Network Enabler SDK 2 Programmer's Guide

Fourth Edition, December 2004

www.moxa.com/product

Moxa Technologies Co., Ltd.

Tel: +886-2-8919-1230
Fax: +886-2-8919-1231
Web: www.moxa.com

MOXA Technical Support
Worldwide: support@moxa.com.tw
The Americas support@moxa.com

Network Enabler SDK 2 Programmer's Guide

Copyright Notice

Trademarks

Disclaimer

MOXA Internet Services

Customer satisfaction is our number one concern. To ensure that customers receive the full benefit of our products, Moxa Internet Services has been set up to provide technical support, driver updates, product information, and user's manual updates.

The following services are provided:

E-mail for technical support

address: support@moxa.com.tw

World Wide Web site for product information

address: http://www.moxa.com
or
http://www.moxa.com.tw

Table of Contents

1

Introduction

Network Enabler SDK 2 (NE SDK2) is designed for users who would like to run their own proprietary application on NE-4100 Series Network Enabler modules. NE SDK2 comes with a Moxa embedded OS (Moxa OS), TCP/IP communication stack. In addition, a Software Development Kit (SDK) and Windows utility allow system integrators to design proprietary solutions that do not use traditional PCs.

Moxa OS Ver. 2.x

Moxa NE SDK 2 has an embedded, small footprint, multi-tasking OS developed by Moxa Technologies Co. Ltd. The OS was developed originally for TCP/IP-based Terminal Server products in 1992. It is a powerful and reliable software platform with a user-friendly SDK. The major features of Moxa OS Ver. 2.0 are:

- 16-bit, Unix-like embedded operating system
- Small footprint (< 300 KB, including the TCP/IP protocol stack)
- Non-preemptive multi-thread system
- Stream I/O
- Standard BSD Sockets for TCP/IP programming with multi-TCP session support

NE SDK 2

To assist in the development of Network Enabler applications, Moxa provides a comprehensive and easy-to-use SDK designed for use on Windows 95/98/ME/NT/2000/XP platforms. The main features of NE SDK 2 are:

- Borland™ Turbo C 2.01 compiler
- SDK libraries with more than 100 function calls
- SDK Manager and EXE2AP utilities for software download and troubleshooting
- PComm Terminal Emulator
- More than 20 example programs
- Documentation

NE SDK V2.0 has the following programming features:

- Supports NE-4110S-P, 4110A-P, 4120S-P, 4120A-P, 4100T-P
- Up to 192 KB of user program space (large mode in Turbo C)
- Up to 160 KB of flash memory access space
- Up to 10 TCP sessions

Other advanced features include:

- EXE2FRM utility for advanced application deployment
- NECI library for utility development

Programmable Version of Network Enabler

Network Enabler models NE-4110S, NE-4110A, NE-4120S, NE-4120A and NE-4100T are the standard versions that run serial-to-Ethernet firmware for regular TCP Server, TCP Client, UDP, and Real COM operation modes.

The programmable versions of Network Enabler, denoted by adding "-P" to the model name, are NE-4110S-P, NE-4110A-P, NE-4120S-P, NE-4120A-P, and NE-4100T-P.

The Network Enabler SDK works only with the "-P" versions of Network Enabler.

SDK Function Group Overview

The SDK Library contains six groups of function calls. The following table lists the six function call groups along with their primary function.

SDK Library Function Group Overview	
Function Call Group	**Description**
Serial I/O API	Serial communication function calls that follow the same style as PComm Library. This API includes communication parameters, character read/write, block read/write, I/O control, break generation, and more.
BSD Socket API	Standard BSD Socket API for TCP/IP programming. Supports TCP and UDP communication.
Simplified Socket API	Simplified TCP/IP programming. Supports TCP and UDP communication.
System Control API	Overall system control for Network Enabler Module, including system configuration, system restart, system timeout counter.
Flash ROM Access API	Read, write, erase user storage space on flash.
Debug API	Sending messages to the debug window in SDK Manager.
Digital I/O API	Setting and retrieve digital I/O status including IN / OUT direction and it's high/low status.
Thread Control API	Multi-thread functions to control thread open, close, suspend, resume, and get thread status.
Time Server API	Set and retrieve time server, time zone, and time zone index.

Refer to "Network Enabler SDK2 API Reference" for detailed usage information for each of these functions.

2

Installing SDK

This chapter covers the following items:

- How to install NE SDK2 utilities and Library
- How to install the Turbo C compiler and environmental variables
- The location of Libraries and Example Files

System Requirements

You will need the following items to use SDK's development tools:

- Windows 9x, NT, ME, 2000, or XP operating system
- At least 64 MB of RAM
- At least 15 MB of hard disk space

Installing NE SDK2

To install NE SDK2 on a Windows system, insert the software CD-ROM included with the product into your computer's CD-ROM drive. The installation program should start automatically. Simply follow the onscreen instructions to complete the installation.

The installation program installs the following files:

Directory	Sub Directory\Files	Description
\NESDK2	\Example\	SDK example files
	\Include\	This directory contains all header files
	sdkconf.h	System configuration declaration
	sdkppp.h	PPP configuration declaration
	sdktask.h	Multi-thread program declaration
	sdkdbg.h	Debug symbol API declarations
	sdkflash.h	Flash access API declaration
	sdknet.h	Simplified Sockets API declaration
	sdksio.h	Serial I/O API declaration
	sdksock.h	BSD socket API declaration
	sdksys.h	System API declaration
	\Library\	This directory contains the SDK Library files
	C0sdk.obj	Object code for all APIs.
	Moxa_sdk.lib	API library for NE SDK.
	\Utility\	This directory contains SDK Manager, EXE2AP,

Directory	Sub Directory\Files	Description
		EXE2FRM, MoxaTerm utilities along with corresponding help files.
	\necilib\Library\	This directory contains the library for utility development under Windows
	neci_sdk.dll	DLL file for NECI (Network Enabler Configuration Interface) API
	neci_sdk.h	Declaration file for NECI API
	neci_sdk.lib	VC Library file for NECI API
	neci_sdk_b.lib	BC Library file for NECI API
	neci_sdk.chm	NECI help file
	\neci\Example\	Example programs for NECI
\TC	tcc.exe	Turbo C 2.01 compiler
	tlink.exe	Turbo C 2.01 linker
	\Include\	Include file for Turbo C.
	\Lib\	Library file for Turbo C

Desktop Icons

The SDK installation places one program icon on your computer's desktop. The icon serves as a shortcut to NE SDK Manager.

Setting up Environment Variables

During the installation procedure, the SDK setup program automatically sets up your environment variables for the Turbo C compiler. You may also set up the environment variables manually from Control panel → System, or you can add a command line to autoexec.bat.

