neoway

Get connected Get smart

N725 OpenLinux

API Notes

Issue 1.2 Date 2023-03-28

Copyright © Neoway Technology Co., Ltd. 2023. All rights reserved.

No part of this document may be reproduced or transmitted in any form or by any means without prior written consent of Neoway Technology Co., Ltd.

neoway有方 is the trademark of Neoway Technology Co., Ltd.

All other trademarks and trade names mentioned in this document are the property of their respective holders.

Notice

This document provides guide for users to use N725 OpenLinux.

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

THIS DOCUMENT 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 with 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

Neoway Confidential

Contents

Neoway Confidential

Copyright © Neoway Technology Co., Ltd.

About This Document

Scope

This document is applicable to N725 OpenLinux.

Audience

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

Change History

Issue	Date	Change	Changed By
1.0	2022-08	Initial draft	Wen Jingshun
1.1	2022-05	Updated 1.4 SDK Directory Structure	Chen Zhirong
1.2	2023-01	Added the OS information	Chen Zhirong

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

Neoway Confidential

Copyright © Neoway Technology Co., Ltd.

1 Introduction to N725 OpenLinux

1.1 Overview

N725 OpenLinux provides the following resources:

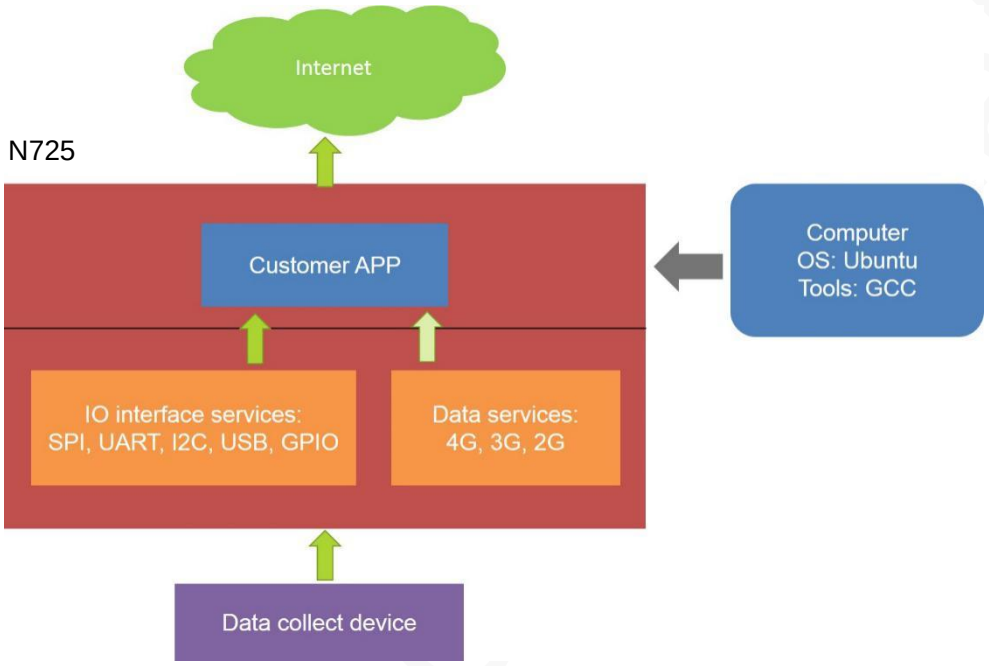
- Supports programming basic on C
- Supports hardware interfaces: ADC, UART, SPI, I2C, and so on.
- Supports ADB debugging

Neoway Confidential

1.2 Software Architecture

The following figure shows the logical components for the module

Figure 1-1 Software architecture



- Operating system

  OpenWRT 5.4.195.

- Programming language

  Provides C API functions that run on Linux and header files which contain the structure definitions.

- Compilation Environment

  It allows developers to build applications on a computer running Ubuntu OS, compile applications using the GCC compiler, and generate executable files.

- Operating method

  Push the executable file to the /usr/data directory through the ADB command. After modifying the file permission, you can manually operate the basic functional module.

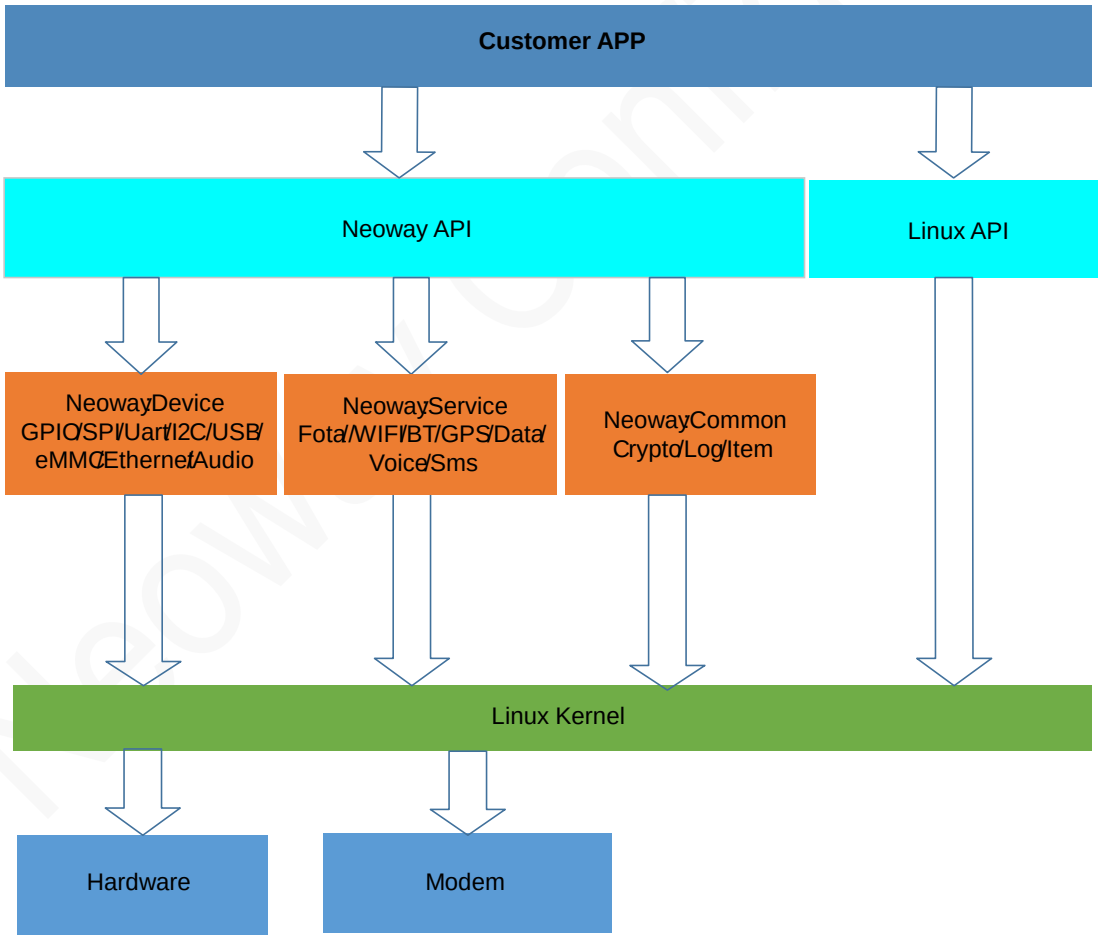
The following items detail the function of each API.

