neoway

Get connected Get smart

OpenLinux

Developer Guide

Issue 1.1 Date 2020-01-07

Notice

This document provides guide for users to use OpenLinux.

This document is intended for system engineers (SEs), development engineers, and test engineers.

THIS GUIDE PROVIDES INSTRUCTIONS FOR CUSTOMERS TO DESIGN THEIR APPLICATIONS. PLEASE FOLLOW THE RULES AND PARAMETERS IN THIS GUIDE TO DESIGN AND COMMISSION. NEOWAY WILL NOT TAKE ANY RESPONSIBILITY OF BODILY HURT OR ASSET LOSS CAUSED BY IMPROPER OPERATIONS.

THE INFORMATION IN THIS DOCUMENT IS SUBJECT TO CHANGE WITHOUT NOTICE DUE TO PRODUCT VERSION UPDATE OR OTHER REASONS.

EVERY EFFORT HAS BEEN MADE IN PREPARATION OF THIS DOCUMENT TO ENSURE ACCURACY OF THE CONTENTS, BUT ALL STATEMENTS, INFORMATION, AND RECOMMENDATIONS IN THIS DOCUMENT DO NOT CONSTITUTE A WARRANTY OF ANY KIND, EXPRESS OR IMPLIED.

Neoway provides customers complete technical support. If you have any question, please contact your account manager or email to the following email addresses:

Sales@neoway.com

Support@neoway.com

Website: http://www.neoway.com

Contents

About This Document

Scope

This document is applicable to OpenLinux modules, including:

- N720 OpenLinux
- A70 OpenLinux

Audience

This document is intended for system engineers (SEs), development engineers, and test engineers.

Change History

Issue	Date	Change	Changed By
1.0	2019-11	Initial draft	Tommy Sun
1.1	2019-12	Modified some steps	Tommy Sun

Conventions

Symbol	Indication
	This warning symbol means danger. You are in a situation that could cause fatal device damage or even bodily damage.
	Means reader be careful. In this situation, you might perform an action that could result in module or product damages.
	Means note or tips for readers to use the module

1 Overview

OpenLinux is an open platform that Neoway developed for users to implement various custom functions.

This document provides guidelines in the following topics about OpenLinux modules:

Business model

how to commission the the Evaluation Board (EVB)

how to prepare drivers and debugging tools

how to set up SDK environments

how to develop applications

how to develop based on source codes

how to upgrade firmware

production recommendations and tools.

2 Hardware Design

Reference documents	*Neoway_N720_OpenLinux_HW_User_Guide* *Neoway_A70 Series_OpenLinux_HW_User_Guide*
Description	Pin definitions of the OpenLinux modules and hardware design guidelines.

3 Commissioning EVB

Reference documents	To commission the EVB, refer to the EVK user guide.
Description	• EVB interfaces and functions • EVK components • Power supply • Serial tool

4 Software Development Process

4.1 Requirements

- Get familiar with the development of the Linux applications and the common Linux system commands.
- Master the basic knowledge of network protocols and get familiar with the usage of TCP/FTP/HTTP protocols in Linux.
- Master the driver configuration by modifying the device tree.

4.2 Development Process

Step 1: Install the drivers and learn about the ADB debugging tool.

Refer to chapter 6 and chapter 7.

Step 2: Establish the SDK environments.

Refer to chapter 8.

- Debugging and developing environments: Windows 7 and later versions.
- SDK and source code compiling: Ubuntu 14.04 or 16.04

Step 3: Develop application programs and generate a file system.

Refer to chapter 9 and chapter 10.

Step 4: Configure the automatic startup of applications with the system.

