ARM926EJ-S™

32-bit Microprocessor

NUC980 NuWriter User Manual

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

www.nuvoton.com

Table of Contents

1 OVERVIEW

The NuWriter is a programing tool for the NUC980 series. The NuWriter application and firmware code are open sourced, and users can add new features or develop new user interfaces per user's application. The NuWriter tool uses chip's USB ISP mode with windows application on the PC by USB device for data transmission to program the image file to different storage devices. On-Board ROM device includes NAND Flash, SPI Flash , eMMC/SD, and SPI NAND Flash.

The NUC980 series MPU supports the following four system boot-up conditions:

- Boot from eMMC/SD device
- Boot from SPI FLASH device
- Boot from NAND-type Flash ROM device
- USB ISP mode

All of these four conditions are selected by the Power-on Setting. After the NUC980 series chip power on reset, the Power-on Setting is enabled or disabled to decide startup flow. For example, watchdog timer enable/disable, QSPI0_CLK Frequency selection and JTAG enable/disable, and so on. The Power-on setting can be set by the DIP Switch (SW2) on the development board. The DIP switches SW2.1~SW2.10 are set to ON/OFF, respectively. The corresponding GPIO PG0~PG9 pin output potential is 0/1. The table shown below describes the definition of each power-on setting bit.

Power-On Setting Pin(GPIO PG)	DIP Switch(SW2)	**Description**	**Power-On Setting Register Bit**
USB0_ID	**N/A**	**USB Port 0 Role Selection** 0 = USB Port 0 act as a USB host. 1 = USB Port 0 act as a USB device.	USBID (SYS_PWRON[16])
PG[1:0]	SW2.2/SW2.1	**Boot Source Selection** 00 = Boot from USB. 01 = Boor from SD/eMMC. 10 = Boot from NAND Flash. 11 = Boot from SPI Flash.	BTSSEL (SYS_PWRON[1:0])
PG.2	SW2.3	**QSPI0_CLK Frequency Selection** 0 = QSPI0_CLK frequency is 30 MHz. 1 = QSPI0_CLK frequency is 50 MHz.	QSPI0CKSEL (SYS_PWRON[2])
PG.3	SW2.4	**Watchdog Timer (WDT) Enabled/Disabled Selection** 0 = After power-on, WDT Disabled. 1 = after power-on WDT Enabled.	WDTON (SYS_PWRON[3])
PG.4	SW2.5	**JTAG Interface Selection** 0 = Pin PA[6:2] used as JTAG interface. 1 = Pin PG[15:11] used as JTAG interface.	JTAGSEL (SYS_PWRON[4])
PG.5	SW2.6	**UART 0 Debug Message Output ON/OFF Selection** 0 = UART 0 debug message output ON and pin PF[12:11] used as the UART0 functionality. 1 = UART 0 debug message output OFF and pin PF[12:11] used as the GPIO functionality.	URDBGON (SYS_PWRON[5])

PG[7:6]	SW2.8/ SW2.7	**NAND Flash Page Size selection** 00 = NAND Flash page size is 2KB. 01 = NAND Flash page size is 4KB. 10 = NAND Flash page size is 8KB. 11 = Ignore Power-On Setting.	NPAGESEL (SYS_PWRON[7:6])
PG[9:8]	SW2.10/ SW2.9	**Miscellaneous Configuration** When BTSSEL = 01, Boot from SD/eMMC, the MISCCFG defines the SD0/eMMC0 or SD1/eMMC1 used as the booting source. 11 = SD0/eMMC0 (GPC group) used as the booting source. Others = SD1/eMMC1 (GPF group) used as the booting source. When BTSSEL = 10, Boot from NAND Flash, the MISCCFG defines the ECC type. 00 = No ECC 01 = ECC is BCH T12 10 = ECC is BCH T24 11 = Ignore power-on setting When BTSEL = 11, Boot from SPI Flash, the MISCCFG defines the SPI Flash type and data width. 00 = SPI-NAND Flash with 1-bit mode. 01 = SPI-NAND Flash with 4-bit mode. 10 = SPI-NOR Flash with 4-bit mode. 11 = SPI-NOR Flash with 1-bit mode.	MISCCFG (SYS_PWRON[9:8])

Table 1 Power On Setting

1.1 Boot from NAND

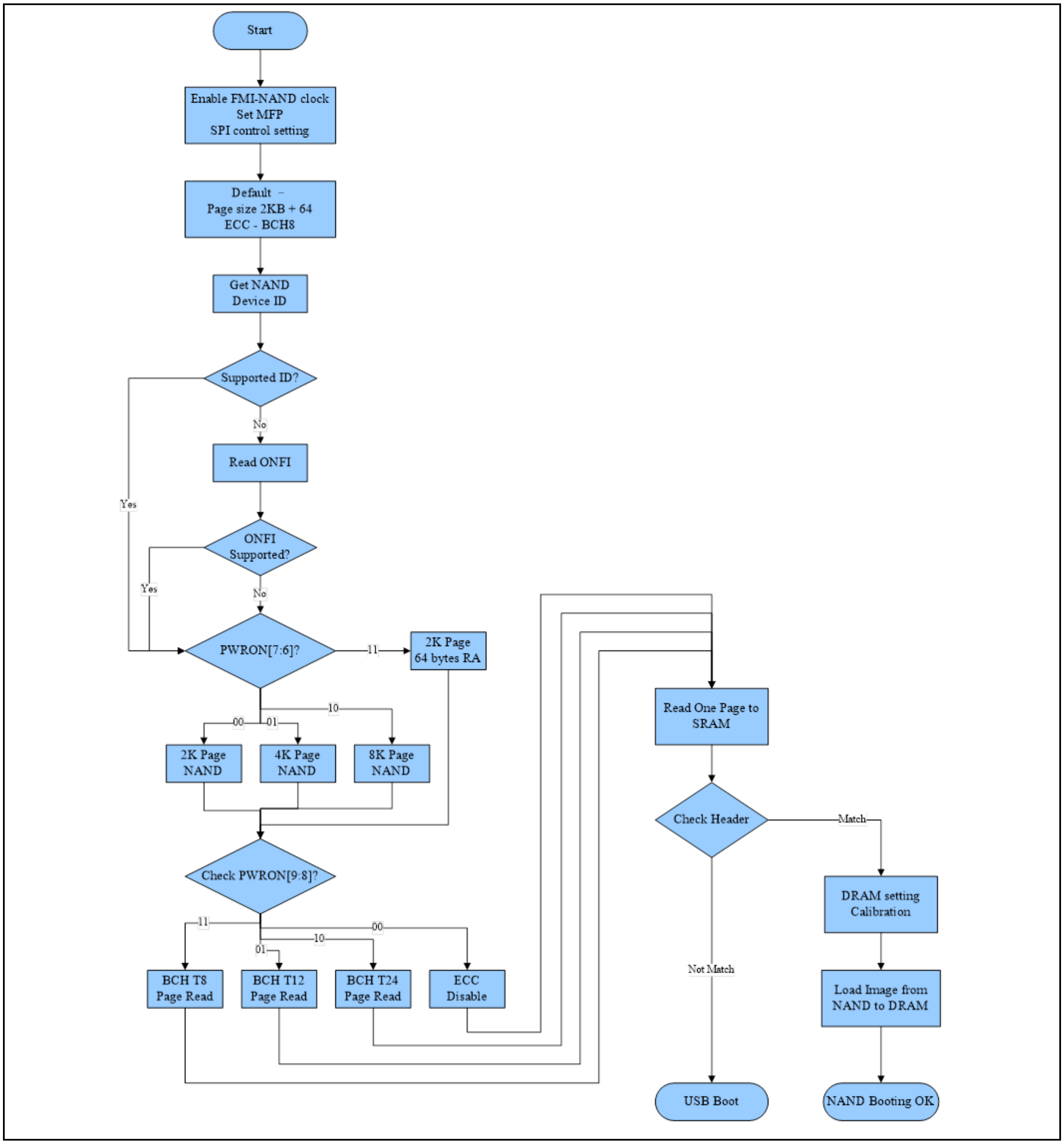


Figure 1-1 Nand Startup Flow

The NAND boot start flow detects PWRON[7:6] to determine page size and PWRON[9:8] to determine NAND Flash ECC type, and then reads the block 0 in NAND Flash to determine boot code marker is correct. Boot code marker is refer to Figure 1-1. If boot code marker cannot be found in the block0, block1, block2 and block3, it is skipped from “Check Header” states to USB boot. When the boot code marker detects that it is correct, it will do DDR initialization. It copies just uboot image from NAND to DDR, and executes Loader image. Note that the loader image of total size plus DDR parameter of total size is equal to or smaller than one block size.

That is “uBoot Image + Header + DDR Parameter ≤ Block Size”.

1.2 Boot from SPI

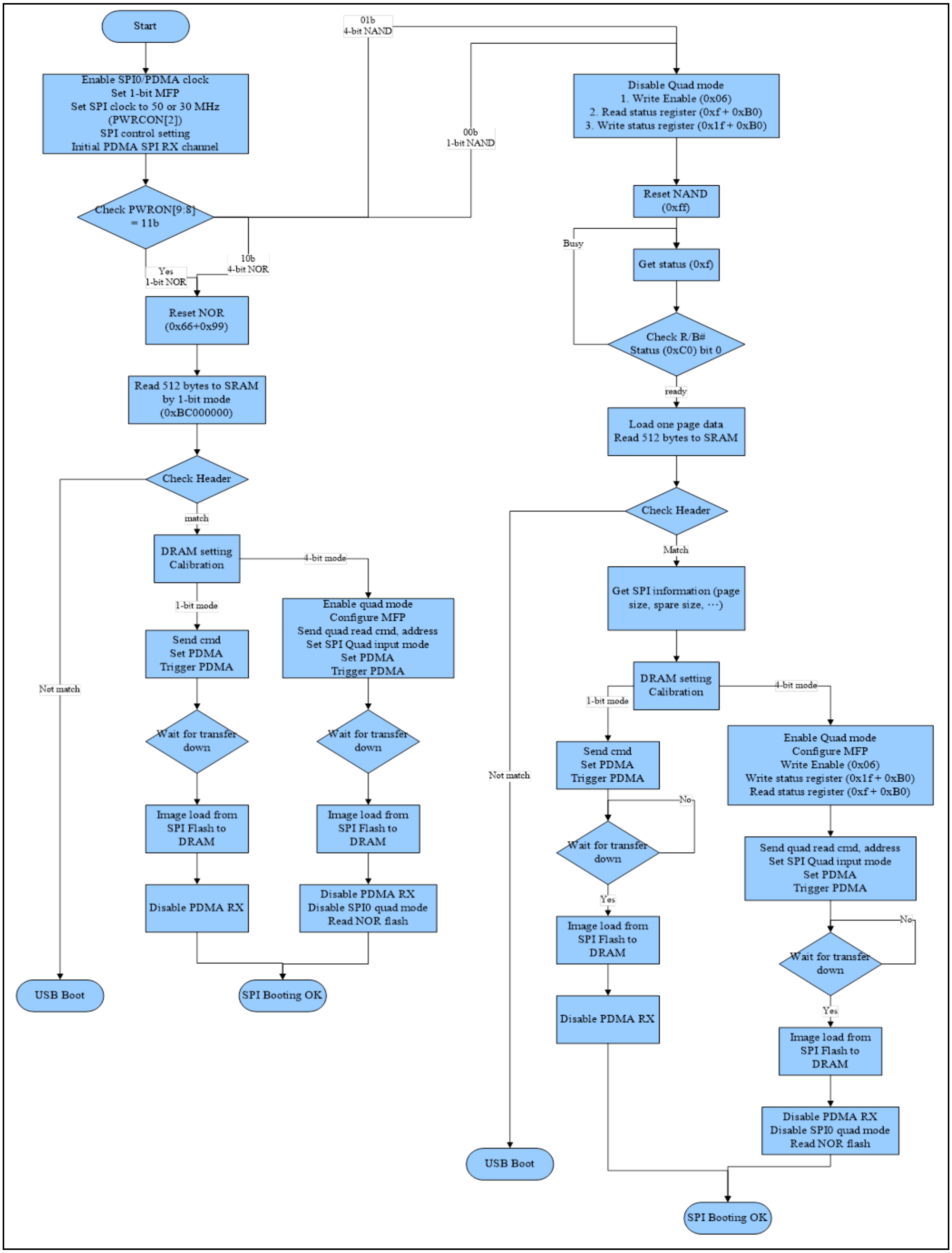


Figure 1-2 SPI Startup Flow

The SPI boot flow reads address 0x0000000 from SPI Flash to check if the boot code marker is correct. For the boot code marker, refer to Figure 1-2. If the boot code marker cannot be found in the SPI Flash, it is skipped from “Check Header” states to USB boot. When it is detected that the boot code marker is correct, it will do DDR initialization. It copies just uboot image from SPI to DDR, and executes uboot image.

1.3 Boot from eMMC/SD

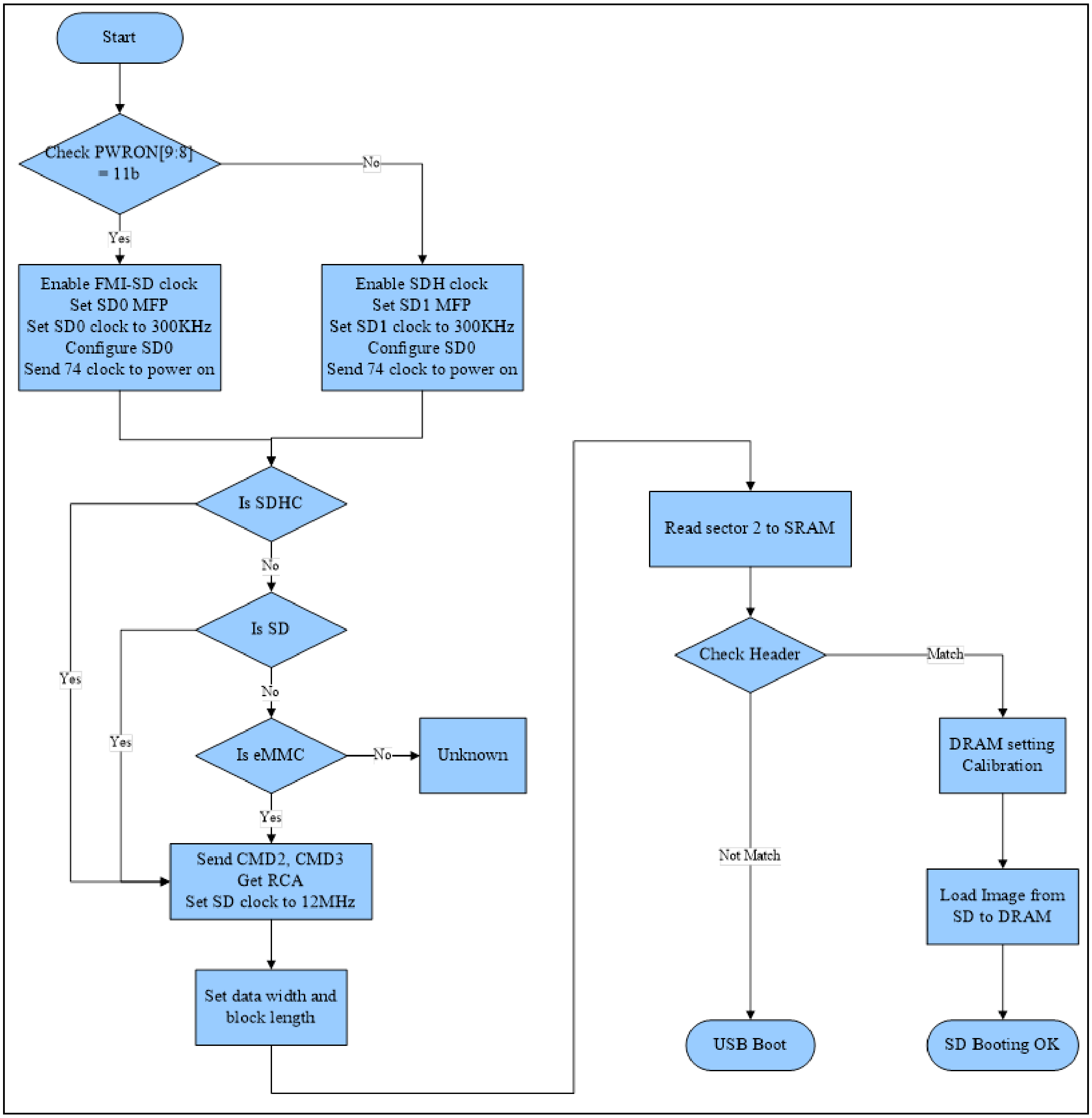


Figure 1-3 eMMC/SD Startup Flow

The eMMC/SD boot flow detects PWRON[9:8] to select eMMC0/SD0 or eMMC1/SD1. Read address 0x0000400 from eMMC/SD to check boot code marker is correct. Boot code marker is refer to Figure 1-3. If boot code marker cannot be found in the eMMC/SD, then it is skipping from “Check Header” states to USB boot. When boot code marker detects correct and then do DDR initialization. It copy just uboot image from eMMC/SD to DDR, and execute uboot image.

1.4 USB ISP

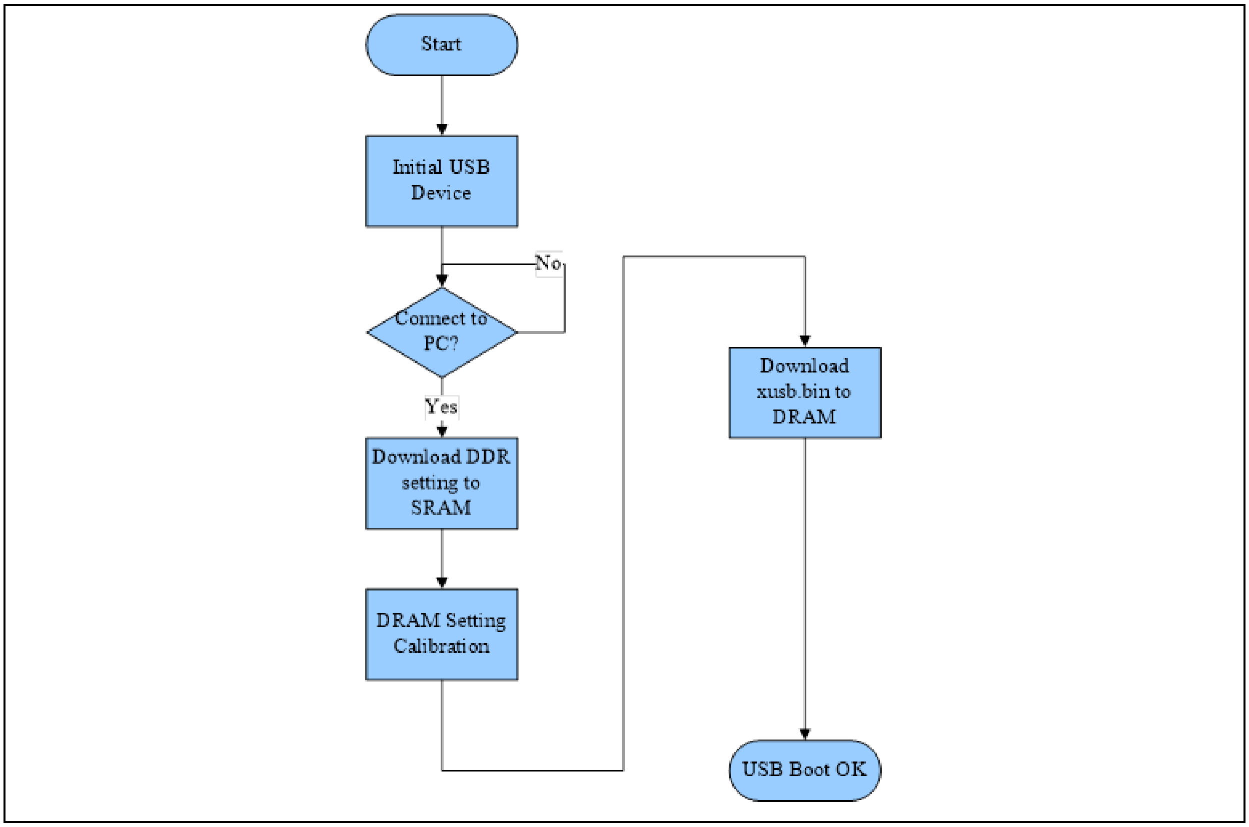


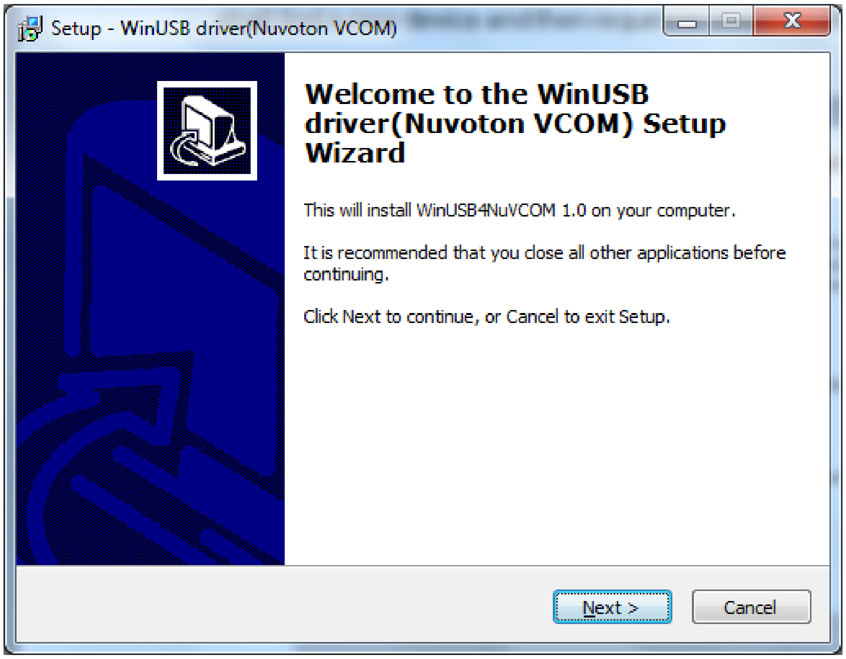
Figure 1-4 USB Startup Flow

The first step of USB boot flow is to wait DDR parameter to complete initialization. The NuWriter is connected to the NUC980 series MPU and transmits DDR parameter. After completion, the NuWriter transmits xusb.bin to the NUC980 series MPU to execute.

2 INSTALL NUWRITER DRIVER

It is required to install a device driver on a PC when using the NuWriter. The installation steps on Windows 7 are listed below

Step1: Double click the "*WinUSB4NuVCOM.exe*" to install the driver. The "WinUSB4NuVCOM.exe" can be found in the "Tool" directory. Plug NUC980 Series MPU EVB board to an USB port on a PC, the Windows shall find a new device and then request to install its driver (Figure 2 1).



Setup - WinUSB driver(Nuvoton VCOM)

Welcome to the WinUSB driver(Nuvoton VCOM) Setup Wizard

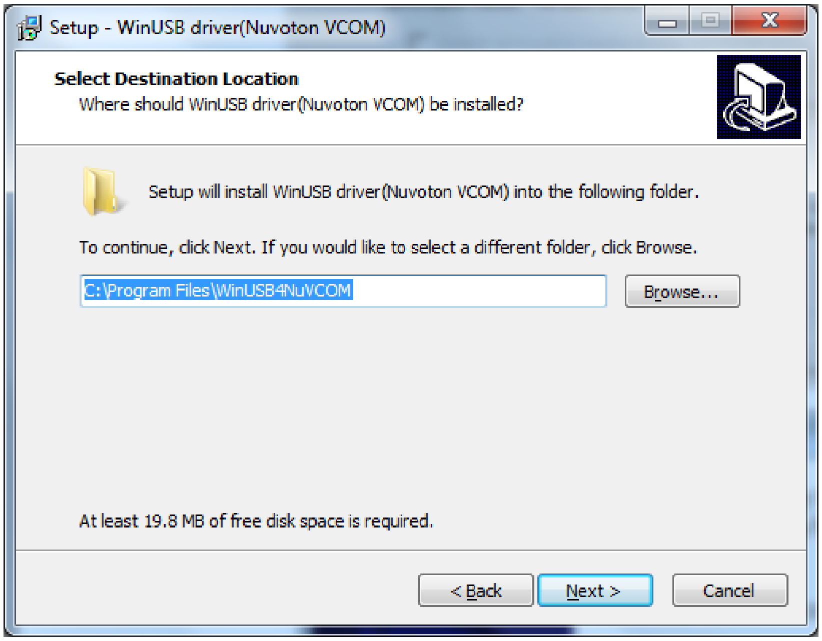
This will install WinUSB4NuVCOM 1.0 on your computer.

