ARM926EJ-S™ Based

32-bit Microprocessor

NuDesign NK-980IoT

User Manual

For additional information or questions, please contact: Nuvoton Technology Corporation.

www.nuvoton.com

Table of Contents

1 OVERVIEW

Nuvoton's NuDesign NK-980IoT demo board is based on Nuvoton NUC980DK61Y microprocessor, and includes a single 10/100 Ethernet port, 2 high-speed USB hosts, 1Gb SPI NAND Flash memory, a microphone input, a stereo headphone output and an Arduino compatible interface. This demo board supports embedded Linux OS which provides all IoT protocols you need, such as AWS Client, mbed-cloud Client, MQTT, and Web Server.

User can start using the NuDesign NK-980IoT demo board following the instruction of this document to develop a plenty of IoTapplications quickly.

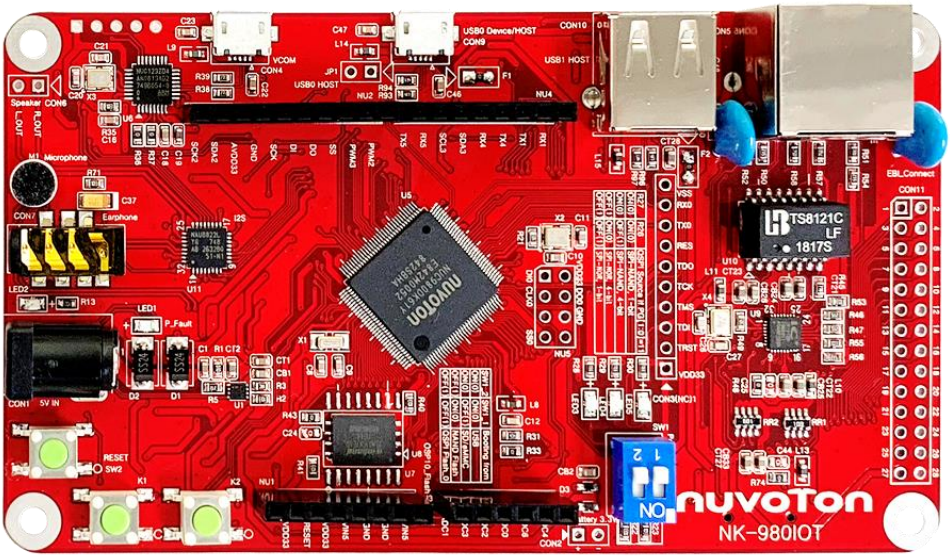

Figure 1-1 NuDesign NK-980IoT Demo Board – Front

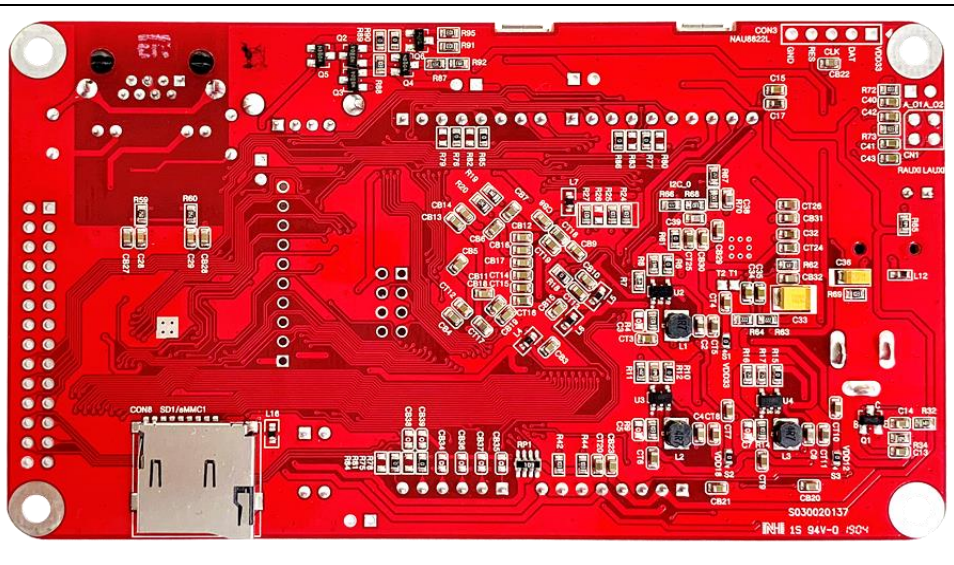

Figure 1-2 NuDesign NK-980IoT Demo Board – Back

2 NUDESIGN NK-980IOT DEMO QUICK START

2.1 Install BSP

Linux BSP contains three directories. Content of each directory is listed in the following table.

Directory Name	Content
BSP	A tar ball contains cross compiler, root file system, and pre-build tool chain.
Documents	BSP related documents.
Tools	NuWriter tool and its driver for Windows and SD writer tool.

Table 2-1 BSP Content

Source code could be downloaded using repo tool. The steps are listed below.

Make sure you have a bin/ directory in your home directory and that is included in your path.

```
$ mkdir ~/bin
$ export PATH=~/bin:$PATH
```

Download the repo tool and ensure it is executable.

```
$ curl https://storage.googleapis.com/git-repo-downloads/repo > ~/bin/repo
$ chmod a+x ~/bin/repo
```

Create an empty directory to hold working directory.

```
$ mkdir WORKING_DIR
$ cd WORKING_DIR
```

Configure git with your real name and email address.

```
$ git config --global user.name "Your Name"
$ git config --global user.email "you@example.com"
```

Use one of following commands to download a manifest file for NUC980 BSP. The first command is downloaded from Github, and the second command is downloaded from Gitee. You can use the command to select the site with faster download speed.

```
$ repo init -u git://github.com/OpenNuvoton/manifest.git -b nuc980-2019.09 –m github.xml
```

Or

```
$ repo init -u https://gitee.com/OpenNuvoton/manifest.git -b nuc980-2019.09 –m gitee.xml
```

Then download source code.

```
$ repo sync
```

After downloading the source code, please copy the tar ball under the BSP directory to Linux machine and use the following command to extract the file.

```
$ tar zxvf nuc980bsp.tar.gz
```

After entering nuc980bsp directory, execute the installation script install.sh. This script requires the administrator privilege to execute. You can use “su” command to switch to root and execute the installation script.

```
$ su
Password: (Enter password of root)
```

```
# ./install.sh
```

Or execute this script as root by using “sudo” command. (This method works for those distributions do not open the root account privilege, such as Ubuntu.)

```
# sudo ./install.sh
```

Below is the console output during installation. The path input should be the same as the WORKING_DIR set previously.

```
Now installing arm_linux_4.8 tool chain to /usr/local/
Setting tool chain environment
Installing arm_linux_4.8 tool chain successfully
Install rootfs, applications, u-boot and Linux kernel
Please enter absolute path for installing(eg:/home/<user name>) :
BSP will be installed in /<path you input>/nuc980bsp
/home/someone
Extract rootfs and pre-build images
...
...
NUC980 BSP installation complete
```

2.2 Build U-Boot

User can compile NuDesign NK-980IoT U-Boot code with following steps.