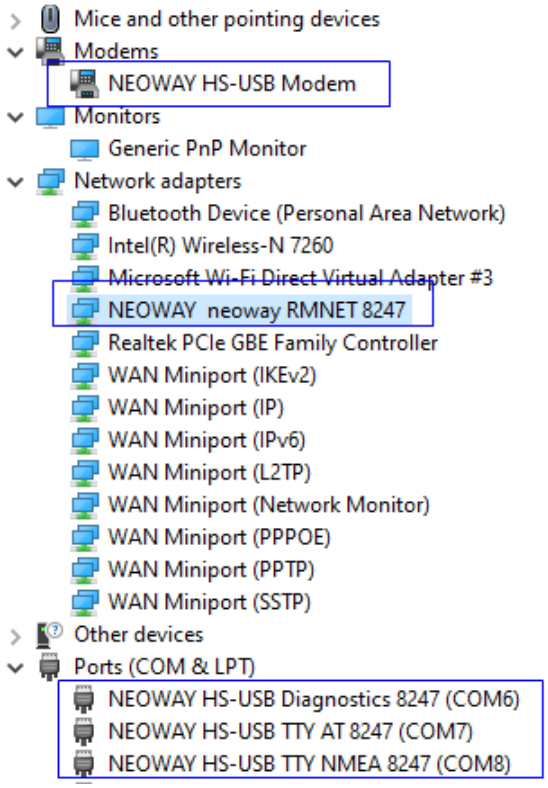
Refer to chapter 11.

Step 5: Upgrade the module firmware.

Refer to chapter 12.

5 Installing USB Driver

Reference documents	*Neoway_Openlinux_USB_Driver_Installation_Guide (Windows)*
Description	Install the USB driver in Windows OS. To facilitate the debugging of programs developed by customers, the OpenLinux modules allow the Device Manager to display the ADB interface by default after the driver is installed.



6 ADB Tool (Windows)

Name	Fastboot.rar
Storage path	In the tool folder of the SDK package
System Requirements	Windows 7/8/10
Functions	• Used to access the module kernel (Linux OS) • Used to push applications into the module for remote debugging • Used upgrade single function

- To obtain the SDKs and relevant documents, you need to sign an NDA agreement with Neoway.
- To use the ADB for debugging in Linux, install the Android-ADB corresponding to the kernel.
- The application that is pushed into the module might be lost if the system recovers from an exception.
- After you finalize your applications for production, embed them into the firmware by customizing the file system in source codes.

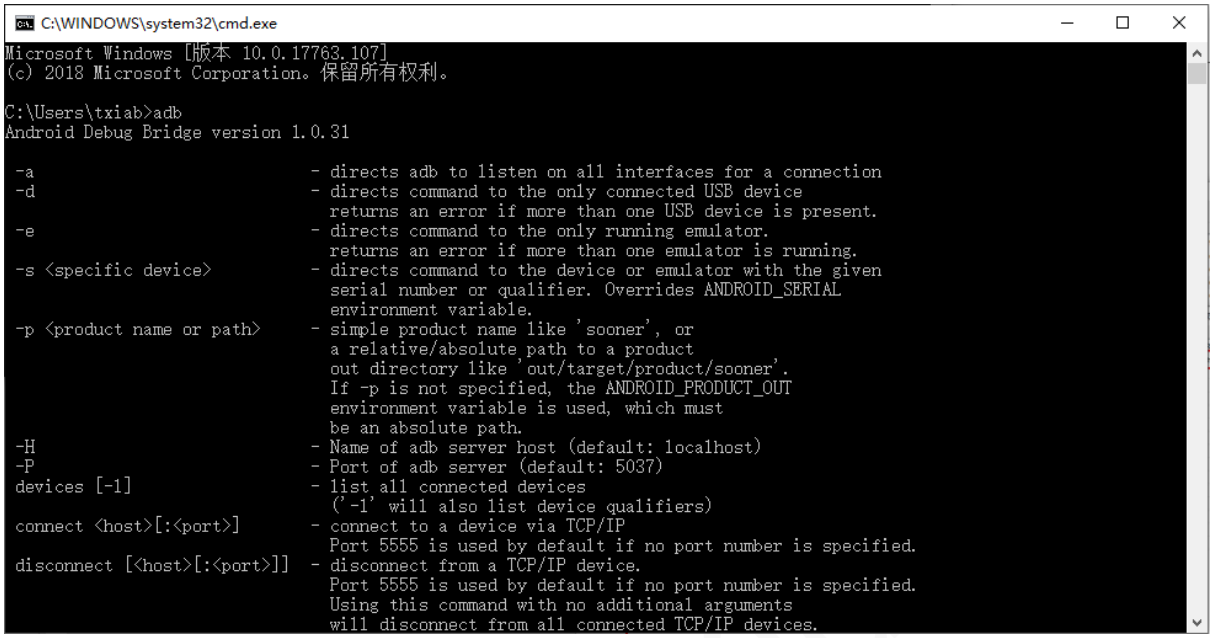
6.1 Using ADB Commands

Step 1: (Optional) To execute ADB commands in the CLI that is not indexed to the folder path of the **adb** script, configure the environment variable for the script.