It is recommended that you close all other applications before continuing.

Click Next to continue, or Cancel to exit Setup.

Figure 2-1 WinUSB4NuVCOM Driver Setup (1)

Step2: Click "**Next**". The software installation will ask you how to install the driver as shown below.



Setup - WinUSB driver(Nuvoton VCOM)

Select Destination Location

Where should WinUSB driver(Nuvoton VCOM) be installed?

Setup will install WinUSB driver(Nuvoton VCOM) into the following folder.

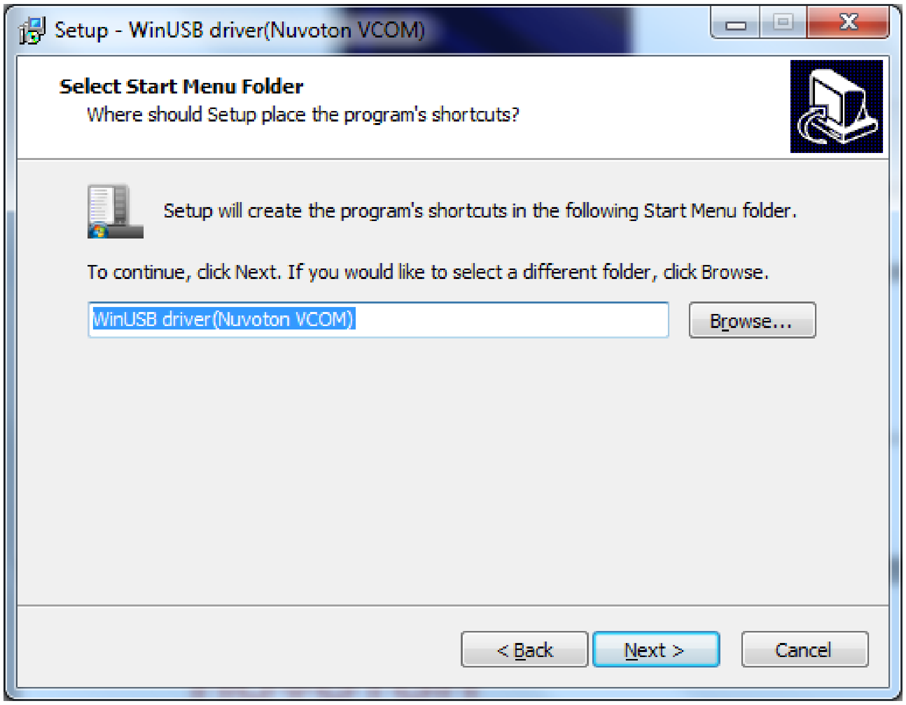
To continue, click Next. If you would like to select a different folder, click Browse.

C:\Program Files\WinUSB4NuVCOM

At least 19.8 MB of free disk space is required.

Figure 2-2 WinUSB4NuVCOM Driver Setup (2)

Step3: Select a folder to install the driver, and then click the "**Next**" button.



Setup - WinUSB driver(Nuvoton VCOM)

Select Start Menu Folder
Where should Setup place the program's shortcuts?

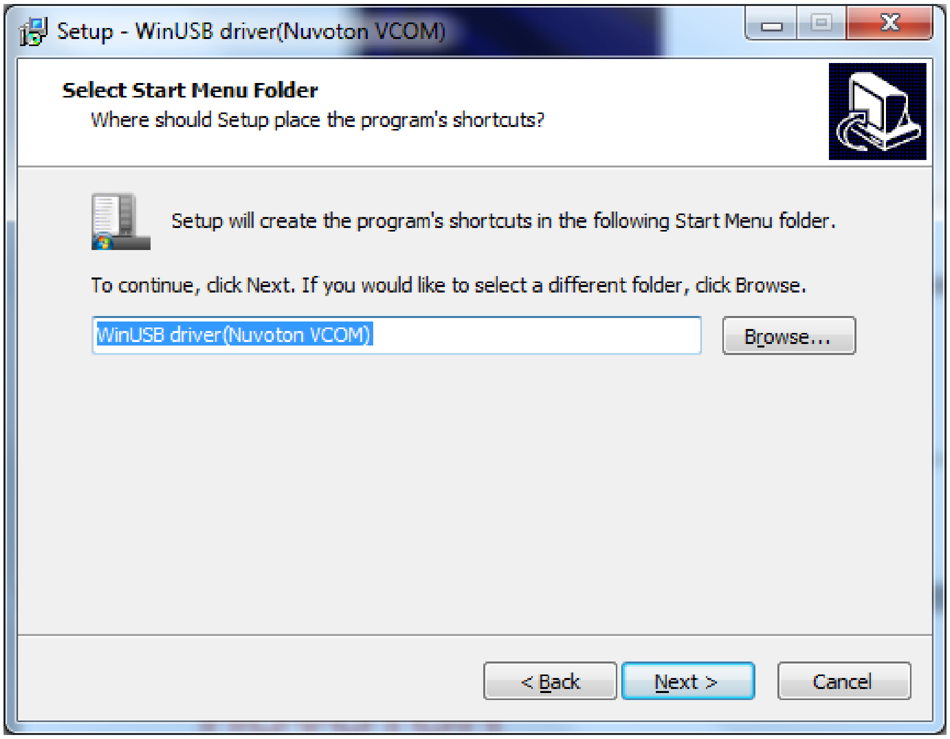
Setup will create the program's shortcuts in the following Start Menu folder.

To continue, click Next. If you would like to select a different folder, click Browse.

WinUSB driver(Nuvoton VCOM)

Figure 2-3 WinUSB4NuVCOM Driver Setup (3)

Step4: Specify the folder name in Start Menu, and then click the "**Next**" button.



Setup - WinUSB driver(Nuvoton VCOM)

Select Start Menu Folder
Where should Setup place the program's shortcuts?

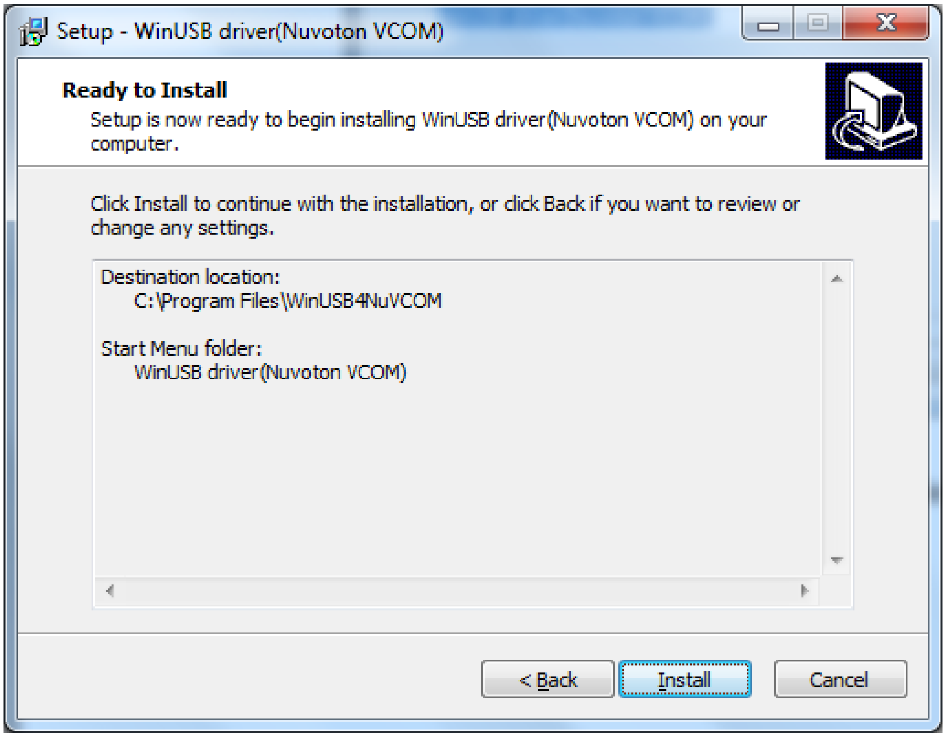
Setup will create the program's shortcuts in the following Start Menu folder.

To continue, click Next. If you would like to select a different folder, click Browse.

WinUSB driver(Nuvoton VCOM)

Figure 2-4 WinUSB4NuVCOM Driver Setup (4)

Step5: Click the "**Install**" button to install driver.



Ready to Install

Setup is now ready to begin installing WinUSB driver(Nuvoton VCOM) on your computer.

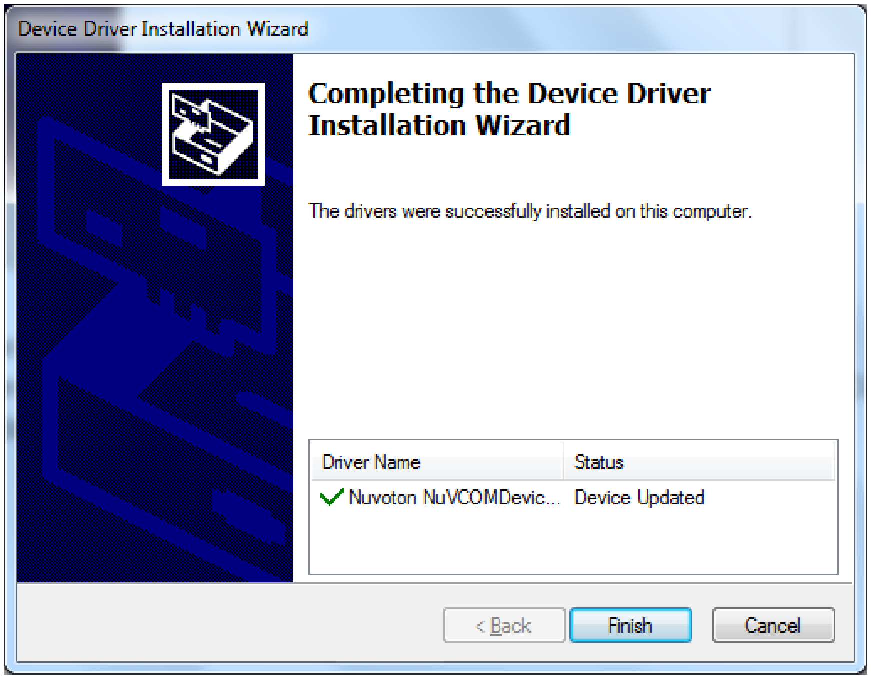
Click Install to continue with the installation, or click Back if you want to review or change any settings.

Destination location:
C:\Program Files\WinUSB4NuVCOM

Start Menu folder:
WinUSB driver(Nuvoton VCOM)

Figure 2-5 WinUSB4NuVCOM Driver Setup (5)

Step6: After driver installation is compete, Device Driver Installation Wizard shows that the device is ready to use. Click the “**Finish**” button.



Completing the Device Driver Installation Wizard

The drivers were successfully installed on this computer.

Driver Name	Status
✓ Nuvoton NuVCOMDevic...	Device Updated

Figure 2-6 WinUSB4NuVCOM Driver Setup (6)

Step7: After driver installation is complete. Users can find the WinUSB driver (Nuvoton VCOM) through “Device Manager”.

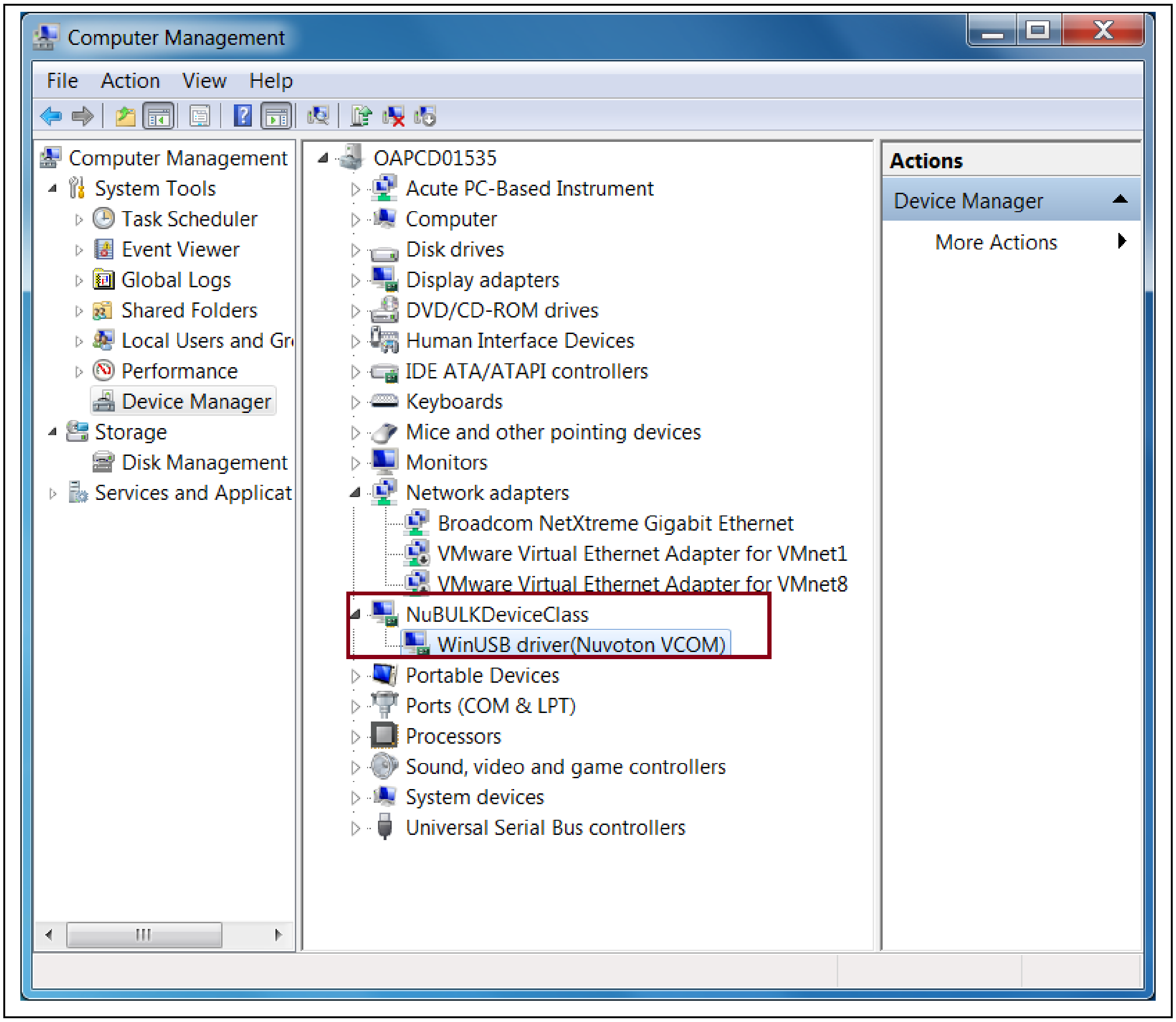


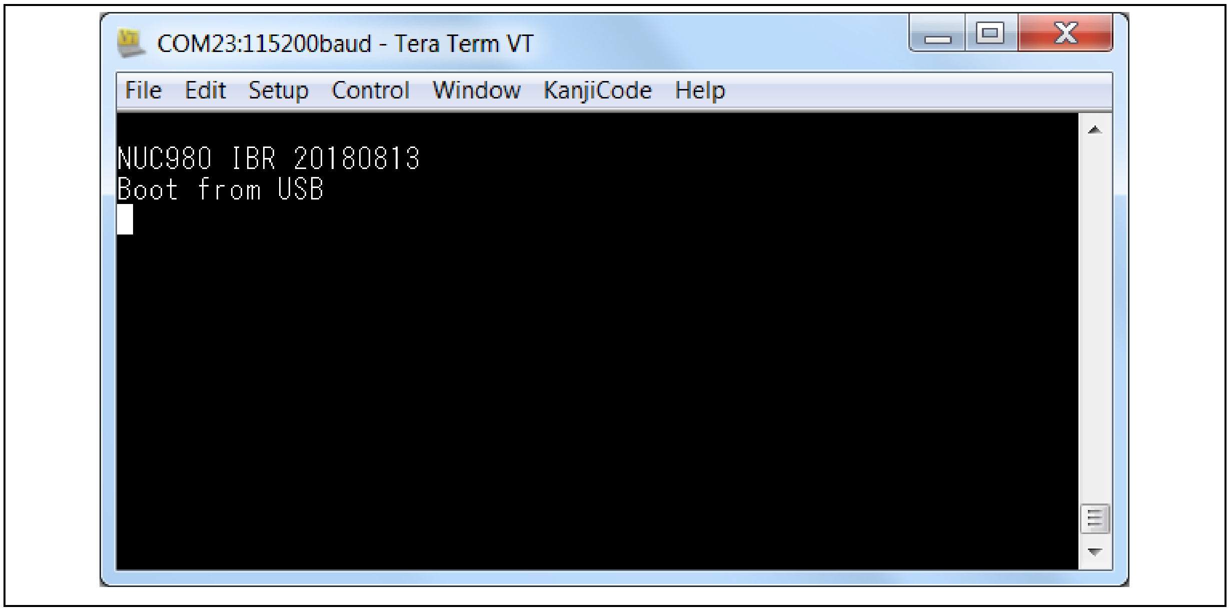
Figure 2-7 Nuwriter VCOM Device

3 USB ISP MODE

The NUC980 series selects the boot method by Power-on Setting. The Power-on Setting can be set by the DIP switch SW2.1~SW2.10 on the development board. For example, if user selected USB ISP mode, the DIP switches SW2.1 and SW2.2 on the development board are all set to On. For other boot settings, please refer to the following table:

Power-on Setting	SW2.2	SW2.1
USB ISP	On	On
Boot from eMMC/SD	On	Off
Boot from NAND	Off	On
Boot from SPI	Off	Off

The Power-on Setting DIP switches SW2.1 and SW2.2 on the development board are set to USB ISP mode. Users can see console message “Boot from USB” (Figure 3 1) in the Tera Term terminal.

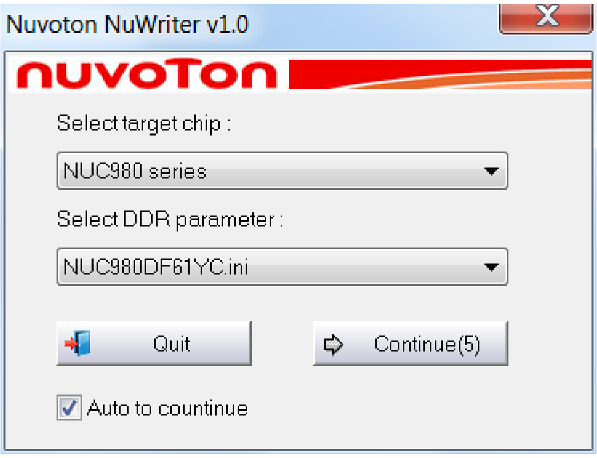

COM23:115200baud - Tera Term VT
File Edit Setup Control Window KanjiCode Help

```
NUC980 IBR 20180813
Boot from USB
```

Figure 3-1 NuWriter USB ISP Mode

4 USER INTERFACE AND FUNCTION INTRODUCTION

Open the NuWriter tool on a PC and double click “*NuWriter.exe*”. Select the target chip (NUC980 series) and select the DDR initialization parameters according to the chip type (*NUC980DF61YC.ini*) (Figure 4 1). After select DDR parameter, click “**Continue**” to use NuWriter tool. When the Nuwriter connection is successful, the user can see the information of all the storage devices on the development board and display “**`Finish get INFO`**” (Figure 4-2) on the terminal to start Nuwriter tool. Note that the tool cannot work if the “WinUSB4NuVCOM” driver is not found.

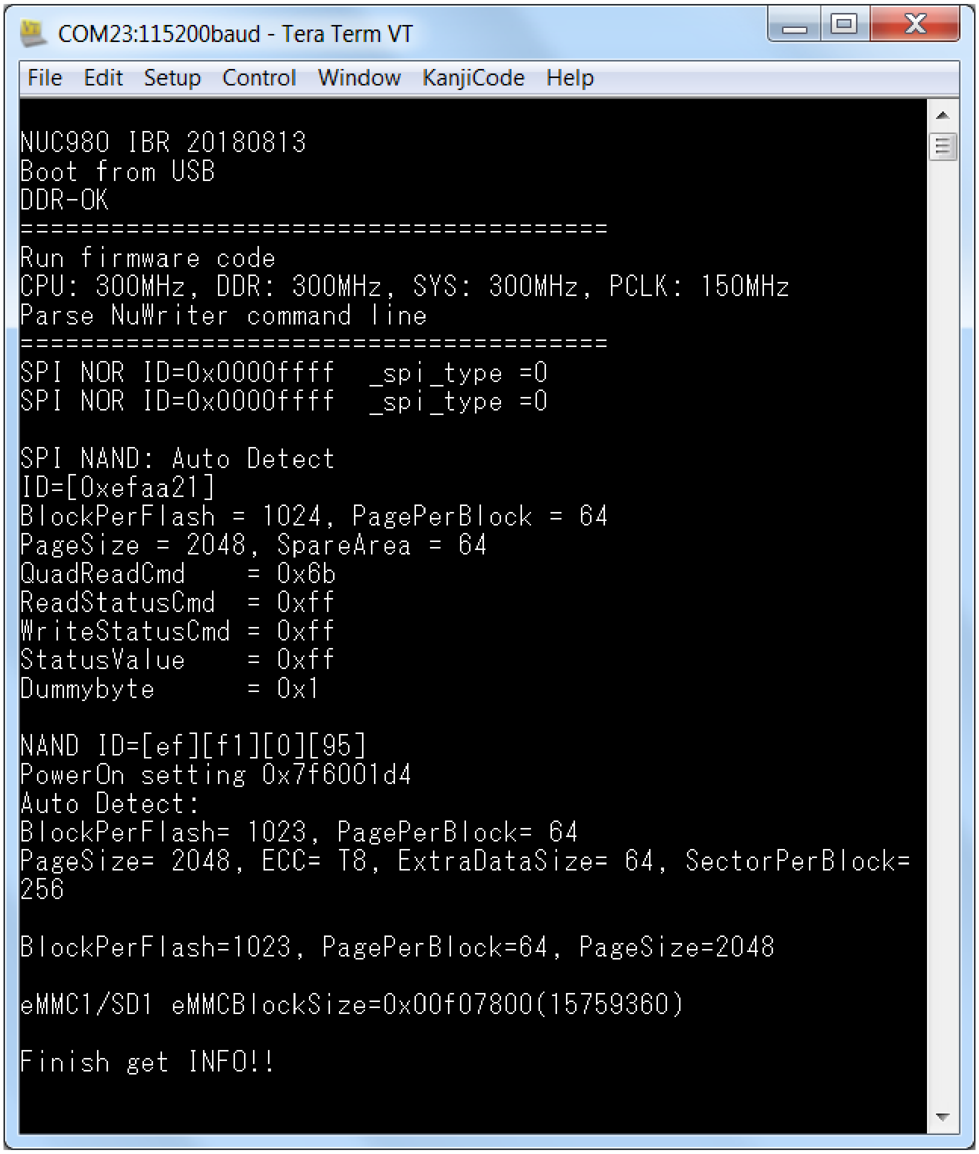


Select target chip : NUC980 series

Select DDR parameter : NUC980DF61YC.ini

☑ Auto to countinue

Figure 4-1 NuWriter Set Chip/DDR


COM23:115200baud - Tera Term VT
File Edit Setup Control Window KanjiCode Help

```
NUC980 IBR 20180813
Boot from USB
DDR-OK
=======================================
Run firmware code
CPU: 300MHz, DDR: 300MHz, SYS: 300MHz, PCLK: 150MHz
Parse NuWriter command line
=======================================
SPI NOR ID=0x0000ffff   _spi_type =0
SPI NOR ID=0x0000ffff   _spi_type =0

SPI NAND: Auto Detect
ID=[0xefaa21]
BlockPerFlash = 1024, PagePerBlock = 64
PageSize = 2048, SpareArea = 64
QuadReadCmd     = 0x6b
ReadStatusCmd   = 0xff
WriteStatusCmd  = 0xff
StatusValue     = 0xff
Dummybyte       = 0x1

NAND ID=[ef][f1][0][95]
PowerOn setting 0x7f6001d4
Auto Detect:
BlockPerFlash= 1023, PagePerBlock= 64
PageSize= 2048, ECC= T8, ExtraDataSize= 64, SectorPerBlock=
256

BlockPerFlash=1023, PagePerBlock=64, PageSize=2048

eMMC1/SD1 eMMCBlockSize=0x00f07800(15759360)

Finish get INFO!!
```

Figure 4-2 NuWriter Console Message

4.1 Operating Window

The user can distinguish PC Tool version number, FW version number and DDR parameter version number from the Nuwriter tool window (Figure 4 3). The software operation window consists of the Flash Type drop-down menu, the Connection State button, and the Function Control window (Figure 4 4). As shown in below, Function Control window for setting parameters according to the Flash type. The Connection State button is used to detect Nuwriter's USB device and check USB driver is installed successfully.