Enter U-Boot folder.

```
# cd u-boot-2016.11
```

Configure U-Boot for NuDesign NK-980IoT demo board.

```
$ make nuc980_iot_defconfig
```

Compile U-Boot. This step will generate u-boot.bin and spi/u-boot-spl.bin.

```
$ make
......
  COPY    u-boot.bin
  SYM     u-boot.sym
./scripts/check-config.sh u-boot.cfg \
                ./scripts/config_whitelist.txt . 1>&2
```

2.3 Build Linux Kernel

User can build kernel image with following steps.

Enter kernel folder.

```
# cd linux-4.4.y
```

Configure kernel for NuDesign NK-980IoT demo board.

```
# make nuc980_iot_defconfig
```

Build kernel.

```
$ make uImage
......
    Kernel: arch/arm/boot/Image is ready
cp arch/arm/boot/Image   ../image/980image
  Kernel: arch/arm/boot/Image is ready
cp arch/arm/boot/Image   ../image/980image
  GZIP    arch/arm/boot/compressed/piggy.gzip
  AS      arch/arm/boot/compressed/piggy.gzip.o
  LD      arch/arm/boot/compressed/vmlinux
  OBJCOPY arch/arm/boot/zImage
  Kernel: arch/arm/boot/zImage is ready
  Kernel: arch/arm/boot/Image is ready
cp arch/arm/boot/Image   ../image/980image
  Kernel: arch/arm/boot/zImage is ready
  UIMAGE  arch/arm/boot/uImage
Image Name:   Linux-4.4.115
Created:      Tue Dec 25 16:15:24 2018
Image Type:   ARM Linux Kernel Image (uncompressed)
Data Size:    8039480 Bytes = 7851.05 kB = 7.67 MB
Load Address: 00008000
Entry Point:  00008000
  Image arch/arm/boot/uImage is ready
cp arch/arm/boot/uImage   ../image/980uimage
$ ls ../image/
980image 980uimage
```

2.4 Program Kernel and U-Boot to SPI NAND Flash

This section introduces how to program U-Boot and kernel to SPI NAND Flash.

1. Install NuWriter Driver. (Please refer to "*NUC980 NuWriter User Manual*")
2. Set SW1 (Power on Setting) to boot from USB. Connect HSUSB Device port to PC.

SW1.2/SW1.1 Switch State	Function
ON/ON	Boot from USB
ON/OFF	Boot from SD/eMMC
OFF/OFF	Boot from SPI Flash

Table 2-2 Power On Setting

3. Open NuWriter. Select "**NUC980 series**" as target chip and select "**NUC980DK61Y.ini**" as DDR parameter. Click the **Continue** button and then select SPI NAND mode using the pull down menu in the main window.

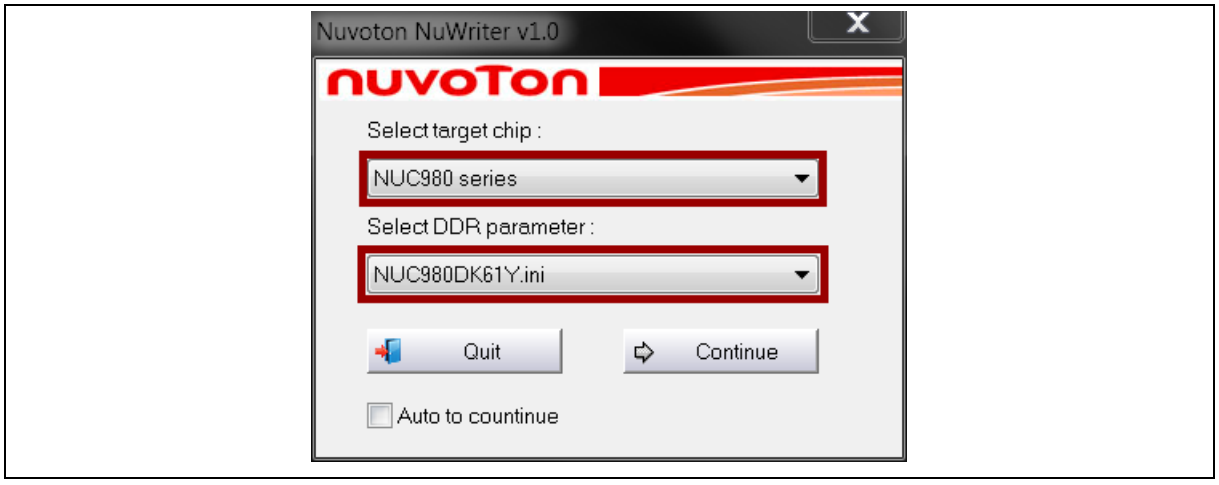

Figure 2-1 NuWriter Setting

1. Program *u-boot-spl.bin* in "u-boot-2016.11/spl" folder:
   1) Input parameter
      - **Image Name:** u-boot-spl.bin (refer to section 2.2)
      - **Image Type:** Loader
      - **Image execute address:** 0x200
   2) Click "**Program**"
   3) Wait until the program is completed.