1. Right-click **This Computer** and choose **Properties** from the shortcut menu.
2. In the System window, click **Advanced system settings** in the left pane.

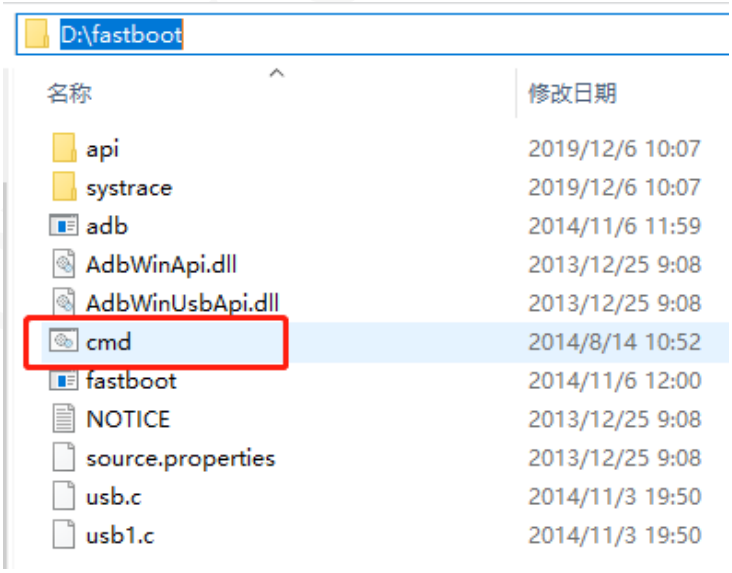
   is displayed.
3. In the **System Properties** dialog box, click the **Advance** tab and then the **Environment Variables** button.
4. In the **Environment Variables** dialog box, double-click the **Path** line in the **User variables for [user profile]** area.
5. Add the folder path of the **adb** script, and click **OK** to save the settings.
6. Open a CLI.
7. Input **adb** and press **Enter**.

If **Android Debug Bridge version** is displayed, the environment variable is configured successfully.



Do NOT save the **adb** script in C:\. Otherwise, the environment variable configured in the above step cannot be identified.

Step 2: Extract fastboot.rar.



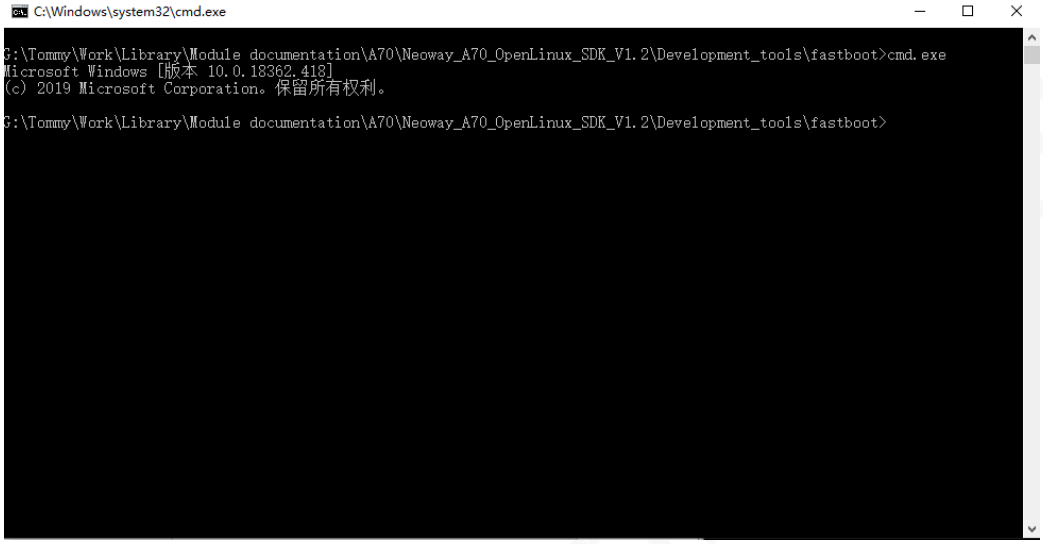
Step 3: Double-click **cmd.bat** in the folder to open CMD.