Assuming the installed directory for SDK is C:\NESDK, the command lines are as follows:

```
path=c:\nesdk\tc;%path%
set INCLUDE=c:\nesdk\tc\include
set LIB=c:\nesdk\tc\lib
```

3

Developing User Applications

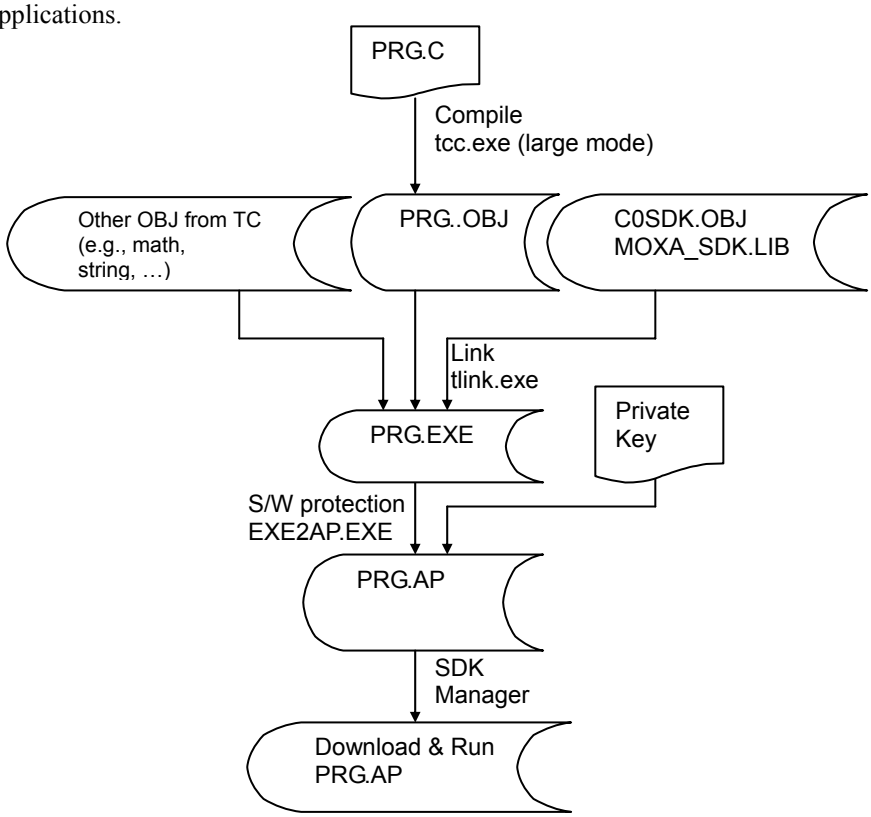
In this chapter, we cover the following items:

- Application Development Flow
- Software tools required for application development
- Software key protection
- Application deployment

To give you a better understanding of the development flow, we use the example program sdkser23.c to illustrate.

Application Development Flow

The following flowchart illustrates the standard process for developing your Network Enabler applications.

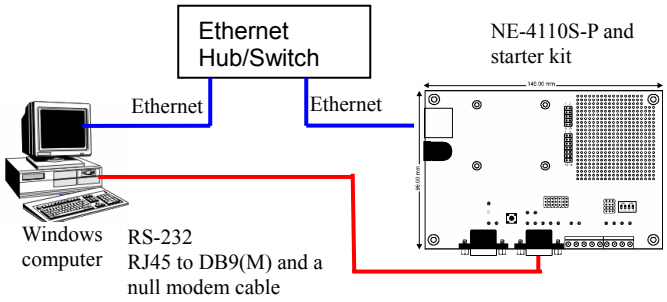


Development System

Before getting started on application development, you need to make sure that all of the required equipment is set up and ready to go. Take NE-4110S-P as an example. The basic configuration needed for application development is as follows:

- Windows 95/98/ME/NT/2000/XP computer with SDK installed
- NE-4110S-P with a starter kit
- Ethernet hub/switch (or use a cross-over Ethernet cable to link the computer and Network Enabler directly)
- Two Ethernet cables
- One RS-232 cable (Male DB9 to Female straight-through cable). This allows you to transmit and receive serial data along with RTS, CTS, DTR, DSR control signals. However, you will not be able to reach the DCD signal.

The basic wiring needed to set up a development system is shown below.



Note that for the serial-to-Ethernet version, P0 is the primary data port and P1 is used for the serial console port (3 wires). For the programmable version, **both P0 and P1 ports are freely programmable with the SDK**.

Source Code Editing

Since the Turbo C compiler and linker work under the traditional DOS environment, we recommend using DOS's **edit** program to edit your source code (simply type "edit" on the DOS command line and then hit Enter to start the editor). This is the approach we use in this manual.

You may also use "notepad" or any other text editor to write and edit source code. Turbo C's IDE environment is another good option. However, the environment variables need to be manually configured since NPort SDK works mainly from command line mode.

For example, to edit the file sdkser23.c, first open a DOS window, use the **cd** (change directory) command to open the folder c:\nesdk\sdk\example\serial, and then type **edit sdkser23.c**. You should now be able to view and edit the file sdkser23.c.

Compiling and Linking your Application

Under NE SDK, compiling and linking is done from DOS command mode. Because of this, you will need to create a make file before compiling and linking a program.

Creating a Make File

For tcc.exe, the following parameters should be enabled or properly assigned.

-I – assign include files directory.
-L – assign library files directory.
-O – optimize jumps.
-Z – maximum register usage.
-1 – 80186/286 instructions.
-ml– large mode
-c – compile only
-w – enable all warnings

For tlink.exe, the following parameters should be enabled or properly assigned.

-s – detailed map of segment.

For instance, the make file for C:\NESDK\EXAMPLE\SERIAL\SDKser23.c is as follows.

```
c:\nesdk\example\serial> type sdkser23.bat
tcc     -I..\..\include -L..\..\library -O -Z -1 -ml -c -w SDKser23.c
tlink /s ..\..\library\C0sdk+SDKser23,SDKser23,SDKser23,..\..\library\moxa_sdk
```

*1. Note that C0sdk.obj must be placed in the first position when linking object codes. Otherwise, the entry point of the program will be incorrect.

Generating an EXE file

You should be able to generate an EXE file after compiling and linking a program without receiving any error messages. For example, after you run the make file, sdkser23.bat, in c:\nesdk\example\serial, the file sdkser23.com is generated in the same directory.

Using EXE2AP

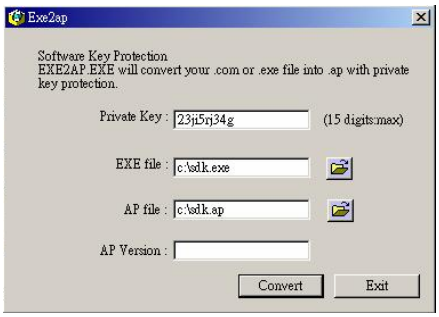
After successfully generating an EXE file, you will need to use EXE2AP.EXE to convert the EXE file into an AP file that has an embedded "Private Key." This must be done before downloading the AP file to the Network Enabler..

Protect Your Software with Private Key

System integrators often provide their proprietary knowledge to solve particular problems for their customers. Once you start providing Network Enabler as a platform for other users, you will need to manage and protect your intellectual property that is embedded in the Network Enabler. For this reason, Moxa Network Enabler provides a private key that you can embed into your application (AP). The target Network Enabler should have the same private key, which can be set up with SDK Manager. The Moxa OS automatically checks to make sure that the private key in the AP and kernel match.

Embedding a Private Key in your Application

To embed a private key in the target AP (e.g., sdkser23.com) first double click on the Exp2ap desktop icon.