Copyright © Neoway Technology Co., Ltd.

- Data call
- Network mode selection, network registration, network status querying
- SIM card services
- FOTA
- Sleep and wakeup
- I/O interfaces including GPIO, ADC, SPI, UART, I2C, etc.
- Standard Linux components and interfaces

1.3 System Framework

Figure 1-2 System framework



Neoway Confidential

User applications can invoke two kinds of APIs: Neway APIs and Linux APIs

Table 1-1 Neoway API

API	Function
APIs that access device drivers	Applications can invoke these APIs to access peripheral devices, such as GPIO and ADC. They are implemented by referring to **libnwy_device.a**.
APIs that access Neoway services	Dial-up networking, GPS, and FOTA upgrade. They are implemented by referring to **libnwy_service.a**.
Common APIs	These APIs provide common functions such as printing log and key data storage. They are implemented by referring to **libnwy_common.a**.

1.4 SDK Directory Structure

This section explains the key files of N725 SDK, the zip file and the unzipped folder provided by Neoway are shown below. The file name is followed by the version number and the release date. Before development, ensure whether you get the latest SDK.

```
drwxrwxr-x 3 zhirong zhirong      4096 3月  17 16:30 N725_OPEN_WRT_SDK_V1.01_2022-02-25
-rwxr----- 1 zhirong zhirong 195964312 3月  16 10:58 N725_OPEN_WRT_SDK_V1.01_2022-02-25.tar.gz
```

The following details the files of the SDK file.

```
drwxrwxr-x 11 zhirong zhirong 4096 2月  25 16:05 example
drwxrwxr-x  2 zhirong zhirong 4096 2月  25 16:05 include
drwxrwxr-x  2 zhirong zhirong 4096 2月  25 16:05 libs
-rwxrwxr-x  1 zhirong zhirong  622 2月  25 16:05 Makefile
-rwxrwxr-x  1 zhirong zhirong  355 2月  25 16:05 readme
drwxrwxr-x  2 zhirong zhirong 4096 2月  25 16:05 tool
```

- example: contains examples of each functionality module
- include: contains header files of each functionality module
- libs: contains library files provided by SDK
- tool: contains the cross compiler

example

Contains folders with sample programs.

Neoway Confidential

```
drwxrwxr-x 2 zhirong zhirong 4096 2月  25 16:05 data
drwxrwxr-x 2 zhirong zhirong 4096 2月  25 16:05 dm
drwxrwxr-x 3 zhirong zhirong 4096 2月  25 16:05 ftp
drwxrwxr-x 5 zhirong zhirong 4096 2月  25 16:05 http
drwxrwxr-x 2 zhirong zhirong 4096 2月  25 16:05 loc
-rwxrwxr-x 1 zhirong zhirong  408 2月  25 16:05 makefile
drwxrwxr-x 2 zhirong zhirong 4096 2月  25 16:05 network
drwxrwxr-x 2 zhirong zhirong 4096 2月  25 16:05 sim
drwxrwxr-x 2 zhirong zhirong 4096 2月  25 16:05 sms
drwxrwxr-x 2 zhirong zhirong 4096 2月  25 16:05 tcp
```

- data: data dialing
- dm: module information and mode setting
- ftp: FTP samples
- http: HTTP samples
- network: network registration
- sim: SIM card operating

Taking data dialing sample as an example, the program contains the following files:

```
-rwxrwxr-x 1 zhirong zhirong    84 2月  25 16:05 install.bat
-rwxrwxr-x 1 zhirong zhirong  1662 2月  25 16:05 makefile
-rwxrwxr-x 1 zhirong zhirong 40823 2月  25 16:05 nwy_data_test
-rwxrwxr-x 1 zhirong zhirong  7609 2月  25 16:05 nwy_data_test.c
```

- install.bat: adb push and file permissions modification reference script
- makefile: a tool to simplify or to organize code for compilation.
- nwy_data_test: executable file generated after compilation
- nwy_data_test.c: source code

tool

Contains a cross-compilation tool, and an environment variable setting script as shown in the followings:

```
-rwxrwxr-x 1 zhirong zhirong       792 2月  25 16:05 neoway-env-init.sh
-rw-rw-r-- 1 zhirong zhirong 193781269 2月  25 16:05 toolchain-arm_cortex-a7+neon-vfpv4_gcc-4.8-linaro_uClibc-1.0.25_eabi.tar.gz
```

Neoway Confidential

- toolchain-arm_cortex-a7+neon-vfpv4_gcc-4.8-linaro_uClibc-1.0.25_eabi.tar: cross-compilation tool
- neoway-env-init.sh: environment variable setting script

Neoway Confidential

2 How to Develop the Program

This chapter details the development processes.

2.1 Environment Configuration

2.1.1 Configuring Development Environments

- Operating System

  Ubuntu 64-bit OS, version 14.04

- Cross-compilation tool

  toolchain-arm_cortex-a7+neon-vfpv4_gcc-4.8-linaro_uClibc-1.0.25_eabi.tar.gz (can be found in the **/tool/** directory of the SDK package.)

- Operating process:

1. Decompress the following file in an Ubuntu PC.

```
tool$tar -xzvf toolchain-arm_cortex-a7+neon-vfpv4_gcc-4.8-linaro_uClibc-1.0.25_eabi.tar.gz
```

2. You can also execute the following scripts to decompress the cross-compilation tool and perform the initialization process.

```
tool$source neoway-env-init.sh
Neoway cross toolchain environment setup success!
```

3. When “Neoway cross toolchain environment setup success!” is displayed, the initiation process is performed successfully.

