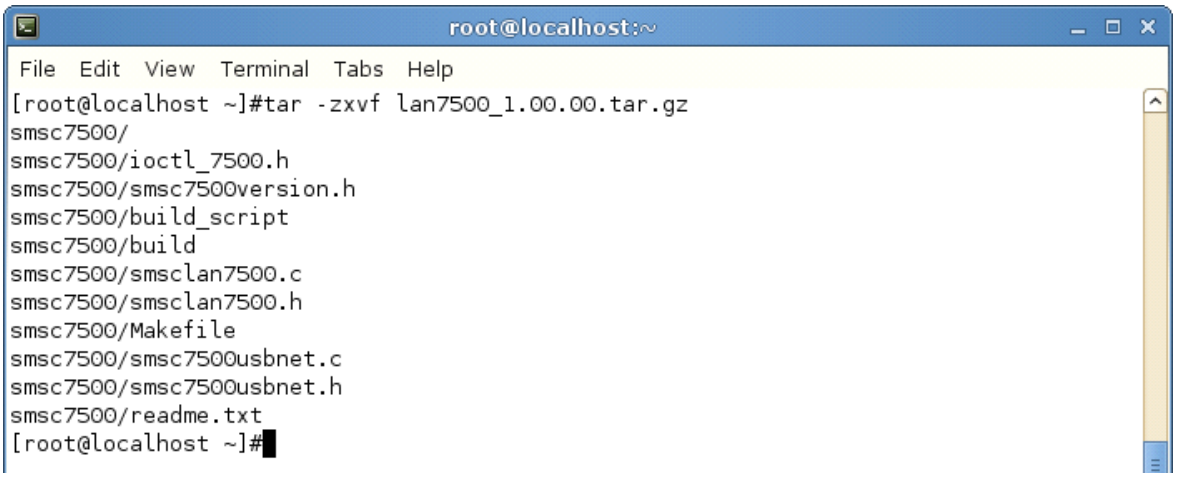

Figure 6.1 Tar of Distribution File

In order to continue the build process, switch to the smsc7500 directory via the cd command and type "`./build`" to invoke the build script, as illustrated in Figure 6.2. This example also illustrates use of the "`ls`" command to list the contents of the smsc7500 directory before invocation of the build script.

Figure 6.2 Execution of Build Script

At the completion of the build script, the module `smsc7500.ko` is generated. The module `smsc7500.ko` can be installed via the "`insmod`" command, as illustrated inFigure 6.3.

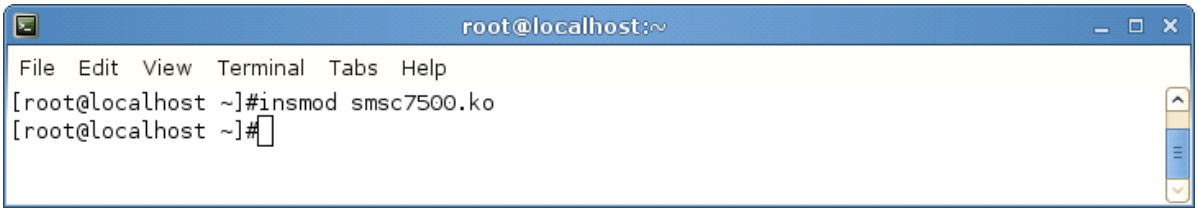

Figure 6.3 Installing the Driver

Software installation is now complete. In order to complete the hardware configuration, the LAN7500/LAN7500i device must now be plugged into an available USB port on the computer. Once

the device is plugged in, click System->Administrator->Network on the task bar to bring up the Network Configuration window, as illustrated in Figure 6.4.

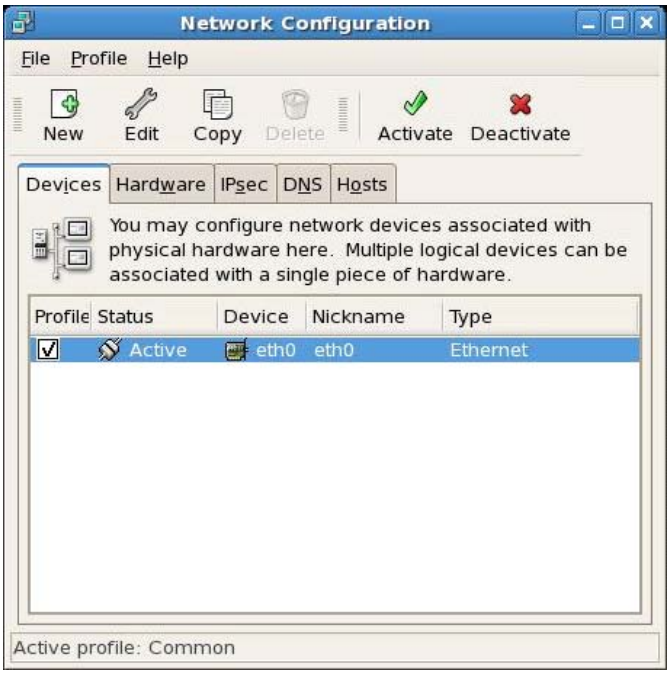

Figure 6.4 Network Configuration Window

Click “New”, to add a new device type. Figure 6.5 illustrates the resulting window.

Figure 6.5 Add New Ethernet Device Type

Select “Ethernet connection” device type and click the “Forward” button. The window will now appear as illustrated in Figure 6.6.

Figure 6.6 Device Selection

Select “Smsc7500” and click the “Forward” button. The window will now appear as illustrated in Figure 6.7.

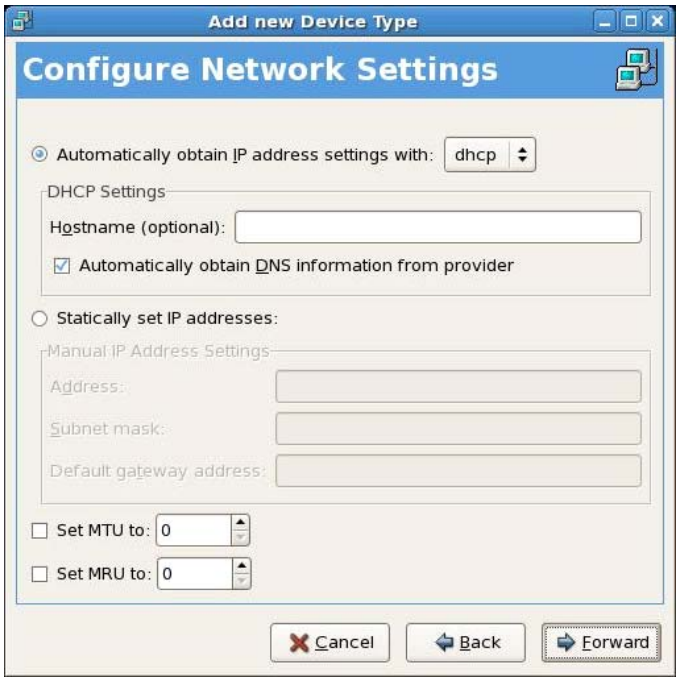

Figure 6.7 IP Address Generation

The new device must be configured to 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 6.7. Upon completion of the configuration, click the “Forward” button. The window will now appear as illustrated in Figure 6.8.

Figure 6.8 Device Creation

Click the “Apply” button to create the device. The Network Configuration screen, illustrated in Figure 6.4 on page 59, 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.

6.2 Linux Driver Uninstallation

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

Open a command terminal and bring the LAN7500 network interface down by typing "`ifconfig <interface> down`", where `<interface>` is the interface associated with the LAN7500.

Remove the LAN7500 ethernet module by typing "`rmmod smsc7500`"

These steps are illustrated in Figure 6.9.

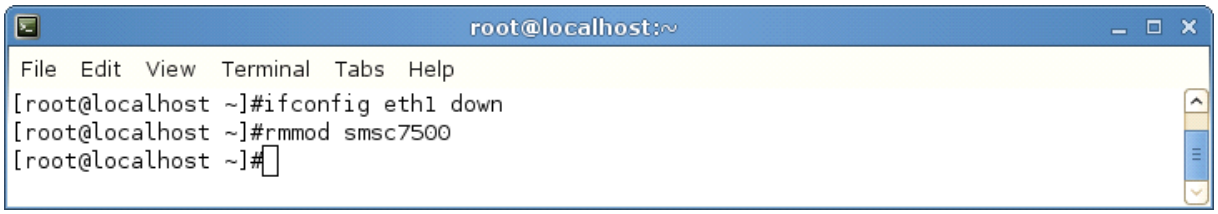

Figure 6.9 Uninstallation Steps

The device uninstallation is now complete.

Chapter 7 Windows CE Driver

This chapter details the methods for building a LAN7500/LAN7500i driver into a Windows CE image. This may be accomplished in two ways:

- Building a CE Image with LAN7500/LAN7500i Source Files
- Building a CE Image with LAN7500/LAN7500i 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

7.1 Building a CE Image with LAN7500/LAN7500i 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 “SMSC7500”. A list of the device driver source files is provided in Table 7.1.

Table 7.1 Windows CE LAN7500/LAN7500i 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.
`lan7500.h`	This module contains LAN7500/LAN7500i 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.
`smsc7500.c`	USB ethernet adapter control/bulk/interrupt transport.
`smsc7500.h`	This module contains structure definitions and function prototypes.
`smsc7500.reg`	This module contains the LAN7500/LAN7500i 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.smsc.com/main.nsf for additional information.

To build the LAN7500/LAN7500i 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 LAN7500/LAN7500i driver to the "DRIVERS" folder, for example, %_TARGETPLATROOT%\src\drivers. The folder name "smsc7500" will then be listed under the "DRIVERS" directory.

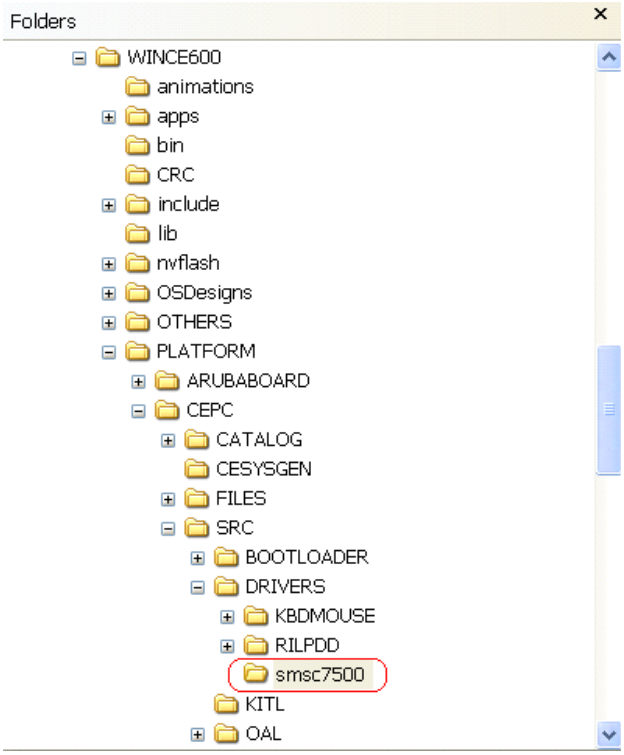

Figure 7.1 Example Unzip to DRIVERS Directory

Name	Size	Type	Date Modified
init.c	185 KB	C Source	1/25/2010 3:00 ...
interrupt.c	55 KB	C Source	1/21/2010 2:01 ...
Ioctl.c	8 KB	C Source	1/19/2010 12:04...
Ioctl.h	3 KB	C/C++ Header	1/19/2010 12:04...
lan7500.h	87 KB	C/C++ Header	2/3/2010 3:17 PM
MAKEFILE	1 KB	File	5/13/2004 9:47 ...
OS.c	1 KB	C Source	1/25/2010 12:30...
os.h	8 KB	C/C++ Header	1/19/2010 12:04...
receive.c	92 KB	C Source	1/25/2010 2:57 ...
request.c	132 KB	C Source	2/3/2010 3:17 PM
send.c	63 KB	C Source	1/20/2010 2:32 ...
smsc7500.c	86 KB	C Source	1/25/2010 2:54 ...
smsc7500.def	1 KB	Export Definition...	1/19/2010 12:04...
smsc7500.h	33 KB	C/C++ Header	1/25/2010 12:13...
smsc7500.reg	3 KB	Registration Entr...	1/25/2010 12:28...
sources	2 KB	File	1/14/2010 12:24...
version.h	2 KB	C/C++ Header	1/19/2010 12:04...

Figure 7.2 smsc7500 Folder Contents

2. The `DIRS` file, under the `%_TARGETPLATROOT%\src\drivers` folder, has a list of drivers to be compiled during the image build procedure. The `smsc7500` directory must be added to this list to allow Platform Builder to build the driver automatically.

```
!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 \
      smsc7500 \
~
-- INSERT --                                                    25,1          Bot
```

Figure 7.3 DIRS List of Drivers to be Built

3. The driver file must then be included into the target image. The `smsc7500.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.

```
        ; @XIPREGION ENDIF PLATFORM_MODULES_RIL
    ENDIF IMGFAKERIL
; @CESYSGEN ENDIF CELLCORE_MODULES_RIL

      smsc7500.dll    $(_FLATRELEASEDIR)\smsc7500.dll    NK    SHK

FILES
;  Name            Path                                          Memory Type
;  --------------  --------------------------------------------  -----------

; @CESYSGEN IF CE_MODULES_DEVICE
; @CESYSGEN ENDIF CE_MODULES_DEVICE
~
~
-- INSERT --                                                  272,36        Bot
```

Figure 7.4 Adding smsc7500.dll into platform.bib

4. The `smsc7500` folder contains the LAN7500/LAN7500i registry keys file `smsc7500.reg`. This file must be included into `%_TARGETPLATROOT%\FILES\Platform.reg` or alternatively, the contents of the `smsc7500.reg` file copied into `%_TARGETPLATROOT%\FILES\Platform.reg`.

```
; @CESYSGEN IF FP_VOIP_MODULES_BOOTSTRAP

; @CESYSGEN IF FP_VOIP_MODULES_PHRESOURCE

[HKEY_LOCAL_MACHINE\System\VoIP\Hotkeys\Bootstrap]
 "sk1"=dword:00000070        ; VK_F1
 "sk2"=dword:00000071        ; VK_F2
 "sk3"=dword:00000072        ; VK_F3
 "sk4"=dword:00000073        ; VK_F4

; @CESYSGEN ENDIF FP_VOIP_MODULES_PHRESOURCE

; @CESYSGEN ENDIF FP_VOIP_MODULES_BOOTSTRAP

#include "$(_TARGETPLATROOT)\src\drivers\smsc7500\smsc7500.reg"
```

Figure 7.5 Including smsc7500.reg in platform.reg

5. The CE image may now be built. This procedure varies, depending 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 LAN7500/LAN7500i driver.

7.2 Building a CE Image with LAN7500/LAN7500i DLL & REG Files

To build the LAN7500/LAN7500i device driver from the smsc7500.dll and smsc7500.reg files into a CE image, the following procedure must be followed:

1. Copy the smsc7500.dll and smsc7500.reg files into the %_TARGETPLATROOT%\FILES folder.

2. Include smsc7500.reg into %_TARGETPLATROOT%\FILES\Platform.reg or alternatively, the contents of the smsc7500.reg file can be copied into %_TARGETPLATROOT%\FILES\Platform.reg.

```
; @CESYSGEN ENDIF FP_VOIP_MODULES_PHRESOURCE
; @CESYSGEN ENDIF FP_VOIP_MODULES_BOOTSTRAP
#include "$(_TARGETPLATROOT)\files\smsc7500.reg"
```

Figure 7.6 Including smsc7500.reg in platform.reg

3. Add smsc7500.dll to the %_TARGETPLATROOT%\files\platform.bib file

```
        ENDIF IMGNORILTSP !
    ; @XIPREGION ENDIF PLATFORM_MODULES_RIL
  ENDIF IMGFAKERIL
; @CESYSGEN ENDIF CELLCORE_MODULES_RIL

FILES
;  Name            Path                                          Memory Type
;  --------------  --------------------------------------------  -----------

  smsc7500.dll    $(_FLATRELEASEDIR)\smsc7500.dll                     NK

-- INSERT --                                              271,1         98%
```

Figure 7.7 Adding smsc7500.dll into platform.bib

4. The CE image may now be built. This procedure varies, depending 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 LAN7500/LAN7500i 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 LAN7500/LAN7500i driver.