Next, take the following steps to complete the process of embedding a private key:

1. Type the private key in the "Private Key" field. E.g., you could choose "23ji5rj34g" as the private key.
2. Click on the folder icon to the right of **EXE** file and then navigate to the EXE file you would like to convert (e.g., sdk.exe).
3. The name of the target AP file (sdk.ap, for this example) will appear in the **Protected User Application** text input box. If desired, you can change the file name, or click on the folder icon and navigate to a different folder in which you would like to store the resulting AP file.
4. Prior to downloading and running the AP file, use SDK Manager to set up the same private key for the Network Enabler. Refer to next section for more details.

Embedding a Private Key from the Command Line

To give program developers a convenient alternative for producing an AP, EXE2AP can also be activated from the DOS command line. Enter the command in the following format:

EXE2AP –Kxxxxxxx –Syyyyyyyy.yyy –Dzzzzzzzz

Argument description

-K Private key. Select a key with at most 15 characters or numbers (e.g., you could choose private key = 23ji5rj34g).
-S Source file. The source COM file.
-D AP file. Note that you should not type the file extension name since it is predefined as "AP".

For example, to generate the AP file from the command line, add the following line after tlink in the batch file.

```
exe2ap -K23ji5rj34g -Ssdk.exe -Dsdk
```

Using SDK Manager

After the application has been successfully compiled and linked, it is time to use SDK Manager. SDK Manager is a utility that provides the following functions.

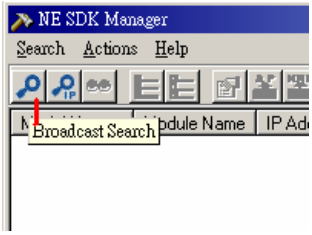
- Search and locate Network Enablers.
- Change the IP, Netmask, Gateway, default serial comm parameters, and operation modes for Network Enabler.
- AP download
- Debug window

Searching for Network Enablers

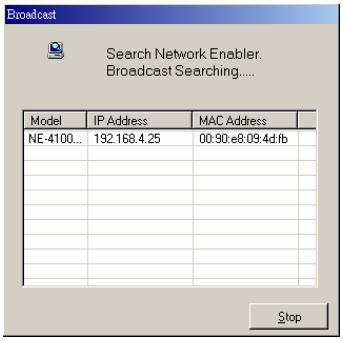
The Search Menu provides two different methods to search the network for a Network Enabler. **Broadcast Search** is used to locate all Network Enablers connected to the same LAN as the host, and **Search by IP** is used to locate a specific Network Enabler, particularly if it is located outside the LAN and can only be accessed by going through a router. The Search Menu also provides the **Locate Gateway** function that can be used to identify a particular Network Enabler.

Broadcast Search

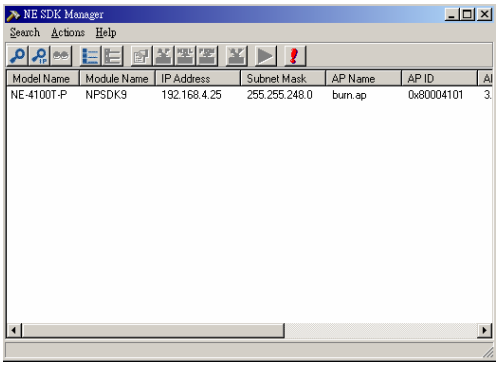
To access this function, select **Broadcast Search** under the **Search** menu, or click on the **Broadcast Search** toolbar icon.



The **Broadcast** window will display the progress of the search.

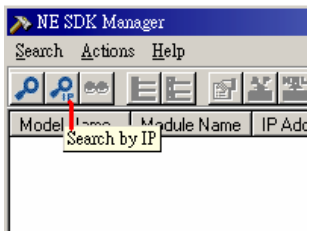


Once the search is complete, the **Model Name**, **Serial No.**, **Server Name**, **AP ID**, **MAC Address**, **IP Address**, **Subnet Mask**, and **OP Mode** of each Network Enabler that was located will be displayed in the SDK Manager window.

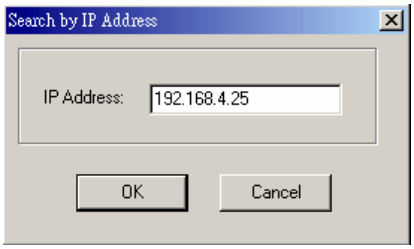


Search by IP

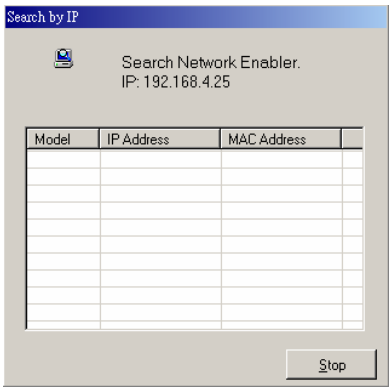
To access this function, select **Search by IP** under the **Search** menu, or click on the **Search** by IP toolbar icon.



When the **Search by IP Address** window opens, type the IP address in the text input box, and then click **OK**.

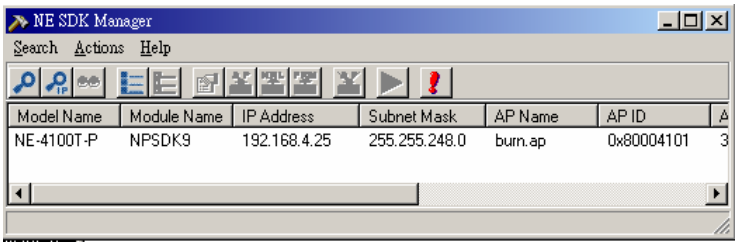


The **Search by IP** window will display the progress of the search.



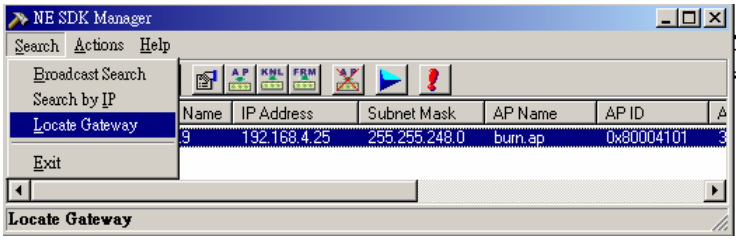
NOTE *If you receive the **Specified device not found** error message, as shown above, and the Network Enabler is located on the same LAN as the host, try using Broadcast Search, or change the host computer's IP address and/or Netmask so that the computer and Network Enabler are on the same subnet.*

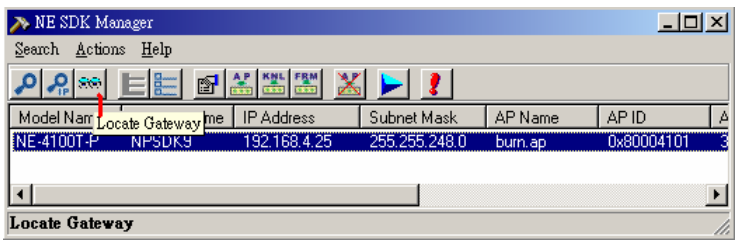
If the search is successful, the **Model Name**, **Serial No.**, **Module Name**, **AP ID**, **MAC Address**, **IP Address**, **Subnet Mask**, and **OP Mode** of the Network Enabler will be displayed in the SDK Manager window.