Do Not repeatedly execute **source neoway-env-init.sh** to perform the initialization process in the same terminal.

Every time the terminal is started, perform the initialization process first.

Neoway Confidential

2.1.2 Configuring Debug Environment

- Operating System

  Windows 7/10

- Debug tool

  ADB -- Android Debug Bridge version 1.0.31 (using Windows OS is recommended). The tool can be found in /tool/fastboot.rar of the SDK package.

- Install USB drivers for the Windows-OS computer.

2.1 Debugging Method

You can debug applications and print log messages through the log system provided in OpenLinux SDK. The log API can be found in **nwy_common.h** and is provided by **libnwy_common.so.0** or **libnwy_common.a**.

Makefile

Add the library file name into **makefile**, e.g. USR_LIB_FILES =-L ../../libs -lnwy_common.

Log APIs

The log system specifies log format, tag, level, and location through macros. The print function is **LOGV**, **LOGE**, **LOGI**, and **LOGD**.

Format	Log printing is controlled by LOG_FORMAT. • To print log messages 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**.
Location	**LOG_MSG_MASK** can be defined to three values to specify the destination locations of log print: • **LOG_MASK_QXDM**: QXDM (TBD) • **LOG_MASK_ADB**: logread buffer

Neoway Confidential

- **LOG_MASK_STD**: Console

For details, see **nwy_common.h**.
Multiple locations can be configured through the logic OR.

2.1.2 Log Capturing Method

Method	Description
Output to a console	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

2.1.3 Reference Case

1. Preparation

```
#include "nwy_common.h"
#define LOG_FORMAT        LOG_SIMPLE     //Output logs without file name, function name, and row number
#define LOG_MSG_TAG     "CRYPTO_TEST"    //Tag
#define LOG_MSG_LEVEL    LOG_DEBUG       //Level
#define LOG_MSG_MASK    LOG_MASK_ADB    //Output to both logread.
```

2. Invoke LOGI(“Welcom to Nwy Crypto test.\n”);

3. Output result

```
[CRYPTO_TEST]: Welcom to Nwy Crypto test.
```

2.2 Compiling and Running

2.2.1 Compiling

- If developers have multiple .c source files or reference multiple library files, modify **makefile** in the corresponding directory.

Neoway Confidential

- **LOCAL_SRC_FILES** is to save source files, **INC_DIR** is to save the header file directory, and **USR_LIB** is to save library files.
- If the compilation fails, execute **make checkenv** in the SDK directory. Check if the value of **COMPILE.c =** is **arm-oe-linux-gnueabi-gcc**.

1. Start a terminal and enter the **SDK/tool/neoway-arm-oe-linux/** directory in the terminal.
2. Execute **source neoway-env-init.sh** to set the environment variables.
3. Compile the library files and the sample code by executing **make** in the SDK directory.

For example, compile an ADC program.

4. Enter the **example/adc** directory in the terminal
5. Execute the **make** command to compile the ADC program.

   An executable file **adc_test** is generated in the directory after a successful compilation.

2.2.2 Running

1. Push the executable file **nwy_adc_test** generated by **make** to **/data** of the module through the ADB shell command.

```
D:\Source\adb_fastboot>adb push C:\Users\20140701058\Desktop\nwy_adc_test /data
363 KB/s (7446 bytes in 0.020s)
```

2. Modify the file permission.

```
D:\Source\adb_fastboot>adb shell chmod 755 /data/nwy_adc_test
```

3. Execute the program in the shell.

```
D:\Source\adb_fastboot>adb shell
/ # /data/nwy_adc_test
/data/nwy_adc_test
**** Adc  Sample ****
```

To facilitate debugging, add the following bat script to the path of the executable program. The previous steps can be operated by executing this script.

Neoway Confidential

install.bat

```
1  adb push adb push C:\Users\20140701058\Desktop\nwy_adc_test /data
2  adb shell chmod 755 /data/nwy_adc_test
3  adb shell /data/bin/nwy_adc_test
```

Neoway Confidential

3 APIs

3.1 Device

This chapter introduces the hardware I/O related APIs. The APIs listed in this chapter are provided by **libnwy_device.so** or **libnwy_device.a**.

3.1.1 UART

UART API definitions can be found in **nwy_uart.h**.

The module can receive/send data from/to PC through the UART port by calling this function.

For the sample code, see **example/uart/nwy_uart_test.c**.

Executing the function without any parameter can print help information.

nwy_uart_open

Function	int nwy_uart_open(const char* port, unsigned int baudrate, nwy_flowctrl_t flowctrl)
Description	Open the UART device and perform the initialization configuration.
Parameter	port: UART device name baudrate: UART baud rate flowctrl: Flow control
Return	Successful: UART device handle Failed: a value < 0; for details see **nwy_error.h**.

nwy_uart_read

Function	int nwy_uart_read (int fd,unsigned char* buf, unsigned int buf_len)
Description	To read UART data
Parameter	fd: UART device handle buf: Points to a pointer that stores read data buf_len: Buffer size

Neoway Confidential

Return	Successful: size of read data Failed: a value < 0; for details see **nwy_error.h**.

nwy_uart_write

Function	int nwy_uart_write(int fd, const unsigned char* buf, unsigned int buf_len)
Description	To write UART data
Parameter	fd: UART device handle buf: Points to a pointer that writes data buf_len: Size of written data
Return	Successful: Size of written data Failed: a value < 0; for details see **nwy_error.h**.

nwy_uart_setdcb

Function	int nwy_uart_setdcb(int fd, struct termios *term);
Description	To set parameters related to the UART term structure
Parameter	fd: UART device handle term: Points to pointer of the term structure
Return	Successful: **NWY_RES_OK** Failed: a value < 0; for details see **nwy_error.h**.

nwy_uart_getdcbconfig

Function	int nwy_uart_getdcbconfig(int fd, struct termios *term);
Description	To obtain UART term structure related parameters
Parameter	fd: UART device handle term: Points to pointer of the term structure
Return	Successful: **NWY_RES_OK** Failed: a value < 0; for details see **nwy_error.h**.