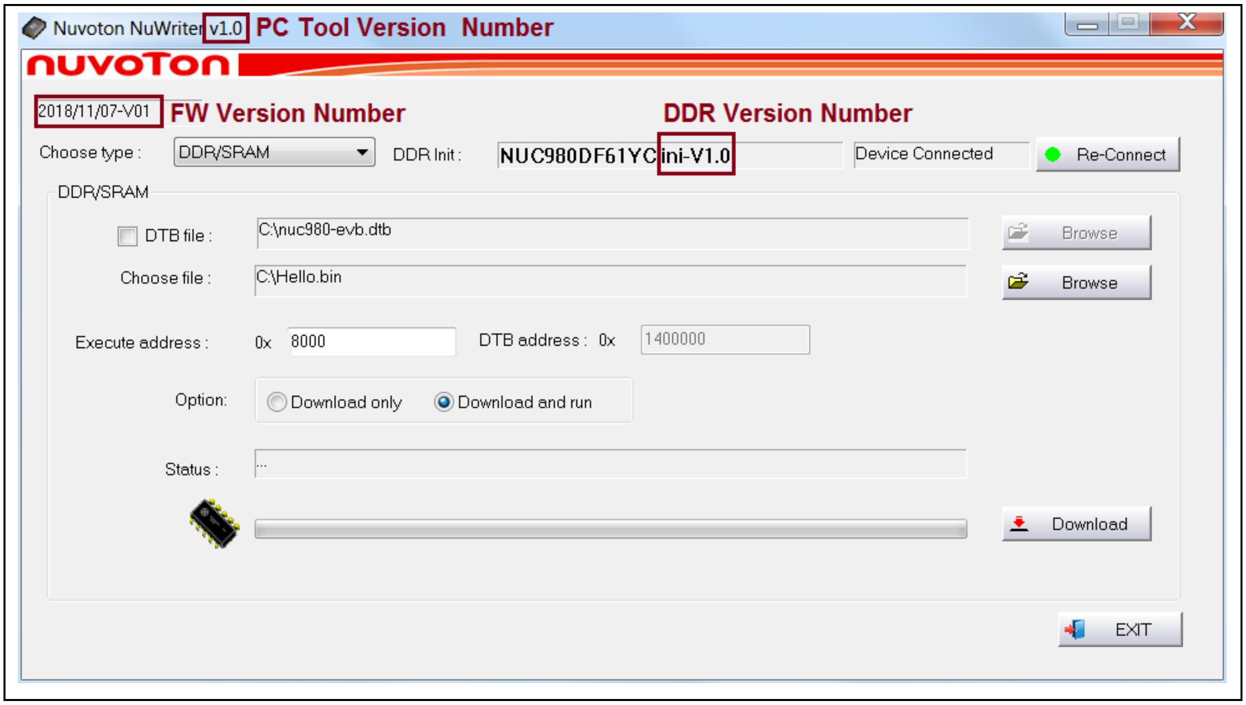


Figure 4-3 Nuwriter Version Number

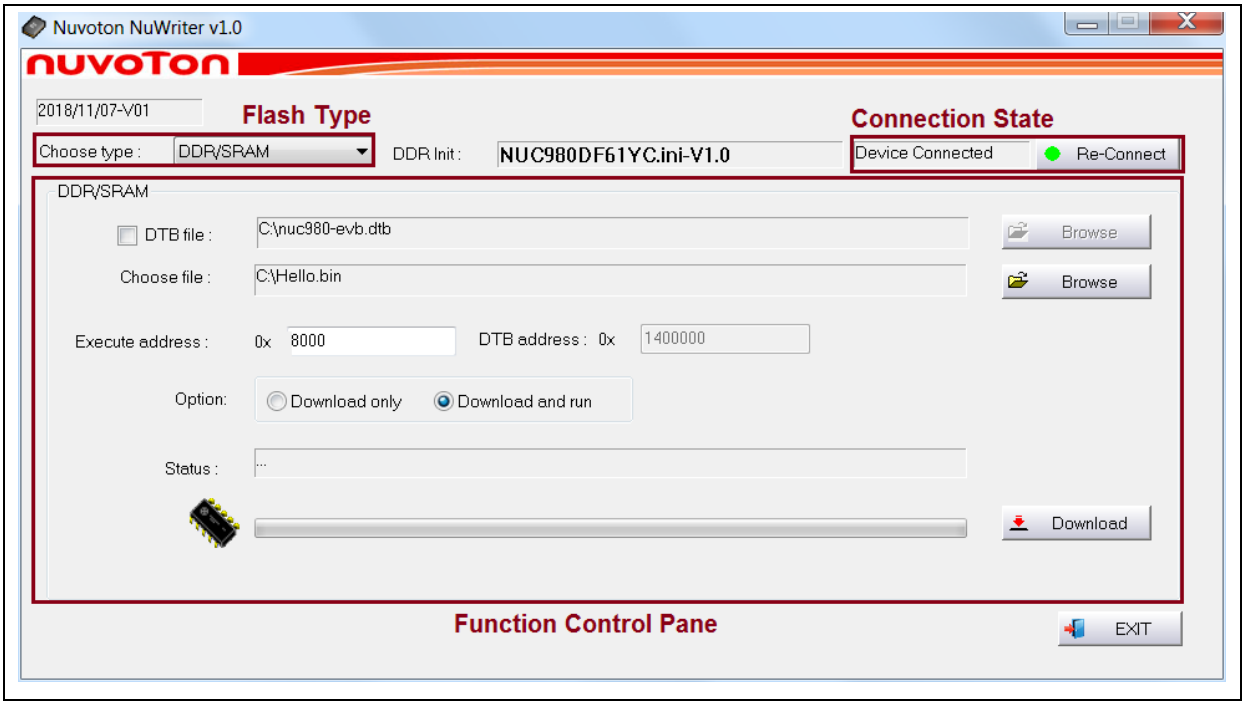
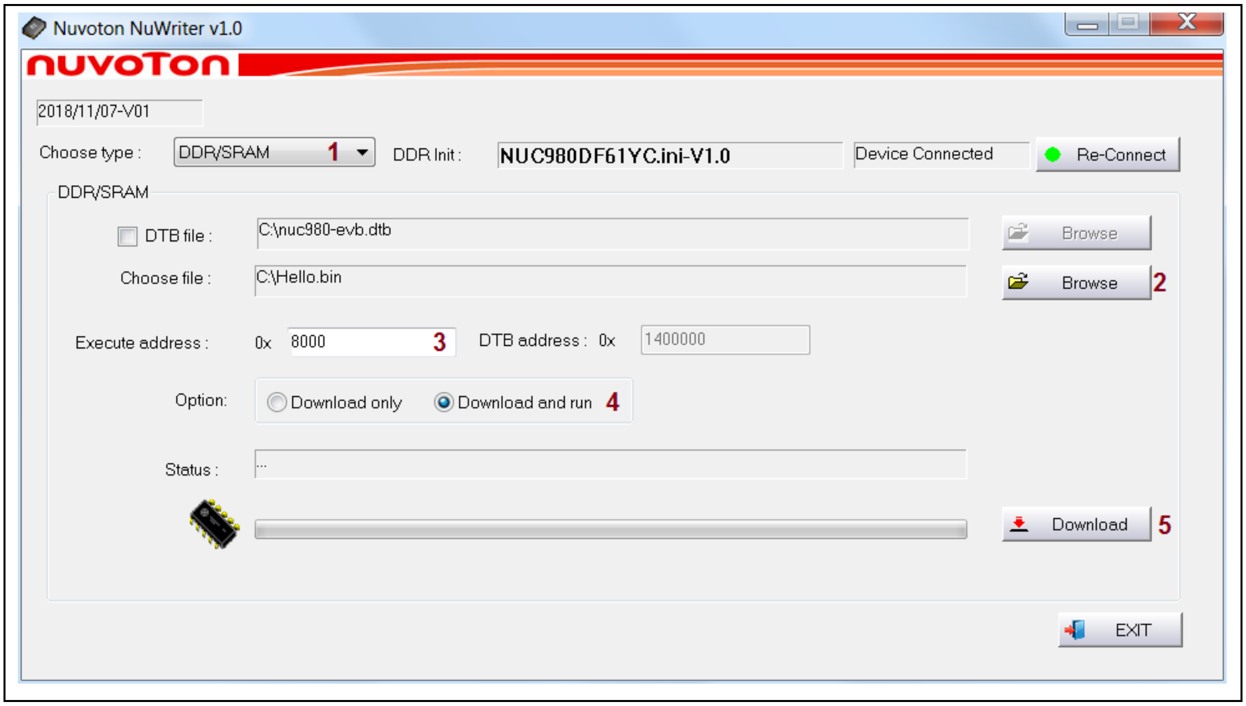


Figure 4-4 Nuwriter Operating Window

4.2 DDR/SRAM Mode

The NUC980 chip provides a Linux kernel and a non-OS BSP. Users can program to the memory by DDR/SRAM mode. In addition, the DTB file for Linux kernel can also be programmed on this mode.

4.2.1 Operation Steps



Choose type : DDR/SRAM

DDR Init : NUC980DF61YC.ini-V1.0

Device Connected

☐ DTB file : C:\nuc980-evb.dtb

Choose file : C:\Hello.bin

Execute address : 0x 8000

DTB address : 0x 1400000

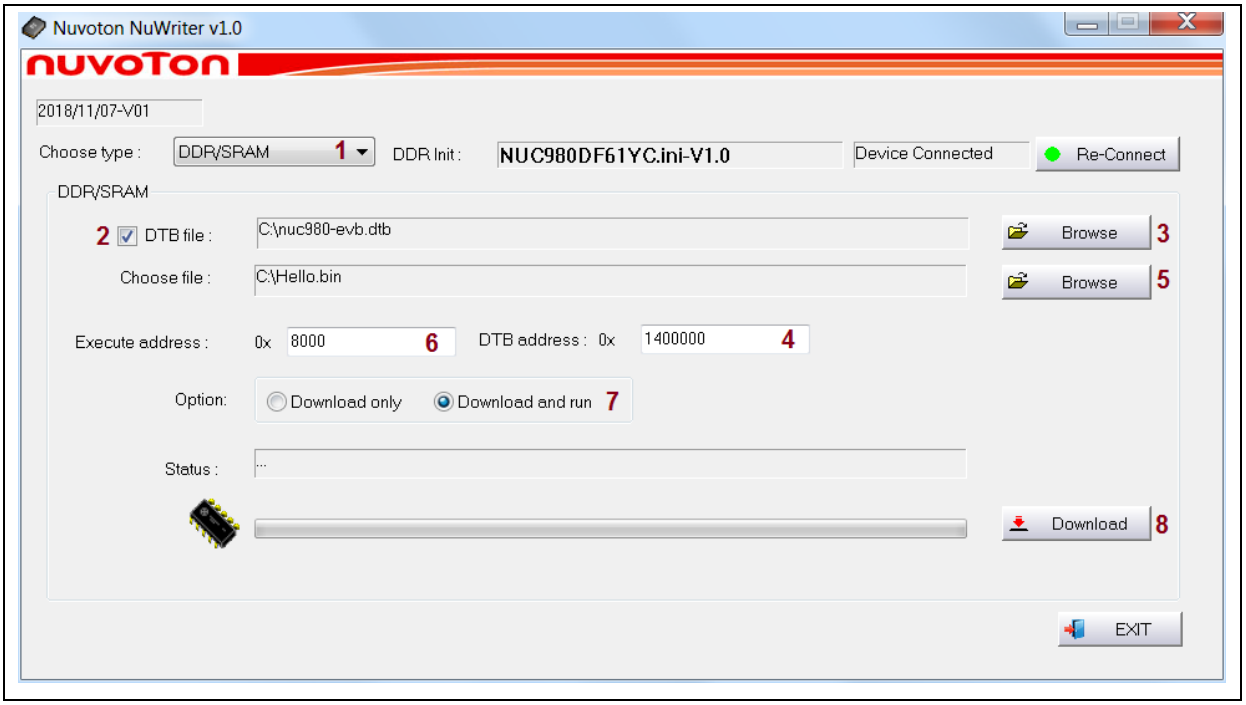
Option: ○ Download only ● Download and run

Status : ...

Figure 4-5 NuWriter – DDR/SDRAM Mode (1)

The DDR/SRAM mode is used to download an image to DDR or SDRAM for debugging purpose, as shown in Figure 4-5. The steps below explain how to use DDR/SRAM mode.

1. Select “**DDR/SRAM**”.
2. Click the “**Browse**” button to load the image file.
3. Input the image execution address. **Note:** All of DDR types, the last 1MB is not available.
4. Select “**Download only**” or ”**Download and run**”.
5. Click the “**Download**” button to program the DDR or SDRAM.


Nuvoton NuWriter v1.0

2018/11/07-V01

Choose type : DDR/SRAM

DDR Init : NUC980DF61YC.ini-V1.0

Device Connected

DDR/SRAM

☑ DTB file : C:\nuc980-evb.dtb

Choose file : C:\Hello.bin

Execute address : 0x 8000

DTB address : 0x 1400000

Option: ○ Download only ● Download and run

Status : ...

Figure 4-6 NuWriter – DDR/SDRAM Mode (2)

A device tree file(*.dtb) is used to Linux kernel BSP. Users can download the Image file and the DDB file to the DDR or SDRAM memory. The detailed steps are listed below.

1. Select “**DDR/SRAM**”.
2. Use check box “**DTB file**” to download device tree file(*.dtb).
3. Click the “**Browse**” button to load the device tree file(*.dtb).
4. Enter the device tree file execution address. Note: Device tree file execution address cannot be covered by image file.Browse the image.
5. Enter the image execution address. Note: All of DDR types, the last 1MB is not available.
6. Select “**Download only**” or ”**Download and run**”.
7. Click the “**Download**” button to program the DDR or SDRAM.

4.3 NAND Mode

This mode can write a new image to NAND Flash and specify the type of the image. These types can be recognized by bootloader or Linux. The Image type is set as uBoot, Data, Environment or Pack.

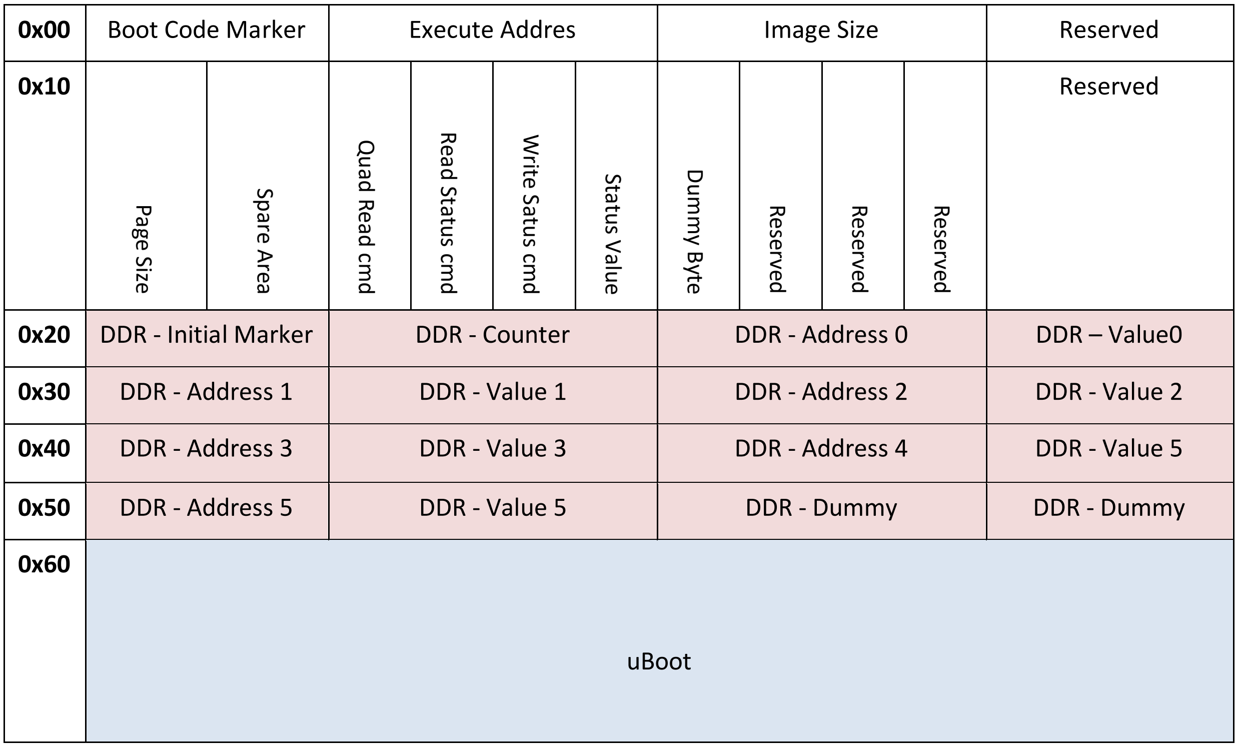
4.3.1 Image Type

4.3.1.1 Loader Type

For the Linux system, Loader Type is used to boot the Linux kernel. To compile NUC980 U-Boot to get Main U-Boot and SPL U-Boot. The Main U-Boot is a fully featured version of U-Boot. The SPL U-Boot is a small binary, it will move Main U-Boot into DDR execution. The SPL U-Boot is only for NAND/SPI NAND boot. SPI or eMMC/SD boot need Main U-Boot only. The default link address of SPL U-Boot is 0x200. The Main U-Boot default link address is 0xE00000.

For Non-OS system, when NUC980 series is powered on, image of Loader type is a first execution program. Users can use Keil compiler to generate binary file as Loader type image. The Loader Type image consists of the 32 bytes header, DDR parameters and binary file. The format is shown in Figure 4-7.

	0x0	0x4	0x8	0xC



<table>
<tr><td><b>0x00</b></td><td colspan="2">Boot Code Marker</td><td colspan="4">Execute Addres</td><td colspan="4">Image Size</td><td>Reserved</td></tr>
<tr><td><b>0x10</b></td><td>Page Size</td><td>Spare Area</td><td>Quad Read cmd</td><td>Read Status cmd</td><td>Write Satus cmd</td><td>Status Value</td><td>Dummy Byte</td><td>Reserved</td><td>Reserved</td><td>Reserved</td><td>Reserved</td></tr>
<tr><td><b>0x20</b></td><td colspan="2">DDR - Initial Marker</td><td colspan="4">DDR - Counter</td><td colspan="4">DDR - Address 0</td><td>DDR – Value0</td></tr>
<tr><td><b>0x30</b></td><td colspan="2">DDR - Address 1</td><td colspan="4">DDR - Value 1</td><td colspan="4">DDR - Address 2</td><td>DDR - Value 2</td></tr>
<tr><td><b>0x40</b></td><td colspan="2">DDR - Address 3</td><td colspan="4">DDR - Value 3</td><td colspan="4">DDR - Address 4</td><td>DDR - Value 5</td></tr>
<tr><td><b>0x50</b></td><td colspan="2">DDR - Address 5</td><td colspan="4">DDR - Value 5</td><td colspan="4">DDR - Dummy</td><td>DDR - Dummy</td></tr>
<tr><td><b>0x60</b></td><td colspan="11">uBoot</td></tr>
</table>

Figure 4-7 Boot Code Header

Boot Code Marker = {0x20,'T','V','N'}.

Execute Address = Copy uboot image to { Execute Address } location.

Image Size = {uBoot image length}.

Page Size = {SPI NAND Flash Page size}.

Spare Area = { SPI NAND Flash Spare area size}.

Quad Read cmd = { SPI NAND Flash Quad Read Command}.

Read Status cmd = { SPI NAND Flash Read Status Command}.

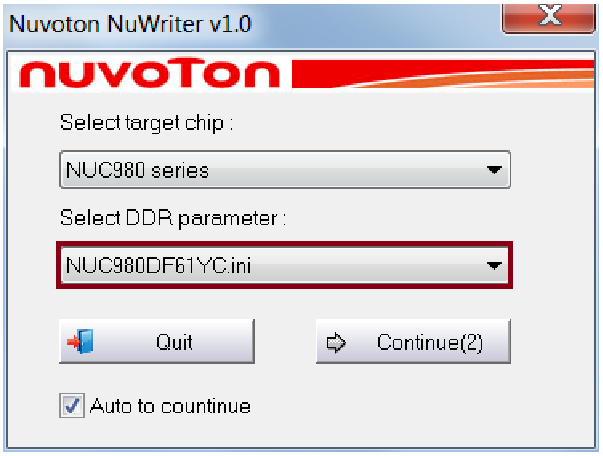
Write Status cmd = { SPI NAND Flash Write Status Command}.

Status Value = { SPI NAND Flash Status value}.

Dummy Byte = { SPI NAND Flash Dummy Byte size}.

DDR - Initial Marker = {0x55AA55AA}.

DDR - Counter = DDR parameter length.