If **cmd.bat** cannot be found in the folder, follow steps below to create one.

1. Create **cmd.txt**.

2. Write **cmd.exe** into **cmd.txt**.
3. Save the **cmd.txt** file.
4. Modify **cmd.txt** to **cmd.bat**.

   In the CLI, the folder path of the **adb** script is indexed already and you can execute the ADB commands directly.



Step 4: Input ADB commands in CLI.

It is recommended to start the execution script by opening **cmd.bat** in the folder of the **adb** script.

6.2 Common ADB Commands

ADB Command	Function
adb devices	To check whether the ADB device is identified after the USB driver is loaded successfully.
adb version	To check the ADB version
adb push [*Local path*]+ [*file name*] [*Module kernel path*]+ [*file name*]	To add the local application into the module. This operation can be performed only after the application permission is modified by executing **adb shell chmod 755** [kernel path](or file name).
adb shell	Enter the kernel
adb reboot	Reboot
adb reboot edl	Enter the burning mode

adb reboot bootloader	Enter the fastboot mode
fastboot devices	Check the device in fastboot mode
fastboot flash [*image name*] [*image path*]	Flash one image (common images: boot, system, modem and etc)
fastboot reboot	The module enters working mode after rebooting.

7 Setting Up SDK Environments

Reference documents	*Neoway_OpenLinux_SDK_Developer_Guide*
Description	• Setting up the environments used to compile, develop and debug apps • Introducing the log printing API • Introducing APIs
Remarks	The SDK offers a compiling tool, environment configuration methods, APIs and the demo of relevant functions. You can use the APIs to modify **makefile** and use the **make** command to compile and generate an application.

7.1 Compilation Environment

- System requirements
  ubuntu 64-bit OS 12.04 or 14.04.
- Cross-compilation tool
  Neoway-arm-oe-linux.tar.gz. It is saved in **tool** directory of the SDK package.

Using Cross-Compilation Tool

The cross-comilation tool can be used in two ways.

- Initializing Environment Variable (Recommended)

Step 1: Extract Neoway-arm-oe-linux.tar.gz in Ubuntu OS.

tar -xzvf neoway-arm-oe-linux.tar.gz

Step 2: Navigate to the neoway-arm-oe-linux directory and execute source neoway-env-init.sh.

The environment variable is initialized successfully if **Neoway cross toolchain environment setup success!** is displayed.

Initialize the environment variable every time you start a terminal.

- Setting Absolute Path

Step 1: Execute **vi ~/.bashrc** to edit the **.bashrc** file in the home directory.

Step 2: Add the source /home/***/tool/neoway-arm-oe-linux/neoway-env-init.sh command at the end of the file.

neoway-env-init.sh is the system absolute path.

Step 3: Save the file.

You can use the cross-compilation tool directly in the terminal after setting its absolute path into **.bashrc**.

7.2 Development Environment

- System requirements

  Windows 7/8/10

- Programming editor &Code browser

  Source insight (or other tools as required)

7.3 Debugging Environment

- System requirements

  Windows 7/8/10

- Debugging tool

  Android Debug Bridge version 1.0.31
  Refer to Chapter 7.

8 Developing Applications

Reference documents	• APIs description: Neoway_OpenLinux_SDK_Developer_Guide • Log printing: Neoway_OpenLinux_SDK_Developer_Guide • Automatic startup: Neoway_OpenLinux_App_Automatic_Startup_Configuration_Guide
Description	• Setting up environments used to compile, develop, and debug apps • Introducing APIs • Introducing the log printing API
Remarks	You can develop your applications based on the SDK APIs provided by Neoway.