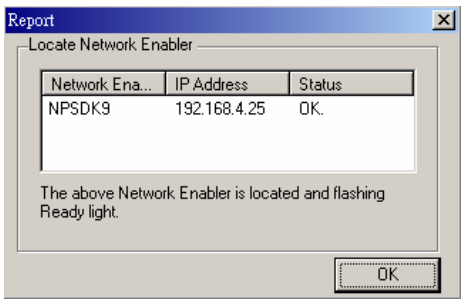
Locate Network Enabler

The **Locate Network Enabler** function is used to find the physical location of a Network Enabler unit when there are multiple Network Enablers on the same network. To use this function, first click on the device you would like to locate to highlight the device's information, and then select **Locate Network Enabler** under the Search menu, or click on the **Locate Network Enabler** toolbar icon.





If the **Locate Result** is **OK**, then the Ready light on the located Network Enabler unit will blink steadily, allowing you to identify the Network Enabler and IP Address.



Click on **OK** to cause the Network Enabler unit's Ready light to stop blinking.

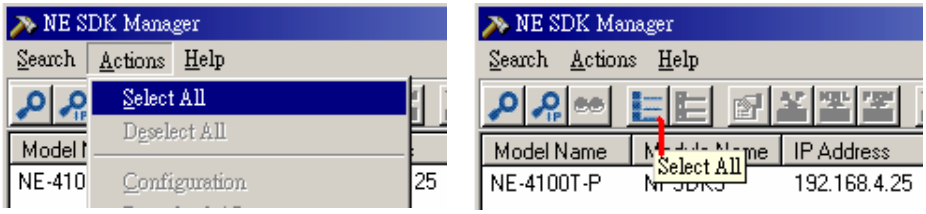
Selecting/Deselecting Network Enablers

The **Select All** and **Deselect All** functions are provided to make it easier to configure large numbers of Network Enablers.

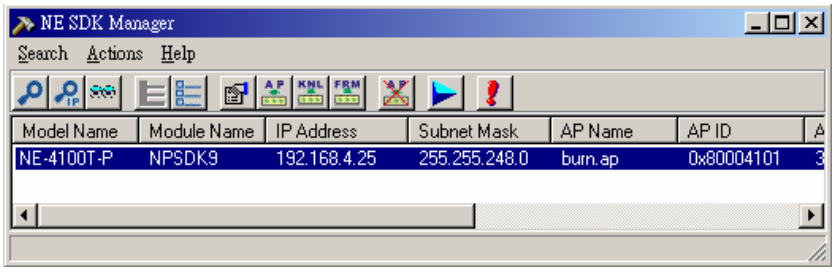
NOTE *SDK Manager includes a **multi-selection** capability. Hold down the Ctrl key to select multiple Network Enablers that are not listed in order, or hold down the Shift key to select all Network Enablers listed between the first and last Network Enabler that you click on.*

Select All

The **Select All** function is used to select all Network Enablers listed in the SDK Manager window. To use this function, select **Select All** under the **Actions** menu, or click on the **Select All** toolbar icon.

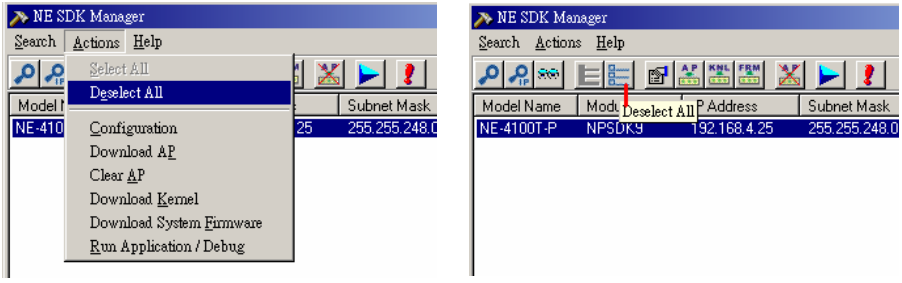


This will cause all Network Enablers listed in NE SDK Manager window to become highlighted.

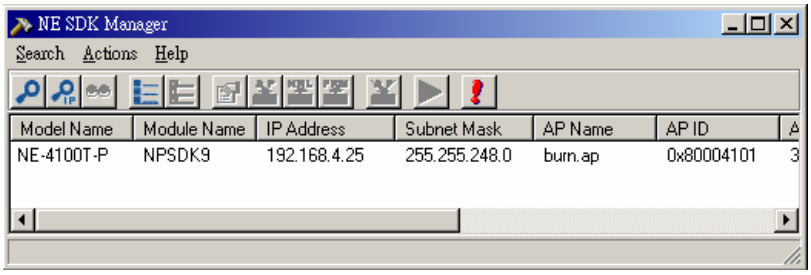


Deselect All

The **Deselect All** function is used to deselect all Network Enablers listed in the SDK Manager window. To use this function, select **Deselect All** under the **Actions** menu, or click on the **Deselect All** toolbar icon.



This will cause all Network Enablers listed in the SDK Manager window to become unhighlighted.

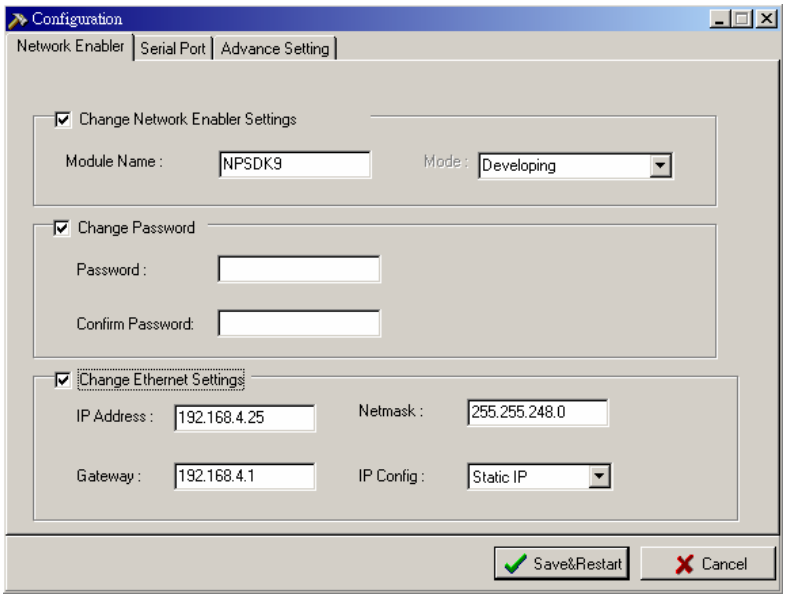


Configuring a Network Enabler

To use this function, select **Configuration** under the **Actions** menu, or click on the **Configuration** toolbar icon. The **Configuration** window opens with the **Comm. Gateway** tab selected. Each of the six **Configuration** window tabs is discussed in the following subsections.

Network Enabler Tab

To make changes, first click in the **Network Enabler** box, and then modify **Network Enabler Name** and network settings.



Change Network Enabler Settings		
Setting	**Options**	**Comments**
Module Name	*Alphanumeric*	Determined by user.
Mode	*Running Mode*	User AP automatically executes after system reboot.
	Developing Mode	User AP doesn't execute after system reboot. You may activate the user AP via SDK Manager. We strongly recommend that users configure the NE module to this mode while developing a program. Doing so can prevent the NE from encountering a dead lock status caused by a programming error.