7.3 USBD 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 USBD interface, a USBD 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;
      }
```

7.4 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 8 Driver Customization

The GUI based Windows (XP/Vista/7) 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.

8.1 Windows Driver Customization and Localization

The Windows driver package is a self extracting executable created with Winzip Self Extractor. The driver EXE needs to be decompressed in the directory structure format. When decompressed, the following file/folder structures will be created as shown in Table 8.1.

Table 8.1 Windows XP/Vista/7 Automated Installer Package Contents

FILE NAME	DESCRIPTION
\install.exe	Executable that checks the platform and launches the appropriate DPInst executable for the architecture
\Readme.txt	Contains the latest release information
\XP\x86\DpInstx86.exe	Microsoft redistributable driver package installer utility for the x86 architecture
\XP\x86\DpInst.xml	DPInst configuration/customization file
\XP\x86\AppData\Eula409.txt	End User License Agreement (EULA) text
\XP\x86\AppData\watermark.bmp	Left margin watermark bitmap
\XP\x86\AppData\logo.bmp	Company or device logo bitmap
\XP\x86\Driver\net7500-x86-n51f.inf	NDIS 5.1 driver installation file for x86
\XP\x86\Driver\lan7500-x86-n51f.cat	NDIS 5.1 driver WHQL-signed catalog file for x86
\XP\x86\Driver\lan7500-x86-n51f.sys	NDIS 5.1 driver file for x86
\XP\x86\Driver\WdfCoInstaller01009.dll	Microsoft Wdf Coinstaller v1.9 for x86
\XP\x64\DpInstx64.exe	Microsoft redistributable driver package installer utility for the x64 architecture
\XP\x64\DpInst.xml	DPInst configuration/customization file
\XP\x64\AppData\Eula409.txt	End User License Agreement (EULA) text
\XP\x64\AppData\watermark.bmp	Left margin watermark bitmap
\XP\x64\AppData\logo.bmp	Company or device logo bitmap
\XP\x64\Driver\net7500-x64-n51f.inf	NDIS 5.1 driver installation file for x64
\XP\x64\Driver\lan7500-x64-n51f.cat	NDIS 5.1 driver WHQL-signed catalog file for x64
\XP\x64\Driver\lan7500-x64-n51f.sys	NDIS 5.1 driver file for x64
\XP\x64\Driver\WdfCoInstaller01009.dll	Microsoft Wdf Coinstaller v1.9 for x64
\Vista\x86\DpInstx86.exe	Microsoft redistributable driver package installer utility for the x86 architecture

Table 8.1 Windows XP/Vista/7 Automated Installer Package Contents

FILE NAME	DESCRIPTION
\Vista\x86\DpInst.xml	DPInst configuration/customization file
\Vista\x86\AppData\Eula409.txt	End User License Agreement (EULA) text
\Vista\x86\AppData\watermark.bmp	Left margin watermark bitmap
\Vista\x86\AppData\logo.bmp	Company or device logo bitmap
\Vista\x86\Driver\net7500-x86-n60f.inf	NDIS 6.0 driver installation file for x86
\Vista\x86\Driver\lan7500-x86-n60f.cat	NDIS 6.0 driver WHQL-signed catalog file for x86
\Vista\x86\Driver\lan7500-x86-n60f.sys	NDIS 6.0 driver file for x86
\Vista\x86\Driver\WdfCoInstaller01009.dll	Microsoft Wdf Coinstaller v1.9 for x86
\Vista\x64\DpInstx64.exe	Microsoft redistributable driver package installer utility for the x64 architecture
\Vista\x64\DpInst.xml	DPInst configuration/customization file
\Vista\x64\AppData\Eula409.txt	End User License Agreement (EULA) text
\Vista\x64\AppData\watermark.bmp	Left margin watermark bitmap
\Vista\x64\AppData\logo.bmp	Company or device logo bitmap
\Vista\x64\Driver\net7500-x64-n60f.inf	NDIS 6.0 driver installation file for x64
\Vista\x64\Driver\lan7500-x64-n60f.cat	NDIS 6.0 driver WHQL-signed catalog file for x64
\Vista\x64\Driver\lan7500-x64-n60f.sys	NDIS 6.0 driver file for x64
\Vista\x64\Driver\WdfCoInstaller01009.dll	Microsoft Wdf Coinstaller v1.9 for x64
\Win7\x86\DpInstx86.exe	Microsoft redistributable driver package installer utility for the x86 architecture
\Win7\x86\DpInst.xml	DPInst configuration/customization file
\Win7\x86\AppData\Eula409.txt	End User License Agreement (EULA) text
\Win7\x86\AppData\watermark.bmp	Left margin watermark bitmap
\Win7\x86\AppData\logo.bmp	Company or device logo bitmap
\Win7\x86\Driver\net7500-x86-n620f.inf	NDIS 6.20 driver installation file for x86
\Win7\x86\Driver\lan7500-x86-n620f.cat	NDIS 6.20 driver WHQL-signed catalog file for x86
\Win7\x86\Driver\lan7500-x86-n620f.sys	NDIS 6.20 driver file for x86
\Win7\x86\Driver\WdfCoInstaller01009.dll	Microsoft Wdf Coinstaller v1.9 for x86
\Win7\x64\DpInstx64.exe	Microsoft redistributable driver package installer utility for the x64 architecture
\Win7\x64\DpInst.xml	DPInst configuration/customization file
\Win7\x64\AppData\Eula409.txt	End User License Agreement (EULA) text
\Win7\x64\AppData\watermark.bmp	Left margin watermark bitmap
\Win7\x64\AppData\logo.bmp	Company or device logo bitmap

Table 8.1 Windows XP/Vista/7 Automated Installer Package Contents

FILE NAME	DESCRIPTION
\Win7\x64\Driver\net7500-x64-n620f.inf	NDIS 6.20 driver installation file for x64
\Win7\x64\Driver\lan7500-x64-n620f.cat	NDIS 6.20 driver WHQL-signed catalog file for x64
\Win7\x64\Driver\lan7500-x64-n620f.sys	NDIS 6.20 driver file for x64
\Win7\x64\Driver\WdfCoInstaller01009.dll	Microsoft Wdf Coinstaller v1.9 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.

8.1.1 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 six times in the package (combination of 3 operating systems and 2 architectures). However, all sets should be identical.

8.1.2 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

8.1.3 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

8.2 Mac OS X Driver Customization

Upon request, SMSC may provide the `LAN7500.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/mac/library/documentation/DeveloperTools/Conceptual/SoftwareDistribution/Introduction/Introduction.html

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

8.2.1 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 8.1 and Figure 8.2, respectively.

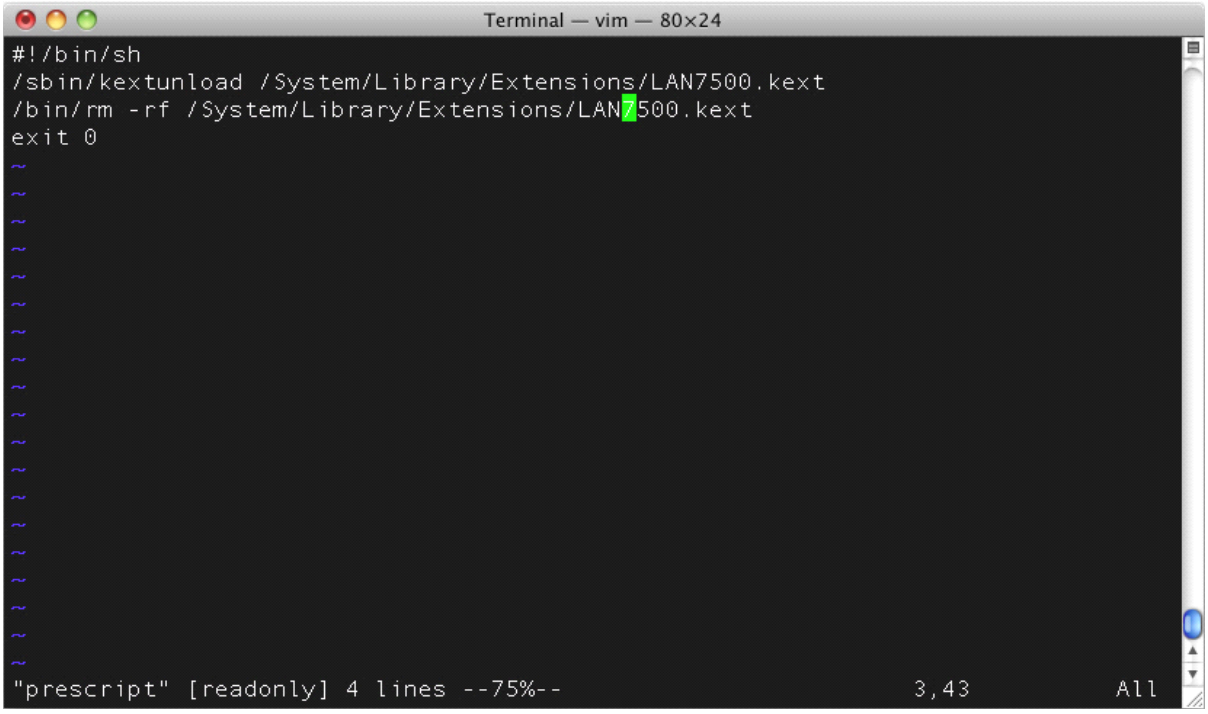

Figure 8.1 Prescript Creation

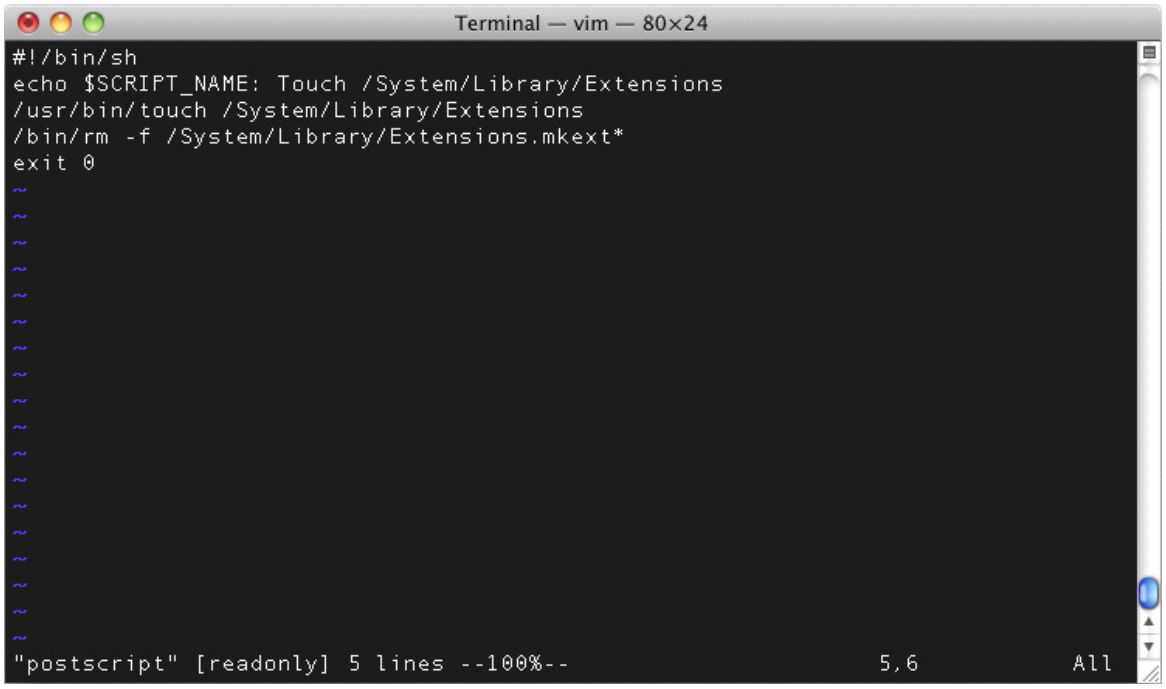

Figure 8.2 Postscript Creation

4. Execute PackageMaker and add the `LAN7500.kext` file by selecting the “+” and browsing to its location, as shown in Figure 8.3. Fill in the “Title” and “Description” fields as desired.

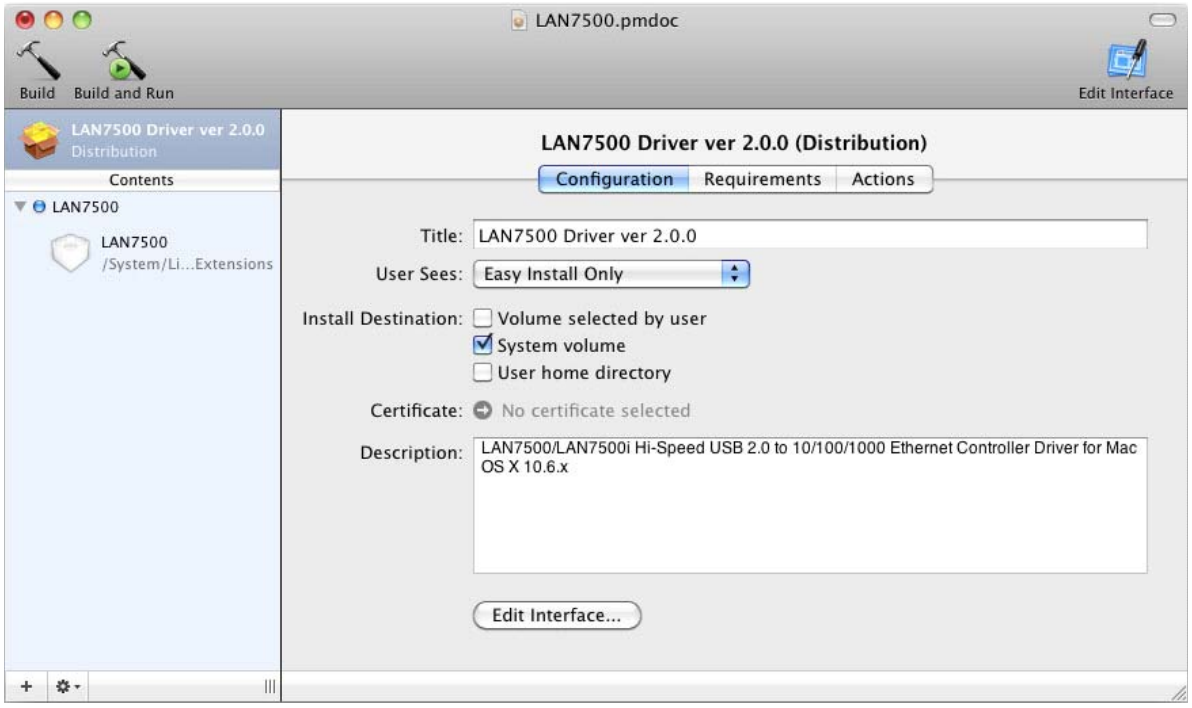

Figure 8.3 Adding LAN7500.kext

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 8.4 through Figure 8.8.

Figure 8.4 Selecting Background Image

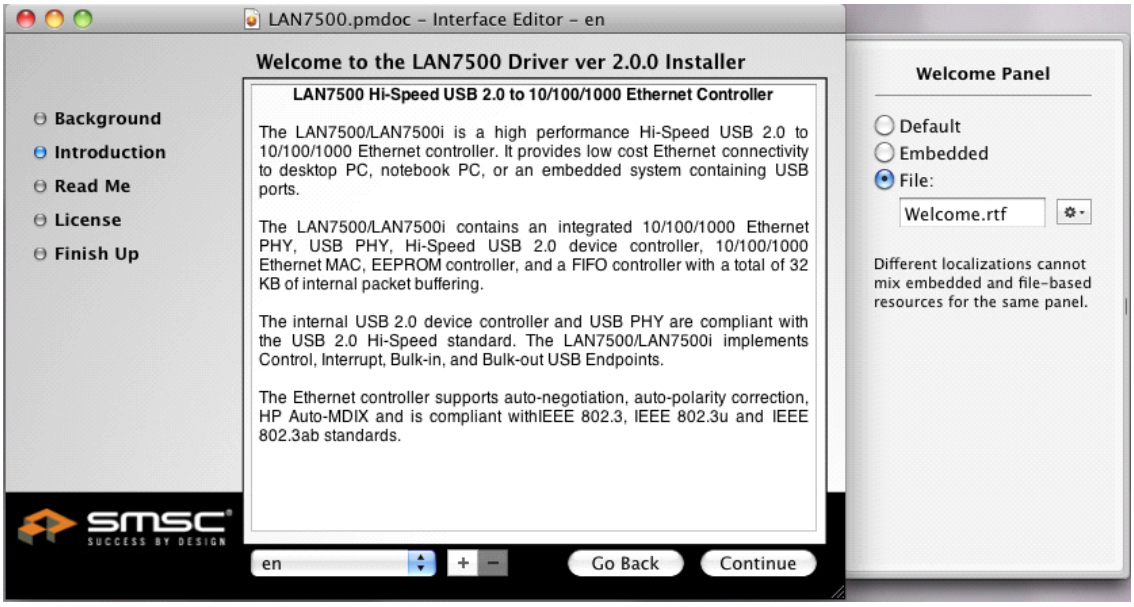

Figure 8.5 Selecting Introduction File

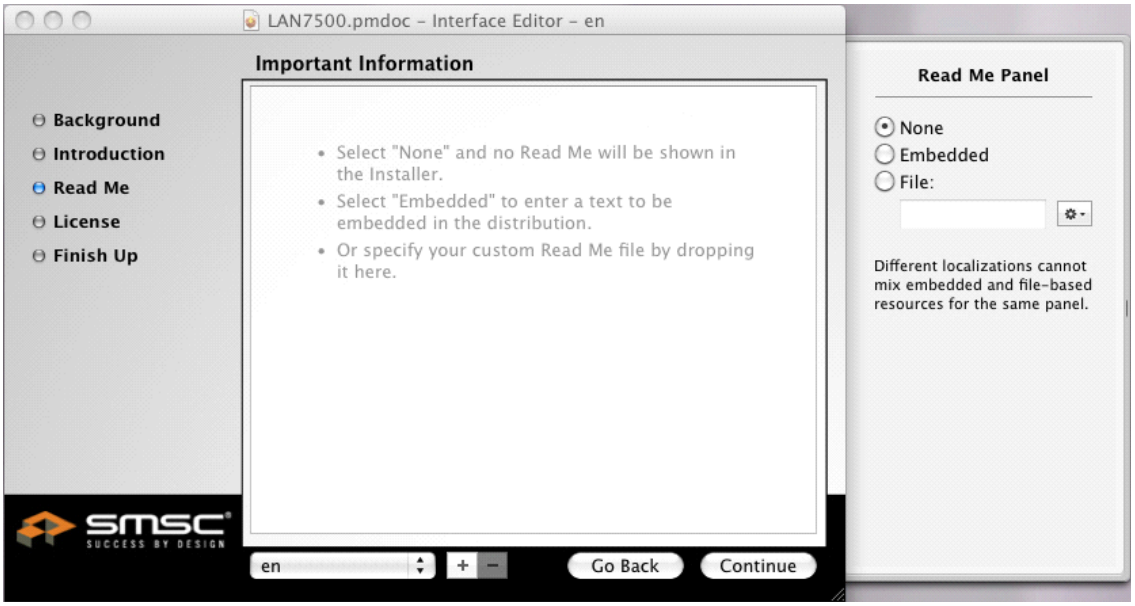

Figure 8.6 Selecting Read Me File

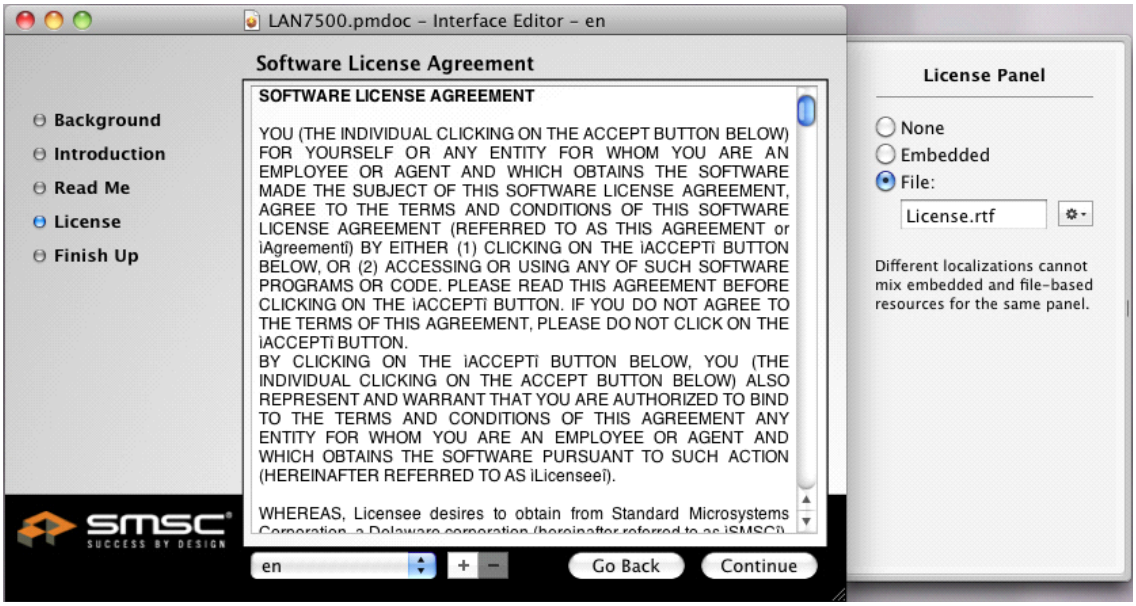

Figure 8.7 Selecting License File

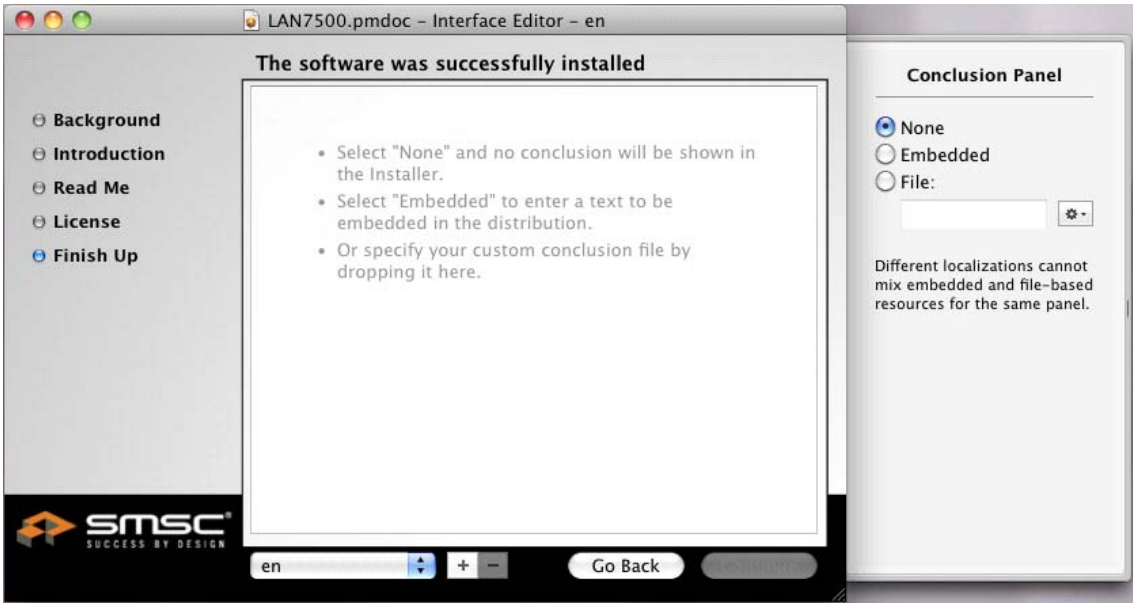

Figure 8.8 Selecting Conclusion File

7. Complete the information under the “Configuration” heading as suggested in Figure 8.9, modifying as necessary for the user’s application.

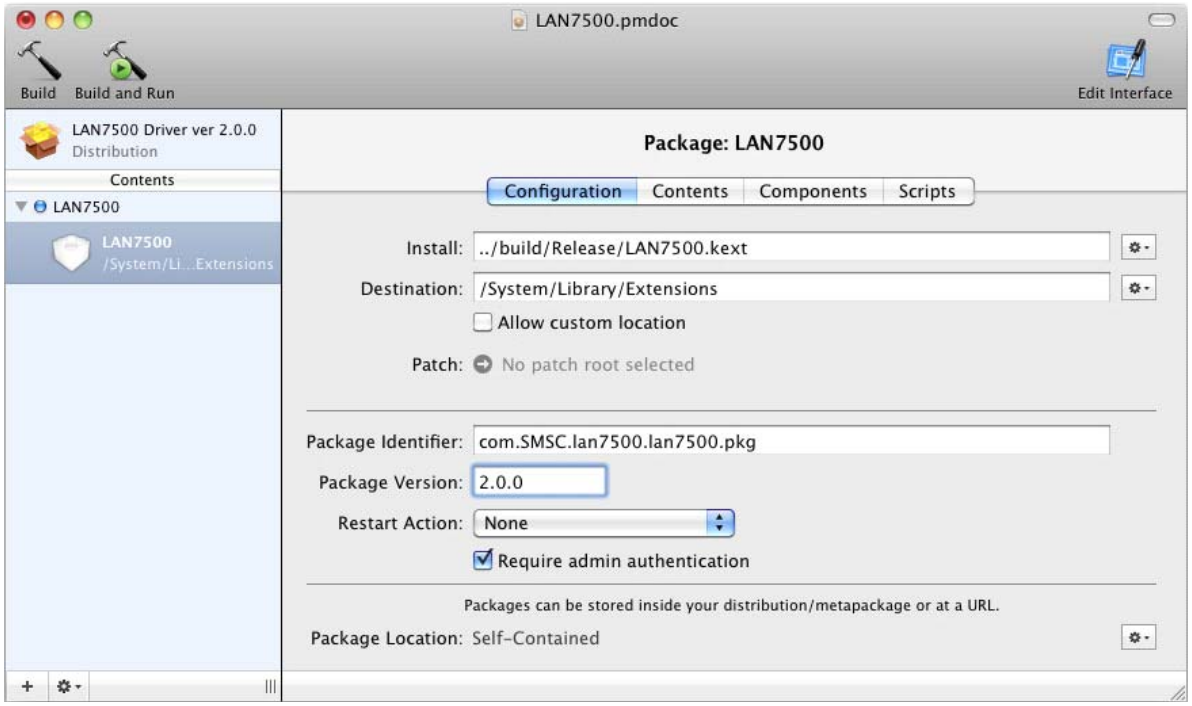

Figure 8.9 Configuration Information

8. Modify the file attributes under “Contents” heading as shown in Figure 8.10.

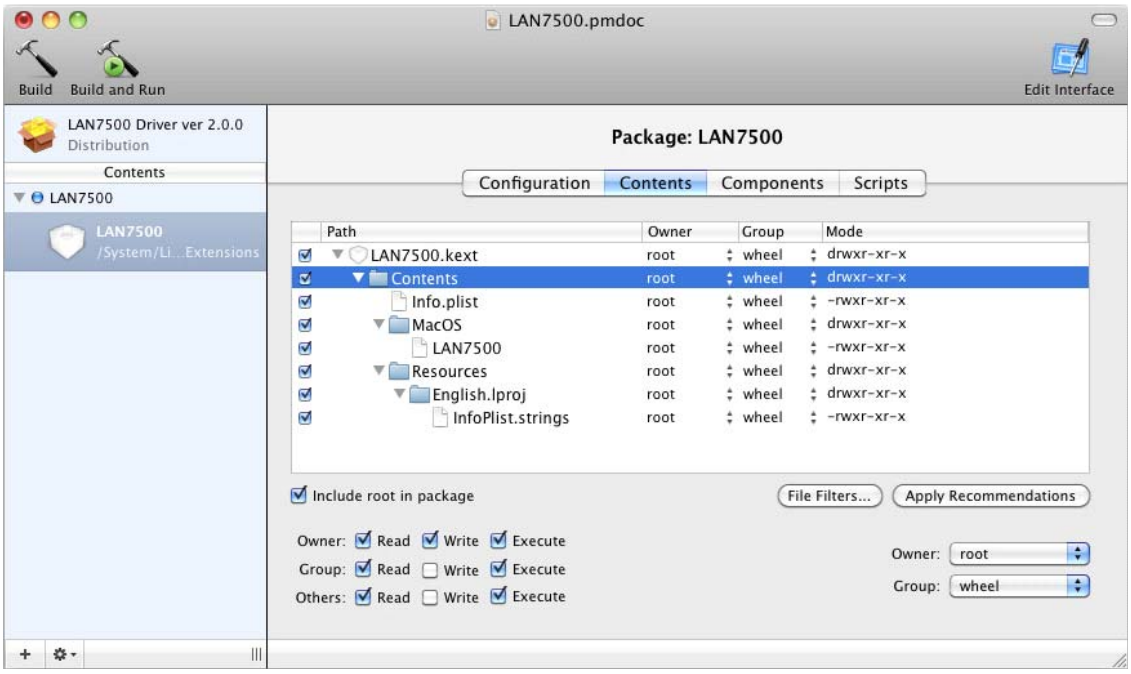

Figure 8.10 Content Information

9. Under the “Scripts” heading, add the prescript and postscript files created earlier to the Preinstall and Postinstall fields as shown in Figure 8.11.

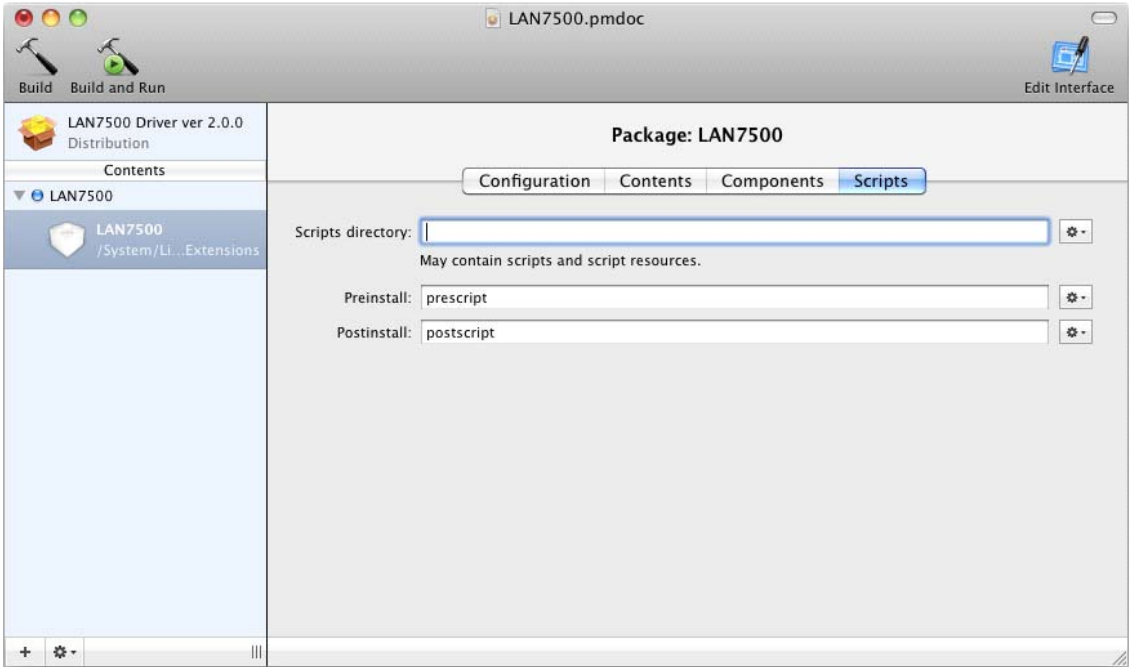

Figure 8.11 Scripts Information

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

Chapter 9 Advanced Driver Parameters

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

9.1 Windows Parameters

Windows parameters are accessible through the “Advanced” tab of the “Device Properties” window. An example of these parameters in Windows XP, Vista, and 7 can be seen in Figure 9.1, Figure 9.2, and Figure 9.3, 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.

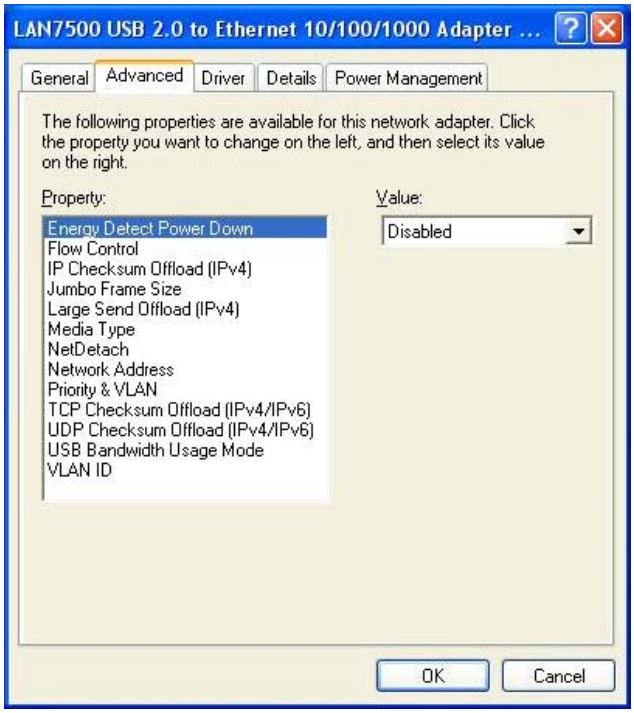

Figure 9.1 Windows XP Advanced Parameters

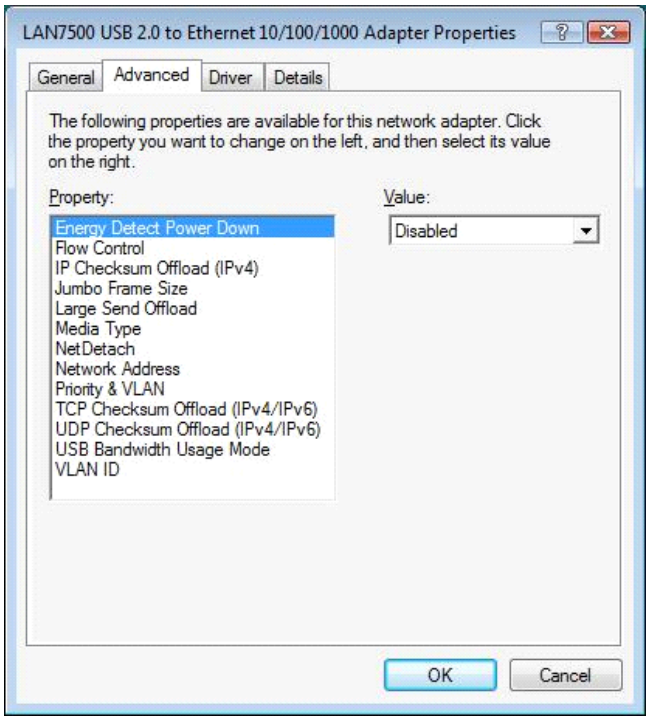

Figure 9.2 Windows Vista Advanced Parameters

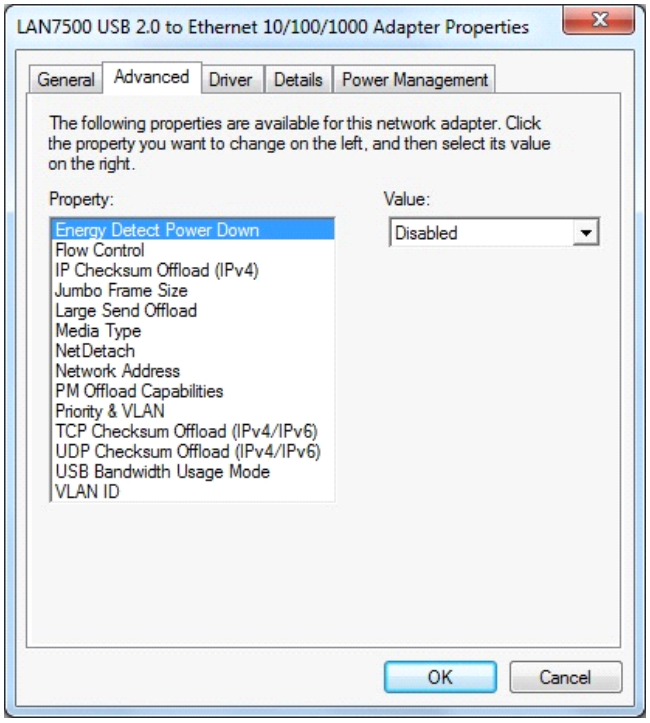

Figure 9.3 Windows 7 Advanced Parameters

9.2 Mac OS X Parameters

Mac OS X device driver parameters are accessible by customizing the `info.plist` file. After installation of the LAN7500/LAN7500i device driver, the following kernal extension (kext) will be created: `/System/Library/Extensions/LAN7500.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 9.4 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.

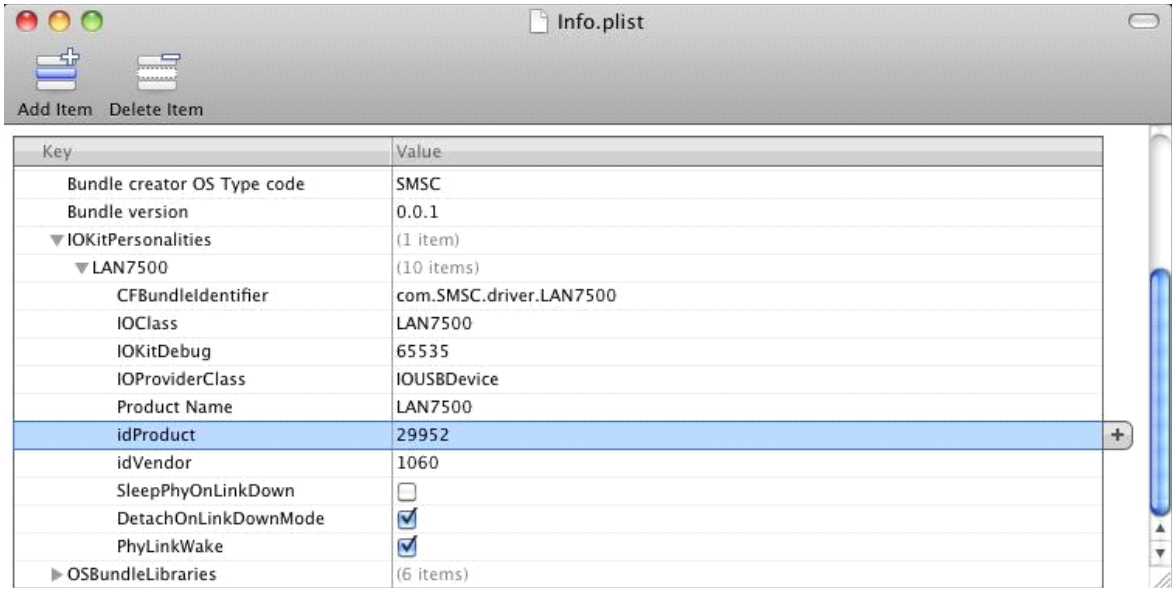

Figure 9.4 Mac OS X info.plist Advanced Parameters

9.3 Linux Parameters

Linux parameters may be modified via the following methods:

- Load Time (insmod)
- Runtime Standard (ethtool)

9.3.1 insmod

The device driver can be loaded with parameters as follows:

```
insmod smsc7500.ko [parameter1 = value1 parameter2 = value2 ... parameterN = valueN]
```

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

```
insmod smsc7500.ko operational_mode = 1
```

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.

9.3.2 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.3.3.1, "ethtool," on page 118 for information on EEPROM programming using ethtool.

9.4 Windows CE Parameters

Windows CE device driver parameters are accessible via the smsc7500.reg file. A sample of a typical smsc7500.reg file is shown below:

```
[HKEY_LOCAL_MACHINE\Comm\SMSC75001\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 10 Pre-Execution Environment (PXE) Support

SMSC provides LAN7500/LAN7500i 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 LAN7500/LAN7500i PXE-compliant firmware, contact your SMSC representative, or visit www.SMSC.com.

10.1 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 11 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: Functionality and appearance may differ between releases. For the latest release information, please refer to the release notes located at the root of the Windows Manufacturing Utility installation.

11.1 Installation

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

11.1.1 Setup and System Requirements

Table 11.1 details the Windows Manufacturing Utility setup and system requirements.

Table 11.1 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 11.4.2, "Sharing the Ping Partner Across Several Assembly Lines," on page 106). ▪ The test partner system (or switch) must be capable of (and configured to) autonegotiate advertising of all valid 10/100/1000 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. ▪ 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 7500eep.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.

11.1.2 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 LAN7500Utility_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 11.1. Click “Next” to continue.

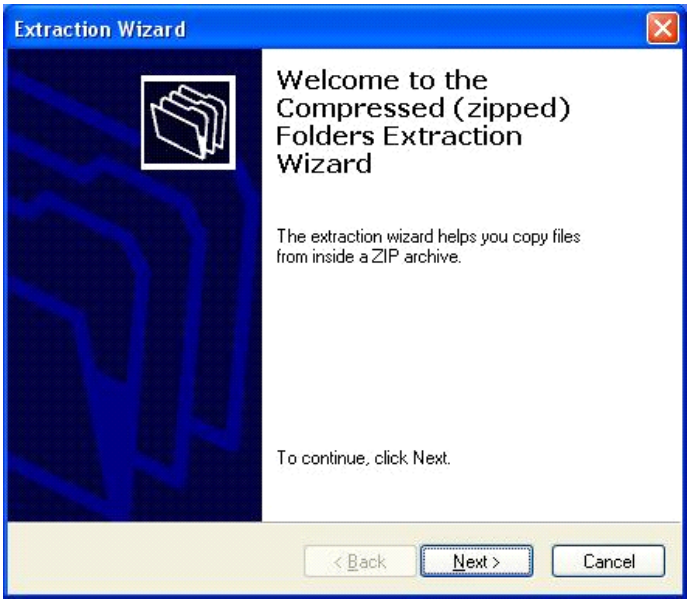

Figure 11.1 Extraction Wizard Window

Indicate the preferred directory for decompression and click “Next”, as shown in Figure 11.2.