8.1 APIs Category

Category	Function
Device	To enable and invoke the hardware pins of peripherals including UART, I2C, SPI, GPIO, ADC, Antenna, Ethernet, Audio and etc.
Service	To enable the relevant network services including: • Data service • Voice service • SIM card service • SMS service • Location service • Wi-Fi Service • FOTA (deferential upgrade - Adups scheme) • Device Manager (device control, version number, working mode, etc.)
Common	To encrypt and save the key data: crypto and items
POSIX	Common protocol APIs: TCP, FTP, HTTP, TLS These APIs meet the Linux standard. For detail, refer to the demo provided by Neoway.
Other APIs	Common Linux system commands including file reading and writing and thread calling. These APIs are not described in this document.

8.2 Printing Log

To facilitate the debugging, Neoway provides a log printing API.

8.2.1 Head File

#include "nwy_common.h"

8.2.2 Library File

- API libraries

  The API to print logs is provided by **libnwy_common.so.0** or **libnwy_common.a**

- Makefile

  Add the library into **makefile**, for example, **USR_LIB_FILES =-L ../../libs -lnwy_common**.

- Log API

  The log system specifies the log format, the tag, the level and the mask through macros. The print functions are respectively **LOGV**, **LOGE**, **LOGI** and **LOGD**.

Log format	To print logs with file name, function name, and row numbers, define **LOG_FORMAT** to **Log_COMPLEX**. If the information is not required, define **LOG_FORMAT** to **LOG_SIMPLE**.
Tag	**LOG_MSG_TAG** can be defined as a character string.
Level	**LOG_MSG_LEVEL** can be defined to 7 levels, including **LOG_EMERG** and **LOG_DEBUG**. For details, see **nwy_common.h**.
Printing positions	**LOG_MSG_MASK** can be defined as the following values: • LOG_MASK_QXDM (QXDM, TBD) • LOG_MASK_ADB (logread buffer) • LOG_MASK_STD (console) For detail, see **nwy_common.h**. Multiple masks can be configured through the logic OR.

8.2.3 Capturing Log

Capturing method	Description
Output to STD	Print logs to **stderr** directly.
Output to logread buffer	Execute the logread command. • Capture all logs in buffer: **adb shell logread**. • Capture logs dynamically: **adb shell logread -f**.
Output to QXDM	TBD

9 Source Code Development

Reference document	*Neoway_OpenLinux_Source_Code_Developer_Guide*
Description	• How to compile the source code • How to customize the file system • How to customize partitions • How to generate the OTA package with the Adups solution (via **./generate_ota.sh**)
Source code functions	• Configure drivers by modifying the device tree. • Customize file system • Modify partitions (via **/partition_nand.xml**) • Generate the OTA package with the Adups solution.
Reference document	Source code developer guide
Source code package	[Version name].des3 (fill the version name)
Compression format	des3
Package size	About 1G
Extracting command	dd if=[Package name].des3 | openssl des3 -d -k [Extracting password]| tar zxf -
Space occupation for compilation	About 7.5G (reserve 10G)
Compilation script	build.sh
After compilation	Module firmware

Configuring the Kernel Version

Follow steps below to control the kernel version:

1. Modify **CONFIG_NWY_SYSCON_KERNEL_VERSION=** [version name] in **mdm9607_N720F_deconfig** and **mdm9607-perf_N720F_deconfig**.

   File path: /kernel/arch/arm/configs

2. Compile the kernel separately.

   ./build.sh perf linux-quic

3. Issue the AT command to query the kernel version.

```
AT+NAPPCHECK?
+NAPPCHECK:<app_ver>,<app_compilatime>
OK
```

<app_ver>: kernel version of the customer's firmware.

<App_compilation>: kernel compilation time, the system clock of the compiler.

10 Automatic Startup of the Application

Reference document	*Neoway_OpenLinux_App_Automatic_Startup_Configuration_Guide*
Description	The automatic startup of an application can be configured in two ways: • Deploy manually (used in the debugging phase) • Modify source code

11 Firmware Upgrade

Reference document	*Neoway_OpenLinux_Firmware_Upgrade_Guide*
Description	The process of firmware upgrade.

12 Production Related

This chapter describes the precautions and tools for SMT reflow soldering.