NOTE *We strongly recommend that users configure the NE module to **Developing Mode** while developing a program. Doing so can prevent the NE from encountering a dead lock status caused by a programming error.*

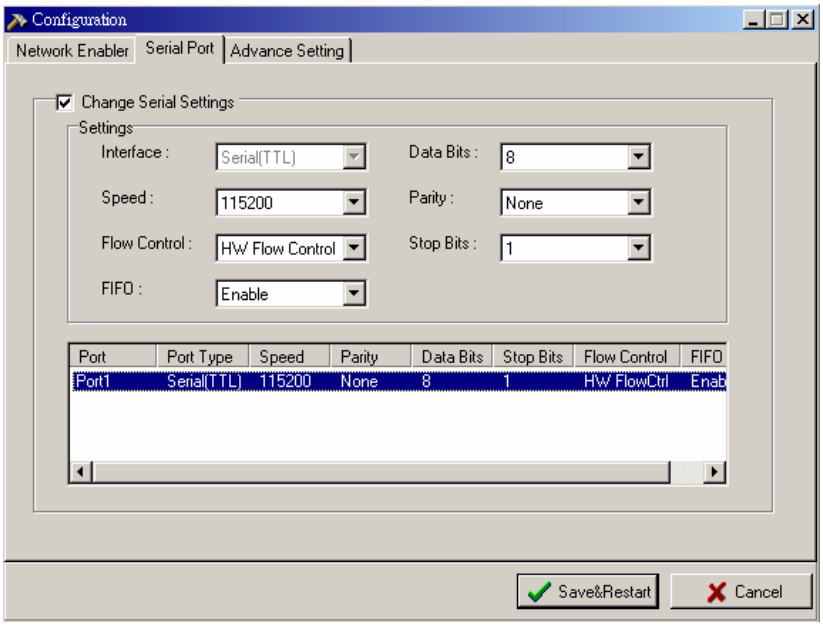
Change Network Enabler Settings		
Setting	**Options**	**Comments**
Password	*Alphanumeric*	Maximum number of characters allowed is 16.
Confirm	*Alphanumeric*	Retype the Password entered in the first box.

Change Network Enabler Settings		
Setting	**Options**	**Comments**
IP Address	*xxx.xxx.xxx.xxx*	*Available with multi-selection* When multiple target Network Enablers are selected, they will all be configured with the same IP address.
Netmask	*xxx.xxx.xxx.xxx*	*Available with multi-selection* Define a range of IPs by inputting Start and End values. These IP addresses will be assigned in sequence to selected Network Enablers, in order of their appearance in the Manager window.
Gateway	*xxx.xxx.xxx.xxx*	User defined IP address
IP Config	*DHCP*	Uses DHCP protocol to request an IP automatically from the DHCP server.

Serial Port Tab

This section of the Configuration window allows you to modify the serial communication parameters. The modified parameters will be saved in the Network Enabler's flash ROM.

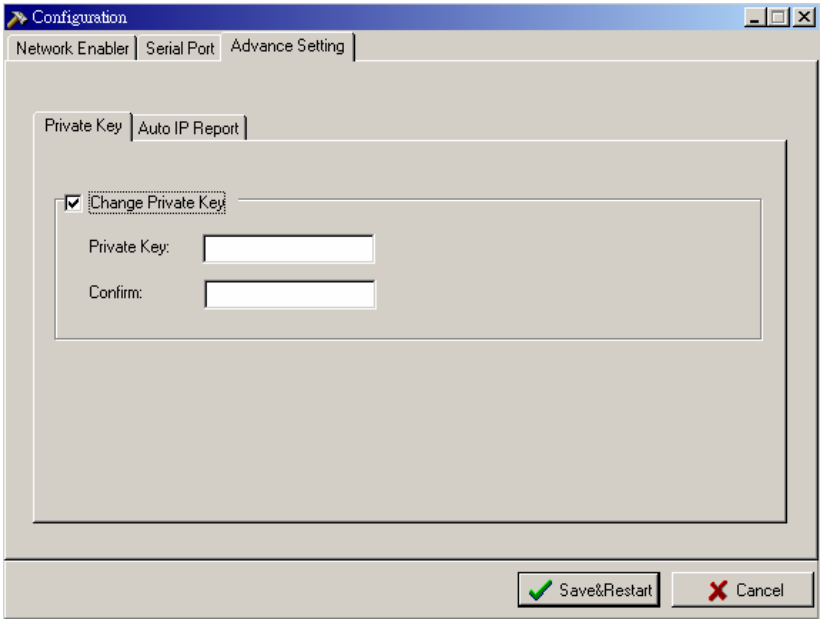
To make changes, first click in the **Change Serial Settings** box, and then click on the port in the display area in the bottom part of the window to make changes to the configuration of the serial ports.



Serial Port Settings	
Setting	**Options**
Port Type	NE-4000T: Serial(TTL) NE-4110S/4120S:RS-232 NE-4110A/4120A: RS-422, RS-485(2w) selected by JP2
Speed	50 to 115.2kbps
Flow Control	None, HW Flow Control, SW Flow Control
FIFO Mode	Enable, Disable
Data Bits	5, 6, 7, 8
Parity	None, Even, Odd, Space, Mark
Stop Bits	1, 2

Private Key Tab

As mentioned above, a private key is required for both the AP and the Network Enabler. To embed the private key in the Network Enabler, first click in the **Change Private Key** box, and then enter the **Private Key** in the Private Key input box. Finally, enter the same Private Key in the **Confirm** box.

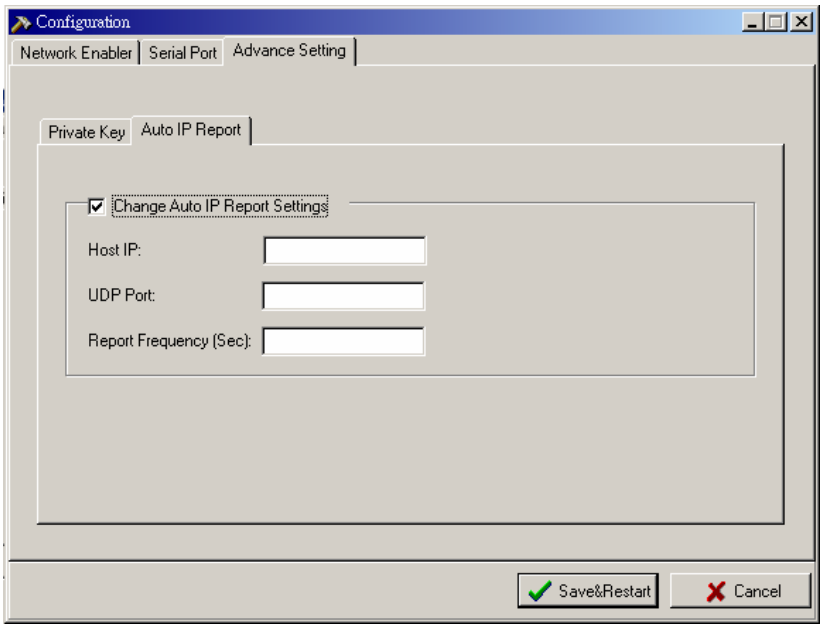


Private Key Settings		
Setting	**Options**	**Comments**
Private Key Input	*Alphanumeric*	Maximum number of alphanumeric characters allowed is 15.
Confirm Private Key	*Alphanumeric*	Retype the Private Key entered in the first box.