Note: The default location is: `C:\Program Files\LAN7500Utility`

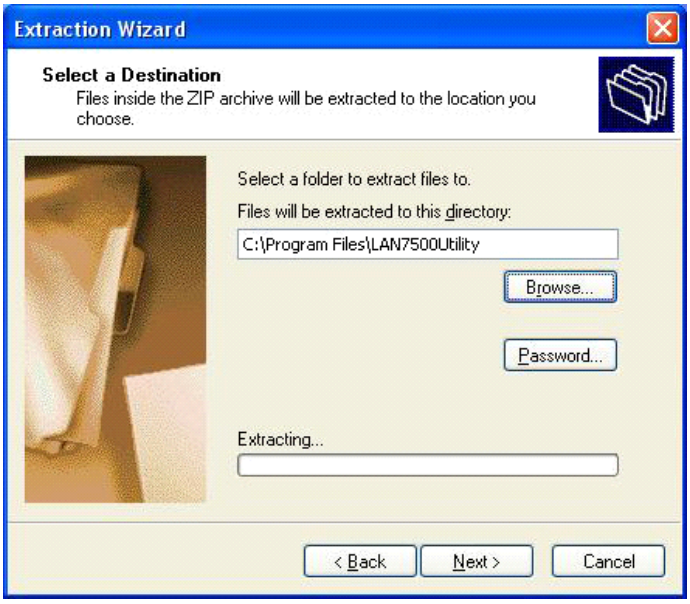

Figure 11.2 Extraction Destination Select Window

Click “Finish” to complete the decompression, as shown in Figure 11.3.

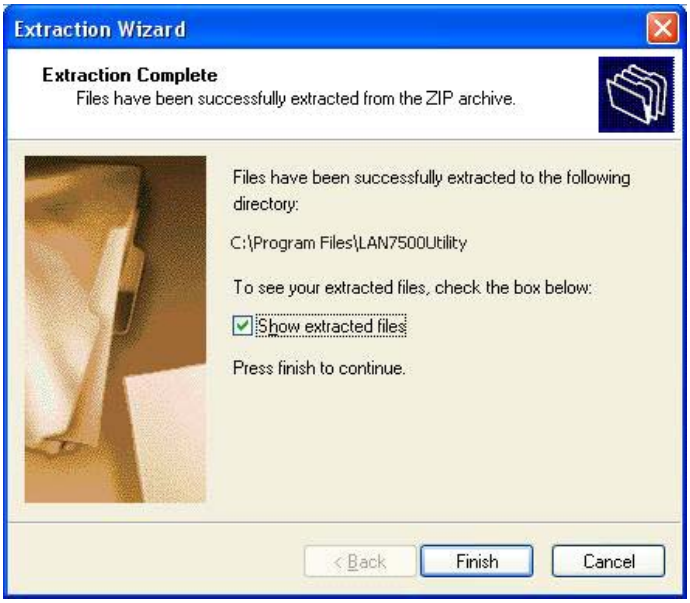

Figure 11.3 Extraction Complete Window

After the decompression is complete, the following files will be located in the chosen directory.

Table 11.2 Decompressed File Descriptions

FILE	DESCRIPTION
README.txt	A readme file detailing release notes and important information
LAN7500Utility.exe	The utility executable
7500eep.ini	The INI file where settings for the utility are saved
Driver\lan7500-x86-n51f_E2P_Utility.inf	Special driver installation script used with the LAN7500 Utility only
Driver\lan7500-x86-n51f.cat	LAN7500 catalog file
Driver\lan7500-x86-n51f.sys	LAN7500 32-bit windows device driver

Figure 11.4 Decompressed Contents

11.1.3 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 `net7500*.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).

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 11.5 Found New Hardware Wizard Welcome Window

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

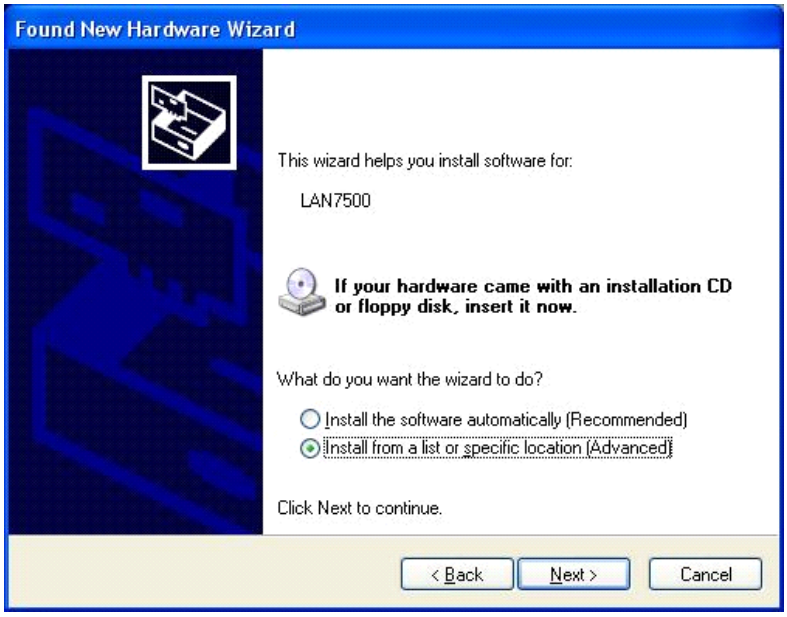

Figure 11.6 Install from a List or Specific Location Option

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

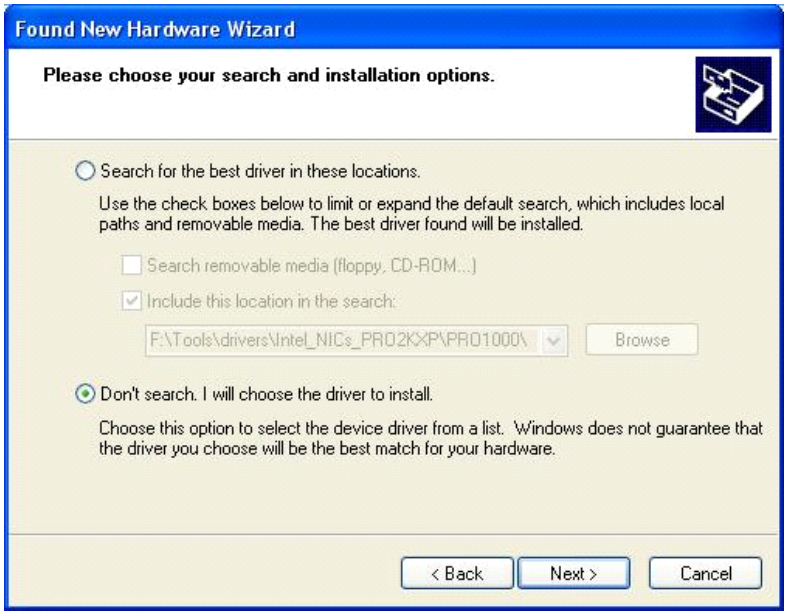

Figure 11.7 Don’t Search Option

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\LAN7500Utility\Driver`) and select “OK”.

Figure 11.8 Install From Disk Option

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

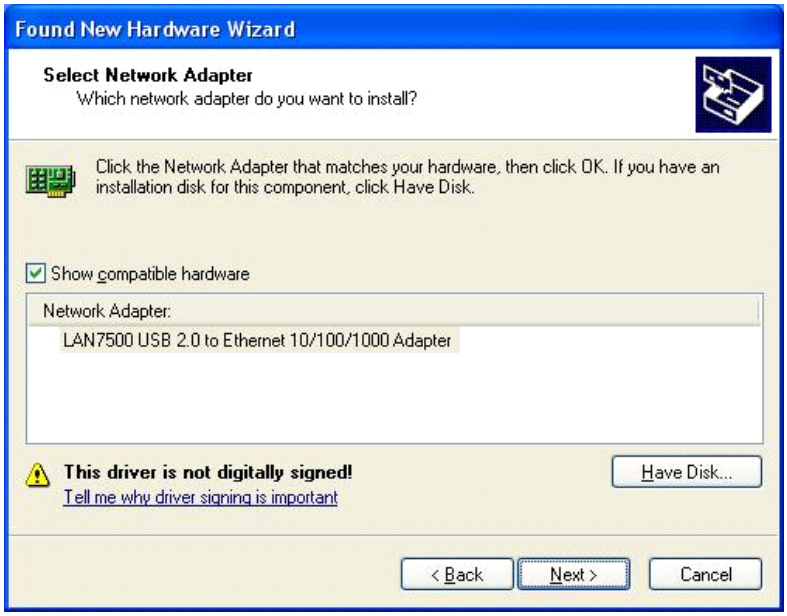

Figure 11.9 Select Network Adapter Window

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

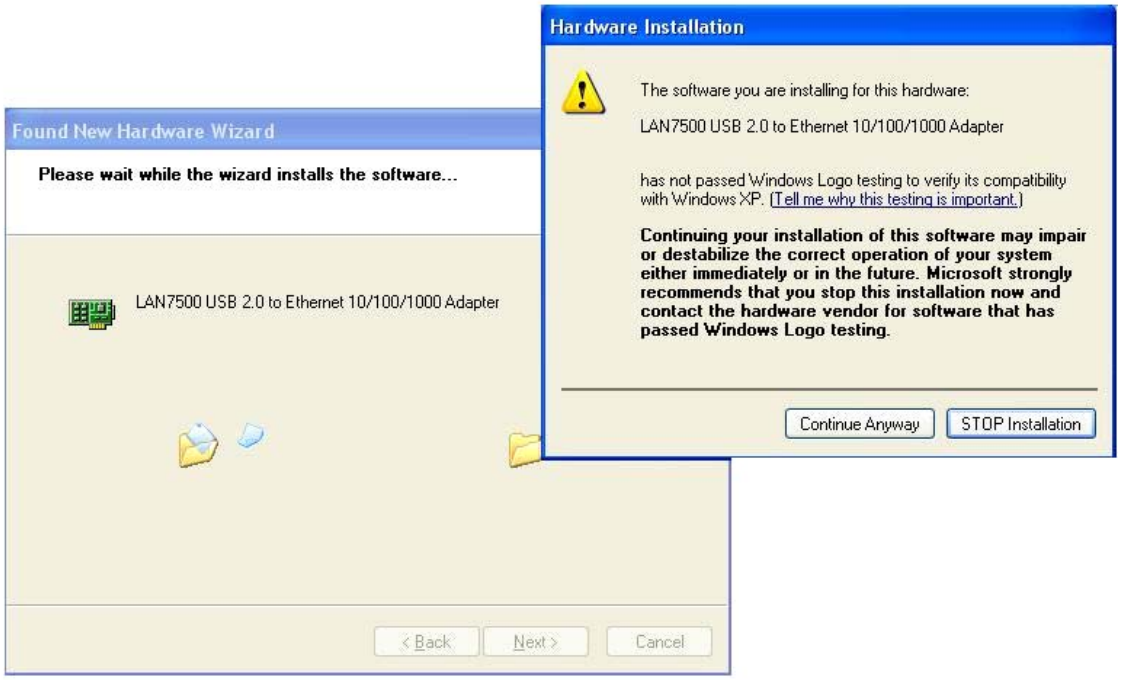

Figure 11.10 Hardware Installation Warning Window

Figure 11.11 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.

11.2 Operation

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

- Starting the Utility
- Using the Utility
- Exiting the Utility

11.2.1 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 LAN7500Utility.exe file.

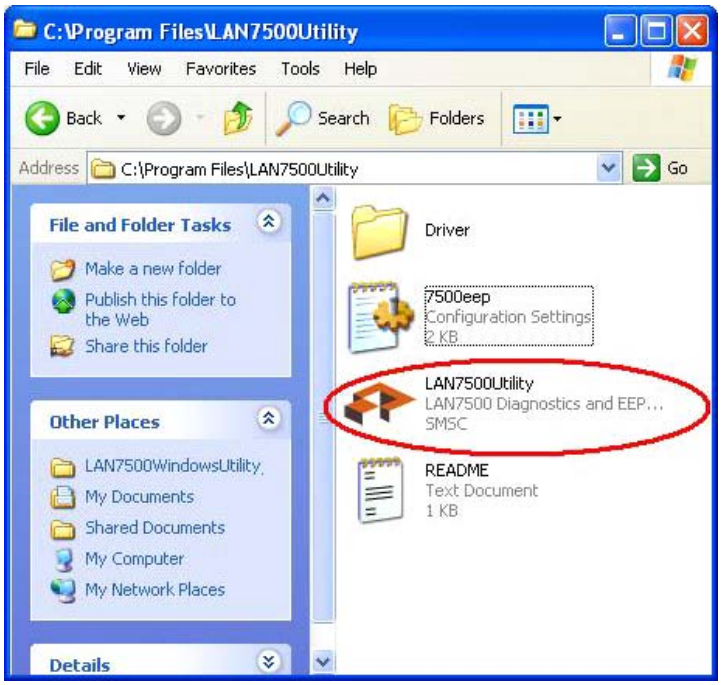

Figure 11.12 Launching the Windows Manufacturing Utility

Note: A desktop shortcut to the LAN7500Utility.exe can be created on the desktop if desired.

11.2.2 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

11.2.2.1 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 `7500eep.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 11.13. Figure 11.14 shows the drop-down GPIO/LED Configurations menu, which is accessed by clicking the "GPIO/LED Configurations" button on the EEPROM Contents Editor tab.

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

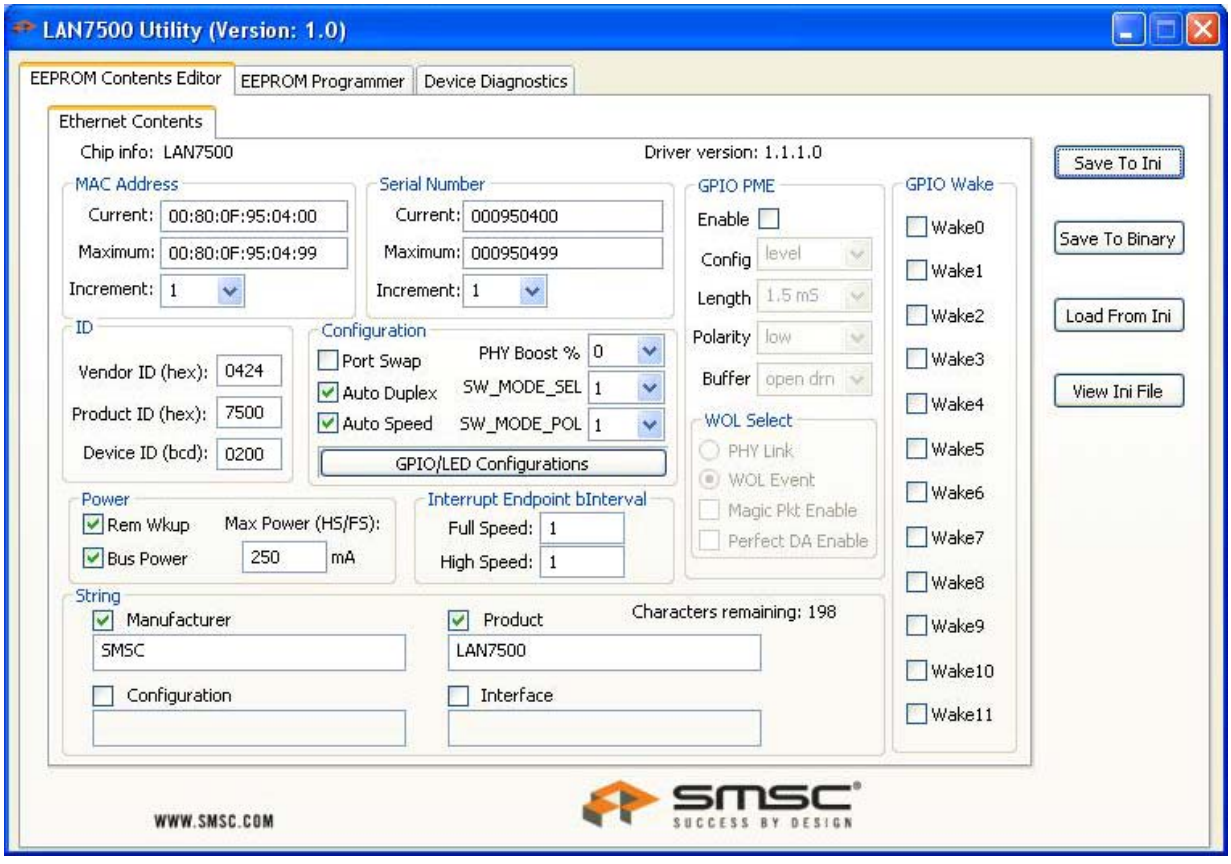

Figure 11.13 EEPROM Contents Editor Tab

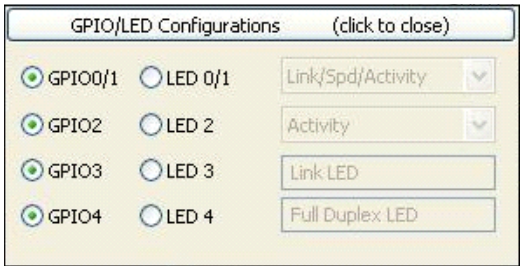

Figure 11.14 GPIO/LED Configurations Drop-Down Menu

11.2.2.1.1 EEPROM CONTENTS EDITOR FIELDS

The following fields are available for editing in this tab:

Table 11.3 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: 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: 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.
	Device ID: 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.

Table 11.3 EEPROM Contents Editor Tab - Fields (continued)

CATEGORY	FIELD DESCRIPTIONS
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.
	Rem Wkup: 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 bInterval	These fields define the interval for polling the interrupt endpoint as defined in the bInterval field of the endpoint descriptor in 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).
String	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 string lengths so all content fits within the EEPROM size (See Note 11.1).
	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.
Configuration	These parameters allow the configuration of various device hardware features.
	Port Swap: Enables/disables the swapping of the USBDP and USBDM signals (Enabled when checked).
	Auto Duplex: Enables/disables automatic duplex detection on the Ethernet port (Enabled when checked).
	Auto Speed: Enables/disables automatic speed detection on the Ethernet port (Enabled when checked).
	PHY Boost %: Selects the electrical drive strength boost percentage of the HS output current to the upstream port. The drop-down menu allows the selection of the following: **0:** 0% boost **4:** 4% boost **8:** 8% boost **12:** 12% boost
	SW_MODE_SEL: Selects the modes of operation during which the SW_MODE pin will be asserted. The drop-down menu allows the selection of the following: **0:** SW_MODE asserted in SUSPEND2 mode **1:** SW_MODE asserted in SUSPEND2, SUSPEND1, and NetDetach modes
	SW_MODE_POL: Selects the polarity of the SW_MODE pin. The drop-down menu allows the selection of the following: **0:** SW_MODE active low **1:** SW_MODE active high

Table 11.3 EEPROM Contents Editor Tab - Fields (continued)

CATEGORY	FIELD DESCRIPTIONS
GPIO/LED Configurations	This button, located in the “Configuration” section, opens a drop-down menu used to configure the GPIO/LED functionality (See Figure 11.14). Each GPIO/LED has two radio buttons which select between GPIO or LED functionality. The fields to the right of each set of radio buttons detail the specific events that a particular LED indicates. When selected for LED functionality, LED0/1 and LED2 can be programmed to indicate different events via the corresponding drop-down boxes. LED3 and LED4 cannot change which events they indicate. When a particular pin is configured for GPIO functionality, these LED indicator fields will be grayed out and ignored. **Note:** GPIO0/LED0 and GPIO1/LED1 share functionality and must be configured together as both GPIOs or LEDs. Because of this, these GPIOs and LEDs share a common set of radio buttons and a single LED indicator drop-down box.
	GPIO0/1 - LED0/1: The GPIO0/LED0 and GPIO1/LED1 functionality is selected via one of the following radio buttons: **GPIO0/1:** Pins function as GPIOs **LED0/1:** Pins function as LEDs When configured as LEDs, the indication events are selectable via the drop-down box directly to the right of the corresponding radio buttons. The following LED0/1 event indicator options are available: **Link/Spd:** LED0 and LED1 indicate link and speed **Link/Spd/Activity:** LED0 and LED1 indicate link, speed, and activity
	GPIO2/LED2: The GPIO2/LED2 functionality is selected via one of the following radio buttons: **GPIO2:** Pin functions as a GPIO **LED2:** Pin functions as an LED When configured as an LED, the indication events are selectable via the drop-down box directly to the right of the corresponding radio buttons. The following LED2 event indicator options are available: **Link/Activity:** LED2 indicates link and activity **Activity:** LED2 indicates activity
	GPIO3/LED3: The GPIO3/LED3 functionality is selected via one of the following radio buttons: **GPIO3:** Pin functions as a GPIO **LED3:** Pin functions as an LED When configured as an LED, the LED3 pin will indicate link.
	GPIO4/LED4: The GPIO4/LED4 functionality is selected via one of the following radio buttons: **GPIO4:** Pin functions as a GPIO **LED4:** Pin functions as an LED When configured as an LED, the LED4 pin will indicate full duplex operation.