12.1 Humidity Sensitive Device Control

Reference document	*Neoway_Description_of_Humidity_Sensitive_Device_Control*
Description	The humidity control and process requirements for the Neoway modules: • Storage conditions • Inspection before production • Baking conditions • Workshop life • Maintenance

12.2 Reflow Soldering Guidelines for Surface-Mount Modules

Reference document	*Neoway_Reflow_Soldering_Guidelines_for_Surface-Mount_Modules*
Description	The recommendations for reflow process and process requirements for Neoway modules. • Storage • Precautions for SMT reflow soldering • Recommendations for Stencil Design • Recommendations for Reflow Process • Reference Oven Temperature Curve

12.3 Upgrade Tool in Factory

Reference document	*Neoway_OpenLinux_Firmware_Upgrade_Guide*
Description	The process of firmware upgrade.

A Business Model

The OpenLinux module requires further development after your obtain it. To ensure that the R&D and production of your products are processed smoothly, manage the firmware versions in compliance with the following guidelines:

- Control the versions of your firmware and develop testing schemes for modules in mass production.
- Mark your firmware versions clearly and deploy a version inspection mechanism to facilitate the remote upgrade.

  Neoway modules employ a mechanism to protect the system from faults. The OpenLinux module restores to its production version if it encounters an abnormality.

Production version refers to the module firmware that is flashed into the module through the production tool.

The firmware upgraded via FOTA is not a production-version firmware.

- Inform Neoway about the requested delivery date at least two weeks before the shipment so that the production can be finished on time.

  The modules will be manufactured with the version that you have confirmed for production last time before you send any notice on version changes to Neoway.
  The delivery date is subject to the date that both sides agree.

B Cooperation Plan

Table 12-1 Cooperation plan

Phase	Input	Output	Remarks
Preliminarily understand	• Specifications • HW design guide • Requirement evaluation form	Requirement evaluation form (filled)	• Offer the evaluation form to Neoway for further communication. • Neoway generates a pin definition form (including the pin multiplexing recommendation) according to your evaluation form.
Confirm requirements	• Pin definition form (including the pin multiplexing recommendation) • Requirement matrix • Product Documentation • Debug software on the EVB	• Pin multiplexing form (confirmed) • Requirement matrix agreed by both sides	• The pin multiplexing form is confirmed by both sides. • The requirement matrix can be prepared by Neoway and confirmed by you.
HW design	• HW design guide • Module packaging/ PCB of the EVB	Schematic diagram and PCB	• It is recommended to send your schematic diagram and PCB to Neoway for technical review. • It is recommended to reserve the USB test point to facilitate log capturing, firmware burning and firmware upgrade.
HW design review	Review report	Schematic diagram and PCB (confirmed)	In this phase, review is conducted multiple times to avoid possible issues that occurs during later development process.
Develop drivers	• Source codes	Basic version (excluding	• This phase is required if any driver is modified.

	• Related documents on driver development • Tool and document for QPST upgrade	the applications, but including necessary drivers)	• Neoway offers support in this phase according to the cooperation plan.
Develop applications	• SDKs • Document for SDK	Application programs	• Develop applications based on the SDK APIs. • To test the customer application, you can manually push the applications to the module.
Debug applications	• QXDM • Log capturing guide	Application programs (confirmed)	• Offer the **QXDM** tool to capture the Qualcomm log if necessary. • Add the log mechanism to the application layer to facilitate issues locating. (Recommended)
Customize the file system	• Source codes • Source code developer guide • Document for QPST upgrade • Version control scheme	Final version of firmware (including drivers and applications).	• The product functions can be implemented after the firmware is burned into the module. • Embed the version number mechanism into the module firmware to facilitate the version control and firmware tests during production.
Retrofit the production lines	• Module (with the initial sampling firmware) • Tools and documents for production • Neoway quality agreement-complementary agreement	Retrofit the production lines and then perform the relevant tests for production.	• To embed your firmware into the modules in our module production, sign the *Neoway Quality Agreement-Complementary Agreement*. • Confirm the firmware version with Neoway at least two weeks before shipment; note the firmware version information in the purchase order to facilitate Neoway's production.
Mass production test	Log capturing document		• Capture the application log first for troubleshooting when an issue occurs in mass production. • Contact Neoway FAE for further analysis if you cannot troubleshoot the issue according to the application log.