Auto IP Report

In a dynamic IP environment, the Network Enabler's IP address changes from time to time, making it hard for the host computer to locate the Network Enabler. To solve this problem, Network Enabler automatically reports it's location to a remote host computer. The Location Report section of the Configuration window allows you to set up the IP address corresponding to a specified UDP port for a host computer. The **Report Frequency (Sec)** setting determines how frequently the location report is issued. You can use the NECI library to develop software that learns the location of a remote Network Enabler. Refer to the NECI Library help file, **..\NESDK\neci\Library\neci_sdk.chm,** for more details.

To make changes, first click on the **Auto IP Report** tab, and then enter **Host IP**, **UDP Port**, and **Report Period (Sec)**.



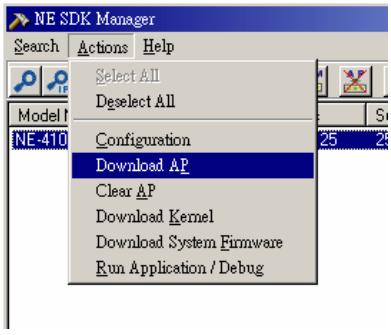
Location Report Settings		
Setting	**Options**	**Comments**
Host IP	*xxx.xxx.xxx.xxx*	IP of a remote host computer. Leave this field blank to disable this function.
Listen Port	0 to 1024	Host computer's UDP listen port.
Report Period (Sec)	0 to 60 sec	Set to 0 to disable this function.

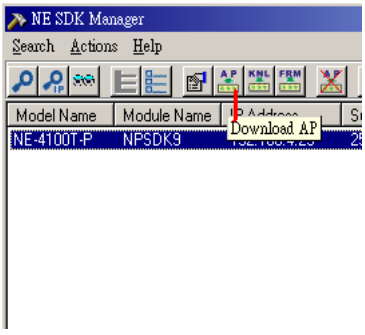
Application Download

Up to this point, we have only discussed how to configure the Network Enabler's network and other parameters. The next thing to do is download the prepared application (AP) to the Network Enabler for testing.

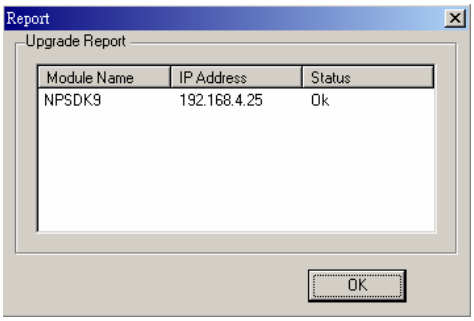
Download AP

To download the AP, first select **Download AP** from the **Actions** menu, or click on the Download AP toolbar icon.



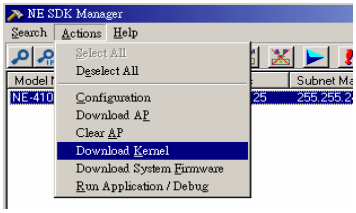


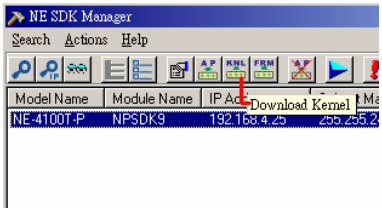
Use the **Open** button to navigate to the folder that contains the AP file, or just type the AP filename directly in the Filename input box. Click **OK** to start downloading the AP. The following window appears when the AP has finished downloading.



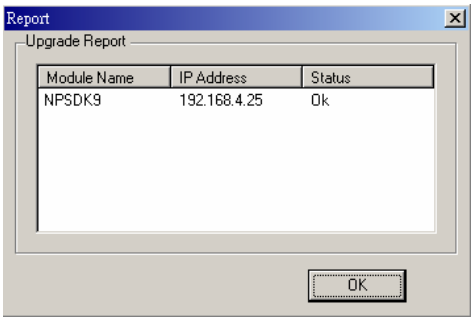
Download Kernel

The kernel for Network Enabler is preloaded before shipment. When you receive an updated kernel from Moxa, you can use the Download Kernel function to update the kernel yourself. To download the kernel to the Network Enabler, first select **Download Kernel** from the **Actions** menu, or click on the **Download Kernel** toolbar icon.





Use the **Open** button to navigate to the folder that contains the kernel file, or just type the kernel filename directly in the Filename input box. Click **OK** to start downloading the kernel. The following window appears when the kernel has finished downloading.



Run Application / Debug

You should now be ready to run your application, but first we need to discuss the two Network Enabler operation modes.

Choose Running Mode or Developing Mode

Network Enabler provides two operation modes.

Developing Mode

When set for **Developing Mode**, the Network Enabler will not start running the application automatically after the system is rebooted. To start the application, select **Run Application /Debug** from the **Actions** menu, or click on the **Run Application /Debug** toolbar icon. We suggest setting the operation mode to Developing Mode when going through the debugging process.

Running Mode

When set for **Running Mode**, the Network Enabler automatically executes the application after the system boots up. However, you can stop the application from within SDK Manager. This mode is suitable for regular shipment.

How to Switch from Running Mode and Developing Mode

You may use NE SDK Manager to switch from running mode and developing mode. To set running and developing mode for selected Network Enabler in NE SDK Manager's list, click on the "Configuration" button to enter another Window.

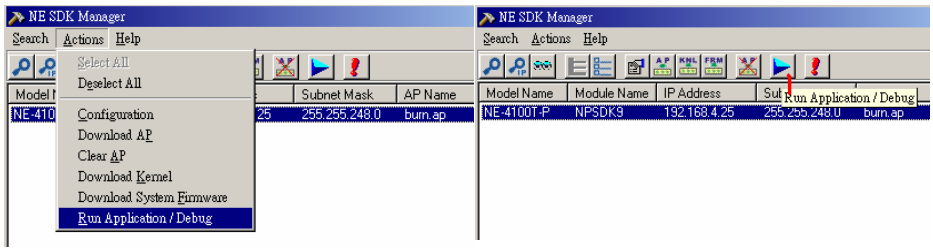
Choose "Running" or "Developing" for "mode" item on the Network Enabler page.

If you find that the NE SDK Manager cannot locate your Network Enabler over the network, there may be a problem with your application. The best way to solve this kind of problem is to set the module to Developing mode from the BIOS. Use these steps to set up Developing mode.

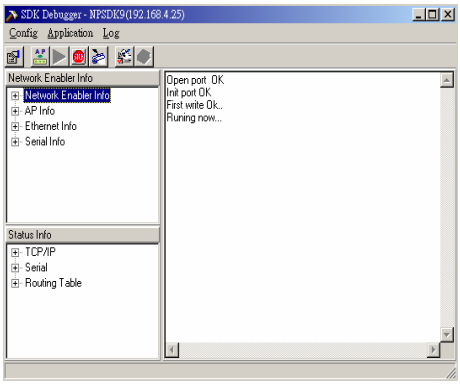
1. Power off the Network Enabler
2. Use a null modem cable to connect P1 of the Network Enabler to your PC's COM1.
3. Use HypterTerminal to open the COM1 using 115200 bps, N,8,1.
4. Use a jumper to short JP1, and then power on the module
5. You should be able to see the BIOS in the terminal emulator screen. Press "N" to set to "Developing" mode.
6. Power off the Network Enabler and remove the jumper from JP1.
7. Power on the Network Enabler, to return to normal operation.