Table 11.3 EEPROM Contents Editor Tab - Fields (continued)

CATEGORY	FIELD DESCRIPTIONS
GPIO PME	These parameters configure the GPIO PME functionality of the device.
	Enable: Enables/disables GPIO PME signalling on the GPIO5 pin (Enabled when checked). **Note:** When enabled, GPIO5 and GPIO6 are used as PME and PME_MODE_SEL, respectively.
	Config: Selects the GPIO PME signal type. The drop-down menu allows the selection of the following signal types: **level:** The GPIO PME is signaled via a level **pulse:** The GPIO PME is signaled via a pulse **Note:** If GPIO PME is disabled, this parameter is ignored.
	Length: When GPIO PME is enabled and configured to signal via a pulse, this parameter selects the GPIO PME pulse length. The drop-down menu allows the selection of the following durations: **0:** 1.5mS pulse length **1:** 150mS pulse length **Note:** If GPIO PME is disabled or configured to signal via a level (not pulse), this parameter is ignored.
	Polarity: Selects the signaling polarity of the GPIO PME pin. The drop-down menu allows the selection of the following: **low:** GPIO PME active low **high:** SW_MODE active high **Note:** If GPIO PME is disabled, this parameter is ignored.
	Buffer: Selects the buffer type of the GPIO PME pin. The drop-down menu allows the selection of the following: **open drn:** Open-drain / open-source driver **push pull:** Push-pull driver **Note:** When “open drn” is selected, setting the polarity low implies open drain and setting the polarity high implies open source. **Note:** If GPIO PME is disabled, this parameter is ignored.

Table 11.3 EEPROM Contents Editor Tab - Fields (continued)

CATEGORY	FIELD DESCRIPTIONS
WOL Select	This sub-section of the “GPIO PME” section selects which events will trigger a wakeup event.
	PHY Link: The PHY Link and WOL Event radio buttons select which wakeup events are supported. When PHY Link is selected, PHY linkup wakeup events are supported. **Note:** Only one of the two radio buttons (PHY Link & WOL Event) may be selected at a time.
	WOL Event: The WOL Event and PHY Link radio buttons select which wakeup events are supported. When WOL Event is selected, Wake-On-LAN (WOL) wakeup events are supported. **Note:** Only one of the two radio buttons (PHY Link & WOL Event) may be selected at a time. When WOL Event is selected, the particular WOL events supported can be selected via the Magic Pkt Enable and Perfect DA Enable checkboxes.
	Magic Pkt Enable: Enables/disables Magic Packet detection and wakeup events (Enabled when checked). **Note:** The Magic Pkt Enable checkbox is only valid when the WOL Event radio button is selected.
	Perfect DA Enable: Enables/disables Perfect DA detection and wakeup events (Enabled when checked). **Note:** The Perfect DA Enable checkbox is only valid when the WOL Event radio button is selected.
GPIO Wake	These parameters configure the GPIO wake abilities.
	Wake0 - Wake11: Enables/disables the corresponding GPIO’s ability to trigger a wake up event (Enabled when checked). **Note:** Wake0-Wake4 cannot be enabled if the corresponding GPIO is also enabled in the “GPIO/LED Configurations” sub-section. Wake5-Wake6 cannot be enabled if PME functionality is enabled in the “GPIO PME” section.

Note 11.1 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.

11.2.2.1.2 EEPROM CONTENTS EDITOR ACTIONS

The following action buttons are available in this tab:

Table 11.4 EEPROM Contents Editor Tab - Actions

BUTTON	ACTION DESCRIPTION
Save To Ini	Saves the information currently displayed in the fields detailed in Section 11.2.2.1.1, "EEPROM Contents Editor Fields" to the `7500eep.ini` file.
Save To Binary	Saves the information currently displayed in the fields detailed in Section 11.2.2.1.1, "EEPROM Contents Editor Fields" to a binary data file to be burned to the EEPROM outside of the Windows Manufacturing Utility.
Load From Ini	Reads the contents of `7500eep.ini` file and refreshes the fields detailed in Section 11.2.2.1.1, "EEPROM Contents Editor Fields" with the file contents.
View Ini File	Opens and displays the `7500eep.ini` file.

11.2.2.2 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 11.15.

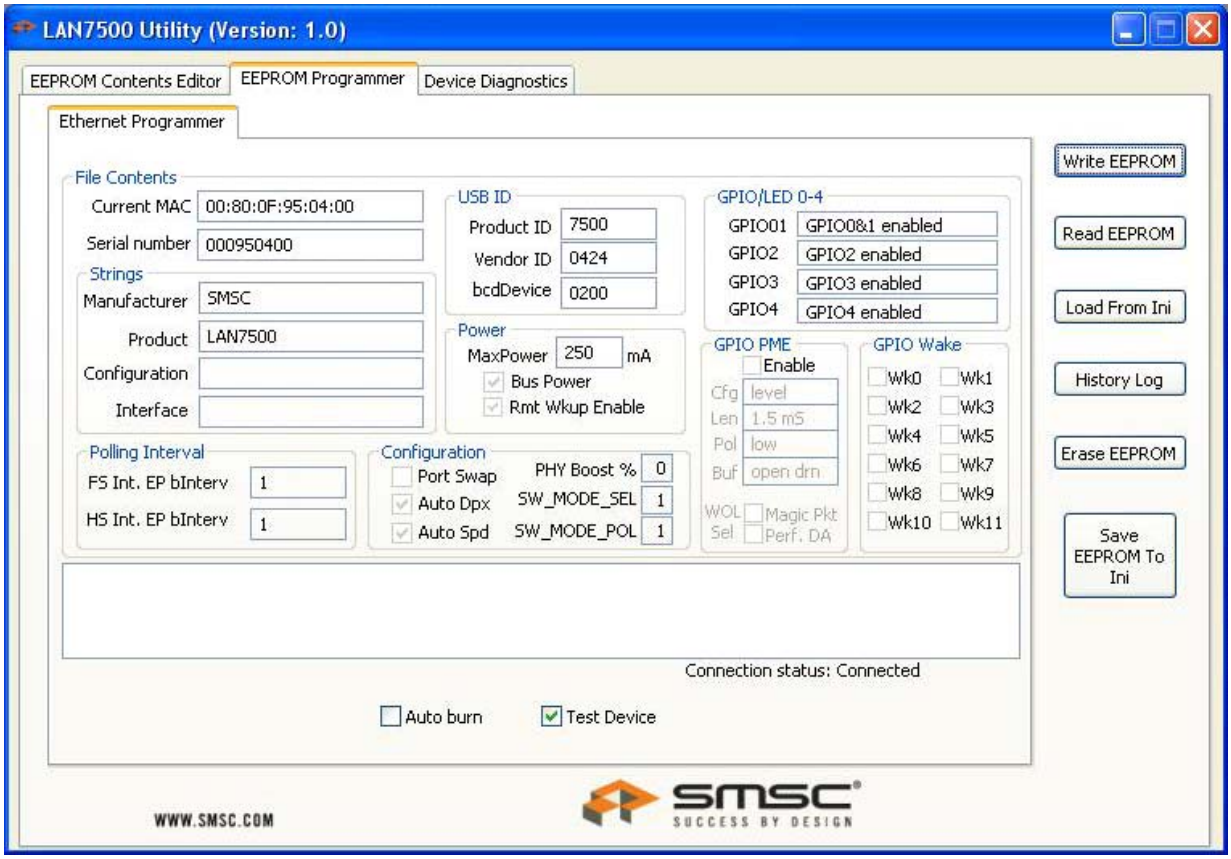

Figure 11.15 EEPROM Programmer Tab

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.

11.2.2.2.1 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 11.3, “EEPROM Contents Editor Tab - Fields,” on page 94.

Note: The “GPIO/LED 0-4” section of the EEPROM Programmer tab corresponds to the GPIO/LED Configurations drop-down sub-section of the EEPROM Contents Editor tab.

11.2.2.2.2 EEPROM PROGRAMMER TAB ACTIONS & STATUS

The following action buttons are available in this tab:

Table 11.5 EEPROM Programmer Tab - Actions

BUTTON	ACTION DESCRIPTION
Write EEPROM	This button burns the information from the `7500eep.ini` file into the EEPROM of the attached device. The status is reported in the list box and also saved in the `LAN7500Logs.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 `LAN7500Logs.txt` file.
Load From Ini	This button reads the `7500eep.ini` file and displays it's contents in the EEPROM/File Contents Fields.
History Log	This button opens the `LAN7500Logs.txt` file in text format.
Erase EEPROM	This button erases the EEPROM of the attached device.
Save EEPROM To Ini	This button reads the attached device's EEPROM and saves it to the `7500eep.ini` file.

The following checkboxes are available in this tab:

Table 11.6 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 11.7 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 `LAN7500Logs.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.
Connection Status	This text 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.

11.2.2.3 Device Diagnostics Tab

The Device Diagnostics tab allows diagnostic tests to be performed on the device. The desired tests are selected using the provided checkboxes. Status information is shown both during and after the tests are completed. The Device Diagnostics tab can be seen in Figure 11.16.

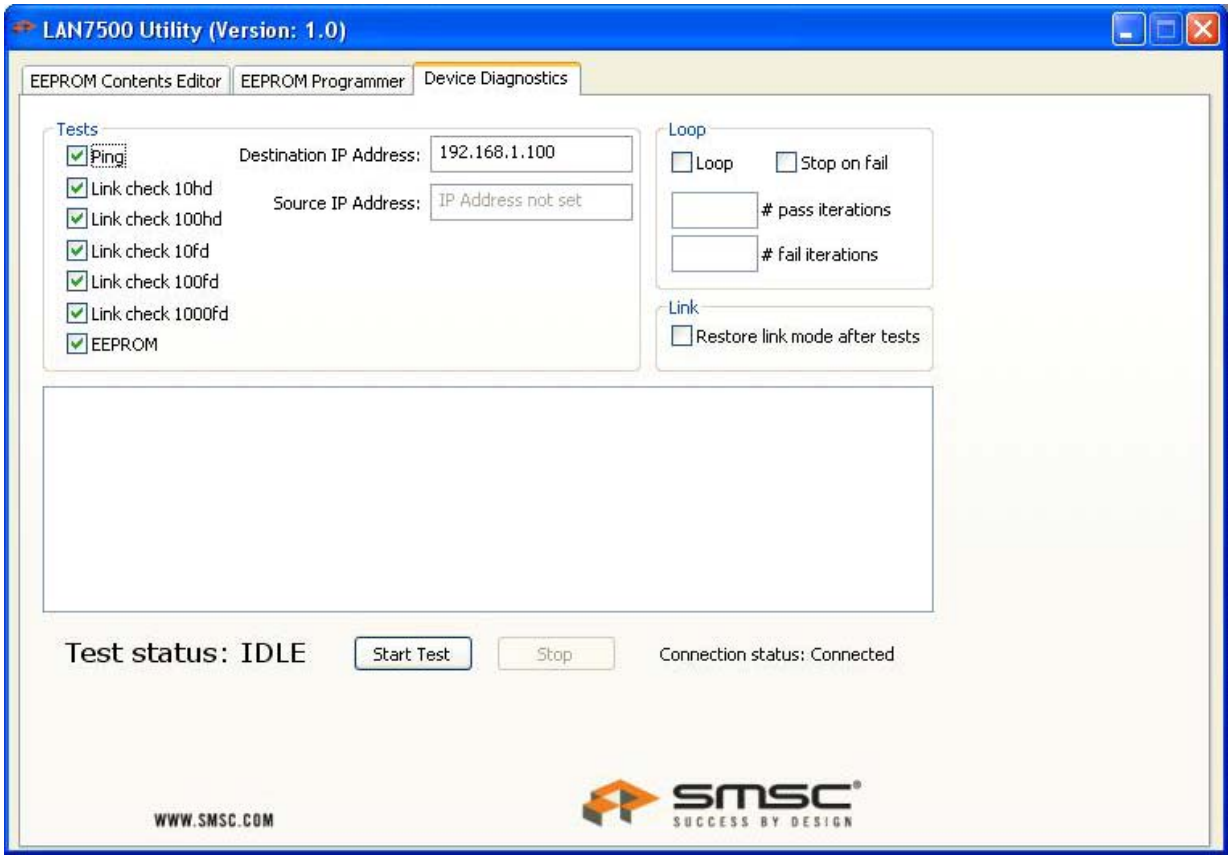

Figure 11.16 Device Diagnostics Tab

11.2.2.3.1 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 11.8 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 11.8 Device Diagnostics Tab - Tests (continued)

TESTS	DESCRIPTION
Source IP Address	In this box, enter the IP address of the device. **Note:** THE SOURCE IP MUST BE ON THE SAME SUBNET AS THE TEST PARTNER PC. This must be verified prior to testing. Otherwise, the stated results may be invalid.
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.
Link Check 1000fd	When checked, the utility will attempt to establish a 1000Mbps/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 `7500eep.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 `7500eep.ini` file.

11.2.2.3.2 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 11.9 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.

11.2.2.3.3 LINK SECTION

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

Table 11.10 Device Diagnostics Tab - Links

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

11.2.2.3.4 STATUS INFORMATION

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

Table 11.11 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 `LAN7500Logs.txt` 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.

11.2.2.3.5 DEVICE DIAGNOSTICS TAB ACTIONS

The following action buttons are available in this tab:

Table 11.12 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

11.2.3 Exiting the Utility

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

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

11.3 Production Testing Methodology and Coverage

11.3.1 Test Methodology

A Windows driver identifies a device by it's 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 it's final EEPROM contents being utilized. The production workflow assumption is that all devices are plugged into to 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 it's cycle) and connect the next one.

To address the concern of the production line not testing each device using it's 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.

11.3.2 Test Time

Performing all available tests takes approximately 15 seconds. The majority of the time is due to the autonegotiation cycles, which take approximately 2 seconds each with 5 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 + 1000FD Link test: approximately 5 seconds per device.

Note: The times provided above are with respect to a Linux PC partner. The ping response time may vary with the particular partner system utilized.

11.4 Examples

This section details examples of typical Windows Manufacturing Utility operations.

11.4.1 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.

11.4.2 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 11.1.3, "Installing the Manufacturing Utility Device Driver," on page 87.
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 11.13 for an example.

Table 11.13 Network Address Assignment Example

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

Chapter 12 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:

- 7500EEP.EXE - EEPROM programming and contents verification utility
- 7500TEST.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.

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

12.1 Environment Requirements and Limitations

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

12.1.1 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.

12.1.2 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.

12.1.3 Networking Partner / Loopback Plug

1. To use 7500TEST.EXE on a network, a network partner is REQUIRED. The partner must be on the same network as the 7500TEST.EXE application and be capable of auto-negotiation advertisement of all 10/100 half/full-duplex combinations. If utilizing 1000Mbps full-duplex testing, the partner must be capable of supporting this mode as well.
2. 7500TEST.EXE also supports external loopback mode, which requires a loopback plug.

12.2 7500EEP.EXE

The 7500EEP.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 7500EEP.EXE command line switches are used as follows:

```
7500EEP (-b <bin_filename> -i <ini_filename>) (-w -r -v)
[-M <mac_address> -S <serial_num> -m <mac_increment> -s <serial_increment>]
[-N] [-V <Vendor_ID>] [-P <Product_ID>] [-L] [-q] [-h] [-e]
```

Table 12.1 7500EEP.EXE Command Switch Definitions

COMMAND SWITCH	DESCRIPTION
-b <bin_filename>	Uses the binary file bin_filename 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 bin_filename can be overwritten using the -m, -s or -M, -S options.
-i <ini_filename>	Uses the options file ini_filename 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 ini ini_filename can be overwritten using the -m, -s or -M, -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 7500.bin 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 mac_address provided. The supported mac_address 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_increment>	Overrides the increment for the MAC address from the ini_filename 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 serial _num provided. The serial_num 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 ini_filename if the -i option is used, or the default of 0 (no increment) if the -b option is used.
-N	Updates the ini or bin file with the next MAC address and serial number. These values are computed using mac_increment and serial_increment.
-V <Vendor_ID>	Specifies the Vendor ID of the device. This command switch is optional. If specified, it must be used in conjunction with the -P command switch. If not specified, the internal chip ID register will be used to identify the device.

Table 12.1 7500EEP.EXE Command Switch Definitions

COMMAND SWITCH	DESCRIPTION
`-P <Product_ID>`	Specifies the Product ID of the device. This command switch is optional. If specified, it must be used in conjunction with the `-V` command switch. If not specified, the internal chip ID register will be used to identify the device.
`-L`	Enables software loop to program multiple devices. If not specified, the application will exit after programming one device.
`-q`	Quiet operation. This option will disable some messages.
`-h`	Displays help menu.
`-e`	Displays PASS or FAIL message in large letters before exiting the program.

12.2.1 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:

```
7500EEP -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:

```
7500EEP -b filename.bin -w -M
```

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

```
-S
```

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

```
7500EEP -b filename.bin -w -M 00:11:22:33:44:55 -S 012345678
```

12.2.2 7500EEP ini Format

The items in the 7500EEP ini are self-explanatory based upon their names. An illustrative example `7500eep.ini` is shown for reference:

Note: The 7500EEP ini format is identical to the format used by the LAN7500/LAN7500i Windows Manufacturing Utility. See Section 11.2.2.1.1, "EEPROM Contents Editor Fields," on page 94 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=LAN7500
ProductEnable=1
ConfigurationString=
ConfigurationEnable=0
InterfaceString=
InterfaceEnable=0
[ID]
VendorID=0424
ProductID=7500
BcdDevice=0200
[PollingInterval]
HighSpeed=1
FullSpeed=1
[Power]
BusPower=1
RemoteWakeupEnable=1
MaxPower=250
[ConfigFlags0]
PortSwap=0
PHYBoost=0
DuplexDetection=1
SpeedDetection=1
SpeedLEDFunction=1
[GPIOWake]
GPIOWake0=0
GPIOWake1=0
GPIOWake2=0
GPIOWake3=0
GPIOWake4=0
GPIOWake5=0
GPIOWake6=0
GPIOWake7=0
GPIOWake8=0
GPIOWake9=0
GPIOWake10=0
GPIOWake11=0
[GPIOPME]
GPIOPMEEnable=0
GPIOPMEConfiguration=0
GPIOPMELength=0
GPIOPMEPolarity=0
GPIOBufferType=0
GPIOPMEWOLSelect=0
PMEMagicPacketEnable=0
PMEPerfectDAEnable=0
[ConfigFlags1]
LED2Function=0
SWModeSelect=0
SWModePolarity=0
GPIOEnable0=1
GPIOEnable1=1
GPIOEnable2=1
GPIOEnable3=1
GPIOEnable4=1
```

12.3 7500TEST.EXE

The 7500TEST.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 required for the `-d` ping option. The partner device's system is required to be in the same network segment as 7500TEST.EXE:

The 7500TEST.EXE command line switches are used as follows:

```
7500TEST [-l <link_mode>] [-d <test_mode>] [-i <ip_address> -I <ip_address>
-V <Vendor_ID> -P <Product_ID> -M <MAC_address>] [-q] [-h] [-e] [-r]
[-g <gpio_pin> -s <gpio_state> -T <gpio_type> -t <gpio_assertion_time>]
```

Table 12.2 7500TEST.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 "`1000fd`" - for 1000BASE-T 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 <test_mode>`	The utility will perform data passing tests. Valid `test_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 LAN7500/LAN7500i device. It is required when the "`-d ping`" option is present. The IP address used must be a valid unicast address on the same subnet as the test partner.
`-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.
`-M <MAC_address>`	Specifies the MAC address to be written into the MAC address register instead of reading it from EEPROM.