Neoway Confidential

nwy_uart_ioctl

Function	int nwy_uart_ioctl(int fd, unsigned int cmd, void* pvalue)
Description	UART ioctl
Parameter	fd: UART device handle cmd: Control command pvalue: Supplementary parameter
Return	Successful: **NWY_RES_OK** Failed: a value < 0; for details see **nwy_error.h**.

nwy_uart_set_databit

Function	int nwy_uart_set_databit(unsigned int databit)
Description	To set the serial port data bit
Parameter	databit: Data bit
Return	Successful: **NWY_RES_OK** Failed: a value < 0; for details see **nwy_error.h**.

nwy_uart_set_stopbit

Function	int nwy_uart_set_stopbit(unsigned int stopbit)
Description	To set the serial port stop bit
Parameter	stopbit: Stop bit
Return	Successful: **NWY_RES_OK** Failed: a value < 0; for details see **nwy_error.h**.

nwy_uart_close

Function	int nwy_uart_close(int fd)
Description	To close the UART device
Parameter	fd: UART device handle
Return	Successful: **NWY_RES_OK** Failed: a value < 0; for details see **nwy_error.h**.

Neoway Confidential

3.1.2 I2C

The I2C API definitions can be found in **nwy_i2c.h**. The API can be used to read or write data via the I2C device. There is a corresponding device mounted on the I2C bus, reading the value of one of the registers of the device, and the value of the registers can be changed by writing. For the sample code, see **example/i2c/nwy_i2c_test.c**.

nwy_i2c_init

Function	int nwy_i2c_init(const char* i2cDev, unsigned char slaveAddr)
Description	To open the I2C device and write the slave address.
Parameter	i2cDev: I2C device name slaveAddr: slave address
Return	Successful: I2C device handle Failed: a value < 0; for details see **nwy_error.h**.

nwy_i2c_read

Function	int nwy_i2c_read(int fd, unsigned char slaveAddr, unsigned char ofstAddr, unsigned char* ptrBuff, unsigned short length)
Description	To read data in specified length from the destination address via the specified I2C slave device.
Parameter	fd: I2C device handle slaveAddr: slave address ofstAddr: Register address ptrBuff: Pointer points to a buffer that stores data length: Length of the read data
Return	Successful: 0: no data a value >0: there is data Failed: a value < 0; for details see **nwy_error.h**.

nwy_i2c_write

Function	int nwy_i2c_write(int fd, unsigned char slaveAddr, unsigned char ofstAddr, unsigned

Neoway Confidential

	char* ptrData, unsigned short length)
Description	To write data in specified length to the destination address via the specified I2C slave device.
Parameter	fd: I2C device handle slaveAddr: slave address ofstAddr: Register address ptrData: Pointer points to a buffer that is written data length: Length of the written data
Return	Successful: **NWY_RES_OK** Failed: a value < 0; for details see **nwy_error.h**.

3.1.3 SPI

SPI API definitions can be found in **nwy_spi.h**. For the sample codes, see **example/spi/nwy_spi_test.c**. To receive data, you should short connect the MOSI and MISO pins of the SPI interface, and then execute the application program.

nwy_spi_init

Function	int nwy_spi_init(char* spidev, uint8_t mode, uint32_t speed, uint8_t bits)
Description	To initialize the SPI devive.
Parameter	Spidev: SPI device name mode: SPI mode, specified by the CPHA and CPOL values (both values can be 0 or 1). They can form four combinations that correspond to four SPI modes. SPI_MODE_0 CPOL=0, CPHA=0 SPI_MODE_1 CPOL=0, CPHA=1 SPI_MODE_2 CPOL=1, CPHA=0 SPI_MODE_3 CPOL=1, CPHA=1 speed: the max clock frequency the SPI interface supports, 50 MHz. bits: The length of each communication word
Return	Successful: SPI device handle Failed: a value < 0; for details see **nwy_error.h**.

Neoway Confidential

nwy_spi_transfer

Function	int nwy_spi_transfer(int fd, uint8_t* tx, uint32_t tx_size, uint8_t* rx,uint32_t speed, uint8_t bits, uint16_t delay)
Description	SPI data transmission
Parameter	fd: SPI device handle tx: Points to a pointer that sends data tx_size: Size of sent data rx: Points to a pointer that receives data speed: clock frequency the SPI interface supports. bits: Data bits delay: Delay between the two **spi_ioc_transfer**
Return	Successful: NWY_RES_OK Failed: a value < 0; for details see **nwy_error.h**.

nwy_spi_deinit

Function	void nwy_spi_deinit(int fd)
Description	To close the SPI device
Parameter	fd: SPI device handle
Return	N/A.

3.1.4 GPIO

GPIO API definitions can be found in **nwy_gpio.h**.

For the sample code, see **example/gpio/nwy_gpio_test.c**.

- Sets the GPIO pin to input

  The pin can read the input level.
- Sets the GPIO pin to output in high or low level.
- Sets the the GPIO pin to interrupt and configures the interrupt mode.

  When the pin is set to interrupt that is used for wake-up, the module wakes up from sleep mode every time the interrupt is triggered. 0 indicates trigger by falling edge. 1 indicates trigger by rising edge.

Neoway Confidential

There are two interrupt detection methods: automatic detection by kernel or execution by user.

- automatic detection is enabled

  Falling edge triggers sleep and rising edge triggers wakeup.

- automatic detection is disabled

  Falling edge triggers the report of sleep mode and rising edge triggers the report of wakeup mode. You can configure the detection mode as required.

For details refer to **example/gpio/nwy_gpio_test.c**.

nwy_gpio_set_val

Function	int nwy_gpio_set_val(int gpio_n, int value)
Description	To set the GPIO output level.
Parameter	gpio_n: GPIO number, see the GPIO definition. value: Output level: 0: low level 1: high level
Return	Successful: the set output level Failed: a value < 0; for details see **nwy_error.h**.

nwy_gpio_get_val

Function	int nwy_gpio_get_val(int gpio_n)
Description	To obtain the GPIO input/output level.
Parameter	gpio_n: GPIO number, see the GPIO definition.
Return	Successful: the obtained input/output level Failed: a value < 0; for details see **nwy_error.h**.

nwy_gpio_set_dir

Function	int nwy_gpio_set_dir(int gpio_n, int dir)
Description	To set the GPIO input/output direction
Parameter	gpio_n: GPIO number, see the GPIO definition. dir: Input/output direction

Neoway Confidential

	0: input 1: output
Return	Successful: the set input/output direction Failed: a value < 0; for details see **nwy_error.h**.