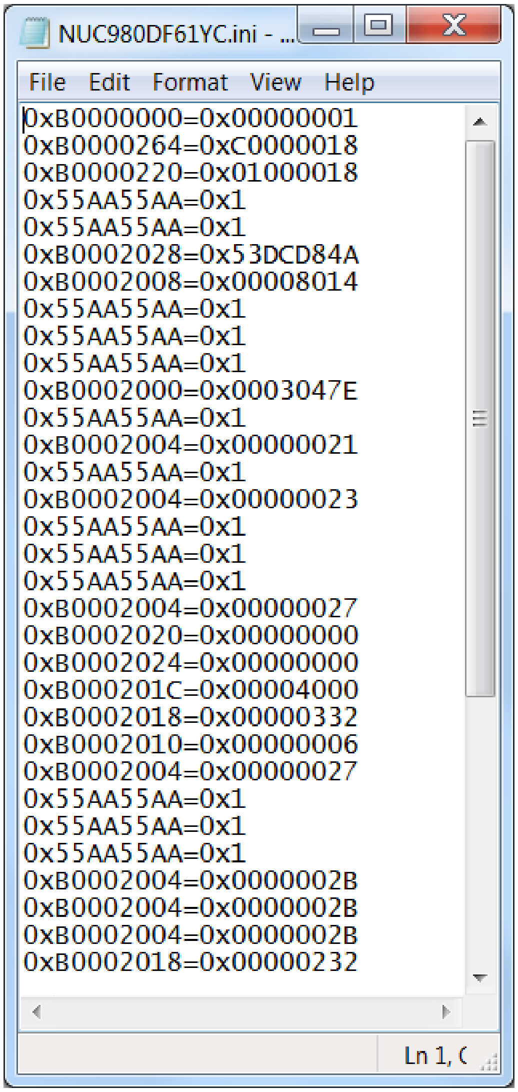
Select target chip : NUC980 series

Select DDR parameter : NUC980DF61YC.ini

☑ Auto to countinue

Figure 4-8 NuWriter – Select DDR Parameter

There are DDR parameters in *NUC980DF61YC.ini* as shown in Figure 4-9.



```
0xB0000000=0x00000001
0xB0000264=0xC0000018
0xB0000220=0x01000018
0x55AA55AA=0x1
0x55AA55AA=0x1
0xB0002028=0x53DCD84A
0xB0002008=0x00008014
0x55AA55AA=0x1
0x55AA55AA=0x1
0x55AA55AA=0x1
0xB0002000=0x0003047E
0x55AA55AA=0x1
0xB0002004=0x00000021
0x55AA55AA=0x1
0xB0002004=0x00000023
0x55AA55AA=0x1
0x55AA55AA=0x1
0x55AA55AA=0x1
0xB0002004=0x00000027
0xB0002020=0x00000000
0xB0002024=0x00000000
0xB000201C=0x00004000
0xB0002018=0x00000332
0xB0002010=0x00000006
0xB0002004=0x00000027
0x55AA55AA=0x1
0x55AA55AA=0x1
0x55AA55AA=0x1
0xB0002004=0x0000002B
0xB0002004=0x0000002B
0xB0002004=0x0000002B
0xB0002018=0x00000232
```

Figure 4-9 NuWriter –NUC980DF61YC.ini

This section uses *NUC980DF61YC.ini* as an example to describe the meaning of the format. (The actual DDR parameters as shown in Figure 4-9, without “line number”.)

```
line 1 : 0xB0000000=0x00000001
line 2 : 0xB0000264=0xC0000018
line 3 : 0xB0000220=0x01000018
line 4 : 0x55AA55AA=0x1
line 5 : 0x55AA55AA=0x1
line 6 : 0xB0002028=0x53DCD84A
line 7 : 0xB0002008=0x00008014
line 8 : 0x55AA55AA=0x1
line 9 : 0x55AA55AA=0x1
line 10: 0x55AA55AA=0x1
line 11: 0xB0002000=0x0003047E
line 12: 0x55AA55AA=0x1
line 13: 0xB0002004=0x00000021
line 14: 0x55AA55AA=0x1
line 15: 0xB0002004=0x00000023
line 16: 0x55AA55AA=0x1
line 17: 0x55AA55AA=0x1
line 18: 0x55AA55AA=0x1
line 19: 0xB0002004=0x00000027
line 20: 0xB0002020=0x00000000
line 21: 0xB0002024=0x00000000
line 22: 0xB000201C=0x00004000
line 23: 0xB0002018=0x00000332
line 24: 0xB0002010=0x00000006
line 25: 0xB0002004=0x00000027
line 26: 0x55AA55AA=0x1
line 27: 0x55AA55AA=0x1
line 28: 0x55AA55AA=0x1
line 29: 0xB0002004=0x0000002B
line 30: 0xB0002004=0x0000002B
line 31: 0xB0002004=0x0000002B
line 32: 0xB0002018=0x00000232
line 33: 0xB000201C=0x00004781
line 34: 0xB000201C=0x00004401
line 35: 0xB0002004=0x00000020
line 36: 0xB0002034=0x00888828
line 37: 0x55AA55AA=0x1
line 38: 0x55AA55AA=0x1
line 39: 0x55AA55AA=0x1
line 40: 0x55AA55AA=0x1
line 41: 0x55AA55AA=0x1
line 42: 0x55AA55AA=0x1
line 43: 0x55AA55AA=0x1
line 44: 0x55AA55AA=0x1
line 45: 0x55AA55AA=0x1
line 46: 0x55AA55AA=0x1
line 47: 0x55AA55AA=0x1
```

The NUC980DF61YC.ini can be calculated DDR - Counter = 46. The first set(line 1) is the version number of the DDR and DDR - Counter is not included this value.

DDR - Address = {0xB0000000}. DDR – Value = {0x00000001}． The DDR version number is V1.0.

DDR - Address 0 = {0xB0000264}. DDR – Value0 = {0xC0000018}．

DDR - Address 1 = {0xB0000220}. DDR – Value1 = {0xC0000018}．

DDR - Address 2 = {0x55AA55AA}． DDR – Value2 = {0x1}．

DDR - Address 3 = {0x55AA55AA}． DDR – Value3 = {0x1}．

DDR - Address 4 = {0xB0002028}． DDR – Value4 = {0x53DCD84A}．

And so on · · ·

DDR - Address 43 = {0x55AA55AA}． DDR – Value2 = {0x1}．

DDR - Address 44 = {0x55AA55AA}． DDR – Value2 = {0x1}．

DDR - Address 45 = {0x55AA55AA}． DDR – Value2 = {0x1}．

DDR - Dummy = {0x00000000}，DDR parameters stored in the storage must be aligned to 16bytes，the NuWriter will help to align.

uBoot = uboot binary file.

Then burned image of Loader Type into Nand Flash in the block0, block1, Block2 and Block3, as shown in Figure 4-10.

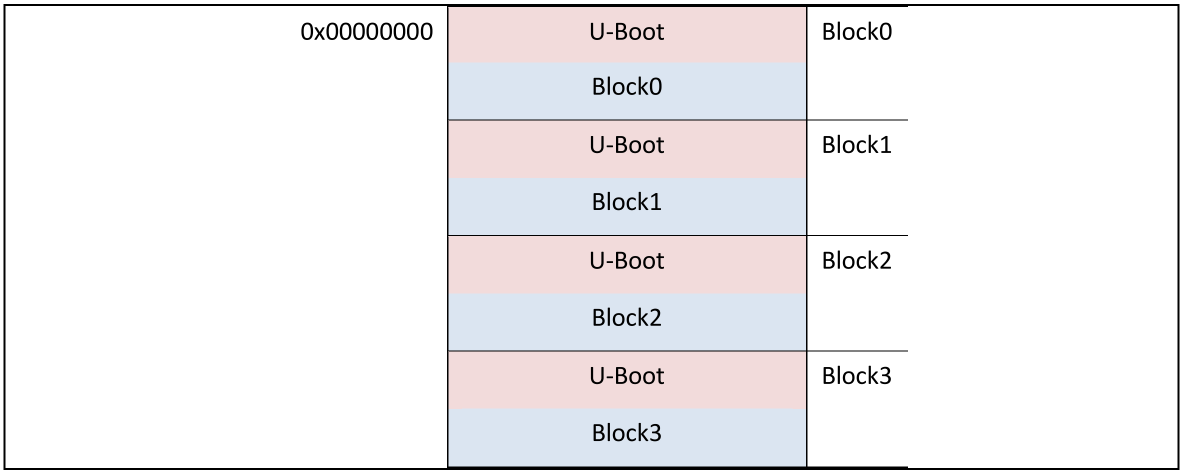


Figure 4-10 NAND Flash Block 0~3

Read out from the NAND Flash are listed below and as shown in Figure 4-11.

Boot Code Marker = {0x20,'T','V','N'}．

Execute Address = 0x00000200．

Image Size = 0x00003F48．

Reserved = 0xFFFFFFFF．

Page Size = 0x800．

Spare Area = 0x40．

Quad Read cmd = 0xFF．

Read Status cmd = 0xFF．

Write Status cmd = 0xFF．

Status Value = 0xFF．

Dummy Byte = 0xFF．

Reserved = 0xFF．

Reserved = 0xFF．

Reserved = 0xFF．

Reserved = 0xFFFFFFFF．

DDR - Initial Marker = 0x55AA55AA．

DDR - Counter = 0x0000002E．

DDR - Address 0 = 0xB0000264． DDR - Value0 = 0xC0000018．

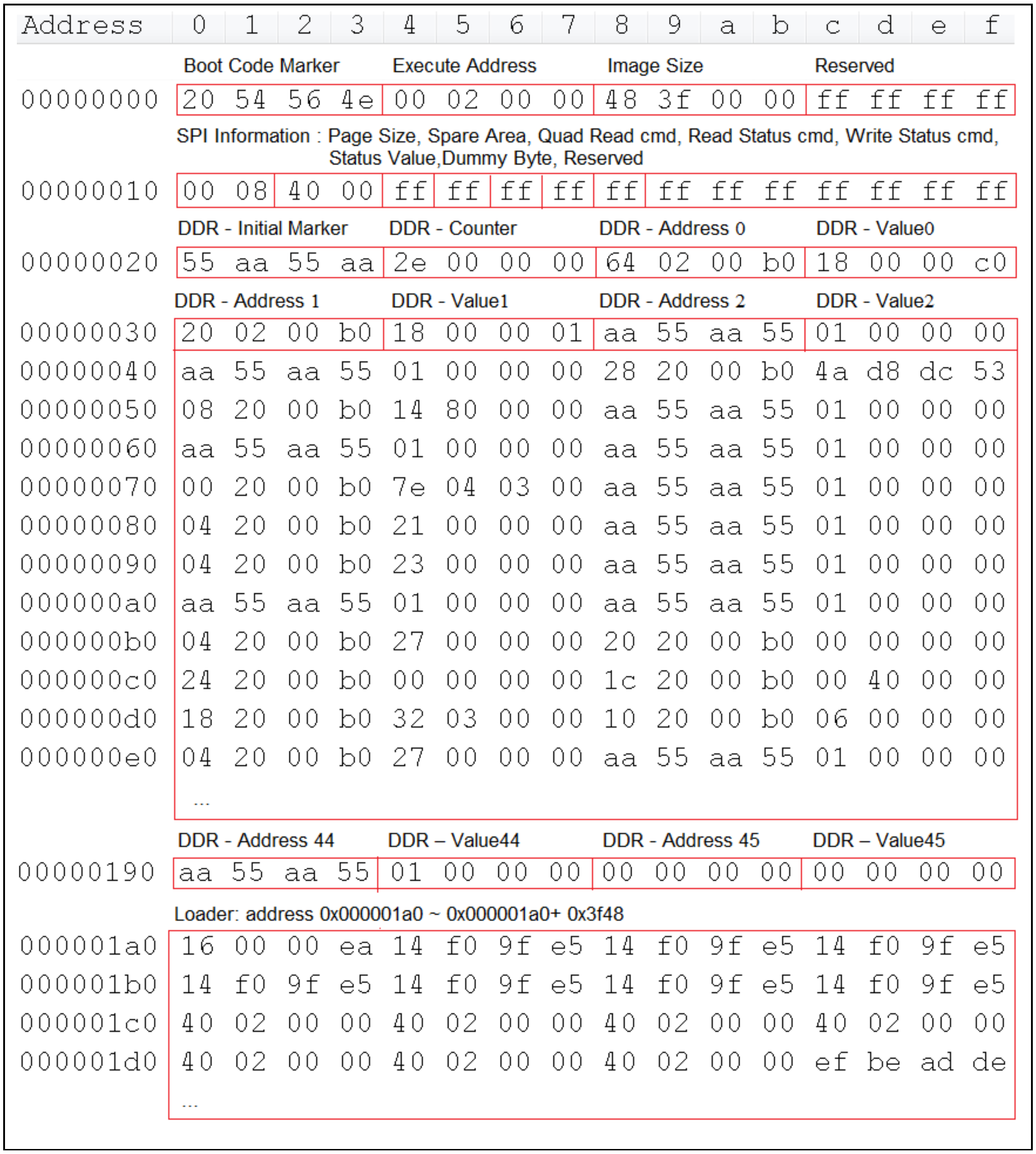
And so on ．．．

DDR - Address 45 = 0x55AA55AA． DDR – Value45 = 0x1．

Loader = address 0x00001a0 ~ 0x00001a0 + 0x00003F48．

The image of Loader Type limitation is listed below.

uBoot Image + Header + DDR parameter ≤ Block Size



```
Address   0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
          Boot Code Marker Execute Address  Image Size   Reserved
00000000  20 54 56 4e     00 02 00 00      48 3f 00 00  ff ff ff ff
          SPI Information : Page Size, Spare Area, Quad Read cmd, Read Status cmd, Write Status cmd,
                            Status Value,Dummy Byte, Reserved
00000010  00 08 40 00 ff ff ff ff ff ff ff ff ff ff ff ff
          DDR - Initial Marker DDR - Counter DDR - Address 0 DDR - Value0
00000020  55 aa 55 aa          2e 00 00 00   64 02 00 b0     18 00 00 c0
          DDR - Address 1 DDR - Value1 DDR - Address 2 DDR - Value2
00000030  20 02 00 b0     18 00 00 01  aa 55 aa 55     01 00 00 00
00000040  aa 55 aa 55 01 00 00 00 28 20 00 b0 4a d8 dc 53
00000050  08 20 00 b0 14 80 00 00 aa 55 aa 55 01 00 00 00
00000060  aa 55 aa 55 01 00 00 00 aa 55 aa 55 01 00 00 00
00000070  00 20 00 b0 7e 04 03 00 aa 55 aa 55 01 00 00 00
00000080  04 20 00 b0 21 00 00 00 aa 55 aa 55 01 00 00 00
00000090  04 20 00 b0 23 00 00 00 aa 55 aa 55 01 00 00 00
000000a0  aa 55 aa 55 01 00 00 00 aa 55 aa 55 01 00 00 00
000000b0  04 20 00 b0 27 00 00 00 20 20 00 b0 00 00 00 00
000000c0  24 20 00 b0 00 00 00 00 1c 20 00 b0 00 40 00 00
000000d0  18 20 00 b0 32 03 00 00 10 20 00 b0 06 00 00 00
000000e0  04 20 00 b0 27 00 00 00 aa 55 aa 55 01 00 00 00
          ...
          DDR - Address 44 DDR – Value44 DDR - Address 45 DDR – Value45
00000190  aa 55 aa 55      01 00 00 00   00 00 00 00      00 00 00 00
          Loader: address 0x000001a0 ~ 0x000001a0+ 0x3f48
000001a0  16 00 00 ea 14 f0 9f e5 14 f0 9f e5 14 f0 9f e5
000001b0  14 f0 9f e5 14 f0 9f e5 14 f0 9f e5 14 f0 9f e5
000001c0  40 02 00 00 40 02 00 00 40 02 00 00 40 02 00 00
000001d0  40 02 00 00 40 02 00 00 40 02 00 00 ef be ad de
          ...
```

Figure 4-11 Address 0x00000000~0x000001a0 of NAND Flash

4.3.1.2 Data Type

Mainly the image of data type into NAND Flash in the specified address. Depending on the value of image start offset (aligned by page size boundary, page size is based on NAND specifications). If image start offset equal then 0x10000, image of data into NAND Flash in the 0x10000 address, it can help user to configure NAND Flash.

Data type supports YAFFS (inband tags mode) and UBIFS. Both file system formats can be selected data type. Those file system can be identified by uboot or Linux.

Using the following command to crate a new yaffs2 image with inband tags.(yaffs2 tags stored in data blocks)

```
# mkyaffs2 --inband-tags -p 2048 rootfs rootfs_yaffs2.img
```

- --inband-tags：yaffs2 tags stored in data blocks.
- -p：Set NAND Flash page size.
- rootfs folder can be compressed into rootfs_yaffs2.img, user can use the NuWriter tool to burn rootfs_yafffs2.img into NAND Flash.

Enter the following command will mount yaffs2 file system on the Linux. yaffs2 command can be found in *yaffs2utils.tar.gz*

```
mount -t yaffs2 –o “inband-tags” /dev/mtdblock2 /flash
```

Users need mkfs.ubifs and ubinize utilities to create UBIFS images. The below example demonstrates how to create an UBIFS image for a 128MiB NAND Flash.

```
# mkfs.ubifs -F -x lzo -m 2048 -e 126976 -c 732 -o rootfs_ubifs.img -
d ./rootfs

# ubinize -o ubi.img -m 2048 -p 131072 –O 2048 -s 2048 rootfs_ubinize.cfg
```

mkfs.ubifs

-r, -d, --root=DIR	build file system from directory DIR
-m, --min-io-size=SIZE	minimum I/O unit size
-e, --leb-size=SIZE	logical erase block size
-c, --max-leb-cnt=COUNT	maximum logical erase block count
-o, --output=FILE	output to FILE
-j, --jrn-size=SIZE	journal size
-R, --reserved=SIZE	how much space should be reserved for the super-user
-x, --compr=TYPE	compression type - "lzo", "favor_lzo", "zlib" or "none" (default: "lzo")
-X, --favor-percent	may only be used with favor LZO compression and defines how many percent better zlib should compress to make mkfs.ubifs use zlib instead of LZO (default 20%)
-f, --fanout=NUM	fanout NUM (default: 8)
-F, --space-fixup	file-system free space has to be fixed up on first mount (requires kernel version 3.0 or greater)
-k, --keyhash=TYPE	key hash type - "r5" or "test" (default: "r5")
-p, --orph-lebs=COUNT	count of erase blocks for orphans (default: 1)
-D, --devtable=FILE	use device table FILE
-U, --squash-uids	squash owners making all files owned by root
-l, --log-lebs=COUNT	count of erase blocks for the log (used only for debugging)
-v, --verbose	verbose operation
-V, --version	display version information
-g, --debug=LEVEL	display debug information (0 - none, 1 - statistics, 2 - files, 3 - more details)

ubinize

-o, --output=	output file name
-p, --peb-size=	size of the physical eraseblock of the Flash this UBI image is created for in bytes, kilobytes (KiB), or megabytes (MiB) (mandatory parameter)

- -m, --min-io-size= minimum input/output unit size of the Flash in bytes
- -s, --sub-page-size= minimum input/output unit used for UBI headers, e.g. sub-page size in case of NAND Flash (equivalent to the minimum input/output unit size by default)
- -O, --vid-hdr-offset= offset if the VID header from start of the physical eraseblock (default is the next minimum I/O unit or sub-page after the EC header)
- -e, --erase-counter= the erase counter value to put to EC headers (default is 0)
- -x, --ubi-ver= UBI version number to put to EC headers (default is 1)
- -Q, --image-seq= 32-bit UBI image sequence number to use (by default a random number is picked)
- -v, --verbose be verbose

The *rootfs_ubinize.cfg* is listed below. The rootfs folder can be compressed into *ubi.img*, user can use the NuWriter tool to brun ubi.img into NAND Flash.

```
[rootfs-volume]
mode=ubi
image=rootfs_ubifs.img
vol_id=0
vol_size=92946432
vol_type=dynamic
vol_name=system
vol_flags=autoresize
```

Users could refer to “/sys/class/misc/ubi_ctrl/dev” for setting “10 56”. Enter the following commands will mount UBIFS file system on the Linux.

```
mknod /dev/ubi_ctrl c 10 56
ubiattach /dev/ubi_ctrl -p /dev/mtd2
mount -t ubifs ubi0:system /flash
```

UBIFS command can be found in mtd-utils.tar.gz

The following options on Linux kernel must also be configured.

YAFFS2：

```
File systems  --->
     [*] Miscellaneous filesystems  --->
          <*>   yaffs2 file system support
          <*>   Autoselect yaffs2 format
          <*>   Enable yaffs2 xattr support
```

UBIFS：

```
Device Drivers  --->
     -*- Memory Technology Device (MTD) support  --->
           <*>   Enable UBI - Unsorted block images  --->
File systems  --->
     [*] Miscellaneous filesystems  --->
          <*>   UBIFS file system support
```

```
           [*]     Advanced compression options
           [*]       LZO compression support
           [*]       ZLIB compression support
```

Pakages	Description

lzo-2.09.tar.gz	The compression / decompression tool.
	Cross compiler command is as follows:
	$ cd lizo-2.09
	$ ./configure --host=arm-linux --prefix=$PWD/../install
	$ make
	$ make install

libuuid-1.0.3.tar.gz	A universally unique identifier tool.
	Cross compiler command is as follows:
	$ cd libuuid-1.0.3
	$ ./configure --host=arm-linux --prefix=$PWD/../install
	$ make
	$ make install

mtd-utils.tar.gz	mtd-utils source code.
	Cross compiler command is as follows:
	Thie packages need to use lzo-2.09.tar.gz and libuuid-1.0.3.tar.gz.
	$ cd mtd-utils
	$ export CROSS=arm-linux-
	$ export WITHOUT_XATTR=1
	$ export DESTDR=$PWD/../install
	$ export LZOCPPFLAGS=-I/home/install/include
	$ export LZOLDFLAGS=-L/home/install/lib
	$ make
	$ make install

yaffs2utils.tar.gz	yaffs2 command tool $ make

4.3.1.3 Environment Type

Loader Type is set uboot environment variables, the image of environment type into NAND Flash in the specified address. U-Boot reads environment variables file to set the environment. If image start offset equal then 0x10000, image of data into NAND Flash in the 0x10000 address, it can help user to configure NAND Flash.

U-Boot environment variable in “env.txt” is listed below.

```
baudrate=115200
```

```
bootdelay=3
ethact=emac
ethaddr=00:00:00:11:66:88
stderr=serial
stdin=serial
stdout=serial
```

4.3.1.4 Pack Type

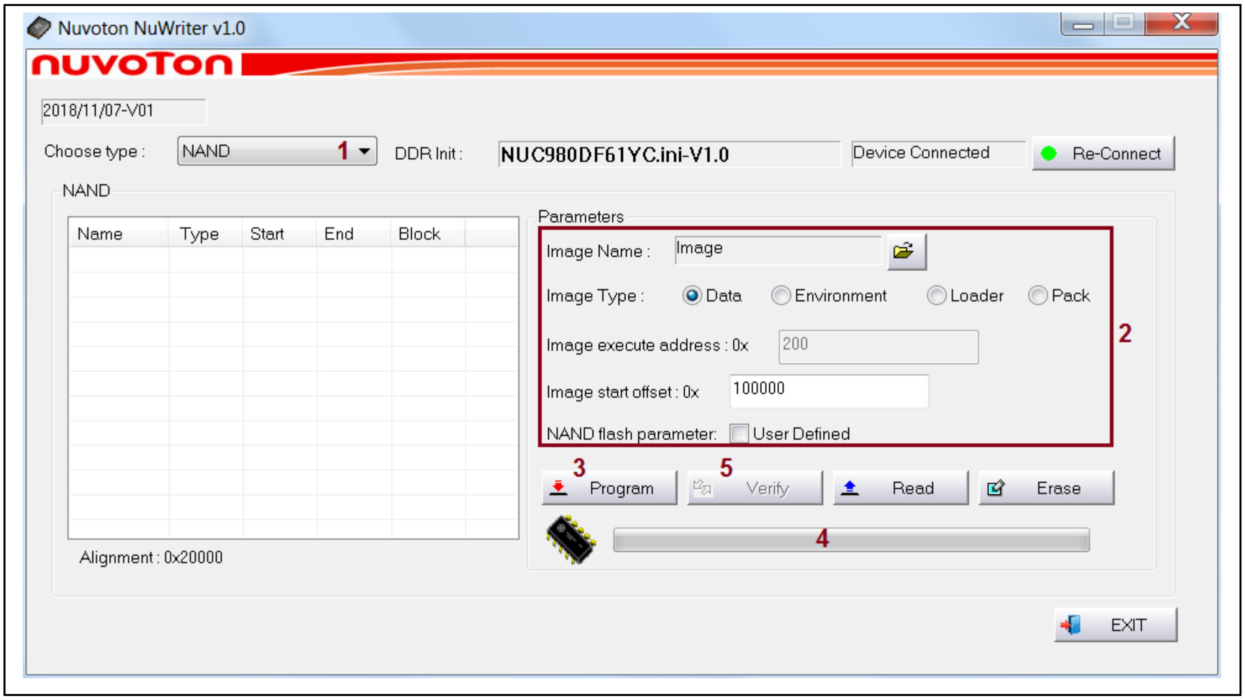
Pack type can burn pack image to NAND Flash. The image of pack type into NAND Flash in the specified address. Depending on the value of image start offset (aligned by block size boundary, block size is based on NAND specifications). User can refer to section 4.7 for creating a pack image.