Table 12.2 7500TEST.EXE Command Switch Definitions (continued)

COMMAND SWITCH	DESCRIPTION
-q	Quiet operation.
-h	Displays help menu.
-e	Displays PASS or FAIL message in large letters before exiting the program.
-r	Reloads the Ethernet EEPROM contents before beginning the test.
-g <gpio_pin>	Specifies the GPIO pin number. Valid gpio_pin values are "0" through "11".
-s <gpio_state>	Specifies the GPIO state. Valid gpio_state values are "0" (off) or "1" (on). **Note:** When starting the utility, the device will be reset. After the rest, the GPIO configuration registers will be set to their default values.
-T <gpio_type>	Specifies the GPIO type. Valid gpio_type values are "0" (open drain) or "1" (push-pull).
-t <gpio_assertion_time>	Specifies the GPIO pin assertion time in seconds (0 by default). If gpio_assertion_time is 0, the state will not be altered after exiting the utility. If gpio_assertion_time is greater than 0, the state will be toggled after the specified time.

APPENDIX A: EEPROM

A.1 EEPROM Format

This section details the EEPROM contents for the LAN7500/LAN7500i. 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 the device descriptor, the only valid values for the length are 0 and 18.

Note: For the configuration and interface descriptor, 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.

Table A.1 EEPROM Format

EEPROM ADDRESS	EEPROM CONTENTS
00h	A5h (EEPROM Programmed Indicator)
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 0
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)

Table A.1 EEPROM Format (continued)

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	GPIO[7:0] Wakeup Enables Bit x = 0 -> GPIOx Pin Disabled for Wakeup Use. Bit x = 1 -> GPIOx Pin Enabled for Wakeup Use.
1Fh	GPI0[11:8] 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 7:4 Unused.
20h	GPIO PME Flags
21h	Configuration Flags 1

Note: EEPROM byte addresses past 21h can be used to store data for any purpose assuming these addresses are not used for descriptor storage.

Table A.2 describes the Configuration Flags 0 byte. If a configuration descriptor exists in the EEPROM, it will override the values in Configuration Flags 0.

Table A.2 Configuration Flags 0

BITS	DESCRIPTION
7	**Port Swap** This bit facilitates swapping the mapping of USBDP and USBDM. 0 = USBDP maps to the USB D+ line and USBDM maps to the USB D- line. 1 = USBDP maps to the USB D- line. USBDM maps to the USB D+ line.
6:5	**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).
4	**Duplex Detection** This bit determines whether duplex operational mode is detected automatically or manually set. 0 = Manual 1 = Automatic

Table A.2 Configuration Flags 0 (continued)

<table>
<tr><th>BITS</th><th>DESCRIPTION</th></tr>
<tr><td>3</td><td>Speed Detection
This bit determines whether operational speed is detected automatically or manually set.

0 = Manual
1 = Automatic</td></tr>
<tr><td>2</td><td>SPD_LED_FUNCTION
This bit specifies the functionality of speed LEDs (LED0 and LED1). The Speed LEDs’ behavior is determined by line speed and the setting of this bit, as indicated in following table:

<table>
<tr><th>SPD_LED_FUNCTION</th><th>SPEED (Mbps)</th><th>LED0</th><th>LED1</th></tr>
<tr><td>0</td><td>No Link</td><td>Off</td><td>Off</td></tr>
<tr><td>0</td><td>10</td><td>On</td><td>Off</td></tr>
<tr><td>0</td><td>100</td><td>Off</td><td>On</td></tr>
<tr><td>0</td><td>1000</td><td>On</td><td>On</td></tr>
<tr><td>1</td><td>No Link</td><td>Off</td><td>Off</td></tr>
<tr><td>1</td><td>10</td><td>Blink</td><td>Off</td></tr>
<tr><td>1</td><td>100</td><td>Off</td><td>Blink</td></tr>
<tr><td>1</td><td>1000</td><td>Blink</td><td>Blink</td></tr>
</table>

When SPD_LED_FUNCTION = 0, the LEDs function solely as Link and Speed LEDs. When SPD_LED_FUNCTION = 1, the LEDs function as Link and Speed and Activity LEDs. In those cases, the table entry “Blink” indicates the LED will remain on when no transmit or receive activity is detected and will blink at an 80 mS rate whenever TX or RX activity is detected.

Note: GPIOEN[1:0] in Configuration Flags 1 must be set in order to properly control speed LED operation. If only one of the bits is set, then untoward operation and unexpected results may occur. If both bits are clear, then SPD_LED_FUNCTION is ignored.</td></tr>
<tr><td>1</td><td>Remote Wakeup Support

0 = Device does not support remote wakeup.
1 = Device supports remote wakeup.</td></tr>
<tr><td>0</td><td>Power Method

0 = Device is bus powered.
1 = Device is self powered.</td></tr>
</table>

Table A.3 describes the Configuration Flags 1.

Table A.3 Configuration Flags 1

BITS	DESCRIPTION
7	**LED2_FUNCTION** This bit specifies the functionality of LED2. 0 = Link and Activity LED. 1 = Activity LED. **Note:** This bit is ignored if GPIOEN2 is not set in this flag byte.
6:2	**GPIOEN[4:0]** This field specifies GPIO/LED functionality for GPIO[4:0]. 0 = GPIOn Pin Functions as GPIO pin. 1 = GPIOn Pin Functions as LED.
1	**SW_MODE_SEL** This bit specifies the modes of operation during which the SW_MODE pin will be asserted. 0 = SW_MODE asserted in SUSPEND2. 1 = SW_MODE asserted in SUSPEND2, SUSPEND1, and NetDetach.
0	**SW_MODE_POL** This bit selects the polarity of the SW_MODE pin. 0 = Active low. 1 = Active high.

Table A.4 describes the GPIO PME flags.

Table A.4 GPIO PME Flags

BITS	DESCRIPTION
7	**GPIO PME Enable** Setting this bit enables the assertion of the GPIO5 pin, as a result of a Wakeup (GPIO) pin, Magic Packet, or PHY Link Up. The host processor may use the GPIO5 pin to asynchronously wake up, in a manner analogous to a PCI PME 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 GPIO5 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 GPIO5 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.

Table A.4 GPIO PME Flags (continued)

BITS	DESCRIPTION
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 GPIO5. 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** Four types of wakeup events are supported; Magic Packet, Perfect DA, 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 WOL events or Link Up wakeup events are supported. 0 = WOL event wakeup supported. 1 = PHY linkup wakeup supported. **Note:** If WOL is selected, the PME Magic Packet Enable and PME Perfect DA Enable bits determine the WOL event(s) that will cause a wakeup. **Note:** If GPIO PME Enable is 0, this bit is ignored.
1	**PME Magic Packet Enable** When GPIO PME WOL Select indicates WOL is selected, this bit enables/disables Magic Packet detection and wakeup. 0 = Magic Packet event wakeup disabled. 1 = Magic Packet event wakeup enabled. **Note:** This bit is ignored if GPIO PME WOL Select indicates WOL event wakeup not supported.
0	**PME Perfect DA Enable** When GPIO PME WOL Select indicates WOL is selected, this bit enables/disables Perfect DA detection and wakeup. 0 = Perfect DA event wakeup disabled. 1 = Perfect DA event wakeup enabled. **Note:** This bit is ignored if GPIO PME WOL Select indicates WOL event wakeup not supported.

A.2 EEPROM Defaults

The signature value of A5h is stored at address 0. A different signature value indicates to the EEPROM controller that no EEPROM or an un-programmed EEPROM is attached to the device. In this case, the hardware default values are used, as shown in Table A.5.

Table A.5 EEPROM Defaults

FIELD	DEFAULT VALUE
MAC Address	FFFFFFFFFFFFh
Full-Speed Polling Interval (mS)	01h
Hi-Speed Polling Interval (mS)	04h
Configuration Flags 0	1Bh
Maximum Power (mA)	FAh
Vendor ID	0424h
Product ID	7500h

A.3 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.3.1 EEPROM Programming in Windows

EEPROM programming in Windows is accomplished via the Windows Manufacturing Utility. Refer to Chapter 11, "Windows Manufacturing Utility," on page 84 for detailed information on the usage of this tool.

A.3.2 EEPROM Programming in DOS

EEPROM programming in DOS is accomplished via the DOS Utility Suite. Refer to Chapter 12, "DOS Utility Suite," on page 107 for detailed information on the usage of these tools.

A.3.3 EEPROM Programming in Linux

A.3.3.1 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 it's new value, respectively. The magic parameter is used to provide the device-specific magic key (0x7500 for the LAN7500/LAN7500i), preventing accidental writing to the EEPROM.

A.3.4 EEPROM Programming in Windows CE

In Windows CE, the device EEPROM can be read or written using the CE7500eep.exe SMSC utility. The CE7500eep 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.smsc.com/main.nsf.

In order to use the CE7500eep utility, the customer's image must include the command console option in the "Catalog Item View", as shown in Figure A.1.

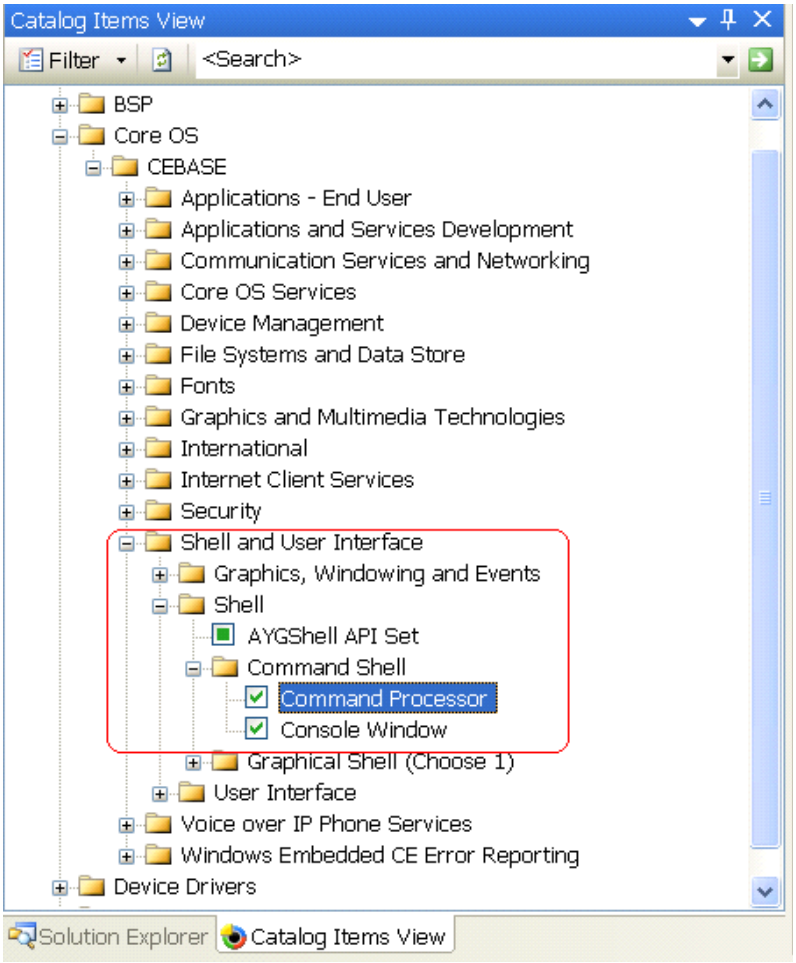

Figure A.1 Command Shell Options in Catalog Items View

To use the CE7500eep utility, double click the CE7500eep 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 `7500.bin`.

Selecting option 3, “Read EEPROM”, will read the EEPROM contents to a binary file, `7500.bin`, located in the `\Windows\` directory.

Selecting Q, will exit the application.

For the latest CE7500eep information, refer to the release notes.

A.3.4.1 Including CE7500eep.exe in a CE Image

The `CE7500eep.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 `CE7500eep.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 `CE7500eep.exe` file to the `%_TARGETPLATROOT%\FILES\` folder.
2. Edit the `Platform.bib` file in the `%_TARGETPLATROOT%\FILES\` folder to include the `CE7500eep.exe` file into the target image, as shown in Figure A.2.

```
    ENDIF IMGFAKERIL
; @CESYSGEN ENDIF CELLCORE_MODULES_RIL

FILES
;  Name            Path                                          Memory Type
;  --------------  --------------------------------------------  -----------

  ce7500eep.exe    $(_FLATRELEASEDIR)\ce7500eep.exe                    NK

-- INSERT --                                              270,53        98%
```

Figure A.2 Inserting CE7500eep.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 `CE7500eep.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 `CE7500eep.exe` utility.

A.4 Customized Operation Without EEPROM

The device provides the capability to customize operation without the use of an EEPROM. Descriptor information and initialization quantities normally fetched from EEPROM and used to initialize descriptors and elements of the System Control and Status Registers may be specified via an alternate mechanism. This alternate mechanism involves the use of the Descriptor RAM in conjunction with the Attribute Registers and select elements of the System Control and Status Registers. The software device driver (See Note A.1 & Note A.2) orchestrates the process by performing the following actions in the order indicated:

- Initialization of SCSR Elements in Lieu of EEPROM Load
- Attribute Register Initialization
- Descriptor RAM Initialization
- Enable Descriptor RAM and Flag Attribute Registers as Source
- Inhibit Reset of Select SCSR Elements

The following subsections explain these actions. The attribute registers must be written prior to initializing the Descriptor RAM. Failure to do this will prevent the Remote Wakeup Support (RMT_WKP) and Power Method (PWR_SEL) flags specified in the Flag Attributes Register (FLAG_ATTR) from being overwritten by the bmAttributes of a Configuration Descriptor written to the Descriptor RAM. The required behavior is analogous to the EEPROM case, where the Power Method and Remote Wakeup Support entries in Configuration Flags 0 are overwritten by the bmAttributes values of a Configuration Descriptor specified in the EEPROM.

Note: Registers which must be used for EEPROM-less operation are defined in Section A.4.6, "EEPROM-less Operation Register Definitions," on page 125

Note A.1 Certain operating systems (e.g., Microsoft Windows) may require some pieces of information from the EEPROM or the equivalent EEPROM-less operation before the device driver is loaded. Therefore, the alternate mechanism described in this section must be provided by pre-OS firmware.

Note A.2 Currently supplied SMSC device drivers and pre-boot execution environment expansion ROMs DO NOT provide the necessary programming mechanisms listed in this chapter.

A.4.1 Initialization of SCSR Elements in Lieu of EEPROM Load

During EEPROM operation, the following register fields are initialized by the hardware using the values contained in the EEPROM. In the absence of an EEPROM, the software device driver must initialize these quantities:

- MAC Receive Address High Register (RX_ADDRH) and MAC Receive Address Low Register (RX_ADDRL)
- PHY Boost (PHY_BOOST) field of the Hardware Configuration Register (HW_CFG)
- Automatic Duplex Detection (ADD) bit of the MAC Control Register (MAC_CR)
- Automatic Speed Detection (ASD) bit of the MAC Control Register (MAC_CR)
- LED 2 Function Select (LED2_FUN_SEL) bit of the LED General Purpose IO Configuration 0 Register (LED_GPIO_CFG0)
- Speed LEDs Function Select (SPDLED_FUN_SEL) bit of the LED General Purpose IO Configuration 0 Register (LED_GPIO_CFG0)
- GPIO Enable 0–3 (GPIOEN[3:0]) of the LED General Purpose IO Configuration 0 Register (LED_GPIO_CFG0) and the GPIOEN4 bit of the GPIO Enable 4-11 (GPIOEN[11:4]) field of the LED General Purpose IO Configuration 1 Register (LED_GPIO_CFG1)
- GPIO Wake 0-11 (GPIOWK[11:0]) field of the General Purpose IO Wake Enable and Polarity Register (GPIO_WAKE)

A.4.2 Attribute Register Initialization

The Attribute Registers are as follows:

- HS Descriptor Attributes Register (HS_ATTR)
- FS Descriptor Attributes Register (FS_ATTR)
- String Descriptor Attributes Register 0 (STRNG_ATTR0)
- String Descriptor Attributes Register 1 (STRNG_ATTR1)
- Flag Attributes Register (FLAG_ATTR)

All of these registers, with the exception of FLAG_ATTR, contain fields defining the lengths of the descriptors written into the Descriptor RAM. If the descriptor is not written into the Descriptor RAM, the associated entry in the Attributes Register must be written as 0. Writing an erroneous or illegal length will result in untoward operation and unexpected results.

The Flag Attributes Register (FLAG_ATTR) provides the mechanism to initialize components of the Configuration Flags and GPIO PME Flags that are stand-alone and not part of any other System Control and Status Register. During EEPROM operation, the analogous fields in this register are read by the hardware from the EEPROM and are not available to the software for read-back or modification.

Note: The software device driver must initialize these registers prior to initializing the Descriptor RAM.

Note: The bmAttributes field of the HS and FS descriptors in descriptor RAM (if present) must be consistent with the contents of the Flag Attributes Register (FLAG_ATTR).

A.4.3 Descriptor RAM Initialization

The Descriptor RAM contents are initialized using the Data Port registers. The Data Port registers are used to select the Descriptor RAM and write the descriptor elements into it. The Descriptor RAM is 512 bytes in length. Every descriptor written into the Descriptor RAM must be DWORD aligned. The Attribute Registers discussed in Section A.4.2 must be written with the length of the descriptors written into the Descriptor RAM. If a descriptor is not used, hence not written into Descriptor RAM, its length must be written as 0 into the associated Attribute Register.

Note: The Attribute Registers must be initialized before the Descriptor RAM.

Note: Address 0 of the Descriptor RAM is always reserved for the Language ID descriptor, even if it will not be supported.

The descriptors must be written in the following order, starting at address 0 of the RAM and observing the DWORD alignment rule:

- Language ID Descriptor
- Manufacturing String Descriptor (String Index 1)
- Product Name String Descriptor (String Index 2)
- Serial Number String Descriptor (String Index 3)
- Configuration String Descriptor (String Index 4)
- Interface String Descriptor (String Index 5)
- HS Device Descriptor
- HS Configuration Descriptor
- FS Device Descriptor
- FS Configuration Descriptor

An example of Descriptor RAM use is illustrated in Figure A.3.

As in the case of descriptors specified in EEPROM, the following restrictions apply to descriptors written into Descriptor RAM:

1. For Device Descriptors, the only valid values for the length are 0 and 18. The descriptor size for the Device Descriptors specified in the Descriptor RAM is a don't care and always overwritten by HW to 0x12 when transmitting the descriptor to the Host.
2. The descriptor type for Device Descriptors specified in the Descriptor RAM is a don't care and is always overwritten by HW to 0x1 when transmitting the descriptor to the Host.
3. For the Configuration and Interface descriptor, the only valid values for the length are 0 and 18. The descriptor size for the Device Descriptors specified in the Descriptor RAM is a don't care and always overwritten by HW to 0x12 when transmitting the descriptor to the Host.
4. The descriptor type for the configuration descriptors specified in the Descriptor RAM is a don't care and always overwritten by HW to 0x2 when transmitting the descriptor to the Host.
5. If a string descriptor does not exist in the Descriptor RAM, the referencing descriptor must contain 00h for the respective string index field.
6. If all string descriptor lengths are zero then a Language ID will not be supported.

Note: The first entry in the Descriptor RAM is always reserved for the Language ID descriptor, even if it will not be supported.

Note: Descriptors specified having bcdUSB, bMaxPacketSize0, and bNumConfigurations fields defined with values other than 0200h, 40h, and 1, respectively, will result in unwanted behavior and untoward results.