Start Application

To start an application loaded in the Network Enabler, first select **Run Application /Debug** from the **Actions** menu, or click on the **Run Application /Debug** toolbar icon.



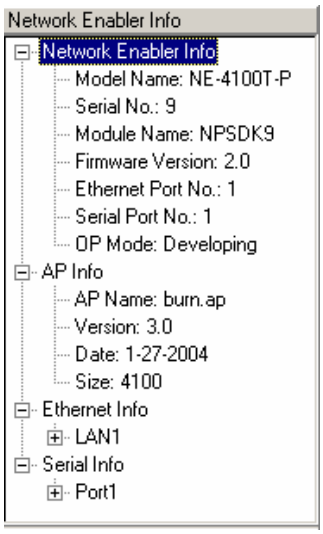
The Debugger window will appear as shown in the figure at the right. From the SDK Debugger window, click on the **Start Application** button to start running your Network Enabler application. There are three message areas in the Debugger window as shown in the figure.



Network Enabler Info Area

The Network Enabler's basic configuration settings are shown in the **Network Enabler Info.** area. The information includes the following:

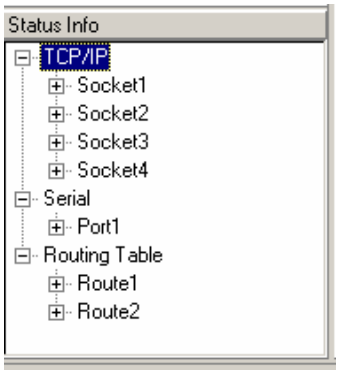
- **Network Enabler Info**—includes Model Name, Serial No., Module Name, Firmware Version, number of Serial Ports, number of Ethernet Ports, and OP Mode.
- **AP Info**—includes latest AP Name, Version, Date, and Size.
- **Ethernet Info**—includes MAC address, IP address, Netmask, and Gateway.
- **Serial Info**—includes serial communication parameters.



Status Info Area

The **Status Info** area shows Network Enabler's online operation status. There are two main items.

- **Ethernet**—Network Enabler supports up to ten user programmable TCP sessions. You can use Remote IP, Remote Port, Local IP, Local Port, Socket Type, and Connect Status to monitor the usage of these TCP sessions from SDK Debugger,
- **Serial**—shows the total Tx and Rx counts starting from when the SDK Debugger was activated. In addition, you can view the line status, including RTS, DTR, CTS, DSR, and DCD.



Debug

The main debug approach for Network Enabler is to put the debug API in your source code as a debug symbol. The debug message will be sent to SDK's debug window via the Ethernet console. The "single step" and "break point" debug methods are currently not supported.

Example Programs

Network Enabler's SDK is a comprehensive utility designed to match a programmer's basic intuition. We've prepared several examples to let you quickly develop your first application. All of the example programs are in the folder \NESDK\EXAMPLE.

NOTE *If you forget the NE-41xx-P's password, enter the Network Enabler's BIOS to reset all settings to their factory default values.*

1. *Use a PC with PComm Terminal and NESDK installed.*
2. *Connect COM1 (115200 bps, N, 8, 1) to P1 on the NE-4100 series Starter Kit.*
3. *Use a jumper to short JP1 on the NE-4100 series module, and then turn on the power of the NE-4100 series SDK. The BIOS screen will appear:*

```
Welcome to Network Enabler Module MainTain program Ver 1.2
Module Type : NE-4100S
Serial No : 00340        MAC Address : 00:90:e8:09:4d:9d
MP Flag : Yes
------------------- >>> Main Menu <<< ----------------------
[D] : Debug                          | [7] : Embedded MAC Test
[H] : Help (Reshow Manual)           | [A] : Jumper & Button & LED Test
[R] : Reboot                         | [B] : Burning Test
[0] : Bios Upgrade                   | [C] : DIO Test
[1] : Firmware Upgrade               | [E] : Jump to Firmware
[2] : FlashRom Test                  | [G] : File's Disk Upgrade
[3] : SDRAM Test                     | [I] : Set/Clean MP-Flag
[4] : Set Serial & MAC Address       | [J] : Terminal
[5] : UART Test                      | [L] : Set SWID
[N] : Set to Development mode        | [F] : Set FirmwareChang-Flag
[T] : Set to Factory Default         |
-------------------------------------------------------------
\>
```

4. *Press "T" or "t" on the keyboard to reset to factory defaults.*
5. *Power the Network Enabler off and then on. The Network Enabler will be operating with the factory default settings. The default IP address is 192.168.127.254, and the default netmask is 255.255.255.0.*

NOTE *The Moxa OS is not a true multi-tasking environment. While waiting for a loop to be processed, we must call at least 1 Moxa SDK function to prevent needing to wait for the OS to respond.*

Do **NOT** write your AP in the following manner:

```
for (;;) {
   if (aa==1)
      break;
   }
```

Do **NOT** write your AP in the following manner:

```
while(1) {
   if (aa==1)
      break;
   }
```

The easiest way to overcome this problem to call the **`sys_sleep_ms()`** function:

```
for (;;) {
   if (aa==1)
      break;
   sys_sleep_ms(1);
   }
```

4

Application Deployment

After your application has been tested and is ready to go, it's time to deliver the Network Enabler with your software. In this chapter, we offer several useful tips and tools that let you organize the production process efficiently.

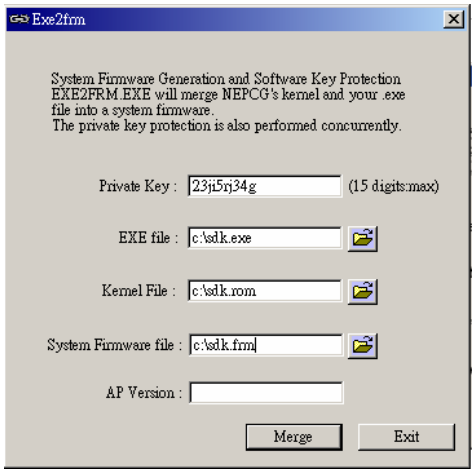
Changing Default Settings for Multiple Network Enablers

The default settings for Network Enabler may be different from the settings you want to use when shipping the product to your customers. To make the job easier, SDK Manager can be used to change multiple Network Enabler settings, all at the same time. The main procedure is as follows:

1. First, connect all Network Enablers and the host computer to the same Ethernet hub or switch.
2. Start SDK Manager, and use Broadcast Search to search all Network Enablers. All of the Network Enablers connected to the hub or switch should appear in the SDK Manager window.
3. Enter the configuration menu, and change each field under each tab to the desired settings.

Creating System Firmware—EXE2FRM

In general, the AP file is downloaded to the Network Enabler separate from the kernel. However, NE SDK provides you with another advanced tool, EXE2FRM, that is used to merge the kernel and AP file into a system firmware file. In this way, product delivery is made easier, and furthermore, the quality of the software is assured.
The EXE2FRM program can be found in the .\UTILITY\EXE2FRM directory. For easy access, simply create a shortcut on your desktop to the EXE2FRM program. After starting the EXE2FRM program, the window shown at the right will appear.