nwy_gpio_get_dir

Function	int nwy_gpio_get_dir(int gpio_n)
Description	To obtain the GPIO input/output direction
Parameter	gpio_n: GPIO number, see the GPIO definition.
Return	Successful: the obtained input/output direction Failed: a value < 0; for details see **nwy_error.h**.

nwy_gpio_irq_config

Function	int nwy_gpio_irq_config(int gpio_n, int irq_mode, void (*irq_callback)(int))
Description	To configure the GPIO configuration
Parameter	gpio_n: GPIO number, see the GPIO definition. irq_mode: interrupt trigger method 2: rising edge triggered 3: falling edge triggered 4: edge-triggered 5: high level triggered 6: low level triggered 7: sleep and wake-up mode Select the 2/3/4/5/6 interface Irq_callback: callback function. The callback function is executed when there is a corresponding interrupt (after the interrupt is configured).
Return	Successful: **NWY_RES_OK** Failed: a value < 0; for details see **nwy_error.h**.

nwy_gpio_pm_config

Function	int nwy_gpio_pm_config(int gpio_n, int irq_mode, int sleep_mode, int sleep_en, void (*irq_callback)(int))

Neoway Confidential

Description	To configure the GPIO sleep and wake-up function.
Parameter	gpio_n: GPIO number, see the GPIO definition. irq_mode: interrupt trigger method 2: rising edge triggered 3: falling edge triggered 5: high level triggered 6: low level triggered 7: wakeup from sleep mode Sleep_mode: set the trigger method to wake up the module from sleep mode. 0: falling edge triggered 1: rising edge triggered sleep_en: whether to enable the kernel detects automatically to wake up from sleep mode 1: enable others: disable Irq_callback: callback function. The callback function is executed when there is a corresponding interrupt (after the interrupt is configured).
Return	Successful: **NWY_RES_OK** Failed: a value < 0; for details see **nwy_error.h**.

nwy_gpio_pm_state_get

Function	int nwy_gpio_pm_state_get(void)
Description	To obtain the reported sleep/wakeup status
Parameter	N/A.
Return	Successful: 0: wake-up mode 1: sleep mode Failed: a value < 0; for details see **nwy_error.h**.

nwy_gpio_irq_close

Function	int nwy_gpio_irq_close(int gpio_n)
Description	To disable the GPIO interrupt and release the GPIO.
Parameter	gpio_n: Used to set the GPIO number, see the GPIO definition.
Return	Successful: **NWY_RES_OK**

Neoway Confidential

	Failed: a value < 0; for details see **nwy_error.h**.

3.1.5 ADC

ADC API definitions can be found in **nwy_adc.h**. For the sample codes, see **example/adc/nwy_adc_test.c**. It is used to obtain the voltage collected by a specified ADC channel. Before obtain the voltage, ensure that each ADC channel is connected.

The detectable voltage ranges from 0 to 1.2 V.

nwy_adc_get

Function	int nwy_adc_get(int channel)
Description	To obtain voltage using a specified ADC channel.
Parameter	channel: Channel number. It can be 1, 2, 3, 4, 5, 6, or 7. 1, 2, 3, 4, and 5 are the numbers of channels collecting the module's internal temperatures. 6 and 7 represent ADC_IN2 and ADC_IN3.
Return	Successful: collected voltage value Failed: a value < 0; for details see nwy_error.h.

3.2 Service

The APIs listed in this chapter are provided by **libnwy_service.so** or **libnwy_service.a**.

3.2.1 Data

Data API definitions can be found in **nwy_data.h**.

They are mainly used to

- set up or close a data dial-up connection,
- set dial-up parameters,
- and obtain status and information of the connection.

Neoway Confidential

Programming Reference

- Invoke **nwy_data_get_srv_handle()** to obtain a resource handle and register with the callback function to obtain the dial-up status.

  The module supports concurrent setup of multiple dial-up connections (at most 6). These dial-up connections can register with their own callback functions or one callback function. If one callback function is registered, these dial-up connections are identified by the resource handles in the callback function.

- Invoke **nwy_data_set_ip_version()** to set the IP protocol version and **nwy_data_set_profile_id()** to set profile ID. To modify profile, invoke **nwy_data_set_profile()**.
- Invoke **nwy_data_start_call()** to set up a dial-up connection and monitor the dialing status using a callback function.
- If a dial-up connection is set up successfully, invoke APIs to obtain the NIC device name and address information for router and NAT settings.

For details, see **example/data/nwy_data_test.c**.

nwy_data_get_srv_handle

Function	int nwy_data_get_srv_handle(nwy_data_cb_func cb_func)
Description	To obtain the resource handle of data dial-up connection and set callback function The handle is used to operate the connection. The callback function is used to notify the connection status.
Parameter	cb_func: The registered callback function which is used to notify the connection status.
Return	Successful: a value >0 (the obtained resource handle) Failed: a value < 0; for details see nwy_error.h.

nwy_data_release_srv_handle

Function	void nwy_data_release_srv_handle(int hndl)
Description	To release resources specified for data dial-up connection and deregister with callback function If the dial-up connection is set up, disconnect it.
Parameter	hndl: Resource handle of the dial-up connection, obtained from **nwy_data_get_srv_handle()**.
Return	N/A.

Neoway Confidential

nwy_data_start_call

Function	int nwy_data_start_call(int hndl, nwy_data_start_call_v02_t *param)
Description	To start dial-up for data connection The connection result is obtained through the callback function that is registered by **nwy_data_get_srv_handle()**.
Parameter	hndl: The specified resource handle of the dial-up connection, obtained from **nwy_data_get_srv_handle()**. para: Dial-up parameters, including profile ID, whether to perform auto-redialing, and the corresponding configurations.
Return	Successful: **NWY_RES_OK** Failed: a value < 0; for details see **nwy_error.h**.

nwy_data_stop_call

Function	int nwy_data_stop_call(int hndl)
Description	To close the specified dial-up connection. The connection result is obtained through the callback function that is registered by **nwy_data_get_srv_handle()**.
Parameter	hnd: The specified resource handle of the dial-up connection, obtained from **nwy_data_get_srv_handle()**.
Return	Successful: **NWY_RES_OK** Failed: a value < 0; for details see **nwy_error.h**.