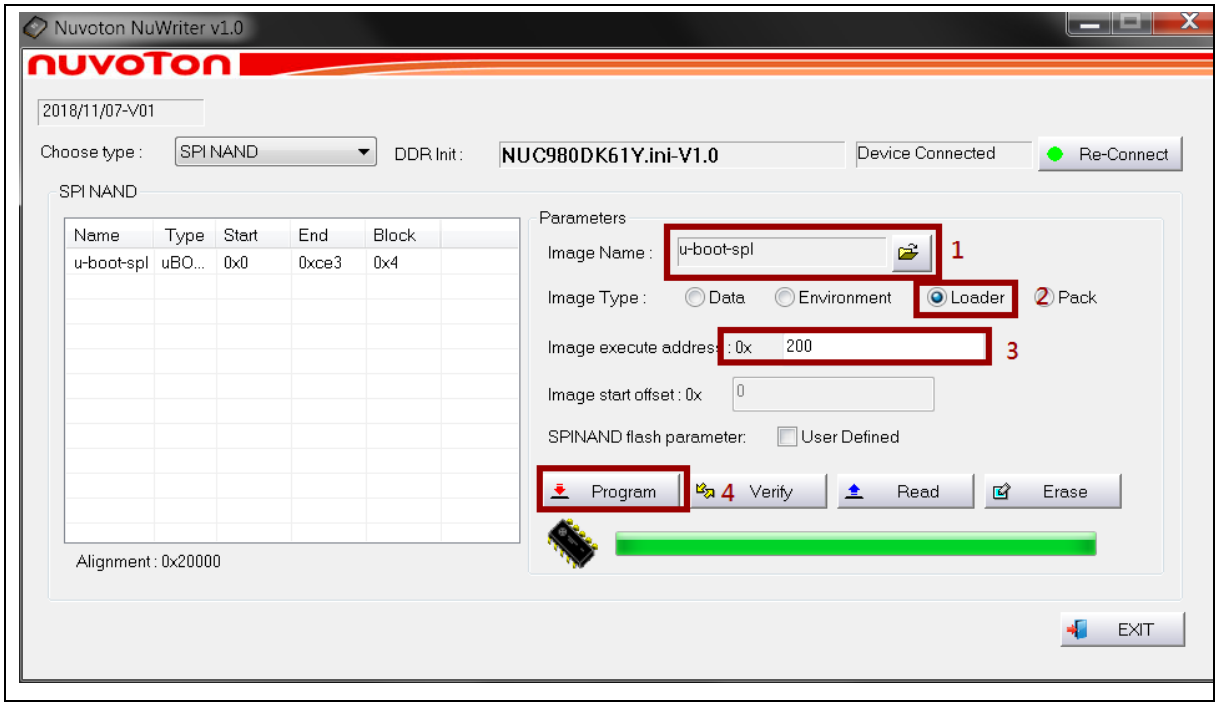

Figure 2-2 Program u-boot-spl

2. Program *u-boot.bin* in "u-boot-2016.11" folder:
   1) Input parameter

- **Image Name:** u-boot.bin (refer to section 2.2)
- **Image Type:** Data
- **Image start offset:** 0x100000

2) Click "**Program**"

3) Wait until the program is completed.

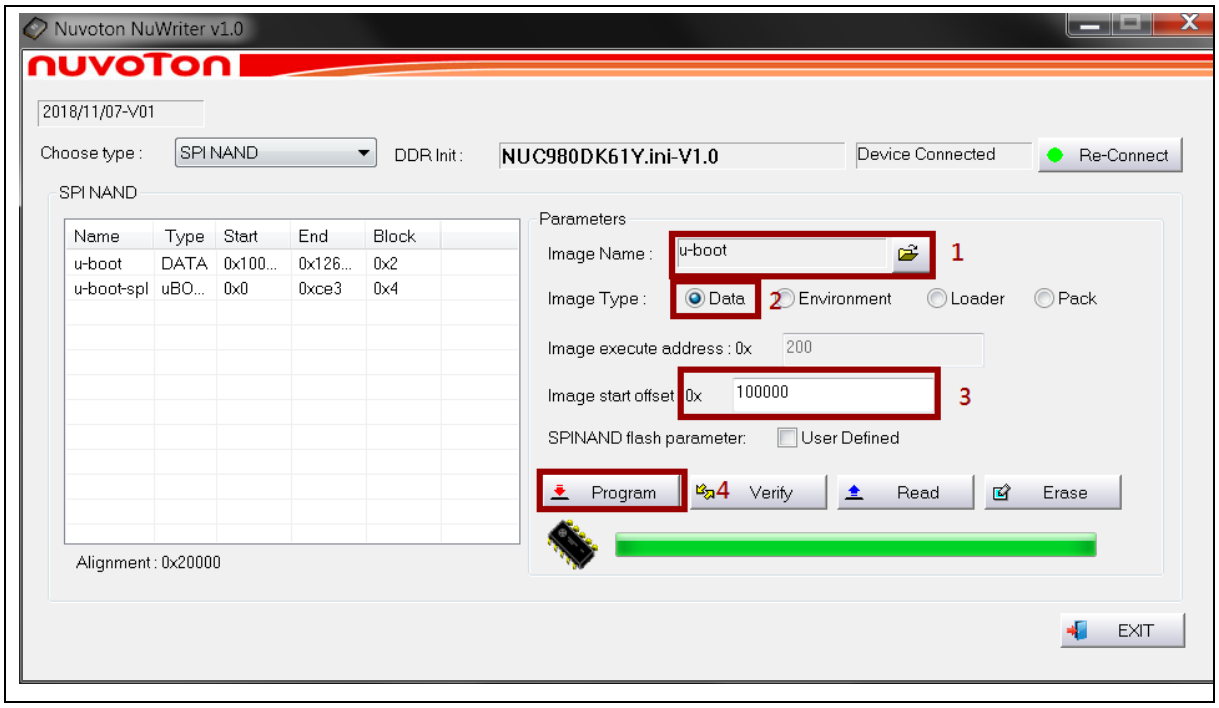

Figure 2-3 Program u-boot

3. Program kernel image:

1) Input parameter

- I**mage Name:** 980uimage (refer to section 2.3)
- **Image Type:** Data
- **Image start offset:** 0x200000

2) Click "**Program**"

3) Wait until the program is completed.