4.3.1.5 Bad Page Process

NAND Flash supports four types image to burn into the NAND Flash. We need to detect and skip bad blocks, it is assumed that NAND is written a data to block 0, block 1, block 2, block 3, but block 3 is bad block. Then write block 0, block 1, block 2 and block 4.

4.3.2 **Operation Steps**

4.3.2.1 Add a New Image



Nuvoton NuWriter v1.0

2018/11/07-V01

Choose type : NAND

DDR Init : NUC980DF61YC.ini-V1.0

Device Connected

Name	Type	Start	End	Block

Alignment : 0x20000

Parameters

Image Name : Image

Image Type : ● Data ○ Environment ○ Loader ○ Pack

Image execute address : 0x 200

Image start offset : 0x 100000

NAND flash parameter: ☐ User Defined

Figure 4-12 NAND – New Image

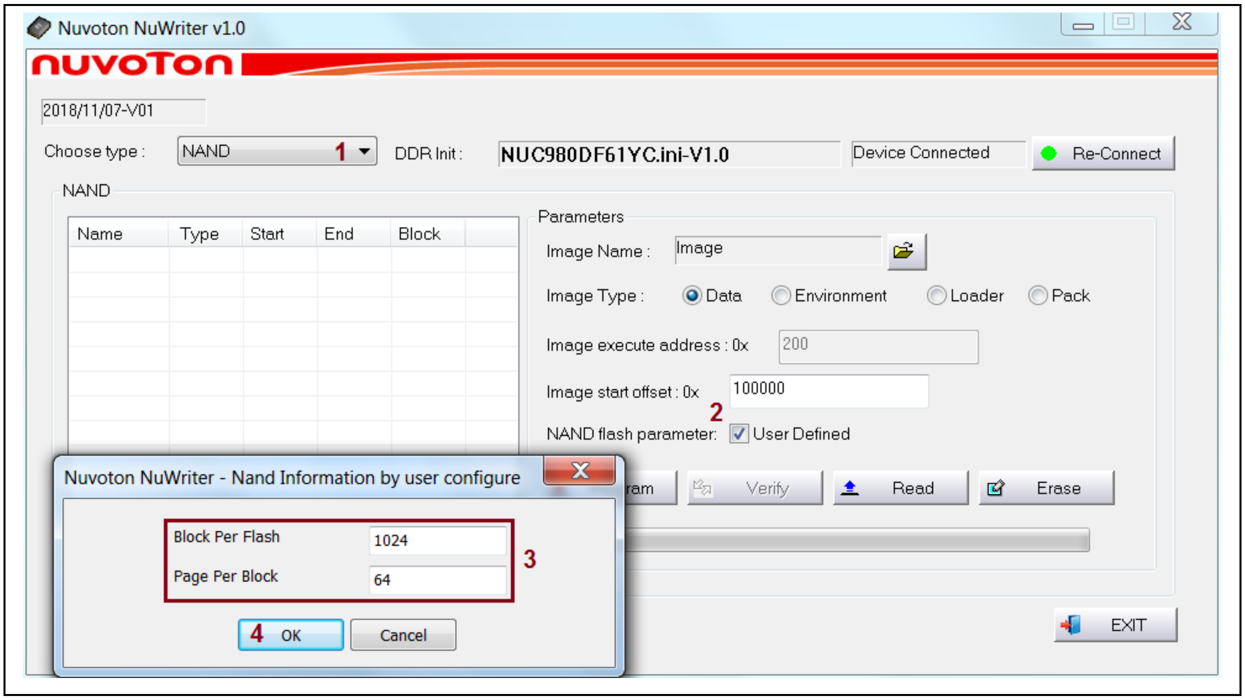
The steps of adding a new image to NAND Flash is shown in Figure 4-12 and the detailed procedures are listed below.

1. Select the “**NAND**” type, which will not list the pre-burned images in the NAND Flash ROM.
2. Fill in the image information：
   - Image Name: Browse the image file
   - Image Type: Select the image type (only one type can be selected)
   - Image execute address: Enter image execute address. Only Loader Type is vaild.
   - Image start offset: Enter image start offset.

- NAND Flash parameter: Check “**User Defined**” to pop up the window for configuring NAND parameters.

3. Click “**Program**” button.
4. Waiting for progress bar shows progress has been completed.
5. After the program is completed, click the “**Verify**” button to read back the image data to make sure the burning status.

4.3.2.2 Configure NAND Flash parameters


Nuvoton NuWriter v1.0

2018/11/07-V01

Choose type : NAND

DDR Init : NUC980DF61YC.ini-V1.0

Device Connected

Image Name : Image

Image Type : ● Data ○ Environment ○ Loader ○ Pack

Image execute address : 0x 200

Image start offset : 0x 100000

NAND flash parameter: ☑ User Defined

Nuvoton NuWriter - Nand Information by user configure

Block Per Flash 1024

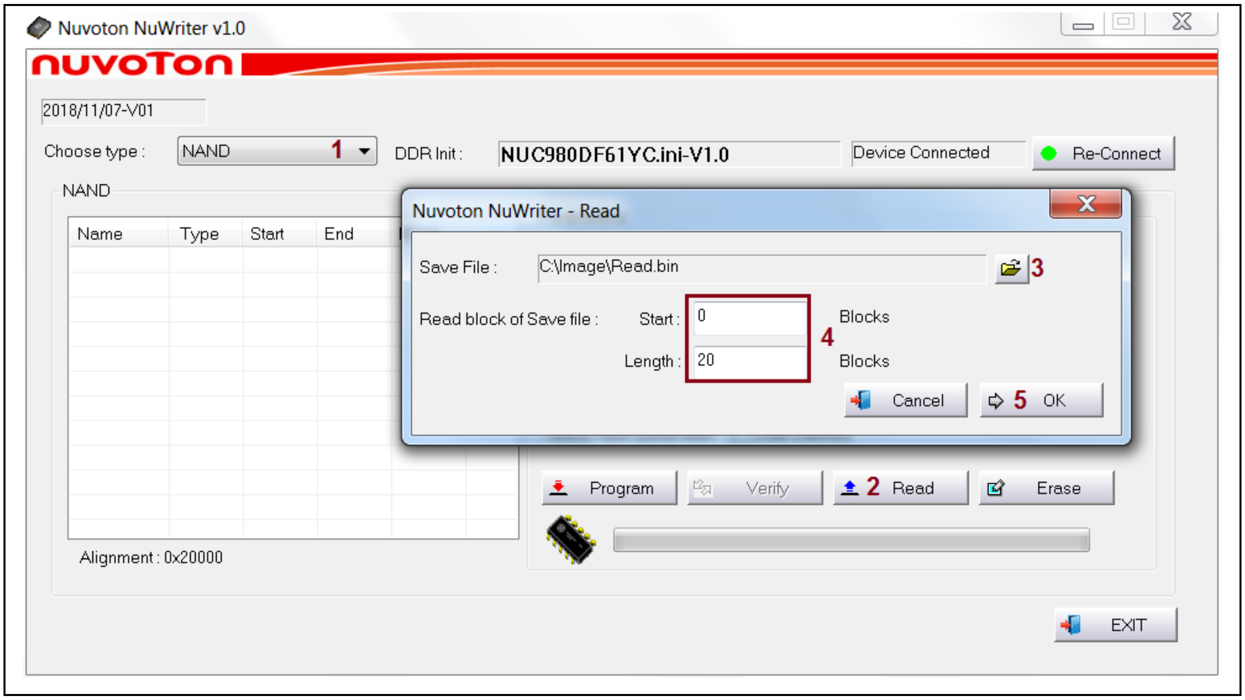
Page Per Block 64

Figure 4-13 NAND – Configure Parameters

The NuWriter will automatically detect the NAND ID to determine its parameters, and users can set the NAND parameters is based on NAND specifications. The following steps describes how to configure NAND parameters.

1. Select the “**NAND**”.
2. Check “**User Defined**” to pop up the window for setting NAND parameter.
3. Fill the following parameters for NAND Flash using.
    - Block Per Flash: Block size per NAND Flash.
    - Page Per Block: Page size per Block.
4. Click “**OK**” button to configure NAND Flash.

4.3.2.3 Read Raw Data



2018/11/07-V01

Choose type : NAND

DDR Init : NUC980DF61YC.ini-V1.0

Device Connected

Save File : C:\Image\Read.bin

Read block of Save file : Start : 0 Blocks

Length : 20 Blocks

Alignment : 0x20000

Figure 4-14 NAND – Read

Refer to the Figure 4-14 to read data from NAND Flash.

1. Select the "**NAND**".
2. Click "**Read**" button.
3. Click the "**Browse**" button to save the image file.
4. Set the values of start addres and block count.(Aligned on block size boundary, Block size is based on NAND Flash specifications).
   - Start: Start addres of blocks
   - Length: Length of blocks
5. Click "**OK**" button to read data.

4.3.2.3.1 Read Data Mode

According to ChipReadWithBad in NuWriter/path.ini to change read function. Read data function is listed below.

1. ChipReadWithBad=0(defaut), Read data region for read data function.
2. ChipReadWithBad=1, Read data and OOB(out of band) region for read data function.

```
[SDRAM]
…


[TARGET]
CHIP=0
DDR=0
DDRADDRESS=16
TimeEnable=1
Timeus=5000
ChipEraseWithBad=0
```

```
ChipReadWithBad=0
ChipWriteWithOOB=0
FirstDelay=500
…
```

4.3.2.4 Erase NAND Flash

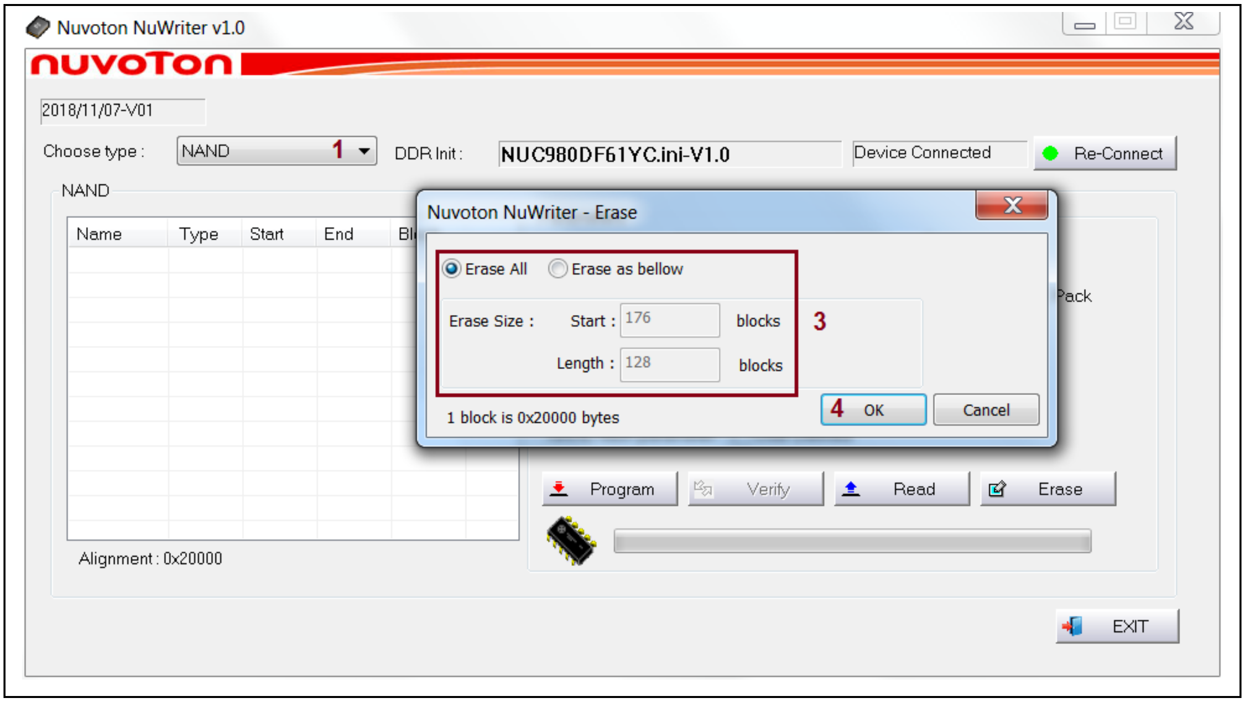


Figure 4-15 NAND – Erase

According to the Figure 4-15 to erase NAND Flash:

1. Select the “**NAND**” type.
2. Click “**Erase**” button.
3. Chose “**Erase All**” or “**Erase as BelowBelow**”
   - Enter the erase blocks, Aligned on block size boundary, Block size is based on NAND Flash specifications.
4. Click “**OK**” button to erase data on the NAND Flash.

4.3.2.4.1 Erase Flash Mode

According to ChipEraseWithBad in Nuwriter/path.ini, It can change erase Flash function.

1. ChipEraseWithBad=0(default), Clear data region for erase Flash function.
2. ChipEraseWithBad=1, Clear data region and OOB region for erase Flash function.

```
[SDRAM]
…

[DEFAULT]
PAGE_IDX=3

[TARGET]
CHIP=0
```

```
DDR=0
DDRADDRESS=16
TimeEnable=1
Timeus=5000
ChipEraseWithBad=0
ChipReadWithBad=0
ChipWriteWithOOB=0
FirstDelay=500
…
```

4.4 SPI Mode

This mode can write a new image to SPI Flash and specify the type of the image. These types can be recognized by bootloader or Linux. The Image type is set Loader, Data, Environment or Pack.

4.4.1 Image Type

4.4.1.1 Loader Type

The detailed introduction of Loader Type format, please refer to section 4.3.1.1. The format of Loader Type is shown in Figure 4-16. Then burned image of Loader Type into SPI Flash in the address 0x00000000, as shown in Figure 4-17.

<table>
<tr><th></th><th colspan="2">0x0</th><th colspan="4">0x4</th><th colspan="4">0x8</th><th>0xC</th></tr>
<tr><td><b>0x00</b></td><td colspan="2">Boot Code Marker</td><td colspan="4">Execute Address</td><td colspan="4">Image Size</td><td>Reserved</td></tr>
<tr><td><b>0x10</b></td><td>Page Size</td><td>Spare Area</td><td>Quad Read cmd</td><td>Read Status cmd</td><td>Write Status cmd</td><td>Status Value</td><td>Dummy Byte</td><td>Reserved</td><td>Reserved</td><td>Reserved</td><td>Reserved</td></tr>
<tr><td><b>0x20</b></td><td colspan="2">DDR - Initial Marker</td><td colspan="4">DDR - Counter</td><td colspan="4">DDR - Address 0</td><td>DDR – Value0</td></tr>
<tr><td><b>0x30</b></td><td colspan="2">DDR - Address 1</td><td colspan="4">DDR - Value 1</td><td colspan="4">DDR - Address 2</td><td>DDR - Value 2</td></tr>
<tr><td><b>0x40</b></td><td colspan="2">DDR - Address 3</td><td colspan="4">DDR - Value 3</td><td colspan="4">DDR - Address 4</td><td>DDR - Value 5</td></tr>
<tr><td><b>0x50</b></td><td colspan="2">DDR - Address 5</td><td colspan="4">DDR - Value 5</td><td colspan="4">DDR - Dummy</td><td>DDR - Dummy</td></tr>
<tr><td><b>0x60</b></td><td colspan="11">uBoot</td></tr>
</table>

Figure 4-16 SPI Boot Code Header

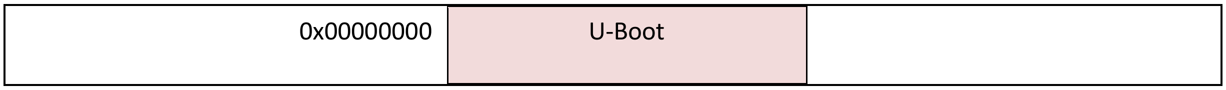


Figure 4-17 SPI Flash

4.4.1.2 *Data Type*

Mainly the image of data type into SPI Flash in the specified address. Depending on the value of image start offset (aligned on block size boundary, block size is based on SPI specifications). If image start offset equal then 0x10000, image of data into SPI Flash in the 0x10000 address, it can help user to configure SPI Flash.

4.4.1.3 *Environment Type*

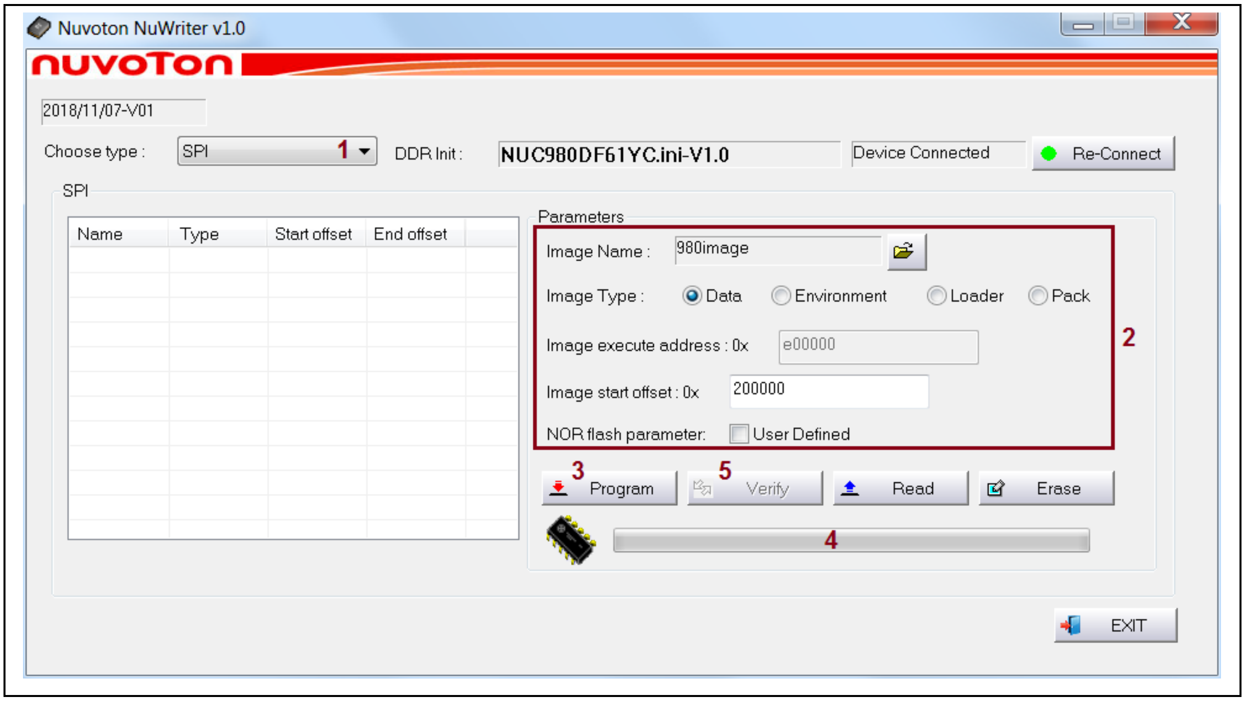
Loader Type is set uboot environment variables, the image of environment type into SPI Flash in the specified address. U-Boot reads environment variables file to set the environment. If image start offset equal then 0x10000, image of data into SPI Flash in the 0x10000 address, it can help user to configure SPI Flash.

4.4.1.4 *Pack Type*

Pack type can burn pack image to SPI Flash. The image of pack type into SPI Flash in the specified address. Depending on the value of image start offset (aligned on block size boundary, block size is based on SPI specifications). User can refer to section 4.7 for creating a pack image.

4.4.2 Operation Steps

4.4.2.1 Add a New Image



2018/11/07-V01

Choose type : SPI

DDR Init : NUC980DF61YC.ini-V1.0

Device Connected

Image Name : 980image

Image Type : ● Data ○ Environment ○ Loader ○ Pack

Image execute address : 0x e00000

Image start offset : 0x 200000

NOR flash parameter: ☐ User Defined

Figure 4-18 SPI – New Image

The steps below explain how to add a new image to SPI Flash.

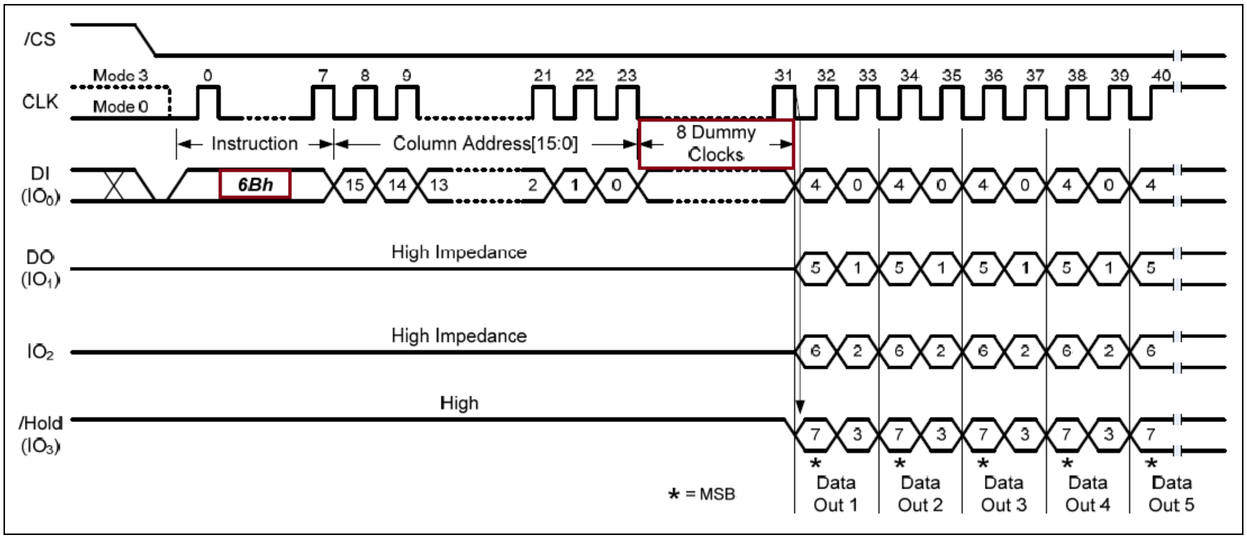
1. Select the “**SPI**” type, which will not list the pre-burned images in the SPI Flash ROM.
2. Fill in the image information：
   - Image Name: Browse the image file.
   - Image Type: Select the image type. (only one type can be selected)
   - Image execute address: Enter image execute address. Only Loader Type is vaild.
   - Image start offset: Enter image start offset.
   - NOR Flash parameter: Check “**User Defined**” to pop up the window for configuring SPI NOR parameters.
3. Click “**Program**” button.
4. Waiting for progress bar shows progress has been completed.
5. After the program is completed, click the “**Verify**” button to read back the image data to make sure the burning status.

4.4.2.2 Configure SPI Flash Parameters

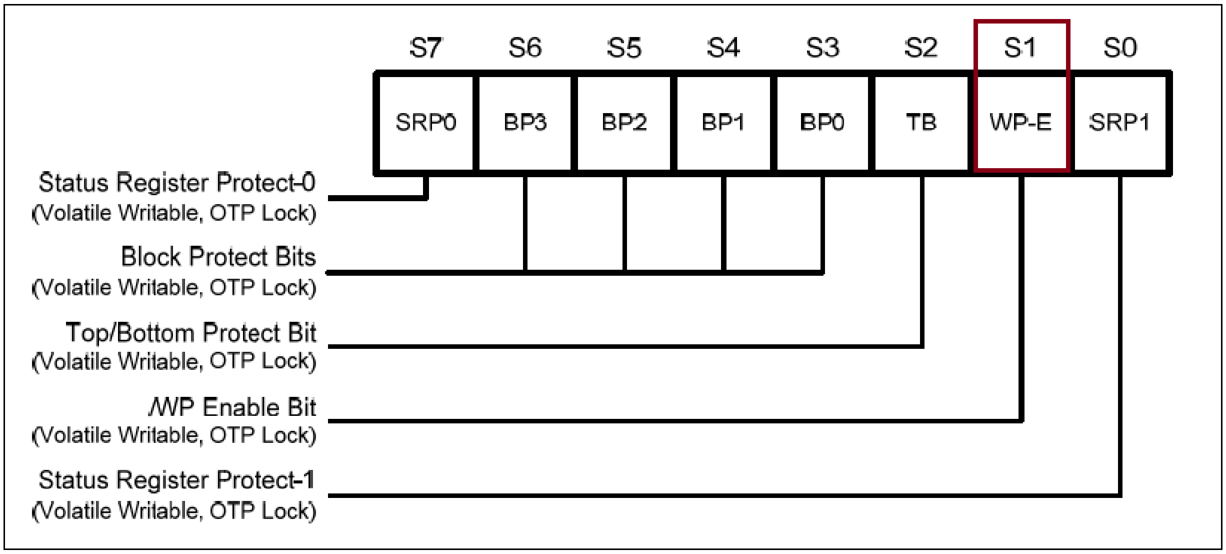
The NuWriter will automatically detect the SPI Flash ID to determine its parameters, and the user can set the SPI Flash parameters is based on SPI Flash specifications. These comands are used for Quad mode operating.