nwy_data_get_ip_addr

Function	int nwy_data_get_ip_addr(int hndl, nwy_data_addr_t_info * info_ptr, int *len)
Description	To obtain the address information of the specified dialing. This API can be invoked only after the dial-up connection is set up successfully.
Parameter	hndl: The specified resource handle of the dial-up connection, obtained from **nwy_data_get_srv_handle()**. info_ptr: A structure or data set of structures used to return address information. If multiple addresses might be returned, reserve a big enough data set of structures. Each piece of address information consists of IP address and subnet mask allocated by the operator, gateway address and mask, primary DNS address and secondary DNS address. See the structure definition in **nwy_data_addr_t_info**. len: Number of addresses obtained

Neoway Confidential

Return	Successful: **NWY_RES_OK** Failed: a value < 0; for details see **nwy_error.h**.

nwy_data_set_profile

Function	int nwy_data_set_profile(int profile_idx, nwy_data_profile_type_t profile_type, nwy_data_profile_info_t * profile_info);
Description	To modify the parameters of the specified profile (with a specified ID and type).
Parameter	profile_idx: Used to specify the ID of the profile to be modified profile_type: Used to specify the type of the profile to be modified. profile_info: Profile parameters, including PDP type, APN, authentication method, user name, and password. See **nwy_data_profile_info_t**.
Return	Successful: **NWY_RES_OK** Failed: a value < 0; for details see **nwy_error.h**.

nwy_data_get_profile

Function	int nwy_data_get_profile(int profile_idx, nwy_data_profile_type_t profile_type, nwy_data_profile_info_t * profile_info);
Description	To obtain the parameters of the specified profile (with a specified ID and type).
Parameter	profile_idx: profile ID to be obtained. profile_type: Profile type to be obtained. profile_info: The returned profile information
Return	Successful: **NWY_RES_OK** Failed: a value < 0; for details see **nwy_error.h**.

Callback Function MSG Type and Structure

ind_state	ind_struct
NWY_DATA_CALL_DISCONNECTED	Reserved
NWY_DATA_CALL_CONNECTED	Reserved

Neoway Confidential

3.2.2 SMS

SMS API definitions can be found in **nwy_sms.h**. The APIs are used to send/receive SMS messages and set the storage location.

Programming Reference

- Invoke nwy_set_report_option to set the SMS report mode.

  The following details the <mt> definitions: For details, see the parameter description of AT+CNMI. For the <mt> parameter:

  1: SMS message is stored rather than directly displayed.
  2: SMS messages are directly displayed rather than stored.

  - If <mt> is set to 1 (only the message index ID will be returned), invoking **nwy_sms_recv_message** can return only the message index ID. To obtain the message content after the operation, you can invoke **nwy_sms_read_message** according to the obtained index ID.
  - If <mt> is set to 2, invoking **nwy_sms_recv_message** can return content except for the message index ID.

- Use the **nwy_sms_unsol_sms_event** function since the SMS function will runloop internally and wait for messages all the time; after **nwy_sms_unsol_sms_event** is used, you can establish a new thread to call the SMS function.

  For the sample examples, see **example/sim/nwy_sms_test.c**.

nwy_init_sms_option

Function	int nwy_init_sms_option(void)
Description	To initialize the SMS parameters
Parameter	N/A.
Return	Successful: 0 Failed: a value < 0; for details see **nwy_error.h**.

nwy_sms_set_storage

Function	int nwy_sms_set_storage(nwy_sms_storage_type_e sms_storage)

Neoway Confidential

Description	To set the SMS message location.
Parameter	nwy_sms_storage_type_e sms_storage 1: stored into the module 2: stored into the SIM card
Return	Successful: 0 Failed: a value < 0; for **details see nwy_error.h**.

nwy_sms_get_storage

Function	nwy_sms_storage_type_e nwy_sms_get_storage(void)
Description	To obtain the SMS message location.
Parameter	N/A.
Return	**nwy_sms_storage_type_e** indicates the message storage location 1: stored into the module 2: stored into the SIM card

nwy_set_report_option

Function	int nwy_set_report_option(uint8_t mode, uint8_t mt, uint8_t bm, uint8_t ds, uint8_t bfr)
Description	To set the SMS report mode
Parameter	mode: mode of notifying the SMS message after it is received. mt: specifies the rules for managing the received SMS according the message's Data Coding Scheme (DCS). bm: specifies the rules for managing the received Cell Broadcast messages (CBM). ds: specifies the rules for managing the Status Report messages (SMS-STATUS-REPORT). bfr: controls the buffering of URCs: Refers to 3GPP 27.005
Return	Successful: 0 Failed: a value < 0; for details see **nwy_error.h**.

nwy_sms_set_sca

Function	int nwy_sms_set_sca(char *sca)
Description	To set the SMSC number.

Neoway Confidential

Parameter	sca: the set SMSC number
Return	Successful: 0 Failed: a value < 0; for details see nwy_error.h.

nwy_sms_get_sca

Function	int nwy_sms_set_sca(char *sca)
Description	To obtain the SMSC number.
Parameter	sca: the obtained SMSC number.
Return	Successful: 0 Failed: a value < 0; for details see **nwy_error.h**.

nwy_sms_send_message

Function	int nwy_sms_send_message(nwy_sms_info_type_t *p_sms_data)
Description	To send SMS messages
Parameter	sca: the obtained SMSC number. phone_num: destination number msg_context_len: messgae length msg_contxet: message content msg_format: message coding format, 0:GSM7 1:8BIT 2:UNICODE For details, see **nwy_sms.h**.
Return	Successful: 0 Failed: a value < 0; for details see **nwy_error.h**.

nwy_sms_recv_message

Function	void nwy_sms_recv_message(nwy_sms_recv_info_type_t *sms_data)
Description	To obtain the received message content
Parameter	nwy_sms_recv_info_type_t *sms_data oa: sender address oa_size: length of the sender address scts: timestamp dcs: message coding format nIndex: message index number

Neoway Confidential

	nDataLen: message data length pData: message data For details, see **nwy_sms.h**.
Return	N/A.

nwy_sms_delete_message

Function	int nwy_sms_delete_message(uint16_t nindex)
Description	To delete a message
Parameter	nIndex: message index number
Return	Successful: 0 Failed: a value < 0; for details see **nwy_error.h**.

nwy_sms_read_message

Function	int nwy_sms_read_message(unsigned nindex, nwy_sms_recv_info_type_t *sms_data)
Description	To read a message
Parameter	nIndex: message index number sms_data: message content
Return	Successful: 0 Failed: a value < 0; for details see **nwy_error.h**.

nwy_sms_reg_recv_cb

Function	int nwy_sms_reg_recv_cb(nwy_sms_recv_cb_t recv_cb)
Description	A callback function of the URC related to SMS message registration.
Parameter	nwy_sms_recv_cb_t recv_cb: callback function
Return	1

nwy_sms_unsol_sms_event

Function	void nwy_sms_unsol_sms_event(void);

Neoway Confidential

Description	URCs related to SMS message registration; wait and process the message.
Parameter	N/A.
Return	N/A.