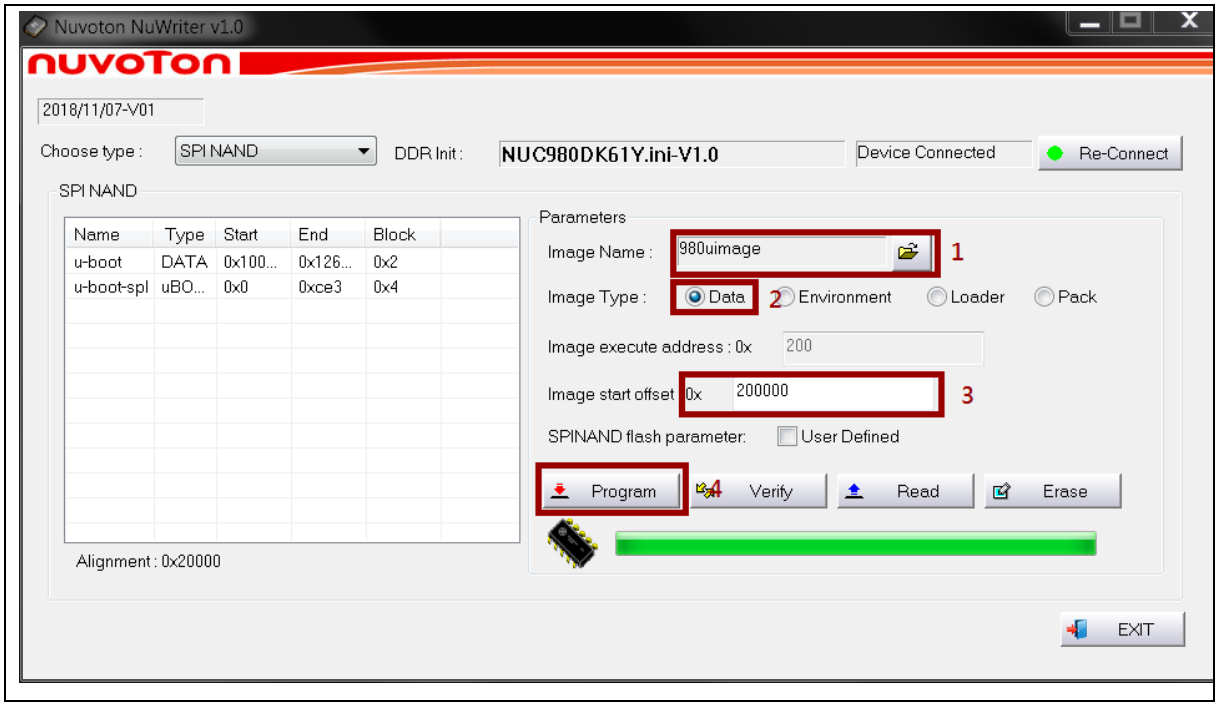

Figure 2-4 Program Kernel Image

2.5 Hardware Setup

1. Set SW1 (Power On Setting) to boot from QSPI 0 Flash (Refer to Table 2-2).
2. Connect USB CDC Debug Port as debug console port. The console setting is 1152008n1.

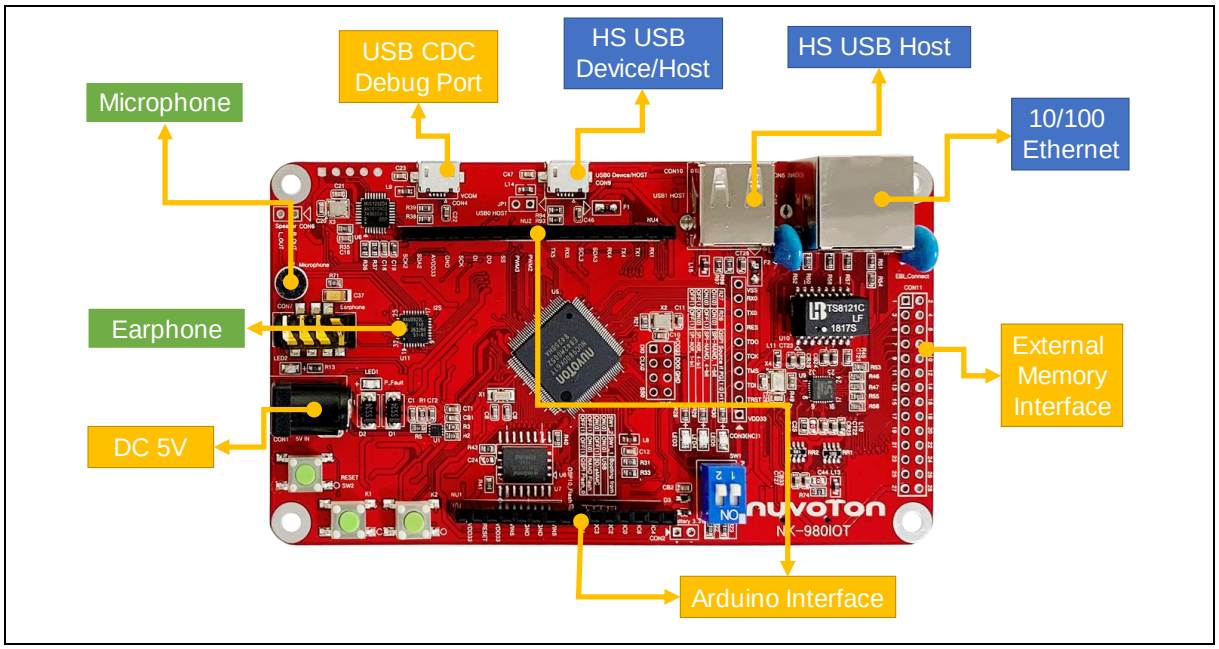

Figure 2-5 NuDesign NK-980IoT Demo Board Setup

2.6 Boot Linux Kernel

This section describes how to boot up Linux kernel.

1. Set SW1 (Power On Setting) to boot from QSPI 0 Flash (Refer to Table 2-2).
2. Press **Reset** button on the demo board. From console user can find that system enters U-Boot. Use following commands to launch Linux kernel after entering U-Boot shell.
   - Type “sf probe 0 75000000” to set SPI speed (optional)
   - Type “sf read 0x7FC0 0x200000 0x800000” to read kernel image from SPI NAND Flash to DDR.
   - Type “bootm 0x7FC0” to boot Linux kernel image.

```
U-Boot 2016.11-g82e8631-dirty (Jan 30 2019 - 14:55:45 +0800)

CPU: NUC980
Board: NUC980
DRAM:  64 MiB
SF: Detected W25N01GV with page size 2 KiB, erase size 128 KiB, total 128 MiB
In:    serial
Out:   serial
Err:   serial
Net:   Net Initialization Skipped
No ethernet found.
=>
=> sf probe 0 75000000
SF: Detected W25N01GV with page size 2 KiB, erase size 128 KiB, total 128 MiB
=> sf read 0x7FC0 0x200000 0x800000
device 0 offset 0x200000, size 0x800000
SF: 7733248 bytes @ 0x200000 Read: OK
=> bootm 0x7FC0
## Booting kernel from Legacy Image at 00007fc0 ...
   Image Name:   Linux-4.4.115+
   Image Type:   ARM Linux Kernel Image (uncompressed)
   Data Size:    7573624 Bytes = 7.2 MiB
   Load Address: 00008000
   Entry Point:  00008000
   Verifying Checksum ... OK
   XIP Kernel Image ... OK

Starting kernel ...
```

3. After booting Linux kernel image, user can see following information from UART console.

```
nuc980-i2c0 nuc980-i2c0: i2c-0: nuc980 I2C adapter
nuc980-i2c2 nuc980-i2c2: i2c-2: nuc980 I2C adapter
nuc980_sd_probe - pdev = nuc980-sdh
usbcore: registered new interface driver usbhid
usbhid: USB HID core driver
nuc980-nadc: nuc980 Normal ADC adapter
nuc980-audio nuc980-audio: nau8822-hifi <-> nuc980-audio-i2s mapping ok
NET: Registered protocol family 17
ALSA device list:
  #0: nuc980_IIS
Freeing unused kernel memory: 5304K

BusyBox v1.22.1 (2016-02-03 14:11:04 CST) built-in shell (ash)
Enter 'help' for a list of built-in commands.

~ # random: nonblocking pool is initialized
```

4. For the detailed kernel compile and setting, please refer to "*NUC980 Linux BSP User Manual*".

2.7 Connect Wi-Fi Access Point

To use Wi-Fi connection, users have to install Wi-Fi dongle driver module in command shell prompt. For example, using following command can install RTL8188EU kernel module.

```
# insmod /lib/modules/8188eu.ko
```