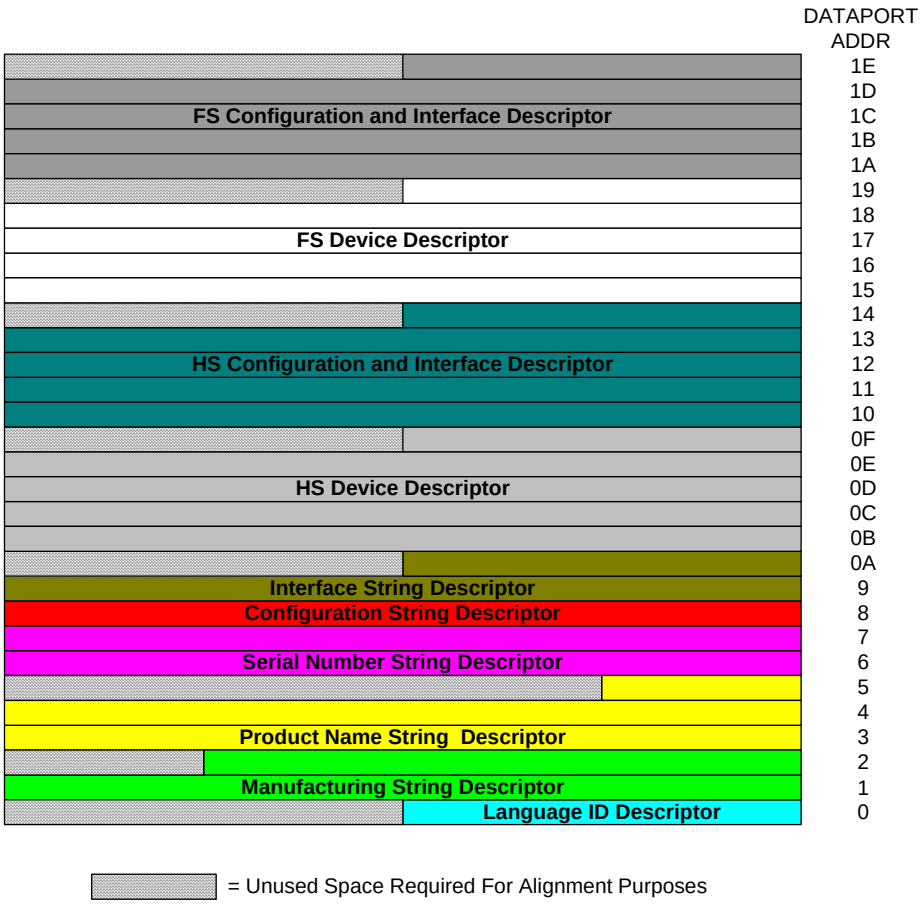

Figure A.3 Descriptor RAM Example

A.4.4 Enable Descriptor RAM and Flag Attribute Registers as Source

The EEPROM Emulation Enable (EEM) bit of the Hardware Configuration Register (HW_CFG) must be configured by the software device driver to use the Descriptor RAM and the Attribute Registers for custom operation. Upon assertion of EEPROM Emulation Enable (EEM), the hardware will utilize the Descriptor information contained in the Descriptor RAM, the Attributes Registers, and the values of the items listed in Section A.4.1 to facilitate custom operation.

A.4.5 Inhibit Reset of Select SCSR Elements

The software device driver must take care to ensure that the contents of the Descriptor RAM and SCSR register content critical to custom operation using Descriptor RAM are preserved across reset operations other than POR. The driver must configure the Reset Protection (RST_PROTECT) bit of the Hardware Configuration Register (HW_CFG) in order to accomplish this.

The following registers have contents that can be preserved across all resets other than POR. Consult the register's description for additional details.

- Descriptor RAM
- Attribute Registers
- MAC Receive Address High Register (RX_ADDRH) and MAC Receive Address Low Register (RX_ADDRL)
- Hardware Configuration Register (HW_CFG)
- LED General Purpose IO Configuration 0 Register (LED_GPIO_CFG0)
- LED General Purpose IO Configuration 1 Register (LED_GPIO_CFG1)
- General Purpose IO Wake Enable and Polarity Register (GPIO_WAKE)

A.4.6 EEPROM-less Operation Register Definitions

This section provides definitions for all registers used to configure the device without an EEPROM. For information on how these registers are used for EEPROM-less operation, refer to Section A.4, "Customized Operation Without EEPROM," on page 121.

A.4.6.1 Hardware Configuration Register (HW_CFG)

Address: 010h Size: 32 bits

BITS	DESCRIPTION	TYPE	DEFAULT
31:16	**RESERVED**	RO	-
15	**NetDetach Status (NETDET_STS)** After the driver loads, this bit is checked to determine whether a NetDetach event occurred.	R/WC	Note A.3
14	**NetDetach Enable (NEDET_EN)** When this bit is set, the device detaches from the USB bus. This results in the driver unloading and no further communication with the device. The device remains detached until PHY link is detected, or a properly configured GPIO pin is asserted. Occurrence of either event causes the device to attach to the USB bus, the driver to be loaded, and the NETDET_STS bit to be asserted.	SC	0b
13	**EEPROM Emulation Enable (EEM)** This bit is used to select the source of descriptor information and configuration flags when no EEPROM is present. 0 = Use defaults as specified in Section A.2, "EEPROM Defaults," on page 118 1 = Use Descriptor RAM and Attributes Registers **Note:** This bit affects operation only when a EEPROM is not present. This bit has no effect when a EEPROM is present. **Note:** This field is protected by Reset Protection (RST_PROTECT).	R/W	0b
12	**Reset Protection (RST_PROTECT)** Setting this bit protects select fields of certain registers from being affected by resets other than POR. **Note:** This field is protected by Reset Protection (RST_PROTECT).	R/W	0b
11	**Port Swap (PORT_SWAP)** Swaps the mapping of USBDP and USBDM. 0 = USBDP maps to the USB D+ line and USBDM maps to the USB D- line. 1 = USBDP maps to the USB D- line. USBDM maps to the USB D+ line.	RO	Note A.4
10:9	**PHY Boost (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) **Note:** This field is protected by Reset Protection (RST_PROTECT).	R/W	Note A.5

BITS	DESCRIPTION	TYPE	DEFAULT
8	**Activity LED 80 ms Bypass (LEDB)** When set, the Activity LED on/off time is reduced to approximately 15us/15us.	R/W	0b
7	**Bulk-In Empty Response (BIR)** This bit controls the response to Bulk-In tokens when the RX FIFO is empty. 0 = Respond to the IN token with a ZLP 1 = Respond to the IN token with a NAK	R/W	0b
6	**Stall Bulk-Out Pipe Disable (SBP)** This bit controls the operation of the Bulk-Out pipe when the FCT detects the loss of sync condition. 0 = Stall the Bulk-Out pipe when loss of sync detected. 1 = Do not stall the Bulk-Out pipe when loss of sync detected.	R/W	0b
5	**RESERVED**	RO	-
4	**Multiple Ethernet Frames per USB Packet (MEF)** This bit enables the USB transmit direction to pack multiple Ethernet frames per USB packet whenever possible. 0 = Support no more than one Ethernet frame per USB packet 1 = Support packing multiple Ethernet frames per USB packet **Note:** The URX supports this mode by default.	R/W	0b
3	**EEPROM Time-out Control (ETC)** This bit controls the length of time used by the EEPOM controller to detect a time-out. 0 = Time-out occurs if no response received from EEPROM after 30 ms. 1 = Time-out occurs if no response received from EEPROM after 1.28 us.	R/W	0b
2	**Burst Cap Enable (BCE)** This register enables use of the burst cap register (BURST_CAP). 0 = Burst Cap register is not used to limit the TX burst size. 1 = Burst Cap register is used to limit the TX burst size.	R/W	0b
1	**Soft Lite Reset (LRST)** Writing 1 generates the lite software reset of the device. A lite reset will not affect the UDC. Additionally, the contents of the EEPROM will not be reloaded. This reset will not cause the USB PHY to be disconnected. This bit clears after the reset sequence has completed. In the case where the Automatic Speed Detection (ASD) bit of the MAC Control Register (MAC_CR) is not set when an EEPROM or EEPROM emulation mode is employed, the MAC Configuration (CFG) setting of the MAC Control Register (MAC_CR) may be out of sync with the actual PHY speed. In this case, LRST will remain high and the reset will not occur.	SC	0b
0	**Soft Reset (SRST)** Writing 1 generates a software initiated reset of the device. If an external Ethernet PHY is used, it will be reset as well. A software reset will result in the contents of the EEPROM being reloaded. While the reset sequence is in progress, the USB PHY will be disconnected. After the device has been reinitialized, it will take the PHY out of the disconnect state and be visible to the Host.	SC	0b

Note A.3 The default value of this bit depends on whether a NetDetach event occurred. If set, the event occurred.

Note A.4 The default value of this bit is determined by the value of the Port Swap bit of the Configuration Flags 0 contained within the EEPROM, if present. If no EEPROM is present, 0b is the default.

Note A.5 The default value of this field is determined by the value of the PHY Boost field of the Configuration Flags 0 contained within the EEPROM, if present. If no EEPROM is present, 00b is the default.

A.4.6.2 LED General Purpose IO Configuration 0 Register (LED_GPIO_CFG0)

Address: 018h Size: 32 bits

This register configures the external GPIO[3:0] pins.

In order for a GPIO to function as a wake event or interrupt source, it must be configured as an input. GPIO pins used to generate wake events must also be enabled by the GPIO_WAKE register, see Section A.4.6.4, "General Purpose IO Wake Enable and Polarity Register (GPIO_WAKE)," on page 132.

BITS	DESCRIPTION	TYPE	DEFAULT
31	**LED 2 Function Select (LED2_FUN_SEL)** The value of this bit determines the operational mode of LED2 when it is enabled via GPIOEN2 of this register. 0 = LED2 is a Link and Activity LED 1 = LED2 is an Activity LED **Note:** This field is protected by Reset Protection (RST_PROTECT).	R/W	Note A.6
30	**Speed LEDs Function Select (SPDLED_FUN_SEL)** The value of this bit determines the operational mode of the speed LEDs (LED0 and LED1) when they are enabled via GPIOEN0 and GPIOEN1 of this register. 0 = LED0 and LED1 are Link and Speed LEDs 1 = LED0 and LED1 are Link and Speed and Activity LEDs **Note:** This field is protected by Reset Protection (RST_PROTECT).	R/W	Note A.7
29:16	**RESERVED**	RO	-

<table>
<tr><th>BITS</th><th>DESCRIPTION</th><th>TYPE</th><th>DEFAULT</th></tr>
<tr><td>15:12</td><td>
GPIO Enable 0–3 (GPIOEN[3:0])
When set, LED functionality is enabled on the corresponding pin. When clear, the pin function as a GPIO.
GPIOEN0 – bit 12
GPIOEN1 – bit 13
GPIOEN2 – bit 14
GPIOEN3 – bit 15

Note: This field is protected by Reset Protection (RST_PROTECT).

These bits, together with GPIO Enable 4-11 (GPIOEN[11:4]) of the LED General Purpose IO Configuration 1 Register (LED_GPIO_CFG1) make GPIOEN[11:0].

Functionality is as follows:

GPIOEN0 and GPIOEN1 = 0, GPIO0 and GPIO1 enabled.
GPIOEN0 and GPIOEN1 = 1, LED0 and LED1 (Speed LEDs) enabled.

Speed LEDs behavior is determined by line speed and the setting of the Speed LEDs Function Select (SPDLED_FUN_SEL) bit, as indicated in following table.:

<table>
<tr><th>SPDLED_FUN_SEL</th><th>SPEED (Mbps)</th><th>LED0</th><th>LED1</th></tr>
<tr><td>0</td><td>No Link</td><td>Off</td><td>Off</td></tr>
<tr><td>0</td><td>10</td><td>On</td><td>Off</td></tr>
<tr><td>0</td><td>100</td><td>Off</td><td>On</td></tr>
<tr><td>0</td><td>1000</td><td>On</td><td>On</td></tr>
<tr><td>1</td><td>No Link</td><td>Off</td><td>Off</td></tr>
<tr><td>1</td><td>10</td><td>Blink</td><td>Off</td></tr>
<tr><td>1</td><td>100</td><td>Off</td><td>Blink</td></tr>
<tr><td>1</td><td>1000</td><td>Blink</td><td>Blink</td></tr>
</table>
When SPDLED_FUN_SEL = 0, the LEDs function solely as Link and Speed LEDs. When SPDLEN_FUN_SEL = 1, the LEDs function as Link and Speed and Activity LEDs. In those cases, the table entry “Blink” indicates the LED will remain on when no transmit or receive activity is detected and will blink at an 80 mS rate whenever TX or RX activity is detected.

GPIOEN2 = 0, GPIO2 enabled
GPIOEN2 = 1, LED2 enabled.

LED2 Functionality is determined by the setting of LED 2 Function Select (LED2_FUN_SEL). If LED_FUN_SEL = 0, LED2 is behaves as a Link and Activity LED. Otherwise, it operates as an Activity indicator.

When configured for link and activity (LED_FUN_SEL = 0), the pin will go low (LED on) for link and then be pulsed high (LED off) for 80msec for transmit or receive activity followed by 80msec of low (LED on).

When configured for activity only (LED_FUN_SEL = 1), the pin is pulsed low (LED on) for 80mS whenever transmit or receive activity is detected. The pin is then driven high (LED off) again for a minimum of 80mS, after which time it will repeat the process if TX or RX activity is detected.
</td><td>R/W</td><td>Note A.8</td></tr>
</table>

BITS	DESCRIPTION	TYPE	DEFAULT
15:12 (cont.)	GPIOEN3 = 0, GPIO3 enabled GPIOEN3 = 1, LED3 (Link LED) enabled. For LED3 operation, this pin is low (LED on) when a link is detected, high (LED off) otherwise. GPIOEN4 = 0, GPIO4 enabled GPIOEN4 = 1, LED4 (Full Duplex LED) enabled. For LED4 operation, this pin is low (LED on) when operating in full-duplex mode, high (LED off) otherwise. **Note:** GPIOEN4 is contained in the GPIO Enable 4-11 (GPIOEN[11:4]) field of the LED General Purpose IO Configuration 1 Register (LED_GPIO_CFG1).		
11:8	**GPIO Buffer Type (GPIOBUF[3:0])** When set, the output buffer for the corresponding GPIO signal is configured as a push/pull driver. When cleared, the corresponding GPIO signal is configured as an open-drain driver. Bits are assigned as follows: GPIOBUF0 – bit 8 GPIOBUF1 – bit 9 GPIOBUF2 – bit 10 GPIOBUF3 – bit 11	R/W	0000b
7:4	**GPIO Direction (GPIODIR[3:0])** When set, enables the corresponding GPIO as an output. When cleared the GPIO is enabled as an input. Bits are assigned as follows: GPIODIR0 – bit 4 GPIODIR1 – bit 5 GPIODIR2 – bit 6 GPIODIR3 – bit 7	R/W	0000b
3:0	**GPIO Data (GPIOD[3:0])** When enabled as an output, the value written is reflected on GPIOn. When read, GPIOn reflects the current state of the corresponding GPIO pin. Bits are assigned as follows: GPIOD0 – bit 0 GPIOD1 – bit 1 GPIOD2 – bit 2 GPIOD3 – bit 3	R/W	Note A.9

Note A.6 The default value of this field is determined by the value of the LED2_FUNCTION field of the Configuration Flags 1 contained within the EEPROM, if present. If no EEPROM is present, 0b is the default.

Note A.7 The default value of this field is determined by the value of the SPD_LED_FUNCTION field of the Configuration Flags 0 contained within the EEPROM, if present. If no EEPROM is present, 0b is the default.

Note A.8 The default value of this field is determined by the value of the GPIOEN[3:0] bits of the GPIOEN[4:0] field of the Configuration Flags 1 contained within the EEPROM, if present. If no EEPROM is present, 0000b is the default.

Note A.9 The default value depends on the state of the GPIO pin.

A.4.6.3 LED General Purpose IO Configuration 1 Register (LED_GPIO_CFG1)

Address: 01Ch Size: 32 bits

This register configures GPIOs 4-11.

In order for a GPIO to function as a wake event or interrupt source, it must be configured as an input. GPIOs used as wake events must also be enabled by the GPIO_WAKE register, see Section A.4.6.4, "General Purpose IO Wake Enable and Polarity Register (GPIO_WAKE)".

BITS	DESCRIPTION	TYPE	DEFAULT
31:24	**GPIO Enable 4-11 (GPIOEN[11:4])** These bits, together with GPIO Enable 0–3 (GPIOEN[3:0]) of the LED General Purpose IO Configuration 0 Register (LED_GPIO_CFG0) make GPIOEN[11:0]. For GPIOEN[11:5]: when set, the corresponding GPIO is disabled. When cleared low, the pin functions as a GPIO. For GPIOEN4: when set, LED4 enabled. When cleared low, GPIO4 enabled. Refer to GPIO Enable 0–3 (GPIOEN[3:0]) field of the LED General Purpose IO Configuration 0 Register (LED_GPIO_CFG0) for definition of LED4 functionality. **Note:** GPIOEN4 is protected by Reset Protection (RST_PROTECT). GPIOEN4 - bit 24 GPIOEN5 - bit 25 GPIOEN6 - bit 26 GPIOEN7 - bit 27 GPIOEN8 - bit 28 GPIOEN9 - bit 29 GPIOEN10 - bit 30 GPIOEN11 - bit 31 **Note:** GPIOs 5-11 are disabled after a reset. Refer to Note A.10 for an explanation of the state of GPIO4 after reset.	R/W	1111111xb Note A.10
23:16	**GPIO Buffer Type 4-11 (GPIOBUF[11:4])** When set, the output buffer for the corresponding GPIO signal is configured as a push/pull driver. When cleared, the corresponding GPIO signal is configured as an open-drain driver. GPIOBUF4 - bit 16 GPIOBUF5 - bit 17 GPIOBUF6 - bit 18 GPIOBUF7 - bit 19 GPIOBUF8 - bit 20 GPIOBUF9 - bit 21 GPIOBUF10 - bit 22 GPIOBUF11 - bit 23	R/W	00h

BITS	DESCRIPTION	TYPE	DEFAULT
15:8	**GPIO Direction 4-11 (GPIODIR[11:4])** When set, enables the corresponding GPIO as output. When cleared, the GPIO is enabled as an input. GPIODIR4 - bit 8 GPIODIR5 - bit 9 GPIODIR6 - bit 10 GPIODIR7 - bit 11 GPIODIR8 - bit 12 GPIODIR9 - bit 13 GPIODIR10 - bit 14 GPIODIR11 - bit 15	R/W	00h
7:0	**GPIO Data 4-11 (GPIOD[11:4])** When enabled as an output, the value written is reflected on GPIOn. When read, GPIOn reflects the current state of the corresponding GPIO pin. GPIOD4 - bit 0 GPIOD5 - bit 1 GPIOD6 - bit 2 GPIOD7 - bit 3 GPIOD8 - bit 4 GPIOD9 - bit 5 GPIOD10 - bit 6 GPIOD11 - bit 7	R/W	Note A.11

Note A.10 The default value of bit x (GPIOEN4) is determined by the value of the GPIOEN4 bit of the GPIOEN[4:0] field of the Configuration Flags 1 contained within the EEPROM, if present. If no EEPROM is present, 0b is the default. The value of this bit determines the state of GPIO4 after reset.

Note A.11 The default value depends on the state of the GPIO pin.

A.4.6.4 General Purpose IO Wake Enable and Polarity Register (GPIO_WAKE)

Address: 020h Size: 32 bits

This register enables the GPIOs to function as wake events for the device when asserted. It also allows the polarity used for a wake event/interrupt to be configured.

Note: GPIOs must not cause a wake event to the device when not configured as a GPIO.