3.2.3 SIM

SIM API definitions can be found in **nwy_sim.h**. To operate or query UICC of SIM/USIM/RUIM, including obtaining card status, operating pin, and reading important files in UICC.

Programming Reference

- Invoke **nwy_sim_get_card_status** to obtain card status and perform the next operation based on the card status.
- If the card status is **NWY_SIM_PIN_REQ/NWY_SIM_PUK_REQ**, the system instructs the user to pass PIN/PUK verification.
- If the card status is **NWY_SIM_READY**, card information, such as the IMSI, ICCID, and MSISDN, can be obtained, and the PIN code can be enabled, disabled, or modified.

  For details, see **example/sim/nwy_sim_test.c**.

nwy_sim_get_card_status

Function	nwy_sim_status_t nwy_sim_get_card_status(nwy_sim_id_t sim_id)
Description	To obtain the status of current UICC, including whether the card is inserted and whether PIN/PUK is required to be inputted.
Parameter	sim_id: Specifies an ID number of the card to be operated.
Return	Successful: return card status; see **nwy_sim_status_t**. Failed: a value < 0; for details see **nwy_error.h**.

nwy_sim_verify_pin

Function	int nwy_sim_verify_pin(nwy_sim_id_t sim_id, const char* pin)
Description	To verify PIN code When the obtained UICC status is NWY_SIN_PIN_REQ, this function is required to

Neoway Confidential

	verify the PIN code.
Parameter	sim_id: Specifies an ID number of the card to be operated. pin: character string, the length should comply with 3GPP specifications.
Return	Successful: **NWY_RES_OK** Failed: a value < 0; for details see **nwy_error.h**.

nwy_sim_unblock

Function	int nwy_sim_unblock (nwy_sim_id_t sim_id, const char* puk,const char* new_pin)
Description	To enter PUK When the SIM card is locked by PIN code, that is, the UICC status is NWY_SIM_PUK_REQ. It is required to unlock.
Parameter	sim_id: Specifies an ID number of the card to be operated. puk: character string, the length should comply with 3GPP specifications. new_pin: new PIN code. Character string, the length should comply with 3GPP specifications.
Return	Successful: **NWY_RES_OK** Failed: a value < 0; for details see **nwy_error.h**.

nwy_sim_get_pin_mode

Function	nwy_sim_pin_mode_t nwy_sim_get_pin_mode(nwy_sim_id_t sim_id)
Description	To obtain the status of current PIN
Parameter	sim_id: Specifies an ID number of the card to be operated.
Return	The current PIN status; for details, see **nwy_sim_pin_mode_t**.

nwy_sim_enable_pin

Function	int nwy_sim_enable_pin(nwy_sim_id_t sim_id, const char* pin)
Description	To enable PIN The function can be called only when the PIN status is **NWY_SIM_PIN_MODE_DISABLED**.
Parameter	sim_id: Specifies an ID number of the card to be operated. pin: character string, the length should comply with 3GPP specifications.

Neoway Confidential

Return	Successful: **NWY_RES_OK** Failed: a value < 0; for details see **nwy_error.h**.

nwy_sim_disable_pin

Function	int nwy_sim_disable_pin(nwy_sim_id_t sim_id, const char* pin)
Description	To disable PIN The function can be called only when the PIN status is **NWY_SIM_PIN_MODE_ENABLED**.
Parameter	sim_id: Specifies an ID number of the card to be operated. pin: character string, the length should comply with 3GPP specifications.
Return	Successful: **NWY_RES_OK** Failed: a value < 0; for details see **nwy_error.h**.

nwy_sim_change_pin

Function	int nwy_sim_change_pin (nwy_sim_id_t sim_id, const char* old_pin, const char* old_pin)
Description	To modify PIN code This function can be invoked only when the PIN code status is **NWY_SIM_PIN_MODE_ENABLED**.
Parameter	sim_id: Specifies an ID number of the card to be operated. old_pin: old PIN code. character string, the length should comply with 3GPP specifications. new_pin: new PIN code. character string, the length should comply with 3GPP specifications.
Return	Successful: **NWY_RES_OK** Failed: a value < 0; for details see **nwy_error.h**.

nwy_sim_get_pin_puk_retry_times

Function	int nwy_sim_get_pin_puk_retry_times (nwy_sim_id_t sim_id, uint8* pin_times, uint8* puk_times)
Description	To obtain the remaining attempts of PIN and PUK

Neoway Confidential

Parameter	sim_id: Specifies an ID number of the card to be operated. pin_times: remaining attempts of PIN puk_times: remaining attempts of PUK
Return	Successful: **NWY_RES_OK** Failed: a value < 0; for details see **nwy_error.h**.

nwy_sim_get_iccid

Function	int nwy_sim_get_iccid (nwy_sim_id_t sim_id, char* iccid_buf, size_t buf_len)
Description	To obtain UICC ID
Parameter	sim_id: Specifies an ID number of the card to be operated. iccid_buf: Buffer for ICCID character string. Reserve enough space to write ICCID. buf_len: Space of iccid_buf
Return	Successful: **NWY_RES_OK** Failed: a value < 0; for details see **nwy_error.h**.

nwy_sim_get_imsi

Function	int nwy_sim_get_imsi (nwy_sim_id_t sim_id, char* imsi_buf, size_t buf_len)
Description	To obtain IMSI
Parameter	sim_id: Specifies an ID number of the card to be operated. imsi_buf: Buffer for IMSI character string. Reserve enough space to write IMSI. buf_len: Space of imsi_buf
Return	Successful: **NWY_RES_OK** Failed: a value < 0; for details see **nwy_error.h**.

nwy_sim_get_msisdn

Function	int nwy_sim_get_msisdn (nwy_sim_id_t sim_id, char* msisdn_buf, size_t buf_len)
Description	To obtain MSISDN
Parameter	sim_id: Specifies an ID number of the card to be operated. msisdn_buf: Buffer for MSISDN character string. Reserve enough space to write MSISDN. buf_len: Space of msisdn_buf

Neoway Confidential

Return	Successful: **NWY_RES_OK** Failed: a value < 0; for details see **nwy_error.h**.

nwy_sim_switch_slot

Function	int nwy_sim_switch_slot(nwy_sim_id_t sim_id)
Description	To perform multi-card switching.
Parameter	sim_id: Specifies an ID number of the card to be operated.
Return	Successful: **NWY_RES_OK** Failed: a value < 0; for details see **nwy_error.h**.