Then execute wpa_supplicant. The parameter in /etc/wpa_supplicant has to provide SSID, PSK… information of access point. After successfully connected with access point, use DHCP to lease a IP address from DHCP server.

```
# /usr/wpa_supplicant –Dwext –iwlan0 –c/etc/wpa_supplicant.conf –B
# udhcpc –i wlan0
```

2.8 Connect NB-IoT

To use NB-IoT, users can cross compile dialer application pppd using buildroot by enabling following options:

```
Target Packages --->
Networking applications --->
[*] pppd
```

Configure the APN name in rootfs/etc/ppp/peers/quectel-chat-connect and set the username, password in rootfs/etc/ppp/peers/quectel-ppp. Exectue the following command to dial up.

```
# /usr/sbin/pppd call quectel-ppp &
```

2.9 Connect LTE

To connect to LTE service provider, execute quectel-CM under /usr directory. This application can takes parameter to configure APN, username, password, authentication protocol, and pin code as shown in the following example. The parameter 0 here indicates no authentication protocol, other valid options are 1(PAP), 2(CHAP), and 3(MSCHAPv2).

```
# /usr/quectel-CM –s my_apn my_username my_password 0 -p 1234 &
```

3 AWS IOT

This chapter will describe how to connect AWS service on the NK-980IoT demo board.

3.1 Build AWS IoT Sample

Download AWS IoT device SDK using embedded C from GitHub URL: https://github.com/aws/aws-iot-device-sdk-embedded-C.

```
# git clone https://github.com/aws/aws-iot-device-sdk-embedded-C.git -b release
# cd aws-iot-device-sdk-embedded-C
```

Copy the certificate, private key, public key, and root CA files to “certs/” folder. These files should have been downloaded while creating a certificate for the IoT thing.

Download the latest MbedTLS library from GitHub URL: https://github.com/ARMmbed/mbedtls to AWS SDK’s “external_libs/mbedTLS” folder and build mbedTLS library.

```
# git clone https://github.com/ARMmbed/mbedtls.git external_libs/mbedTLS
# cd external_libs/mbedTLS
# export CC=arm-linux-gcc
# export AR=arm-linux-ar
# make
```

Enter the “*samples/linux/subscribe_publish_sample*” directory. Edit the “Makefile” to set CC as arm-linux-gcc.

```
CC = arm-linux-gcc
```

Edit “*aws_iot_config.h*” in the same directory to set IoT information. Find the code block surrounded by “Get from console” comment. Fill in these entries.

```
// Get from console
// ================================================
#define AWS_IOT_MQTT_HOST              ""
#define AWS_IOT_MQTT_PORT              443
#define AWS_IOT_MQTT_CLIENT_ID         "c-sdk-client-id"
#define AWS_IOT_MY_THING_NAME           "AWS-IoT-C-SDK"
#define AWS_IOT_ROOT_CA_FILENAME       "rootCA.crt"
#define AWS_IOT_CERTIFICATE_FILENAME   "cert.pem"
#define AWS_IOT_PRIVATE_KEY_FILENAME   "privkey.pem"
// ================================================
```

AWS_IOT_MY_THING_NAME is the name of the IoT thing you created on Amazon cloud IoT service. AWS_IOT_MQTT_HOST is obtained from “interact” of the IoT thing. AWS_IOT_MQTT_CLIENT_ID can be any unique name.

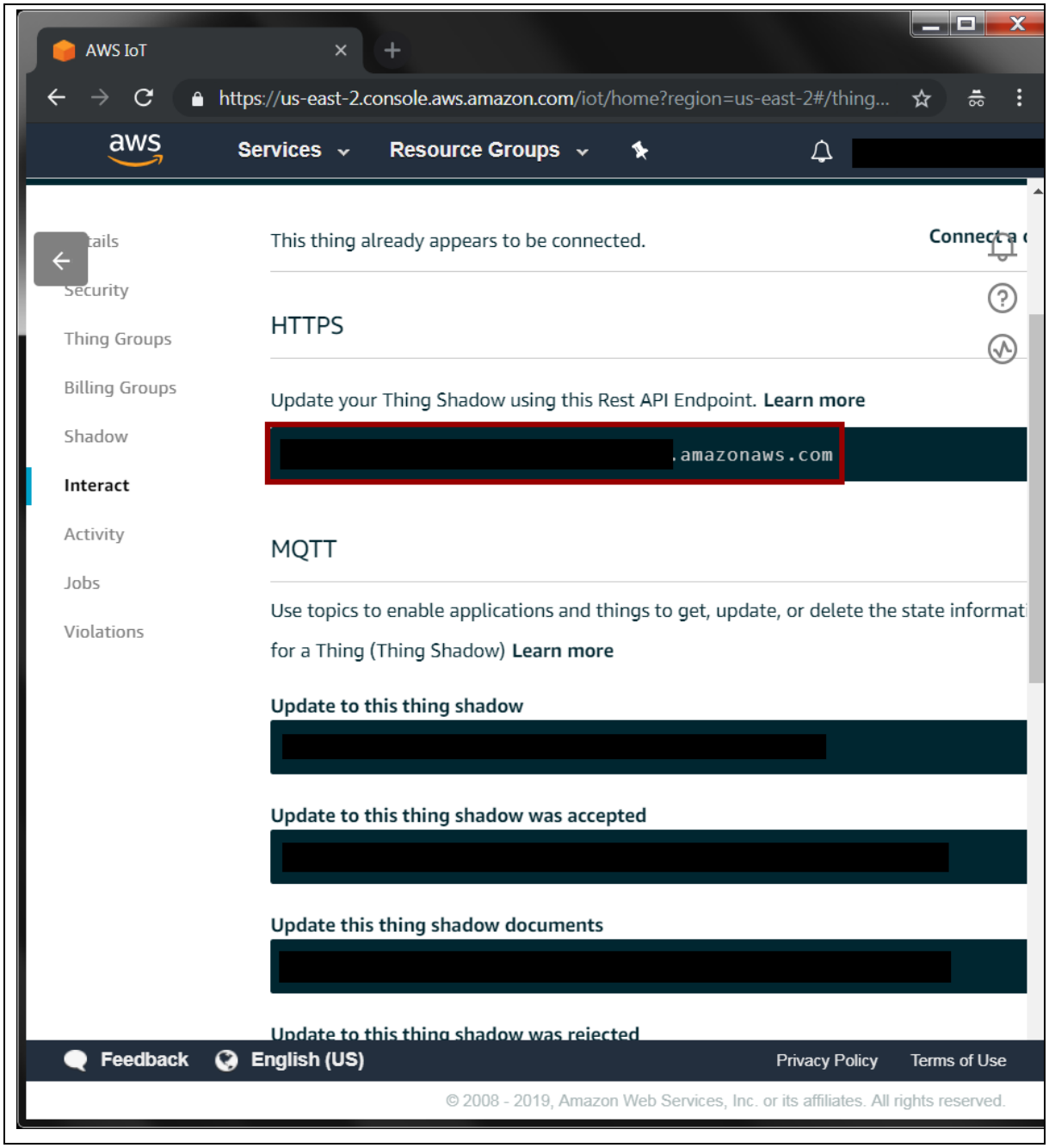

Figure 3-1 Obtain AWS_IOT_MQTT_HOST

For the AWS_IOT_ROOT_CA_FILENAME, AWS_IOT_CERTIFICATE_FILENAME, and AWS_IOT_PRIVATE_KEY_FILENAME, refer to the certificate files obtained while creating a certificate for an IoT thing.