- QuadReadCmd:Quad mode read command. Note that the command here is in hex format.
- ReadStatusCmd=Read status command. Note that the command here is in hex format.
- WriteStatusCmd=Write status command. Note that the command here is in hex format.
- StatusValue=Quad mode enable bit in SPI Flash status register. Note that the command here is in hex format.
- DummyByte=dummy byte length in Quad mode read command. Note that the size is in decimal format.

SPI Quad mode read command and relative informations can also be found in SPI Flash datasheet.

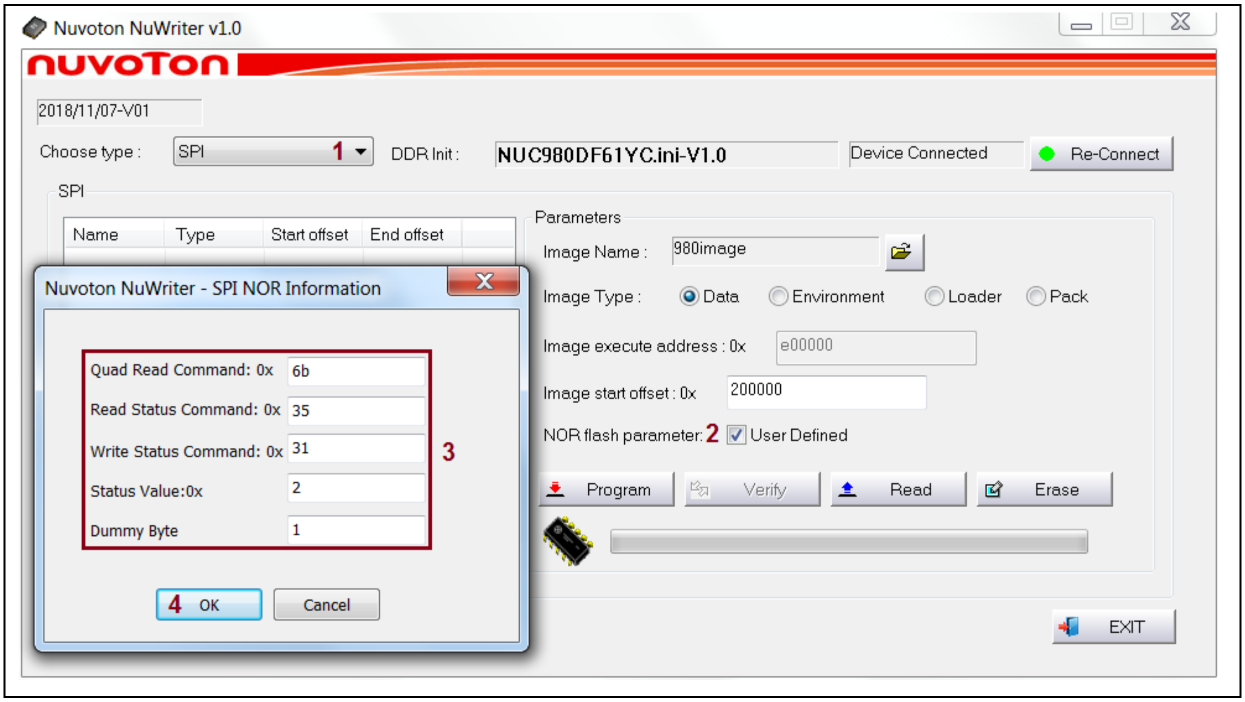


SPI Quad mode needs to set the WP-E bit in the status register to 1, this value corresponds to Nuwriter's StatusValue. The definition of the status register bit needs to refer to the SPI Flash specification. The following figure is captured from a SPI Flash specification.


S7
S6
S5
S4
S3
S2
S1
S0
SRP0
BP3
BP2
BP1
BP0
TB
WP-E
SRP1
Status Register Protect-0 (Volatile Writable, OTP Lock)
Block Protect Bits (Volatile Writable, OTP Lock)
Top/Bottom Protect Bit (Volatile Writable, OTP Lock)
/WP Enable Bit (Volatile Writable, OTP Lock)
Status Register Protect-1 (Volatile Writable, OTP Lock)

Configured SPI Flash parameters as shown in Figure 4-19.

1. Select the “**SPI**” type.
2. Check “**User Defined**” to pop up the window for setting SPI Flash parameter.
3. Fill the Quad Read Command, Read Status Command, Write Status Command, Status Value and Dummy Byte according to the SPI Flash specification.
4. Click “**OK**” button.


Nuvoton NuWriter v1.0

2018/11/07-V01

Choose type : SPI 1

DDR Init : NUC980DF61YC.ini-V1.0

Device Connected

SPI

Name	Type	Start offset	End offset

Parameters

Image Name : 980image

Image Type : ● Data ○ Environment ○ Loader ○ Pack

Image execute address : 0x e00000

Image start offset : 0x 200000

NOR flash parameter: 2 ☑ User Defined

Nuvoton NuWriter - SPI NOR Information

Quad Read Command: 0x 6b

Read Status Command: 0x 35

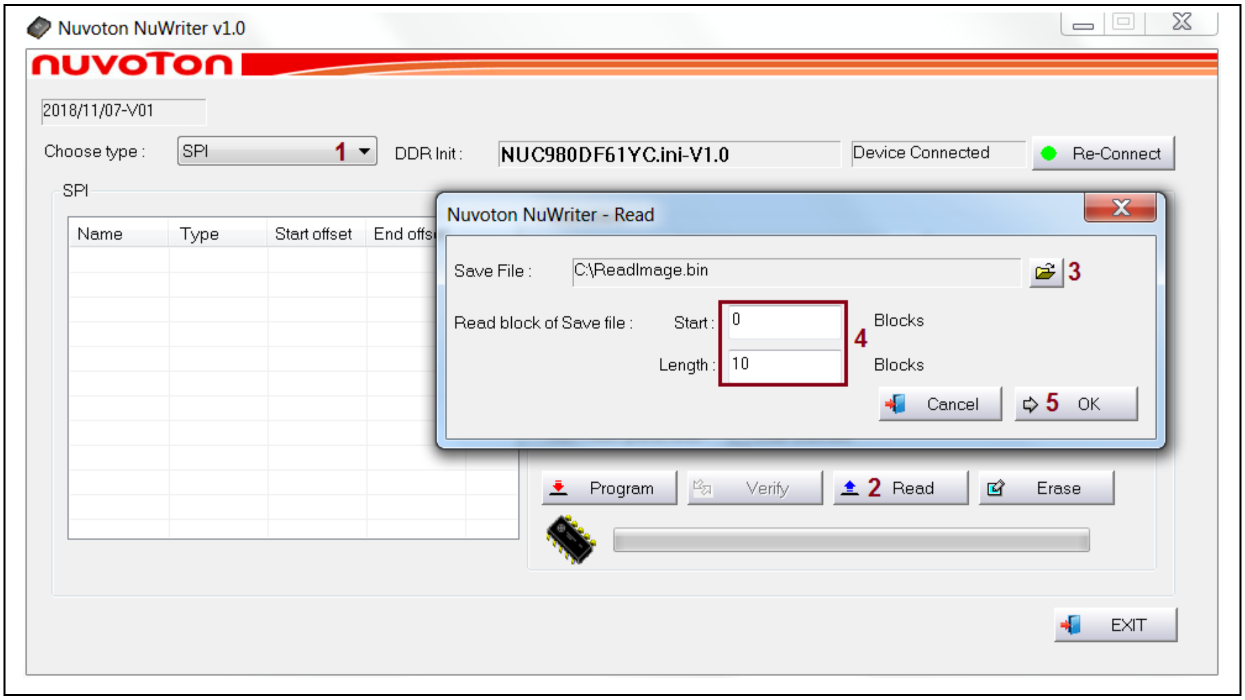
Write Status Command: 0x 31

Status Value:0x 2

Dummy Byte 1

Figure 4-19 SPI– Configure Parameters

4.4.2.3 Read Raw Data


Nuvoton NuWriter v1.0

2018/11/07-V01

Choose type : SPI 1

DDR Init : NUC980DF61YC.ini-V1.0

Device Connected

SPI

Name	Type	Start offset	End offs

Nuvoton NuWriter - Read

Save File : C:\ReadImage.bin 3

Read block of Save file : Start : 0 Blocks

Length : 10 Blocks

Figure 4-20 SPI – Read Image

According to Figure 4-20 to read data from SPI Flash:

1. Select the “**SPI**”.
2. Click “**Read**” button.
3. Click the “**Browse**” button to save the image file.
4. Enter the read back of blocks, Aligned on block size boundary, Block size is based on SPI Flash specifications.

- Start: Start address of blocks
- Length: Length of blocks

5. Click “**OK**” to read data from SPI Flash.

4.4.2.4 Erase SPI Flash

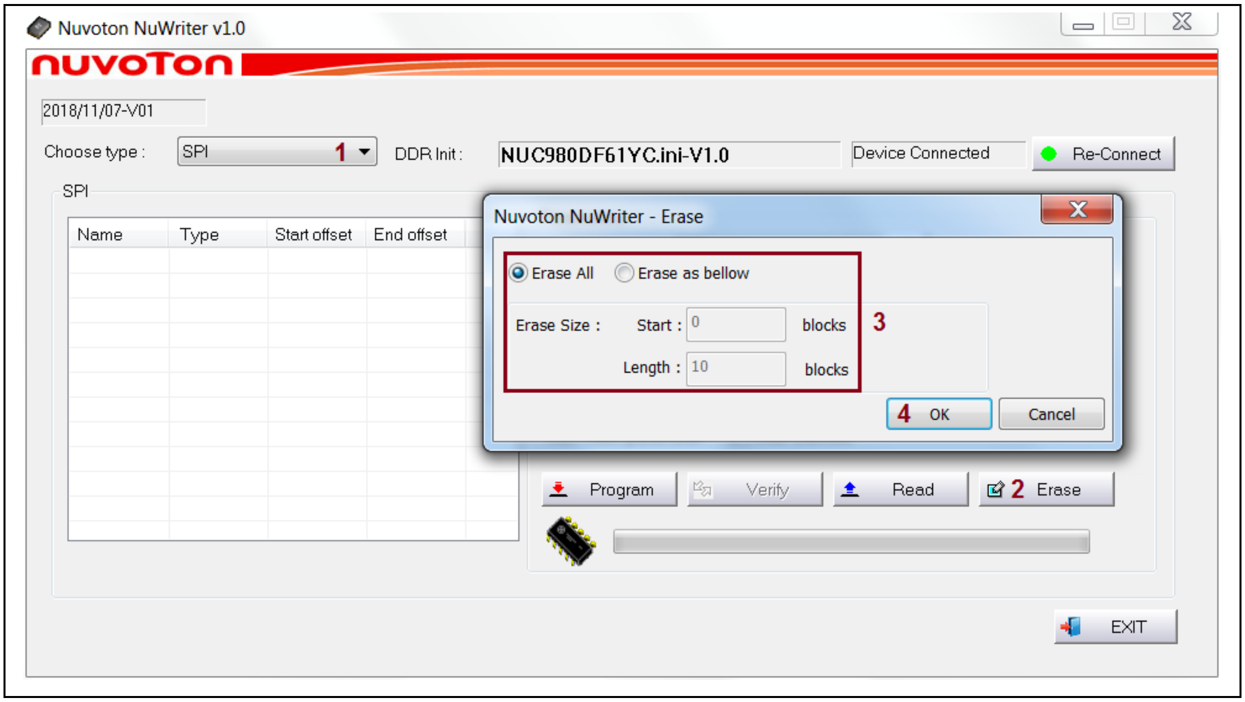


Figure 4-21 SPI – Erase

According to the figure above, Follow the steps below to erase SPI Flash:

1. Select the “**SPI**” type.
2. Click “**Erase**” button
3. Chose “**Erase All**” or “**Erase as Below**”
    - Enter the erase blocks, Aligned on block size boundary, Block size is based on SPI Flash specifications.
4. Click “**OK**” button to erase data on the SPI Flash.

4.5 eMMC/SD Mode

There are two sets of eMMC/SD on the NUC980DF61Y development board, namely eMMC0/SD0 and eMMC1/SD1. Users can select eMMC/SD through the Power-on Setting SW2.9 and SW2.10. Almost all of the functions are placed on the development board, so users use this board need to be aware of certain functions will be shared pin. The eMMC0/SD0 shares the pin with the NAND Flash on the development board, so SW3 and SW4 should be Off state when using the eMMC0/SD0 mode.

This mode can write a new image to eMMC/SD and specify the type of the image. These types can be recognized by bootloader or Linux. The Image type is set Loader, Data, Environment or Pack.

4.5.1 **Image Type**

4.5.1.1 Loader Type

The detailed introduction of Loader Type format, please refer to section 4.3.1.1. The format of Loader Type is shown in Figure 4-22. Then burned image of Loader Type into eMMC/SD in the address 0x400, as shown in Figure 4-23.

	0x0	**0x4**	**0x8**	**0xC**

0x00	Boot Code Maker		Execute Address				Image Size				Reserved
0x10	Page Size	Spare Area	Quad Read cmd	Read Status cmd	Write Status cmd	Status Value	Dummy	Reserved	Reserved	Reserved	Reserved
0x20	DDR - Initial Marker		DDR - Counter				DDR - Address 0				DDR – Value0
0x30	DDR - Address 1		DDR - Value 1				DDR - Address 2				DDR - Value 2
0x40	DDR - Address 3		DDR - Value 3				DDR - Address 4				DDR - Value 5
0x50	DDR - Address 5		DDR - Value 5				DDR - Dummy				DDR - Dummy
0x60	Loader										

Figure 4-22 Boot Code Header

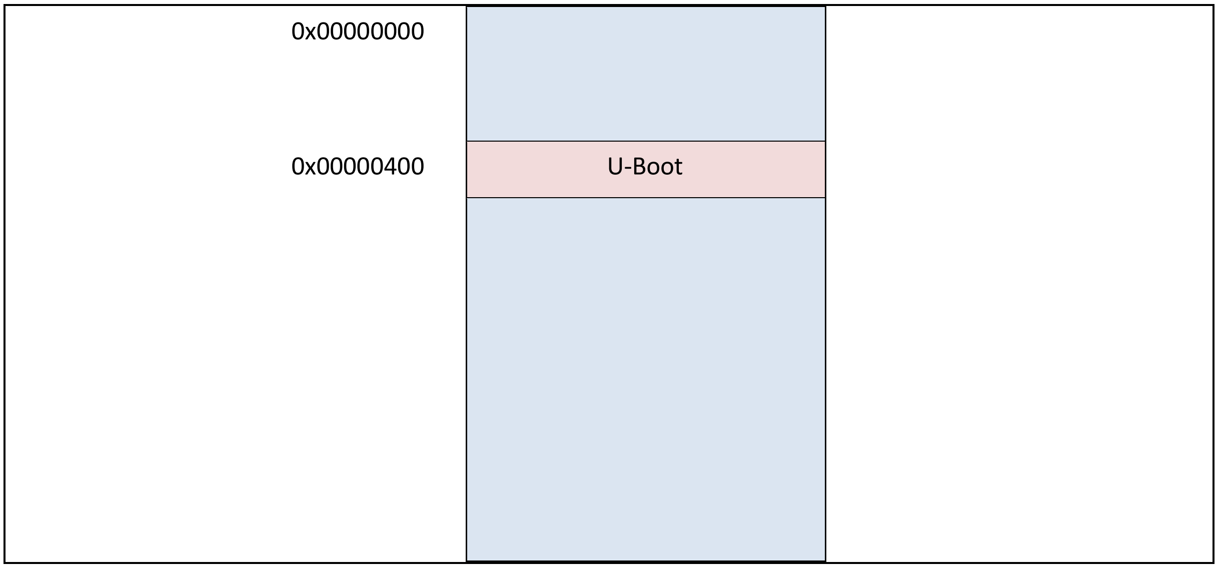


Figure 4-23 eMMC/SD

4.5.1.2 Data Type

Mainly the image of data type into eMMC/SD in the specified address. Depending on the value of image start offset (aligned on 512bytes boundary). If image start offset equal then 0x10000, image of data into eMMC/SD Flash in the 0x10000 address, it can help user to configure eMMC/SD Flash.

4.5.1.3 Environment Type

Loader Type is set uboot environment variables, the image of environment type into eMMC/SD in the specified address. U-Boot reads environment variables file to set the environment. If image start offset equal then 0x10000, image of data into eMMC/SD Flash in the 0x10000 address, it can help user to configure eMMC/SD Flash.

4.5.1.4 Pack Type

Pack type can burn pack image to eMMC/SD. The image of pack type into eMMC/SD in the specified address. Depending on the value of image start offset (aligned on 512bytes boundary). User can refer to section 4.7 for creating a pack image.

4.5.1.5 Format (FAT32)

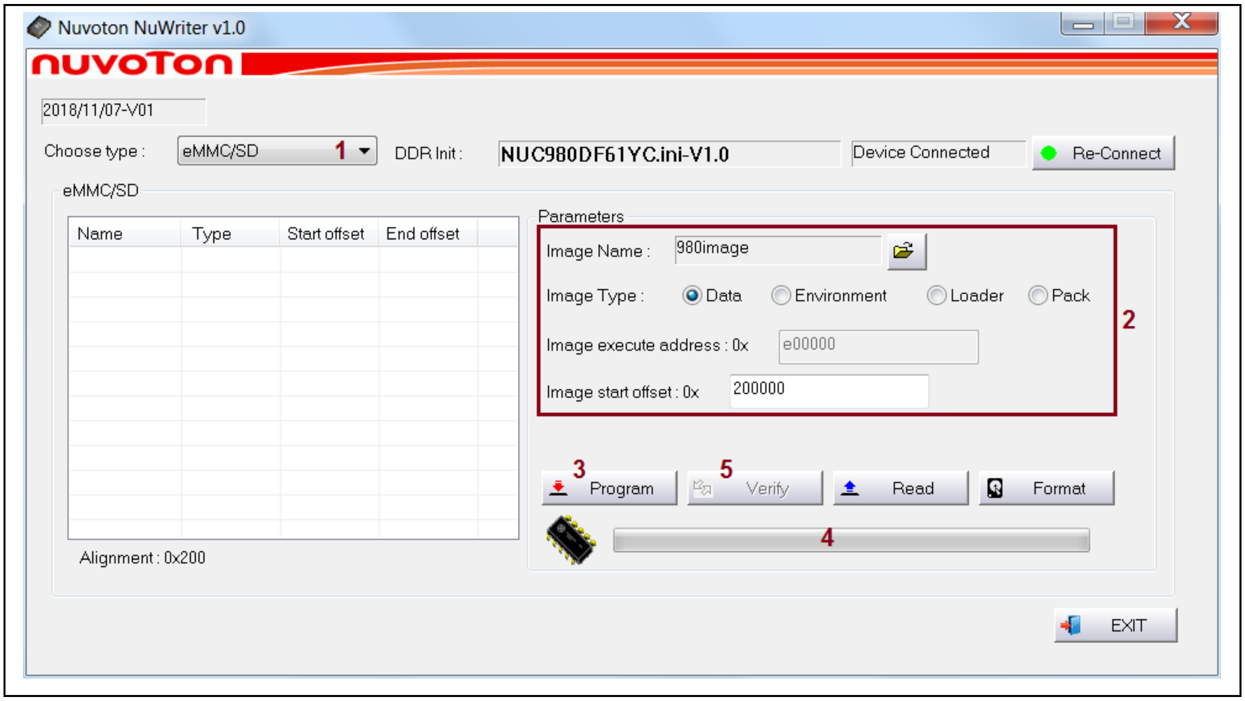
Because eMMC need to store uboot image and other images, so user must set reserved space to store image (aligned on 512bytes boundary). User need to determine the reserved space size.

Note: Modifying this parameter may cause the existing image damage or file system damage.

The FAT32 file system supports multiple disk partitions, but that is limited by the fixed structure of the file allocation table in the main boot record(MBR). A hard disk can only support up to 4 partitions.

4.5.2 **Operation Steps**

4.5.2.1 Add a New Image



Nuvoton NuWriter v1.0

2018/11/07-V01

Choose type : eMMC/SD

DDR Init : NUC980DF61YC.ini-V1.0

Device Connected

Name	Type	Start offset	End offset

Alignment : 0x200

Parameters

Image Name : 980image

Image Type : ● Data ○ Environment ○ Loader ○ Pack

Image execute address : 0x e00000

Image start offset : 0x 200000

Figure 4-24 eMMC/SD – New Image

Figure 4-24 illustrates how to add a new image to eMMC/SD.

1. Select the “**eMMC/SD**” type, which will not list the pre-burned images in the eMMC/SD.
2. Fill in the image information：
    - Image Name: Browse the image file.
    - Image Type Select the image type. (only one type can be selected)
    - Image execute address: Enter image execute address. Only Loader Type is valid.
    - Image start offset: Enter image start offset.

1. Click the “**Program**” button.
2. Waiting for progress bar shows progress has been completed.
3. After the program is completed, click the “**Verify**” button to read back the image data to make sure the burning status.