3.2.4 Network

Network API defintions can be found in **nwy_network.h**. They are used to set network mode, obtain network registration status and network mode, set network mode, and monitor network status according the registered callback function of the Network API.

For the sample examples, see **example/network/nwy_network_test.c**.

nwy_nw_get_register_info

Function	int nwy_nw_get_register_info(nwy_nw_regs_info_type_t *regs_info)
Description	To query the current network information.
Parameter	regs_info: Network informtion, for the definitions, see **nwy_network.h**.
Return	Successful: **NWY_RES_OK** Failed: a value < 0; for details see **nwy_error.h**.

nwy_nw_get_operator_name

Function	int nwy_nw_get_operator_name(nwy_nw_operator_name_t *opt_name)
Description	To obtain information of the current network operator.
Parameter	opt_name: Network operator information, for details, see the definitions in **nwy_network.h**.

Neoway Confidential

Return	Successful: **NWY_RES_OK** Failed: a value < 0; for details see **nwy_error.h**.

nwy_nw_set_rf_moed

Function	int nwy_nw_set_rf_mode(nwy_rf_status_e rf_status)
Description	RF switch
Parameter	rf_status: Used to set the RF status, for the definitions, see **nwy_network.h**.
Return	Successful: **NWY_RES_OK** Failed: a value < 0; for details see nwy_error.h.

nwy_nw_get_signal_csq

Function	int nwy_nw_get_signal_csq(int *csq_val)
Description	To query the signal
Parameter	csq_val: Signal value
Return	Successful: **NWY_RES_OK** Failed: a value < 0; for details see **nwy_error.h**.

3.2.5 DM

DM API definitions can be found in **nwy_dm.h**.

- According to 3GPP and 3GPP2 specifications, IMEI is 15-bit, MEID is 14-bit, and ESN is 8-bit. Reserve buffer long enough for them.
- The firmware version should be not shorter than 32 bytes.

For details, see **example/dm/nwy_dm_test.c**.

nwy_dm_set_dev_op_mode

Function	int nwy_dm_set_dev_op_mode(nwy_dm_operating_mode_top_mode)
Description	To set the operating mode of the device, including Online, Offline, LPM, FTM, and so on. For details, see the definitions in **nwy_dm_operating_mode_t**.

Neoway Confidential

Parameter	op_mode: operating mode
Return	Successful: **NWY_RES_OK** Failed: a value < 0; for details see **nwy_error.h**.

nwy_dm_get_dev_op_mode

Function	int nwy_dm_get_dev_op_mode(nwy_dm_operating_mode_top_mode)
Description	To obtain the operating mode of the device, including Online, Offline, LPM, FTM, and so on. For details, see the definitions in **nwy_dm_operating_mode_t**.
Parameter	op_mode: The returned operating mode
Return	Successful: **NWY_RES_OK** Failed: a value < 0; for details see **nwy_error.h**.

nwy_dm_get_imei

Function	int nwy_dm_get_imei(char* imei_buf, size_t buf_len)
Description	To obtain IMEI of the device.
Parameter	imei_buf: buffer used to return IMEI, ensure that the buffer length meets the length requirement. buf_len: Imei_buf length
Return	Successful: **NWY_RES_OK** Failed: a value < 0; for details see **nwy_error.h**.

nwy_dm_get_fw_version

Function	int nwy_dm_get_fw_version(char* version_buf, size_t buf_len)
Description	To obtain the device’s firmware version.
Parameter	version_buf: buffer used to return the device’s firmware version, ensure that the buffer length meets the length requirement. buf_len: version_buf length
Return	Successful: **NWY_RES_OK** Failed: a value < 0; for details see **nwy_error.h**.

Neoway Confidential

nwy_dm_get_dev_model

Function	int nwy_dm_get_dev_model(char* model_buf, size_t buf_len)
Description	To obtain the device's model
Parameter	model_buf: buffer used to return the device's model, ensure that the buffer length meets the length requirement. buf_len: model_buf length
Return	Successful: **NWY_RES_OK** Failed: a value < 0; for details see **nwy_error.h**.

nwy_dm_get_sdk_version

Function	int nwy_dm_get_sdk_version(char* ver_buf, size_t buf_len)
Description	To obtain the SDK version.
Parameter	buffer used to return the SDK version buf_len: ver_buf length
Return	Successful: **NWY_RES_OK** Failed: a value < 0; for details see **nwy_error.h**.

3.3 Network Protocol Related

This chapter provides tasks and sample code based on the POSIX standard on Linux OS to introduce how to implement the network relate functions.

3.4 Network Related

This chapter describes development tasks and cases Linux open source code based on POSIX.

3.4.1 TCP

A simple TCP client, showing how to use a socket.

Header Files

#include “netdb.h”

Neoway Confidential

#include “sys/types.h”

#include “netinet/in.h”

#include “sys/socket.h”

Programming Reference

1. Open a socket
2. Connect the device to a remote server.
3. Send and receive data over the connection.
4. Close the socket.

For details, see **example/tcp/tcp_client.c**.

3.4.2 FTP

A simple FTP client, implementing basic FTP primitives on Linux APIs

Header Files

#include <sys/types.h>

#include <sys/stat.h>

#include <fcntl.h>

#include <sys/socket.h>

#include <netdb.h>

#include <ctype.h>

#include <sys/ioctl.h>

#include <errno.h>

Programming Reference

1. Connect to an FTP server.

Neoway Confidential

2. Input account.
3. Input password.
4. Check/download/upload files.
5. Exit from the server.

For details, see **example/ftp/ftp_test.c**.

3.4.3 HTTP

A simple HTTP client, implementing HTTP get request on Linux APIs

Header Files

#include“sys/types.h”

#include “sys/stat.h”

#include “fcntl.h”

#include “sys/socket.h”

#include “netdb.h”

#include “ctype.h”

#include “sys/ioctl.h”

#include “errno.h”

Programming Reference

For details, see **example/http/http_test.c**.

Neoway Confidential