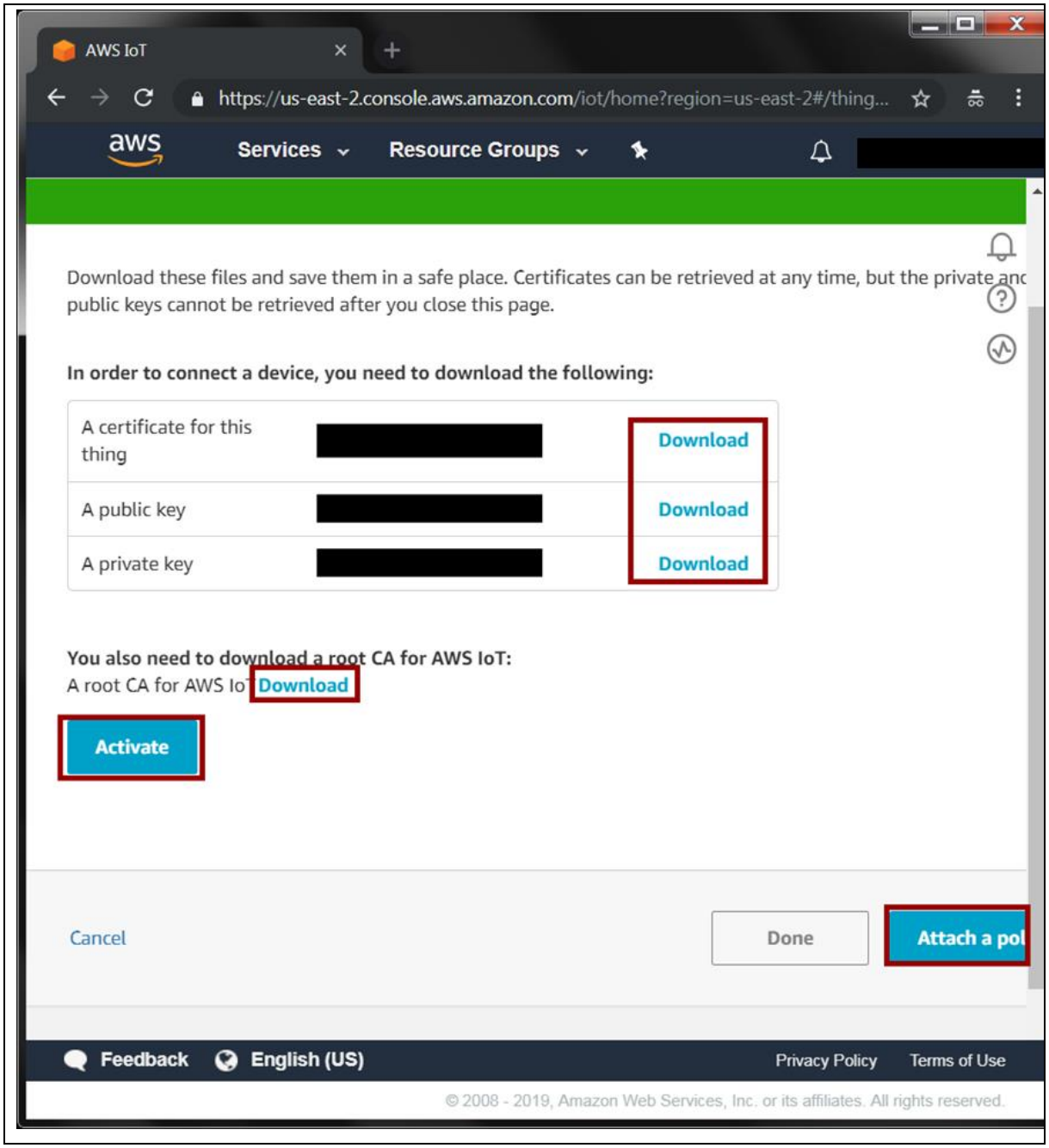

Figure 3-2 Obtain Certificate Files

You can edit "*subscribe_publish_sample.c*" to modify certDirectory[], which refers to the run-time path of your certificate files. Assume that the executable "subscribe_publish_sample" will be executed in "/usr/bin" folder and certificate files will be put in "/usr/certs" folder, then "certDirectory[]" should be "../certs".

```
static char certDirectory[PATH_MAX + 1] = "../certs";
```

In "samples/linux/ubscribe_publish_sample" folder, enter "make" to build the IoT client application "subscribe_publish_sample". Copy this application and certificate file to rootfs.

Please refer to the following URL for more information about AWS IoT Embedded C.

https://docs.aws.amazon.com/iot/latest/developerguide/iot-embedded-c-sdk.html.

3.2 Execute AWS IoT sample

First of all, you must have available internet connection to Amazon cloud. For example, you can use Wi-Fi or NB-IoT mentioned in the previous chapter, or use Ethernet interface. To use Ethernet, you can use following commands to bring up EMAC interface and ask local gateway to assign a IP address to NUC980 and provide a valid DNS server address.

```
# ifconfig eth0 up
# udhcpc –i eth0
```

Build IoT sample “subscribe_publish_sample” as described in section 1.1 of this document. Now you can execute “subscribe_publish_sample” to connect the IoT thing.

Please note that the AWS IoT thing must be created and has an activated certificate with attached policy.

In addition to using AWS IoT device SDK (MQTT client), you can also send messages to AWS service via HTTP connection. Following command line specifies a certificate file and private key file and sends a message to the specified AWS URL.

```
# cd usr
# ./ curl --tlsv1.2 --cacert root-CA.crt --cert 4b7828d2e5-certificate.pem.crt --key 4b7828d2e5-private.pem.key -X POST -d "{ \"serialNumber\": \"G030JF053216F1BS\", \"clickType\": \"SINGLE\", \"batteryVoltage\": \"2000mV\" }" "https://a1pn10j0v8htvw.iot.us-east-1.amazonaws.com:8443/topics/iotbutton/virtualButton?qos=1"
```

Where the parameters are described below:

- --tlsv1.2

Use TLSv1.2 (SSL). curl must be installed with OpenSSL and you must use version 1.2 of TLS.