4.5.2.2 Read Raw Data

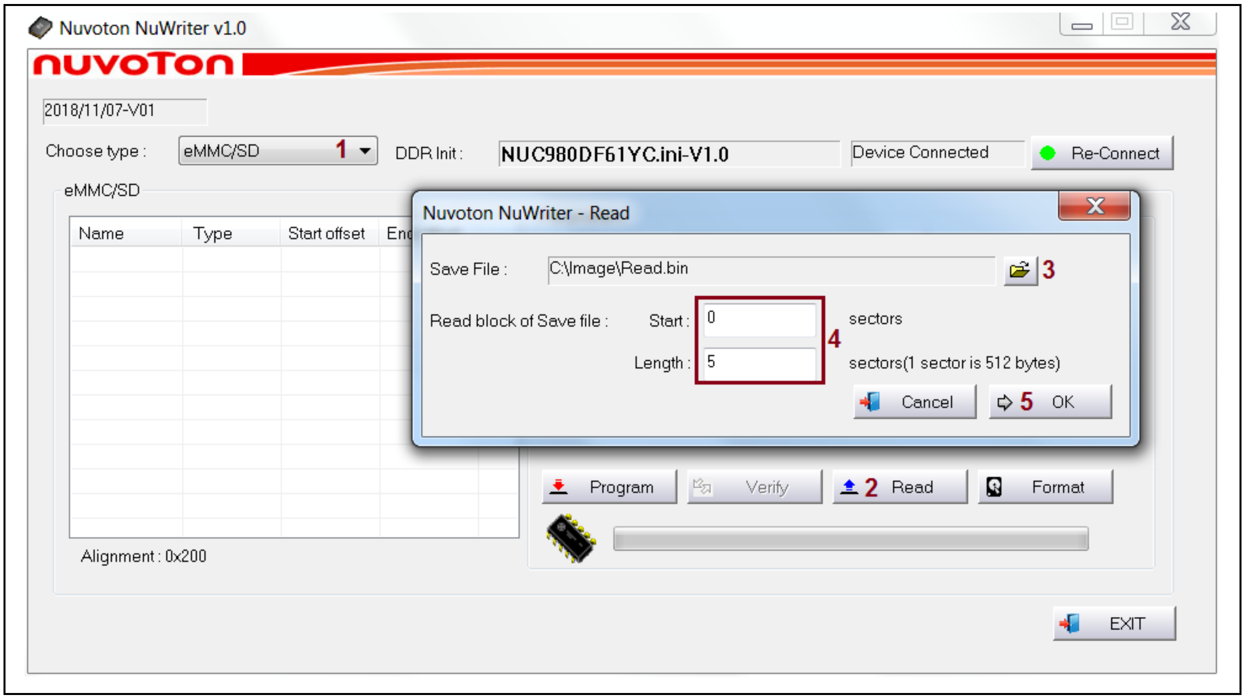


Figure 4-25 eMMC/SD – Read

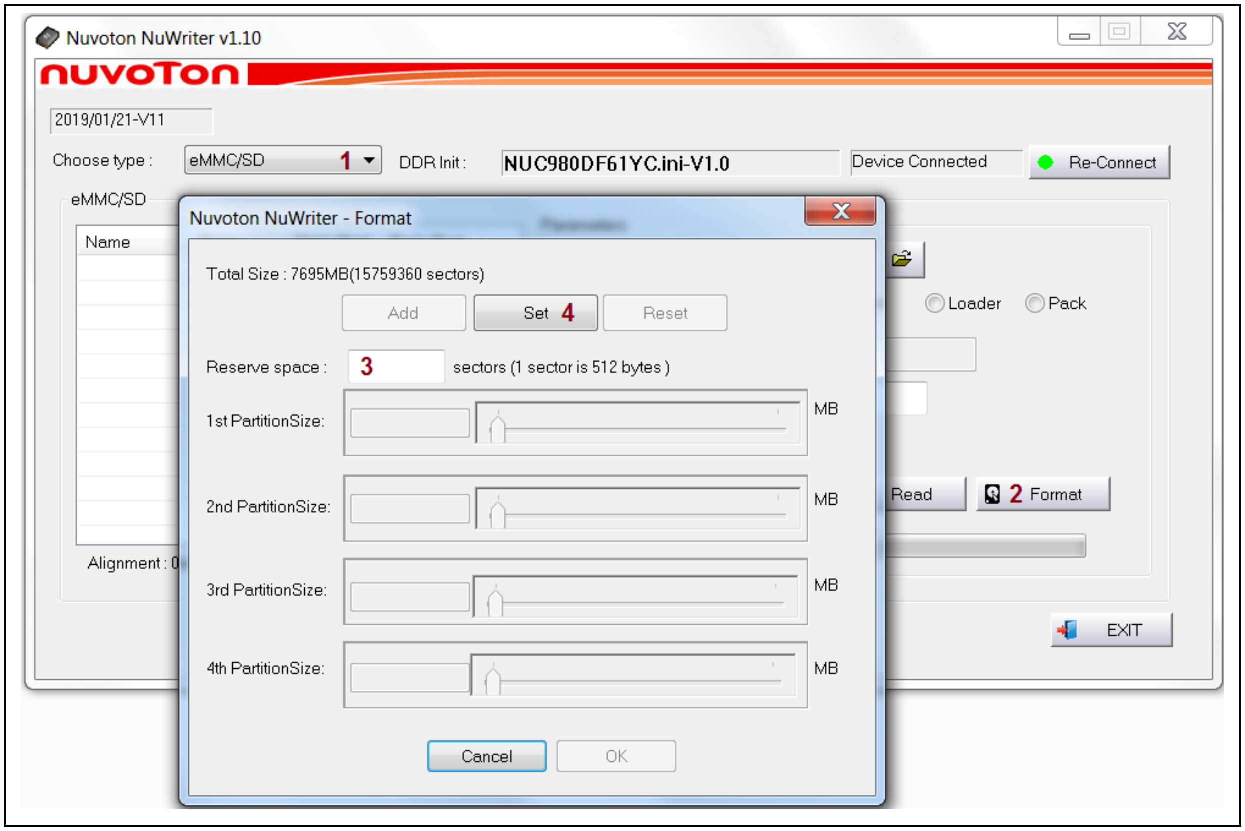
The steps introduce how to read data from eMMC/SD are listed below.

1. Select the “**eMMC/SD**”.
2. Click “**Read**” button.
3. Browse save file.
4. Enter the read back of sectors, Aligned on 512 bytes boundary
   - Start: Start address of sectors
   - Length: Length of sectors
5. Click the “**OK**” button.

4.5.2.3 Format (FAT32)

Figure 4-26, Figure 4-27 and Figure 4-28 illustrate how to format eMMC/SD.

1. Select “**eMMC/SD**”.
2. Click the “**Format**” button.
3. Enter value of Reserve space size(1 sector is 512bytes). Note : This parameter may cause existing image or file system is being destroyed.
4. Click the “**Set**” button to apply reserved space size.
5. After setting the reserved space size, there are four sliders to user configure the eMMC/SD partition size. Users operate these slider to determine each partition size.
   - 1st PartitionSize: First partition size.
   - 2ndPartitionSize: Second partition size.
   - 3rd PartitionSize: Third partition size.
   - 4th PartitionSize: Fourth partition size.
6. Click the "**Set**" button to apply partition size sucessfully.
7. Click the “**Add**” button to add the next partition. Repeat 5~7 to divide the eMMC/SD space into four partitions.
8. Click the "**OK**" button to erase complete.



Nuvoton NuWriter v1.10

2019/01/21-V11

Choose type : eMMC/SD

DDR Init : NUC980DF61YC.ini-V1.0

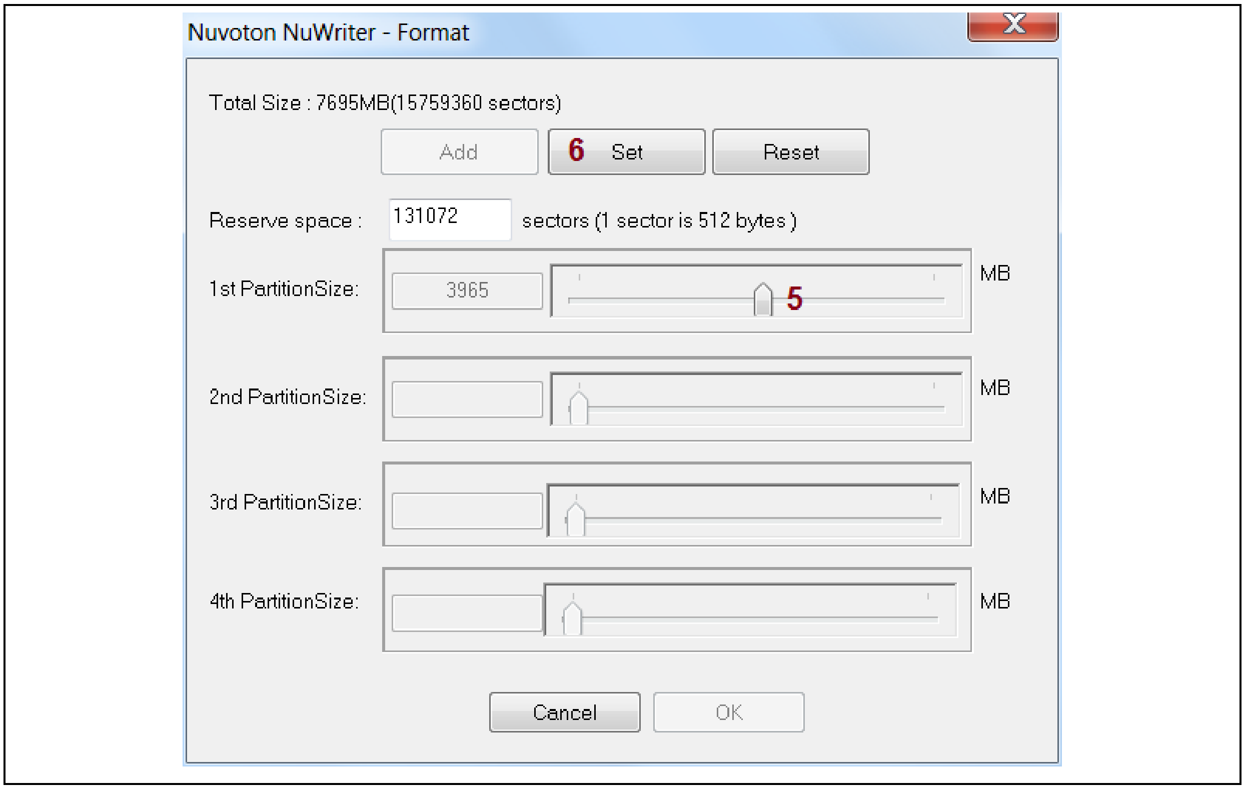
Device Connected

Nuvoton NuWriter - Format

Total Size : 7695MB(15759360 sectors)

Reserve space : sectors (1 sector is 512 bytes)

Figure 4-26 eMMC/SD – Format (1)



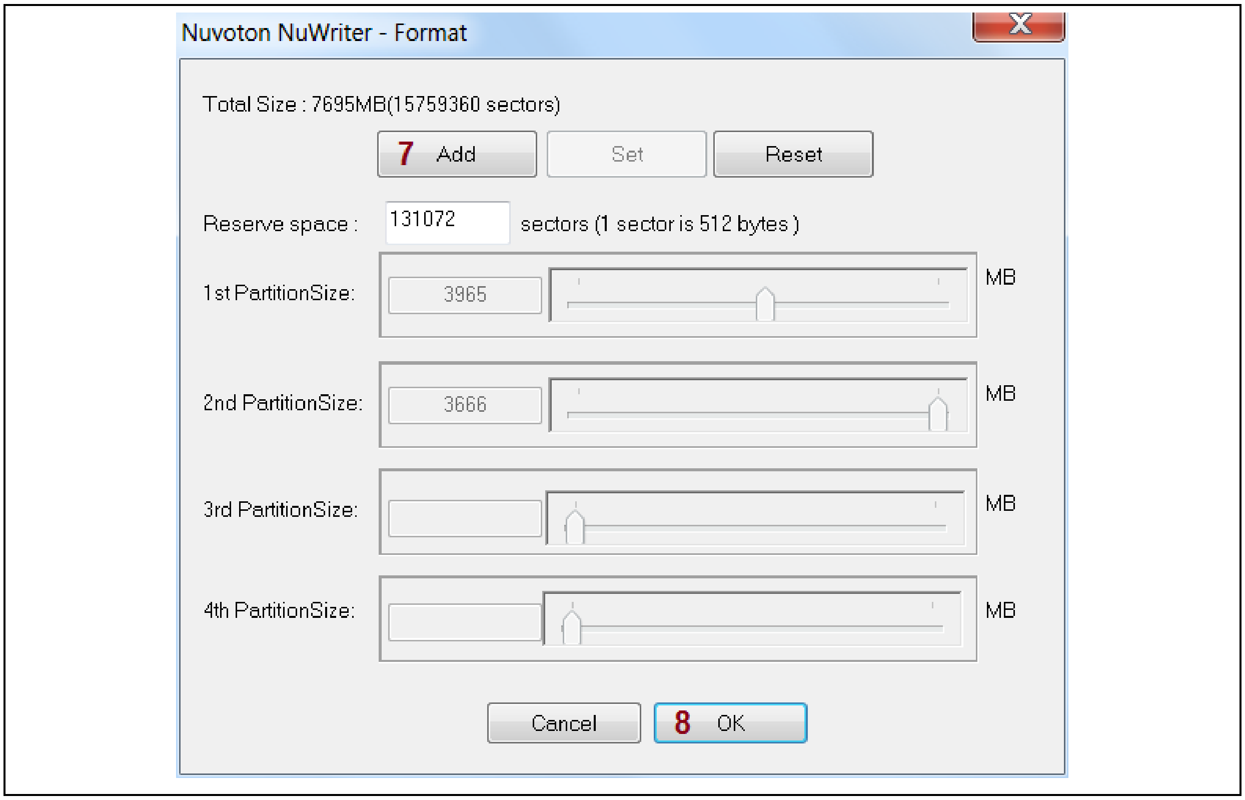
Nuvoton NuWriter - Format

Total Size : 7695MB(15759360 sectors)

Reserve space : 131072 sectors (1 sector is 512 bytes)

1st PartitionSize: 3965 MB

Figure 4-27 eMMC/SD – Format (2)


Nuvoton NuWriter - Format

Total Size : 7695MB(15759360 sectors)

Reserve space : 131072 sectors (1 sector is 512 bytes)

1st PartitionSize: 3965 MB

2nd PartitionSize: 3666 MB

3rd PartitionSize: MB

4th PartitionSize: MB

Figure 4-28 eMMC/SD – Format (3)

4.6 SPI NAND Mode

This mode can write a new image to SPI NAND Flash and specify the type of the image. These types can be recognized by bootloader or Linux. The Image type is set Loader, Data, Environment or Pack.

4.6.1 Image Type

4.6.1.1 Loader Type

The detailed introduction of Loader Type format, please refer to section 4.3.1.1. The format of Loader Type is shown in Figure 4-29. Then burned image of Loader Type into SPI Nand Flash in the address 0x00000000. Figure 4-30 shows the burned image of Loader Type into SPI Nand Flash in the Block0, Block1, Block2 and Block3.

<table>
<tr><th></th><th colspan="2">0x0</th><th colspan="4">0x4</th><th colspan="4">0x8</th><th>0xC</th></tr>
<tr><td><b>0x00</b></td><td colspan="2">Boot Code Marker</td><td colspan="4">Execute Address</td><td colspan="4">Imae Size</td><td>Reserved</td></tr>
<tr><td><b>0x10</b></td><td>Page Size</td><td>Spare Area</td><td>Quad Read cmd</td><td>Read Status cmd</td><td>Write Status cmd</td><td>Status Value</td><td>Dummy</td><td>Reserved</td><td>Reserved</td><td>Reserved</td><td>Reserved</td></tr>
<tr><td><b>0x20</b></td><td colspan="2">DDR - Initial Marker</td><td colspan="4">DDR - Counter</td><td colspan="4">DDR - Address 0</td><td>DDR – Value0</td></tr>
<tr><td><b>0x30</b></td><td colspan="2">DDR - Address 1</td><td colspan="4">DDR - Value 1</td><td colspan="4">DDR - Address 2</td><td>DDR - Value 2</td></tr>
</table>

0x40	DDR - Address 3	DDR - Value 3	DDR - Address 4	DDR - Value 5
0x50	DDR - Address 5	DDR - Value 5	DDR - Dummy	DDR - Dummy
0x60	Loader			

Figure 4-29 Boot Code Header

0x00000000	U-Boot	Block0
	Block0	
	U-Boot	Block1
	Block1	
	U-Boot	Block2
	Block2	
	U-Boot	Block3
	Block3	

Figure 4-30 Block 0~3 of SPI NAND Flash

4.6.1.2 Data Type

Mainly the image of data type into SPI NAND Flash in the specified address. Depending on the value of image start offset (aligned on block size boundary, block size is based on SPI NAND specifications). If image start offset equal then 0x10000, image of data into SPI NAND Flash in the 0x10000 address, it can help user to configure SPI NAND Flash.

4.6.1.3 Environment Type

Loader Type is set uboot environment variables, the image of environment type into SPI NAND Flash in the specified address. U-Boot reads environment variables file to set the environment. If image start offset equal then 0x10000, image of data into SPI NAND Flash in the 0x10000 address, it can help user to configure SPI NAND Flash.

4.6.1.4 Pack Type

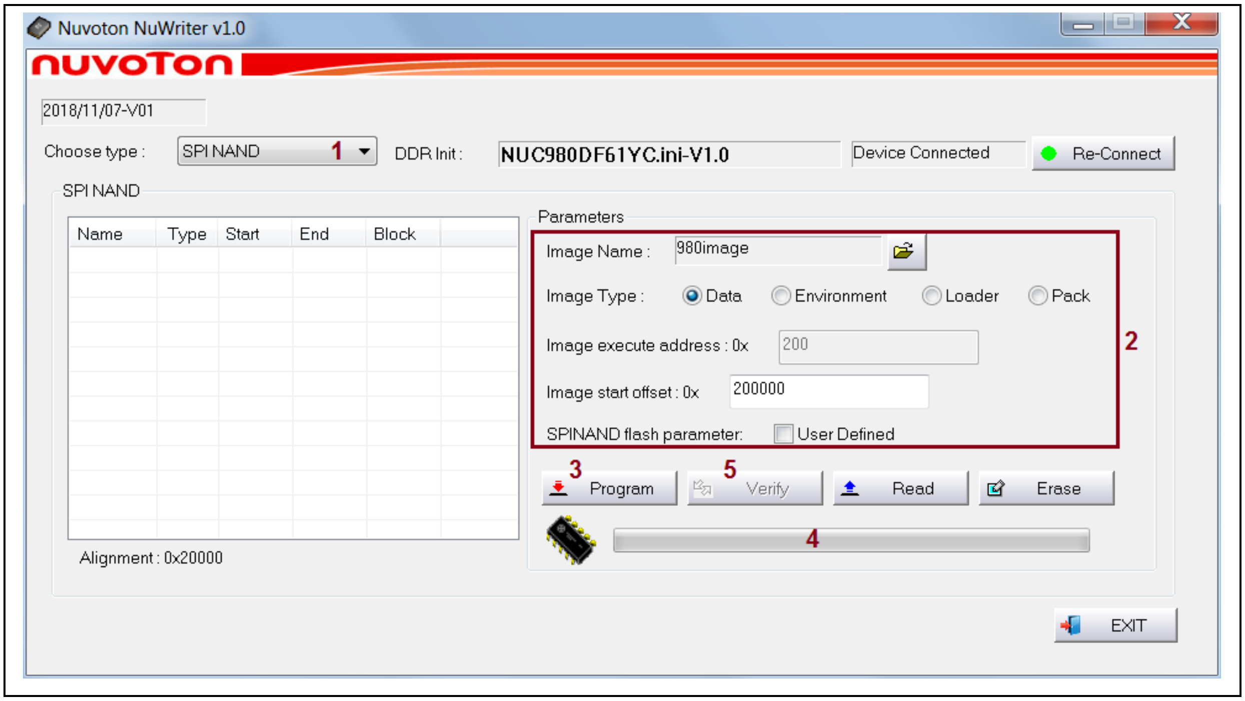
Pack type can burn pack image to SPI NAND Flash. The image of pack type into SPI Flash in the specified address. Depending on the value of image start offset (aligned on block size boundary, block size is based on SPI NAND specifications). User can refer to section 4.7 for creating a pack image.

4.6.1.5 Bad Block Process

SPI NAND Flash supports four types to burn into the SPI NAND Flash. We need to detect and skip bad blocks, it is assumed that SPI NAND is written a data to block 0, block 1, block 2, block 3, but block 3 is bad block. Then write block 0, block 1, block 2 and block 4.

4.6.2 Operation Steps

4.6.2.1 Add a New Image



2018/11/07-V01

Choose type : SPI NAND

DDR Init : NUC980DF61YC.ini-V1.0

Device Connected

Image Name : 980image

Image Type : ● Data ○ Environment ○ Loader ○ Pack

Image execute address : 0x 200

Image start offset : 0x 200000

SPINAND flash parameter: ☐ User Defined

Alignment : 0x20000

Figure 4-31 SPI NAND – New Image

According to Figure 4-31 to add a new image to SPI NAND Flash:

1. Select the “**SPI NAND**” type, which will not list the pre-burned images in the SPI NAND Flash ROM.
2. Fill in the image information:
   - Image Name : Browse the image file
   - Image Type Select the image type (only one type can be selected)
   - Image execute address: Enter image execute address. Only is Loader Type is vaild.
   - Image start offset: Enter image start offset.
   - SPI NAND Flash parameter: Check “**User Defined**” to pop up the window for configuring SPI NAND parameters.
3. Click the “**Program**” button.
4. Waiting for progress bar shows progress has been completed.
5. After the program is completed, click the “**Verify**” button to read back the image data to make sure the burning status.

4.6.2.2 Configure SPI NAND Flash Parameters

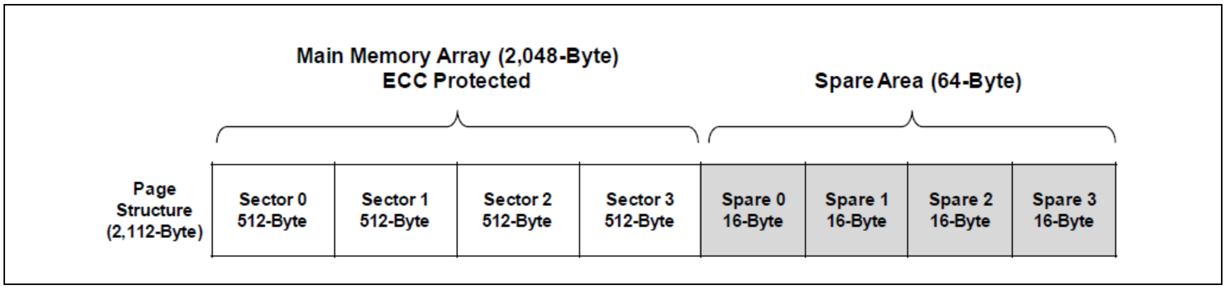
The NuWriter will automatically detect the SPI NAND Flash ID to determine its parameters, and the user can set the SPI NAND Flash parameters is based on SPI NAND Flash specifications. The following content to describe the definition of each SPI NAAD Flash parameters.

- PageSize=SPI NAND page size. Note that the size is in decimal format.
- SpareArea=SPI NAND spare area size. Note that the size is in decimal format.
- QuadReadCmd:Quad mode read command. Note that the command here is in hex format.
- ReadStatusCmd=Read status command. Note that the command here is in hex format.
- WriteStatusCmd=Write status command. Note that the command here is in hex format.
- StatusValue=Quad mode enable bit in SPI Flash status register. Note that the command

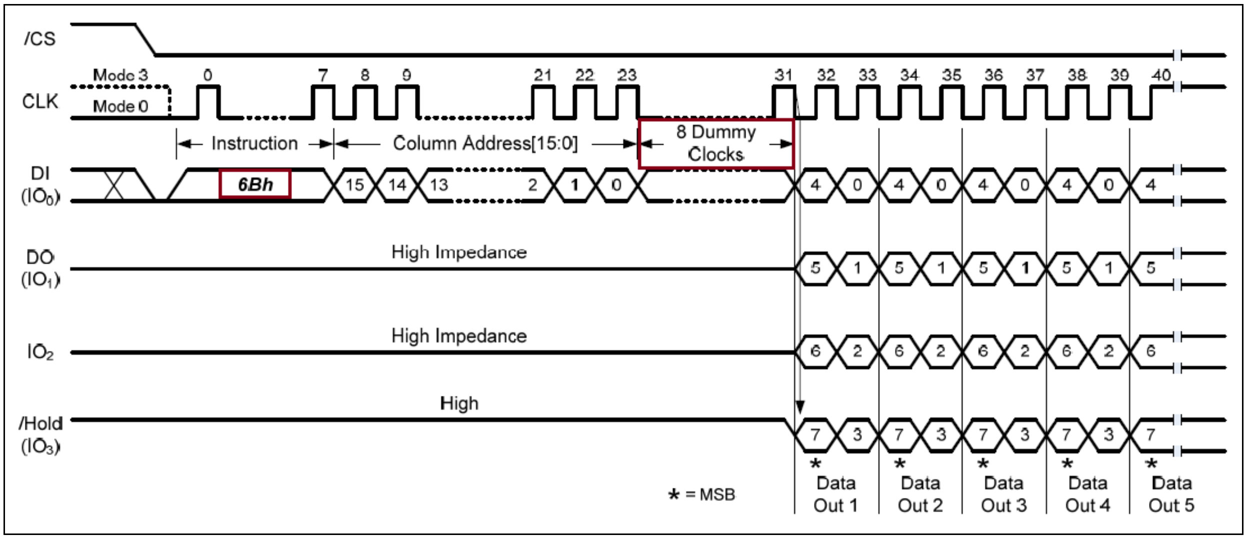
here is in hex format.