To generate a system firmware file, you will need to input the Private Key, original EXE file, and Kernel file. The kernel file can be found in .\FIRMWARE. After entering the above information, enter a file name for the generated System Firmware file. To download the System Firmware, open SDK Manager and locate the target Network Enabler, and then select **Download System Firmware** under the **Actions** menu, or click on the **Download System Firmware** toolbar icon.

Command Line Usage

To speed up program development, EXE2FRM can also be activated from the DOS command line, as shown below. Simply type:

EXE2FRM –Kxxx –Syyyyyyyy.yyy –Fyyyyyyyy.yyy –Dzzzzzzzz

Argument description

-K Private key. Select a key with at most 15 characters or numbers (e.g., you could choose private key = 23ji5rj34g).
-S Source file. The source EXE file.
-F The file name for the kernel.
-D FRM file. Note that you should not type the file extension name since it is predefined as "AP".

For example, to generate an FRM file from the command line, add the following line after tlink in the batch file.

```
exe2frm –K23ji5rj34g –Ssdk.exe –Fsdk.rom –Dsdk
```

Field Utility

SDK Manager is the main field utility for configuration and troubleshooting, but Network Enabler's SDK also provides you with a NECI Library that can be used for developing your own proprietary utility. For more information, refer to next section.

Developing a Network Enabler's Utility

Network Enabler SDK provides the Network Enabler Control Interface (NECI) Library for customers who want to develop their own utility for use on a Windows computer.

NECI Library

NECI (Network Enabler Configuration Interface) is a set of APIs that run on a Windows 95/98/ME/NT/2000/XP system to search, locate, and configure the Network Enabler over the network. The NECI library can be found in the folder .\NECI\LIBRARY. For more information, refer to document neci.chm in that directory. Examples are located in .\NECI\EXAMPLE.

AP ID

Network Enabler has a special parameter, called "AP ID," that is of particular interest to NE SDK programmers who intend to repackage Network Enabler with their own application, new product name, and new model number. The AP ID can be used to distinguish between different application programs.

You can develop several versions of an application for use with different projects. In this case, it is necessary for the host utility to identify which application is running on the Network Enabler. Network Enabler uses the AP ID to identify which module is associated with which NECI API. The AP ID is stored in the System Parameter Block within the firmware.

To set up the AP ID, you will need to insert the following code at the beginning of your own source code (refer to the SDK Library System Control API).

```
void sys_Set_RegisterID( unsigned long id);
```

The AP ID can be read by NE SDK Manager, as well as by your own management utility created with NECI Library.

5

Programming Notes

This chapter provides a more in-depth explanation of the following topics.

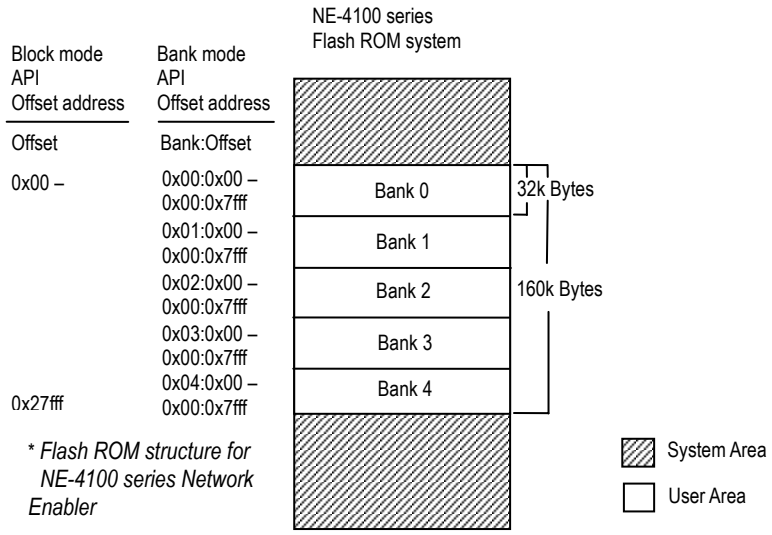
- Flash ROM Access
- Serial I/O Buffer

Flash ROM Access

Network Enabler contains a block of flash memory that is available for data buffering, or to store a small amount of data. To make the operating system more efficient, there is no file system.

Flash ROM Structure

The flash ROM structure is illustrated below.



** Flash ROM structure for NE-4100 series Network Enabler*

NE-4100 Series Network Enablers have 160 KB of user program space. There are two modes for accessing data stored in the flash ROM.

1. **Block mode**—The API "flash_xxxx( )" runs in block mode. The entire user area is treated as a block, with sequential addresses used for each byte of data. The location of data in the flash is specified by an offset address.
2. **Bank mode**—The API "sys_flash_xxx( )" runs in bank mode. Each bank contains 32 KB of flash memory. The location of data in the flash is specified by a combination of bank and offset addresses.

Writing Data

Data is written sequentially into the flash according to your program's instructions. Each time data is written, the offset address automatically moves to the next writing point. For example, if the original offset address is 0x0100, and 20 bytes of data are stored, then the API moves the offset address to 0x0121.

The API flash_length( ) gets the length of the data currently stored in the flash. The return code "-2" indicates that the bank is full. To reuse this bank, you first need to copy the data currently in the bank to another bank. After the bank is cleared by the API, you can again write data, starting at beginning of the bank.

Reading Data

You may use the offset address in block mode or bank:offset in bank mode to retrieve any of the data in the flash memory.

Erasing Data

Due to the characteristic of flash, each byte in the flash can only be written once. To re-write the same byte, you need to erase the entire bank.

Serial I/O Buffer

The internal buffer for serial I/O is located in the kernel. There are 2 KB of buffer space for receiving data, and 4 KB for sending data.

Serial Interfaces

NE-4110A and NE-4120A support RS-485 (2-wire) and RS-485 (4-wire). The interface is determined by the jumper on the Network Enabler Module.

NE-4110A and NE-4120A support ADDC™ (Automatic Data Direction Control) by software. Due to the limitation of software timing, we recommend that you set the RS-485 speed to 19200 bps or less. The module will still operate at a higher baud rate, but data loss may occur if the CPU loading is too great.

6

FAQ

In this chapter, we present a short list of Questions and Answers that you can refer to solve frequently encountered problems.

Q: Why can't SDK Manager be used to configure or start debugging the Network Enabler after locating the Network Enabler over network?

A: SDK Manager uses UDP (which broadcasts packets over the network) to search for Network Enablers installed on the network. The configuration and debugging functions use TCP communication. A typical error code from SDK Manager is "timeout", "-2".The following reasons can cause TCP communication in SDK Manager to fail, while UDP works fine.

1. IP conflict
2. Netmask setting
3. Default gateway

The fastest solution is to reset the IP and netmask so that the PC and Network Enabler are on the same subnetwork. You should also remove the default gateway.

Q: What is the difference between "Download AP" and "Download Firmware?"

A:

1. "Download AP" is for downloading the Application Program to the Network Enabler, but "Download Firmware" involves downloading a file that combines the Application Program and "Kernel."
2. The Firmware contains both the Network Enabler's application program and kernel. This is convenient for developers when distributing the final software package to end users.

Q: Why does SDK Manager fail to download the AP to the Network Enabler, and show error code "-5"?

A:

1. The error code "-5" occurs when the Private Key on the Network Enabler and the AP mismatch. The solution is to reset the Private Key of the Network Enabler to the correct value using SDK Manager.