- --cacert <filename>

The filename of the CA certificate to verify the peer.

- --cert <filename>

The client certificate filename.

- --key <filename>

The private key filename.

- -X POST

The type of request (in this case, POST).

- -d <data>

The HTTP POST data you want to publish. In this case, emulate the data sent by a single button press.

- "https://..."

The URL. In this case, the REST API endpoint for the thing.

4 PELION DEVICE MANAGEMENT

This chapter will describe how to connect Pelion Device Management service on the NK-980IoT demo board.

4.1 Create New Developer Certification

Before connecting to Pelion Device Management, you should create a developer certificate for test. The URL of Pelion Device Management is https://www.pelion.com/. Please login and enter Device Management Console. After login successfully, you can find **Certificates** option in **Device identity**, then click **Certificates**, **New certificate** and **Create a developer certificate** options. In pop-up window, you can enter some description for the certificate optionally. If no description, you can click **Create Certificate** to create. After creating done, you can find the newly created certificate in your certificate list. You can click the newly created certificate on your list and click **Download Developer C file** on the pop-up window. Remember to save the C file which will be used for mbed-cloud-client-example building.

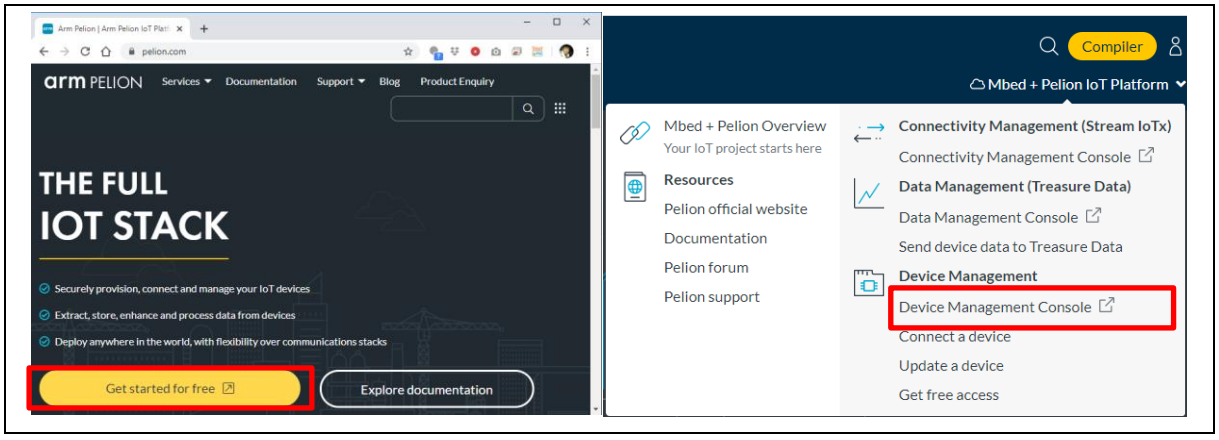

Figure 4-1 Enter Device Management Console

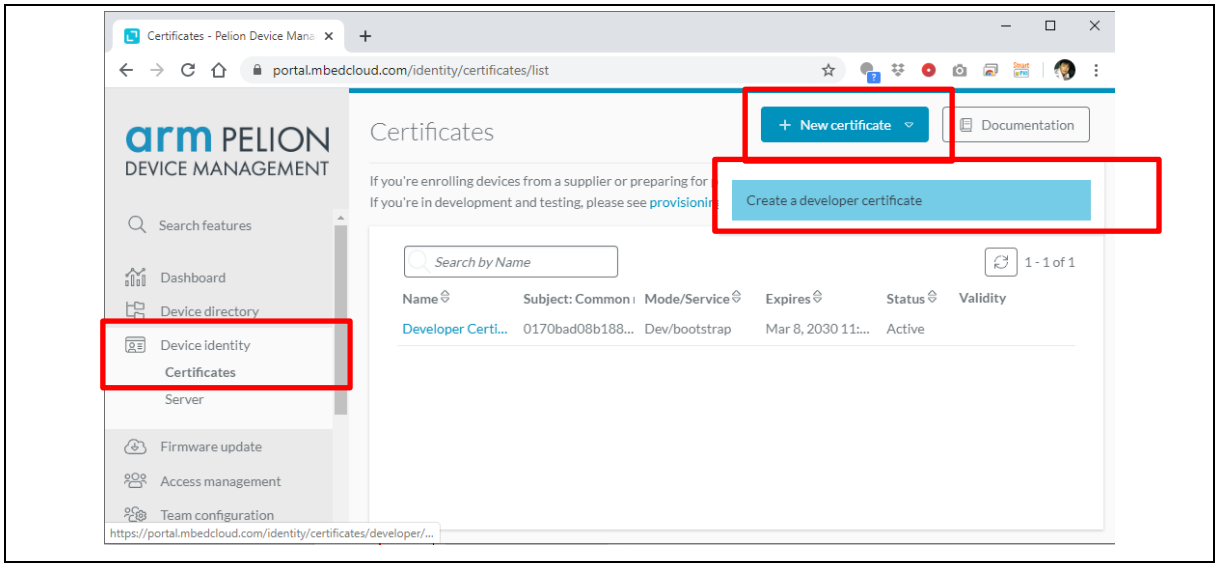

Figure 4-2 Create a Developer Certificate

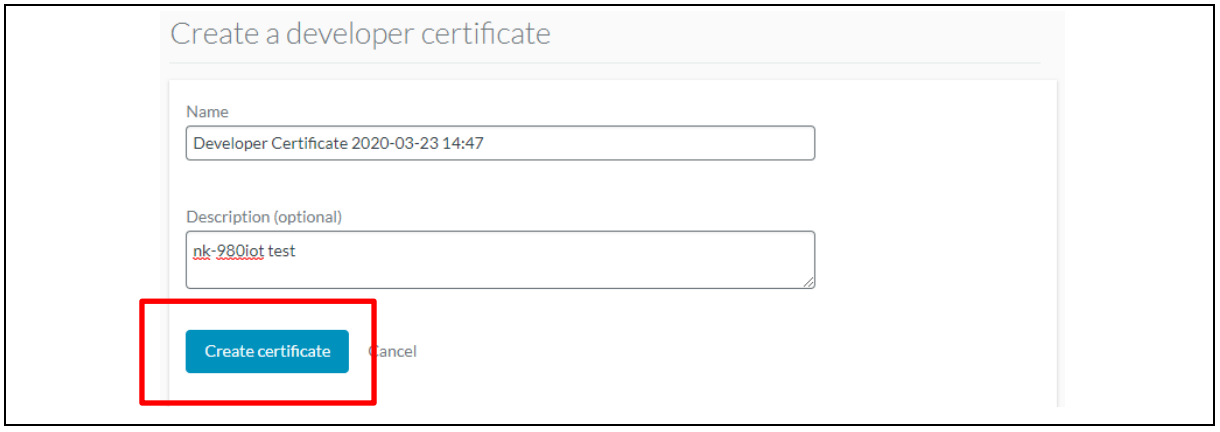

Figure 4-3 Enter Description and Submit

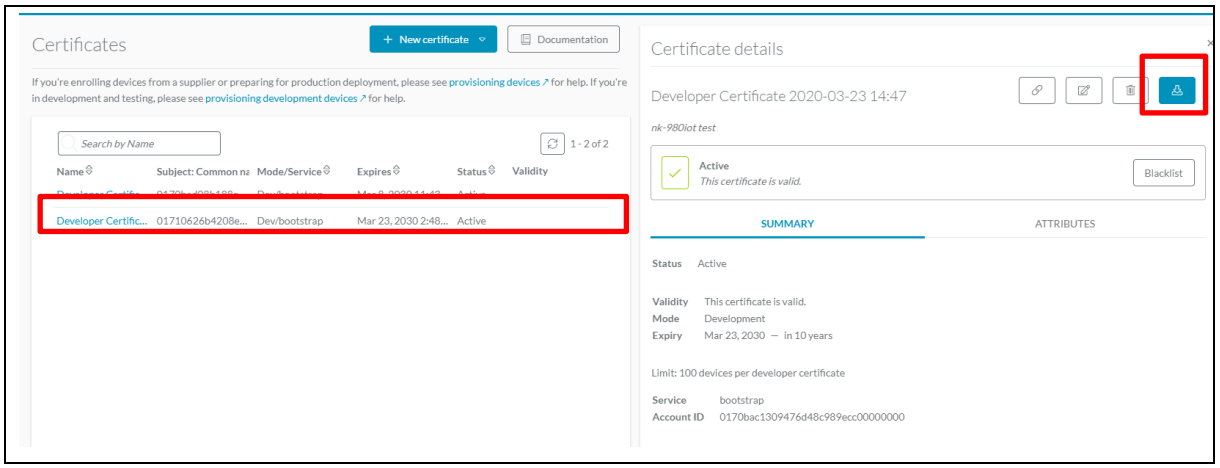

Figure 4-4 Download Developer Certificate C File

4.2 Build mbed-cloud-client-example

Download the mbed-cloud-client-example package from OpenNuvoton on GitHub.

URL: https://github.com/OpenNuvoton/mbed-cloud-client-example

```
# sudo apt install python2.7 python-pip git mercurial virtualenv cmake
# git clone https://github.com/OpenNuvoton/mbed-cloud-client-example -b NK-980IOT
# cd mbed-cloud-client-example
```

Copy the developer certificate C file - *mbed_cloud_dev_credentials.c* to root folder. The file should have been downloaded while creating a developer certificate from Pelion Device Management. Then, execuate commands below to install virtualenv on your host and deploy some dependency libraries. Finally, build the example for the NUC980 platform. After executing ‘./build_nuc980.sh’, its path of outputed file is at *__ARM_NUC980_mbedtls/Debug/mbedCloudClientExample.elf.elf*.

```
# pip2 install virtualenv
# mkdir -p ~/virtualenvs
# virtualenv ~/virtualenvs/nk-980iot
# . ~/virtualenvs/nk-980iot/bin/activate
(nk-980iot)# pip install mbed-cli
```

```
(nk-980iot)# mbed deploy
(nk-980iot)# ./build_nuc980.sh
(nk-980iot)# ls __ARM_NUC980_mbedtls/Debug/mbedCloudClientExample.elf.elf
```

4.3 Execute mbed-cloud-client-example