BITS	DESCRIPTION	TYPE	DEFAULT
31	**PHY Link Up Enable (PHY_LINKUP_EN)** Setting this bit enables the use of GPIO7 to signal a PHY Link Up event when in SUSPEND0 or SUSPEND 3 state. In addition to setting this bit, the parameters for GPIO7 must be properly set, in order for signaling to occur.	R/W	0b
30:28	**RESERVED**	RO	-
27:16	**GPIO Polarity 0-11 (GPIOPOL[11:0])** 0 = Wakeup/interrupt is triggered when GPIO is driven low 1 = Wakeup/interrupt is triggered when GPIO is driven high GPIOPOL0 - bit 16 GPIOPOL1 - bit 17 GPIOPOL2 - bit 18 GPIOPOL3 - bit 19 GPIOPOL4 - bit 20 GPIOPOL5 - bit 21 GPIOPOL6 - bit 22 GPIOPOL7 - bit 23 GPIOPOL8 - bit 24 GPIOPOL9 - bit 25 GPIOPOL10 - bit 26 GPIOPOL11 - bit 27	R/W	000h
15:12	**RESERVED**	RO	-
11:0	**GPIO Wake 0-11 (GPIOWK[11:0])** 0 = The GPIO can not wake up the device. 1 = The GPIO can trigger a wake up event. GPIOWK0 - bit 0 GPIOWK1 - bit 1 GPIOWK2 - bit 2 GPIOWK3 - bit 3 GPIOWK4 - bit 4 GPIOWK5 - bit 5 GPIOWK6 - bit 6 GPIOWK7 - bit 7 GPIOWK8 - bit 8 GPIOWK9 - bit 9 GPIOWK10 - bit 10 GPIOWK11 - bit 11 **Note:** This field is protected by Reset Protection (RST_PROTECT).	R/W	Note A.12

Note A.12 The default value of this field is loaded from the associated bytes of the EEPROM. The high order, unused bits, of the EEPROM are ignored. If no EEPROM is present, the default value of each bit in the field is 0. A USB Reset or Soft Lite Reset (LRST) will cause this field to be restored to the image value last loaded from EEPROM, or will cause the value of each bit to be set to 0 if no EEPROM is present.

A.4.6.5 Data Port Select Register (DP_SEL)

Offset: 024h Size: 32 bits

The Data Port Ready bit indicates when the data port RAM access has completed. In the case of a read operation, this bit indicates when the read data has been stored in the DP_DATA register.

BITS	DESCRIPTION	TYPE	DEFAULT
31	**Data Port Ready (DPRDY)** 1 = Data Port is ready. 0 = Data Port is busy processing a transaction.	RO	1b
30:4	**RESERVED**	RO	-
3:0	**RAM Test Select (RSEL)** Selects which RAM to access. 0000 = URX Buffer RAM (Do not access at run time) 0001 = RFE VLAN and DA Hash Table (VHF RAM) 0010 = LSO Header RAM (Do not access at run time) 0011 = FCT RX RAM (Do not access at run time) 0100 = FCT TX RAM (Do not access at run time) 0101 = Descriptor RAM (Do not access at run time) 0110 = WOL RAM (Do not write at run time) 0111 = RESERVED 1000 = RESERVED 1001 = RESERVED 1010 = RESERVED 1011 = RESERVED 1100 = RESERVED 1101 = RESERVED 1110 = RESERVED 1111 = RESERVED	R/W	0000b

A.4.6.6 Data Port Command Register (DP_CMD)

Offset: 028h Size: 32 bits

This register commences the data port access. Writing a one to this register will enable a write access, while writing a zero will do a read access.

The address and data registers need to be configured appropriately for the desired read or write operation before accessing this register.

BITS	DESCRIPTION	TYPE	DEFAULT
31:1	**RESERVED**	RO	-
0	**Data Port Write**. Selects operation. Writing to this bit initiates the dataport access. 1 = write operation 0 = read operation	R/W	0b

A.4.6.7 Data Port Address Register (DP_ADDR)

Offset: 02Ch Size: 32 bits

Indicates the address to be used for the dataport access.

BITS	DESCRIPTION	TYPE	DEFAULT
31:14	**RESERVED**	RO	-
13:0	**Data Port Address[13:0]**	R/W	00 0000 0000 0000b

A.4.6.8 Data Port Data Register (DP_DATA)

Offset: 030h Size: 32 bits

The Data Port Data register holds the write data for a write access and the resultant read data for a read access.

Before reading this register for the result of a read operation, the Data Port Ready bit should be checked. The Data Port Ready bit must indicate the data port is ready. Otherwise the read operation is still in progress.

BITS	DESCRIPTION	TYPE	DEFAULT
31:0	**Data Port Data [31:0]**	R/W	0000_0000h

A.4.6.9 HS Descriptor Attributes Register (HS_ATTR)

Address: 04Ch Size: 32 bits

This register sets the length values for HS descriptors that have been loaded into Descriptor RAM via the Data Port registers. The HS Polling interval is also defined by a field within this register. The Descriptor RAM images may be used, in conjunction with this register, to facilitate customized operation when no EEPROM is present.

Note: If a descriptor does not exist in Descriptor RAM, its size value must be written as 00h.

Note: This register only affects system operation when an EEPROM is not present and the EEPROM Emulation Enable (EEM) bit indicates Descriptor RAM and the Attributes Registers are to be used for descriptor processing.

Note: Writing to this register when an EEPROM is present is prohibited and will result in untoward operation and unexpected results.

Note: This register is protected by Reset Protection (RST_PROTECT).

BITS	DESCRIPTION	TYPE	DEFAULT
31:24	**RESERVED**	RO	-
23:16	**HS Polling Interval (HS_POLL_INT)**	R/W	04h
15:8	**HS Device Descriptor Size (HS_DEV_DESC_SIZE)** Note A.13	R/W	00h
7:0	**HS Configuration Descriptor Size (HS_CFG_DESC_SIZE)** Note A.14	R/W	00h

Note A.13 The only legal values are 0 and 12h. Writing any other values will result in untoward behavior and unexpected results.

Note A.14 The only legal values are 0 and 12h. Writing any other values will result in untoward behavior and unexpected results.

A.4.6.10 FS Descriptor Attributes Register (FS_ATTR)

Address: 050h Size: 32 bits

This register sets the length values for FS descriptors that have been loaded into Descriptor RAM via the Data Port registers. The FS Polling interval is also defined by a field within this register. The Descriptor RAM images may be used, in conjunction with this register, to facilitate customized operation when no EEPROM is present.

Note: If a descriptor does not exist in Descriptor RAM, its size value must be written as 00h.

Note: This register only affects system operation when an EEPROM is not present and the EEPROM Emulation Enable (EEM) bit indicates Descriptor RAM and the Attributes Registers are to be used for descriptor processing.

Note: Writing to this register when an EEPROM is present is prohibited and will result in untoward operation and unexpected results.

Note: This register is protected by Reset Protection (RST_PROTECT).

BITS	DESCRIPTION	TYPE	DEFAULT
31:24	**RESERVED**	RO	-
23:16	**FS Polling Interval (FS_POLL_INT)**	R/W	01h
15:8	**FS Device Descriptor Size (FS_DEV_DESC_SIZE)** Note A.15	R/W	00h
7:0	**FS Configuration Descriptor Size (FS_CFG_DESC_SIZE)** Note A.16	R/W	00h

Note A.15 The only legal values are 0 and 12h. Writing any other values will result in untoward behavior and unexpected results.

Note A.16 The only legal values are 0 and 12h. Writing any other values will result in untoward behavior and unexpected results.

A.4.6.11 String Descriptor Attributes Register 0 (STRNG_ATTR0)

Address: 054h Size: 32 bits

This register sets the length values for the named string descriptors that have been loaded into Descriptor RAM via the Data Port registers. The Descriptor RAM images may be used, in conjunction with this register, to facilitate customized operation when no EEPROM is present.

Note: If a descriptor does not exist in Descriptor RAM, its size value must be written as 00h.

Note: This register only affects system operation when an EEPROM is not present and the EEPROM Emulation Enable (EEM) bit indicates Descriptor RAM and the Attributes Registers are to be used for descriptor processing.

Note: Writing to this register when an EEPROM is present is prohibited and will result in untoward operation and unexpected results.

Note: This register is protected by Reset Protection (RST_PROTECT).

BITS	DESCRIPTION	TYPE	DEFAULT
31:24	**Configuration String Descriptor Size (CFGSTR_DESC_SIZE)**	R/W	00h
23:16	**Serial Number String Descriptor Size (SERSTR_DESC_SIZE)**	R/W	00h
15:8	**Product Name String Descriptor Size (PRODSTR_DESC_SIZE)**	R/W	00h
7:0	**Manufacturing String Descriptor Size (MANUF_DESC_SIZE)**	R/W	00h

A.4.6.12 String Descriptor Attributes Register 1 (STRNG_ATTR1)

Address: 058h Size: 32 bits

This register sets the length values for the named string descriptors that have been loaded into Descriptor RAM via the Data Port registers. The Descriptor RAM images may be used, in conjunction with this register, to facilitate customized operation when no EEPROM is present.

Note: If a descriptor does not exist in Descriptor RAM, its size value must be written as 00h.

Note: This register only affects system operation when an EEPROM is not present and the EEPROM Emulation Enable (EEM) bit indicates Descriptor RAM and the Attributes Registers are to be used for descriptor processing.

Note: Writing to this register when an EEPROM is present is prohibited and will result in untoward operation and unexpected results.

Note: This register is protected by Reset Protection (RST_PROTECT).

BITS	DESCRIPTION	TYPE	DEFAULT
31:8	**RESERVED**	RO	-
7:0	**Interface String Descriptor Size (INTSTR_DESC_SIZE)**	R/W	00h

A.4.6.13 Flag Attributes Register (FLAG_ATTR)

Address: 05Ch Size: 32 bits

This register sets the values of elements of the Configuration Flags 0, Configuration Flags 1, and GPIO PME Flags when no EEPROM is present and customized operation, using Descriptor RAM images, is to occur. This register does not contain Configuration Flag elements that are Read/Write components of other registers. Those elements will be programmed by the driver software directly, prior to initiating customized operation via Descriptor RAM. The value of Configuration Flag elements that are Read Only components of other registers may be programmed via this register.

Note: This register only affects system operation when an EEPROM is not present and the EEPROM Emulation Enable (EEM) bit indicates Descriptor RAM and the Attributes Registers are to be used for descriptor processing.

Note: Writing to this register when an EEPROM is present is prohibited and will result in untoward operation and unexpected results.

Note: This register is protected by Reset Protection (RST_PROTECT).

BITS	DESCRIPTION	TYPE	DEFAULT
31:24	**RESERVED**	RO	-
23	**Port Swap (PORT_SWP)** Refer to the Port Swap bit in Table A.2, “Configuration Flags 0,” on page 114 for definition.	R/W	0b
22:18	**RESERVED**	RO	-
17	**Remote Wakeup Support (RMT_WKP)** Refer to the Remote Wakeup Support bit in Table A.2, “Configuration Flags 0,” on page 114 for definition.	R/W	1b
16	**Power Method (PWR_SEL)** Refer to Power Method bit in Table A.2, “Configuration Flags 0,” on page 114 for definition.	R/W	1b
15:10	**RESERVED**	RO	-
9	**SW_MODE Select (SW_MODE_SEL)** Refer to the SW_MODE_SEL bit in Table A.3, “Configuration Flags 1,” on page 116 for definition.	R/W	0b
8	**SW_MODE Polarity (SW_MODE_POL)** Refer to the SW_MODE_POL bit in Table A.3, “Configuration Flags 1,” on page 116 for definition.	R/W	0b
7:0	**GPIO PME Flags (PME_FLAGS)** Refer to Table A.4, “GPIO PME Flags,” on page 116 for bit definitions.	R/W	00h

A.4.6.14 MAC Control Register (MAC_CR)

Offset: 100h Size: 32 bits

This register establishes the RX and TX operating modes.

BITS	DESCRIPTION	TYPE	DEFAULT
31:14	**RESERVED**	RO	-
13	**Automatic Duplex Polarity (ADP)** This bit indicate the polarity of the FDUPLEX PHY LED. 0: FDUPLEX asserted low indicates the PHY is in full duplex mode. 1: FDUPLEX asserted high indicates the PHY is in full duplex mode.	R/W	1b
12	**Automatic Duplex Detection (ADD)** When set, the MAC ignores the setting of the Duplex Mode (DPX) bit and automatically determines the duplex operational mode. The MAC uses a PHY LED/signal to accomplish mode detection and reports the last determined status via the Duplex Mode (DPX) bit. When reset, the setting of the Duplex Mode (DPX) bit determines Duplex operation. **Note:** This bit should not be modified while the MAC's receiver or transmitter is enabled. **Note:** This field is protected by Reset Protection (RST_PROTECT).	R/W	Note A.17
11	**Automatic Speed Detection (ASD)** When set, the MAC ignores the setting of the MAC Configuration (CFG) field and automatically determines the speed of operation. The MAC samples the RX_CLK input to accomplish speed detection and reports the last determined speed via the MAC Configuration (CFG) field. When reset, the setting of the MAC Configuration (CFG) field determines operational speed. **Note:** This bit should not be modified while the MAC's receiver or transmitter is enabled. **Note:** This field is protected by Reset Protection (RST_PROTECT).	R/W	Note A.18
10	**Internal Loopback Operation Mode (INT_LOOP)** Loops back data between the TX datapath and RX datapath interfaces. This is only for full duplex mode. In internal loopback mode, the TX frame is received by the Internal GMII interface, and sent back to the MAC without being sent to the PHY. 0: Normal mode 1: Internal loopback enabled **Note:** This bit should not be modified while the MAC's receiver or transmitter is enabled.	R/W	0b
9:8	**RESERVED**	RO	-

<table>
<tr><th>BITS</th><th>DESCRIPTION</th><th>TYPE</th><th>DEFAULT</th></tr>
<tr><td>7:6</td><td>BackOff Limit (BOLMT)
The BOLMT bits allow the user to set its back-off limit in a relaxed or aggressive mode. According to IEEE 802.3, the MAC has to wait for a random number [r] of slot-times () after it detects a collision, where:

(eq.1)0 < r < ₂K

The exponent K is dependent on how many times the current frame to be transmitted has been retried, as follows:

(eq.2)K = min (n, 10) where n is the current number of retries.

If a frame has been retried three times, then K = 3 and r= 8 slot-times maximum. If it has been retried 12 times, then K = 10, and r = 1024 slot-times maximum.
An LFSR (linear feedback shift register) counter emulates a random number generator, from which r is obtained. Once a collision is detected, the number of the current retry of the current frame is used to obtain K (eq.2). This value of K translates into the number of bits to use from the LFSR counter. If the value of K is 3, the MAC takes the value in the first three bits of the LFSR counter and uses it to count down to zero on every slot-time. This effectively causes the MAC to wait eight slot-times. To give the user more flexibility, the BOLMT value forces the number of bits to be used from the LFSR counter to a predetermined value as in the table below.

<table>
<tr><th>BOLMT Value</th><th># Bits Used from LFSR Counter</th></tr>
<tr><td>2’b00</td><td>10</td></tr>
<tr><td>2’b01</td><td>8</td></tr>
<tr><td>2’b10</td><td>4</td></tr>
<tr><td>2’b11</td><td>1</td></tr>
</table>
Thus, if the value of K = 10, the MAC will look at the BOLMT if it is 00, then use the lower ten bits of the LFSR counter for the wait countdown. If the BOLMT is 10, then it will only use the value in the first four bits for the wait countdown, etc.

Slot-time = 512 bit times. (See IEEE 802.3 Spec., Sections 4.2.3.2.5 and 4.4.2.1).

Note: This bit should not be modified while the MAC's receiver or transmitter is enabled.</td><td>R/W</td><td>00b</td></tr>
<tr><td>5:4</td><td>RESERVED</td><td>RO</td><td>-</td></tr>
<tr><td>3</td><td>Duplex Mode (DPX)
This bit determines the duplex operational mode of the MAC when the Automatic Duplex Detection (ADD) bit is reset. When the Automatic Duplex Detection (ADD) bit is set, this bit is read-only and reports the last determined duplex operational mode.

When set, the MAC is operating in Full-Duplex mode, in which it can transmit and receive simultaneously.

0: MAC is in half duplex mode
1: MAC is in full duplex mode

Note: This bit should not be modified while the MAC's receiver or transmitter is enabled.

Note: Half duplex mode is disabled if the detected or manually set speed is 1000Mbps, regardless of the setting of this bit.</td><td>Note A.19</td><td>0b</td></tr>
</table>

BITS	DESCRIPTION	TYPE	DEFAULT
2:1	**MAC Configuration (CFG)** This field determines the operational speed of the MAC when the Automatic Speed Detection (ASD) bit is reset. When the Automatic Speed Detection (ASD) bit is set, this field is read-only and reports the last determined operational speed. 0: MII Mode - 10 Mbps 1: MII Mode - 100 Mbps 2: GMII Mode - 1000 Mbps 3: RESERVED **Note:** This bit should not be modified while the MAC's receiver or transmitter is enabled.	Note A.20	00b
0	**MAC Reset (MRST)** 0: MAC is enabled 1: MAC is reset	SC	0b

Note A.17 Defaults to 1 when no EEPROM is present, otherwise defaults to the value of the Duplex Detection bit of the Configuration Flags 0 byte in the EEPROM.

Note A.18 Defaults to 1 when no EEPROM is present, otherwise defaults to the value of the Speed Detection bit of the Configuration Flags 0 byte in the EEPROM.

Note A.19 When Automatic Duplex Detection (ADD) is reset, this bit is R/W and determines duplex operation. When Automatic Duplex Detection (ADD) is set, this field is RO and reports the last duplex operational mode determined by the MAC.

Note A.20 When Automatic Speed Detection (ASD) is reset, this field is R/W and determines operational speed. When Automatic Speed Detection (ASD) is set, this field is RO and reports the last operational speed determined by the MAC.

A.4.6.15 MAC Receive Address High Register (RX_ADDRH)

Address: 118h Size: 32 bits

This register contains the upper 16 bits of the physical address of the MAC, where RX_ADDRH[15:8] is the 6th octet of the received frame.

This register used to specify the address used for Perfect DA, Magic Packet and Wakeup frames, the unicast destination address for received pause frames, and the source address for transmitted pause frames. This register is not used for packet filtering.

Note: This register is protected by Reset Protection (RST_PROTECT).

BITS	DESCRIPTION	TYPE	DEFAULT
31:16	**RESERVED**	RO	-
15-0	**Physical Address [47:32]** This field contains the upper 16 bits [47:32] of the physical address of the device.	R/W	FFFFh Note A.21

Note A.21 The default value of this register is loaded from the EEPROM. If no EEPROM is present, the indicated value is used as the default.

A.4.6.16 MAC Receive Address Low Register (RX_ADDRL)

Address: 11Ch Size: 32 bits

Note: This register contains the lower 32 bits of the physical address of the MAC, where RX_ADDRL[7:0] is the first octet of the Ethernet frame.

This register used to specify the address used for Perfect DA, Magic Packet, and Wakeup frames, the unicast destination address for received pause frames, and the source address for transmitted pause frames. This register is not used for packet filtering.

Note: This register is protected by Reset Protection (RST_PROTECT).

BITS	DESCRIPTION	TYPE	DEFAULT
31:0	**Physical Address [31:0]** This field contains the lower 32 bits [31:0] of the physical address of the device.	R/W	FFFF_FFFFh Note A.22

Note A.22 The default value of this register is loaded from the EEPROM, if present. If no EEPROM is present, the indicated value is used as the default.

Table A.6 illustrates the byte ordering of the RX_ADDRL and RX_ADDRH registers with respect to the reception of the Ethernet physical address.

Table A.6 RX_ADDRL, RX_ADDRH Byte Ordering

RX_ADDRL, RX_ADDRH	ORDER OF RECEPTION ON ETHERNET
RX_ADDRL[7:0]	1st
RX_ADDRL[15:8]	2nd
RX_ADDRL[23:16]	3rd
RX_ADDRL[31:24]	4th
RX_ADDRH[7:0]	5th
RX_ADDRH[15:8]	6th

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
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 a 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.