First of all, you must have available internet connection to mbed-pelion. For example you can use Wi-Fi or use Ethernet interface. To use Ethernet, you can use following commands to bring up EMAC interface and ask local gateway to assign a IP address to NUC980 and provide a valid DNS server address. Before executing the program, you should set system time to avoid authenticating wrong.

```
# ifconfig eth0 up
# udhcpc –i eth0
# date -s "2020-03-23 15:16:00"
Mon Mar 23 15:16:00 UTC 2020
```

You can copy mbedCloudClientExample.elf.elf program in a SD card. After booting done, you can insert the card and mount the partition volume on a folder. Finally, you can execute the mbedCloudClientExample.elf.elf program and observe the “Client registered” message on console. The message implies it is connected to mbed pelion successfully.

```
mmc0: new SD card at address e624
mmcblk0: mmc0:e624 SU128 120 MiB

# mount -t vfat /dev/mmcblk0 /mnt/mbed-pelion/
# cd /mnt/mbed-pelion/
# ./mbedCloudClientExample.elf.elf
In single-partition mode.
Creating path ./pal
Start Device Management Client
Using hardcoded Root of Trust, not suitable for production use.
Starting developer flow
Generating random from /dev/random, this can take a long time!
Developer credentials already exist, continuing..
Application ready. Build at: Mar 20 2020 16:41:22
Network initialized, registering...
Client registered
Endpoint Name: 0170e7feb99e0000000000010010ea50
Device ID: 0170e7feb99e0000000000010010ea50
Button resource automatically updated. Value 1
......
......
```

You can also observe device event on mbed-cloud. In **Device directory** menu, click **Devices**, then find out the corresponding Device ID and click it. In the *Device details* window, click ***I*3200** in **RESOURCES** page. You can see the digital input event device sent.

Figure 4-5 Device – Digital Input Event

5 CONCLUSION

Users can connect to cloud service easily following the procedure described in this document, and develop IoT gateway system using the NuDesign NK-980IoT demo board.

6 REVISION HISTORY

Date	Reversion	Description
2019.05.20	1.00	1. Initial release.
2019.04.22	1.01	1. Editorial change.
2019.09.09	1.02	1. Added LTE setting.
2020.04.06	1.03	1. Added mbed pelion device management chapter.

Important Notice

Nuvoton Products are neither intended nor warranted for usage in systems or equipment, any malfunction or failure of which may cause loss of human life, bodily injury or severe property damage. Such applications are deemed, "Insecure Usage".

Insecure usage includes, but is not limited to: equipment for surgical implementation, atomic energy control instruments, airplane or spaceship instruments, the control or operation of dynamic, brake or safety systems designed for vehicular use, traffic signal instruments, all types of safety devices, and other applications intended to support or sustain life.

All Insecure Usage shall be made at customer's risk, and in the event that third parties lay claims to Nuvoton as a result of customer's Insecure Usage, customer shall indemnify the damages and liabilities thus incurred by Nuvoton.