- DummyByte=dummy byte length in Quad mode read command. Note that the size is in decimal format.
- Block Per Flash: Block size per SPI NAND Flash. Note that the size is in decimal format.
- Page Per Block: Page size per Block. Note that the size is in decimal format.

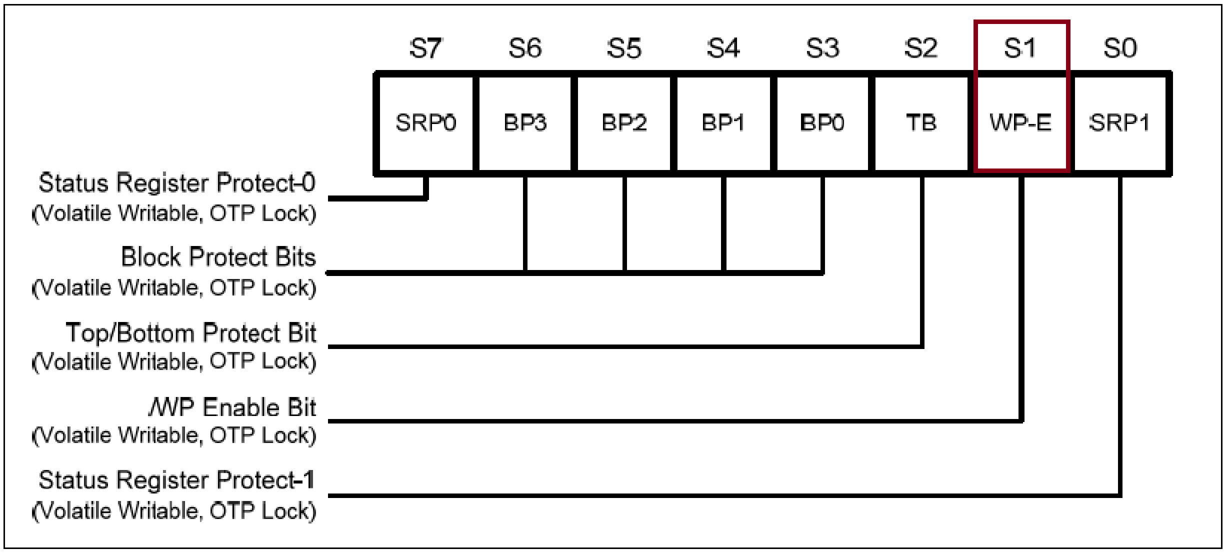
The SPI NAND page size and spare area size differs from SPI NAND Flash model. These information can be found in SPI NAND's datasheet. The below example that SPI NAND's dayasheets describes its page size and spare area size.

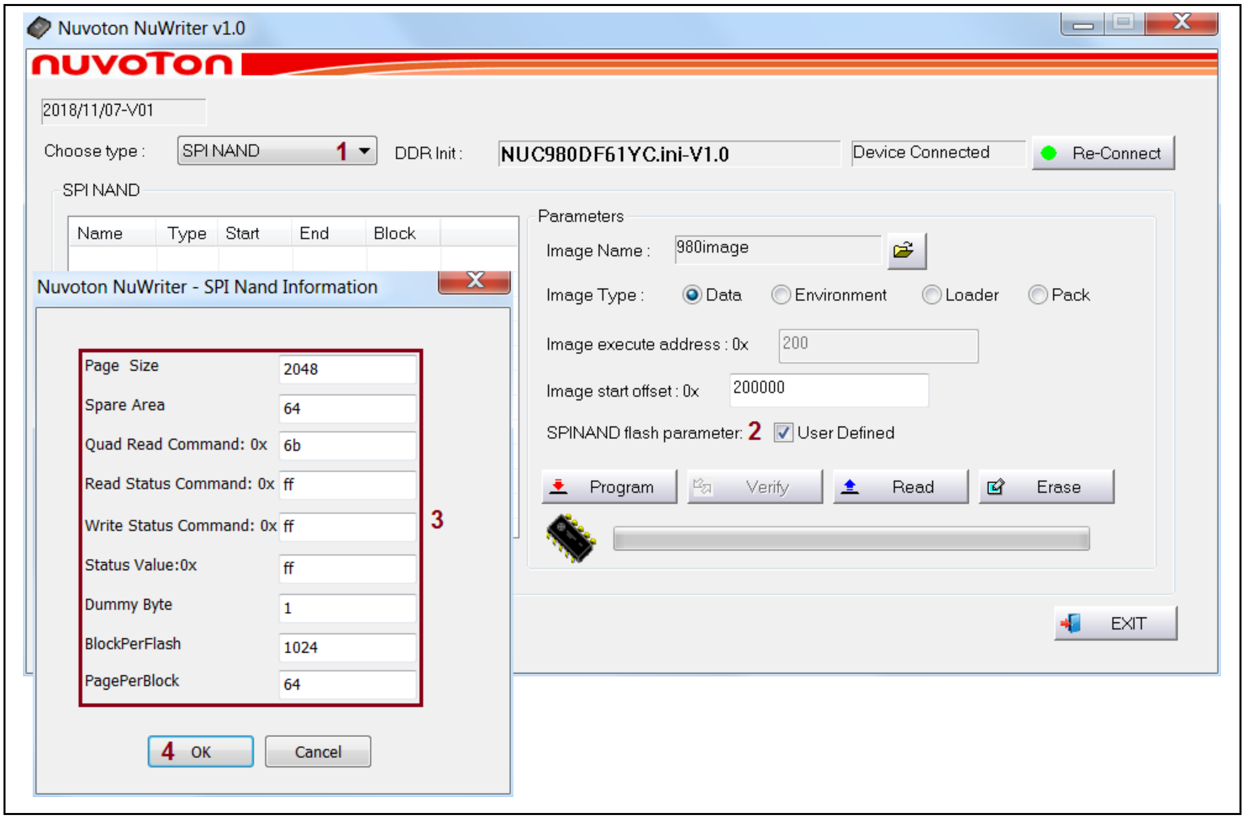


SPI Quad mode read command and relative informations can also be found in SPI NAND Flash datasheet.



SPI Quad mode needs to set the WP-E bit in the status register to 1, this value corresponds to Nuwriter's StatusValue. The definition of the status register bit needs to refer to the SPI NAND Flash specification. The following figure is captured from a SPI NAND Flash specification.


S7
S6
S5
S4
S3
S2
S1
S0
SRP0
BP3
BP2
BP1
BP0
TB
WP-E
SRP1
Status Register Protect-0
(Volatile Writable, OTP Lock)
Block Protect Bits
(Volatile Writable, OTP Lock)
Top/Bottom Protect Bit
(Volatile Writable, OTP Lock)
/WP Enable Bit
(Volatile Writable, OTP Lock)
Status Register Protect-1
(Volatile Writable, OTP Lock)



2018/11/07-V01

Choose type : SPI NAND

DDR Init : NUC980DF61YC.ini-V1.0

Device Connected

SPI NAND

Name	Type	Start	End	Block

Parameters

Image Name : 980image

Image Type : ● Data ○ Environment ○ Loader ○ Pack

Image execute address : 0x 200

Image start offset : 0x 200000

SPINAND flash parameter: ☑ User Defined

Nuvoton NuWriter - SPI Nand Information

Page Size	2048
Spare Area	64
Quad Read Command: 0x	6b
Read Status Command: 0x	ff
Write Status Command: 0x	ff
Status Value:0x	ff
Dummy Byte	1
BlockPerFlash	1024
PagePerBlock	64

Figure 4-32 SPI NAND – Configure Parameters

The NuWriter will automatically detect the SPI NAND ID to determine its parameters, and the user can set the SPI NAND parameters is based on SPI NAND specifications. Refer to Figure 4-32 to set SPI NAND Flash parameters:

1. Select the “**SPI NAND**”.
2. Check “**User Defined**” to pop up the window for setting SPI NAND parameter.
3. Fill the Page Size, Spare Area, Quad Read Command, Read Status Command, Write Status Command, Status Value, Dummy Byte, Block Per Flash, and Page Per Block according to the SPI NAND Flash specification.

4. Click the “**OK**” button.

4.6.2.3 Read Raw Data

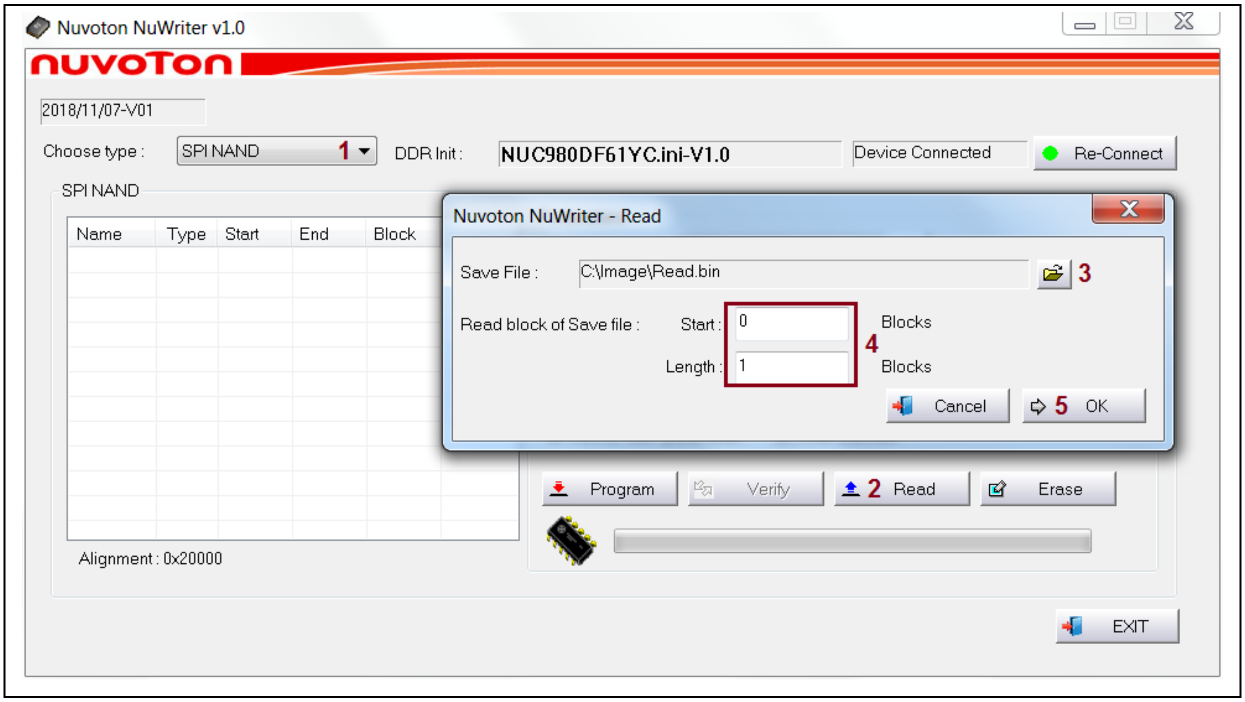
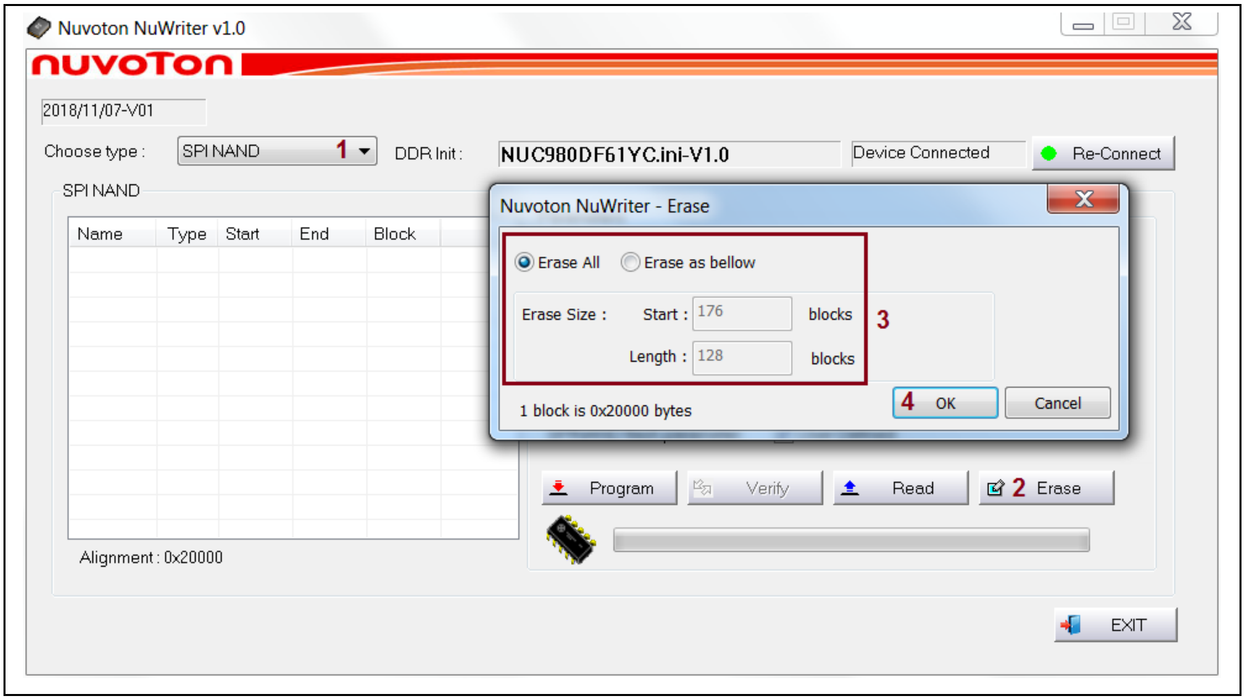


Figure 4-33 SPI NAND – Read

Follow the steps below to read data from SPI NAND Flash.

1. Select the “**SPI NAND**”.
2. Click the “**Read**” button.
3. Browse save file.
4. Set the values of start addres and block count.(Aligned on block size boundary, Block size is based on SPI NAND Flash specifications).
   - Start: Start addres of blocks
   - Length: Length of blocks
5. Click the “**OK**” button.

4.6.2.4 *Erase NAND Flash*



2018/11/07-V01

Choose type : SPI NAND 1

DDR Init : NUC980DF61YC.ini-V1.0

Device Connected

● Re-Connect

Alignment : 0x20000

● Erase All ○ Erase as bellow

Erase Size : Start : 176 blocks

Length : 128 blocks

1 block is 0x20000 bytes

Figure 4-34 SPI NAND – Erase

According to Figure 4-34 to erase SPI NAND Flash:

1. Select the "**SPI NAND**" type.
2. Click the "**Erase**" button.
3. Chose "**Erase All**" or "**Erase as Below**"
   - Enter the erase blocks, Aligned on block size boundary, Block size is based on NAND Flash specifications.
4. Click "**OK**" , Erase data on the SPI NAND Flash.

4.7 Pack Mode

This section introduced how to merge many images into a pack image. The pack image format as shown in Figure 4-35.

	0x0	**0x0**	**0x0**	**0x0**
0x00	Initial Marker	File Length	File Number	Reserve
0x10	Child0-File Length	Child0-File Address	Child0-Reserve	Child0-Reserve
	Child0-data			
	Child1-File Length	Child1-File Address	Child1-Reserve	Child1-Reserve
	Child1-data			

	Child2-data	Child2-data	Child2-data	Child2-data
	Child2-data			

Figure 4-35 Pack Header/Pack Child Header

Initial Marker = {0x00000005}.

File Length = { Length of pack image }, Aligned on 64k bytes boundary.

File Number = { Number of image }.

Child0-File Length = {Length of image 0 }.

Child0-File Address = { Address of image 0 }.

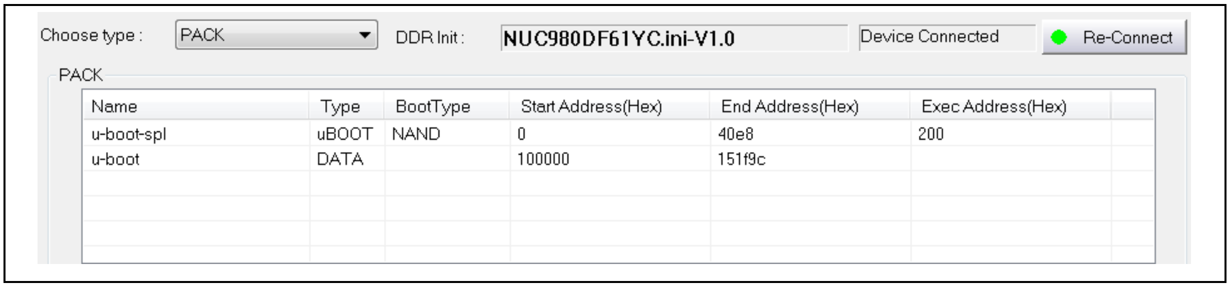
Child0-data = { Content of image 0 }.

Child1-File Length = { Length of image 1 }.

Child1-File Address = { Address of image 1 }.

Child1-data = { Content of image 1 }.

Assuming users provide "u-boot-spl.bin" and "u-boot.bin" to merge into a pack image.



Choose type : PACK

DDR Init : NUC980DF61YC.ini-V1.0

Device Connected

Name	Type	BootType	Start Address(Hex)	End Address(Hex)	Exec Address(Hex)
u-boot-spl	uBOOT	NAND	0	40e8	200
u-boot	DATA		100000	151f9c	

Figure 4-36 Pack Image

Pack image content is listed below.

Initial Marker = 0x00000005.

File Length = (0x000040e8+0x00051f9c) ≅ 0x00070000.

File Number = 2.

Child0-File Length = 0x000040e8.

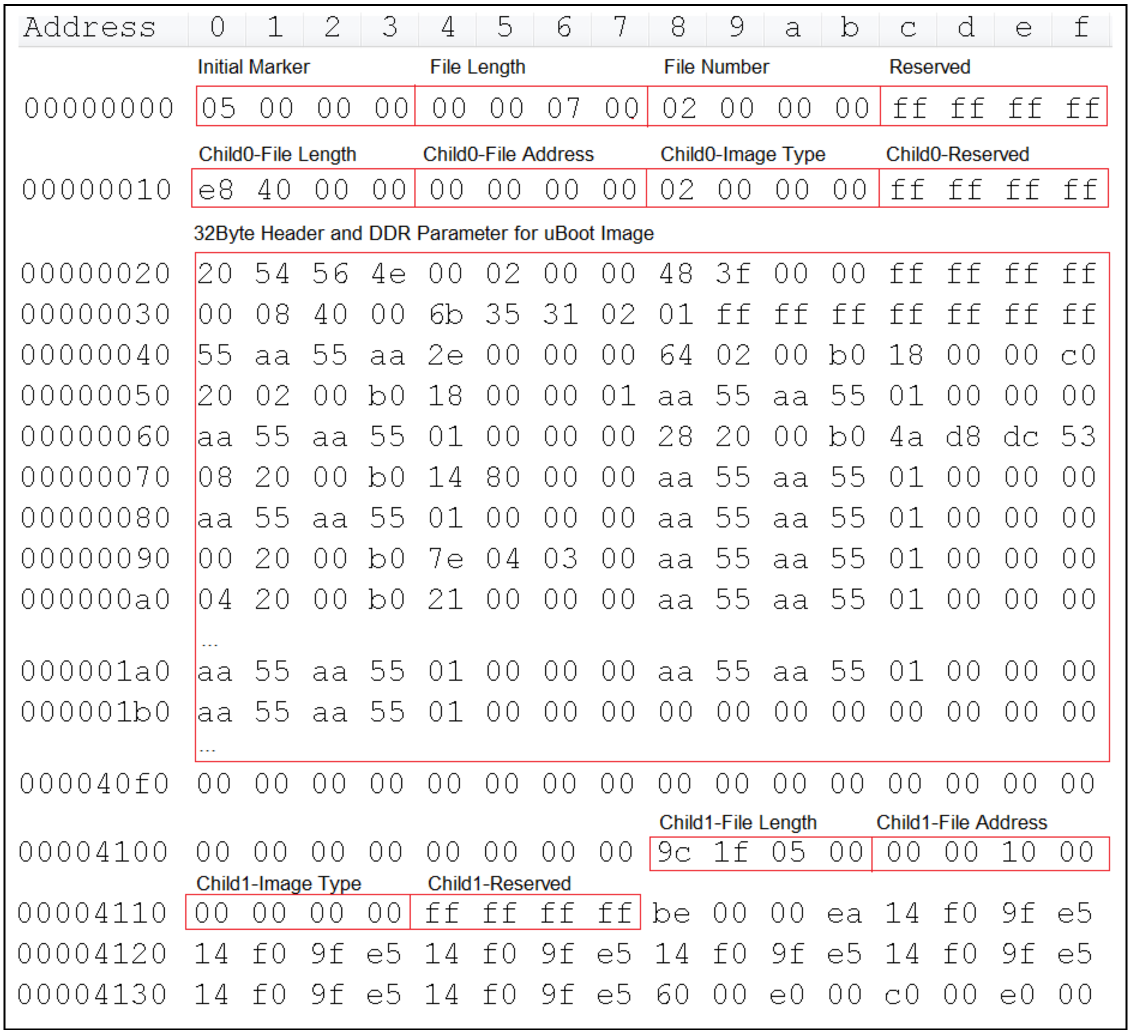
Child0-File Address = 0.

Child0-data = = { address 0x00000000 + 0x00000020 ~ 0x00000020 + 0x000040e8.

Child1-File Length =0x51f9c.

Child1-File Address =0x100000.

Child1-data = = { address 0x00004108 + 0x0000010 ~ 0x0004118 + 0x00051f9c}.



```
Address   0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
          Initial Marker File Length File Number Reserved
00000000  05 00 00 00 00 00 07 00 02 00 00 00 ff ff ff ff
          Child0-File Length Child0-File Address Child0-Image Type Child0-Reserved
00000010  e8 40 00 00 00 00 00 00 02 00 00 00 ff ff ff ff
          32Byte Header and DDR Parameter for uBoot Image
00000020  20 54 56 4e 00 02 00 00 48 3f 00 00 ff ff ff ff
00000030  00 08 40 00 6b 35 31 02 01 ff ff ff ff ff ff ff
00000040  55 aa 55 aa 2e 00 00 00 64 02 00 b0 18 00 00 c0
00000050  20 02 00 b0 18 00 00 01 aa 55 aa 55 01 00 00 00
00000060  aa 55 aa 55 01 00 00 00 28 20 00 b0 4a d8 dc 53
00000070  08 20 00 b0 14 80 00 00 aa 55 aa 55 01 00 00 00
00000080  aa 55 aa 55 01 00 00 00 aa 55 aa 55 01 00 00 00
00000090  00 20 00 b0 7e 04 03 00 aa 55 aa 55 01 00 00 00
000000a0  04 20 00 b0 21 00 00 00 aa 55 aa 55 01 00 00 00
          ...
000001a0  aa 55 aa 55 01 00 00 00 aa 55 aa 55 01 00 00 00
000001b0  aa 55 aa 55 01 00 00 00 00 00 00 00 00 00 00 00
          ...
000040f0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
                                  Child1-File Length Child1-File Address
00004100  00 00 00 00 00 00 00 00 9c 1f 05 00 00 00 10 00
          Child1-Image Type Child1-Reserved
00004110  00 00 00 00 ff ff ff ff be 00 00 ea 14 f0 9f e5
00004120  14 f0 9f e5 14 f0 9f e5 14 f0 9f e5 14 f0 9f e5
00004130  14 f0 9f e5 14 f0 9f e5 60 00 e0 00 c0 00 e0 00
```

Figure 4-37 Output Pack.bin

4.7.1 Image Type

The Pack mode provides four file types, Loader, Data, Environment and eMMC/SD Partition. The eMMC/SD Partition image type is used to record the partition information. This image type is only for eMMC/SD format using. The eMMC/SD Partition image type consists of the 48 bytes (Header). The format is shown in Figure 4-38.

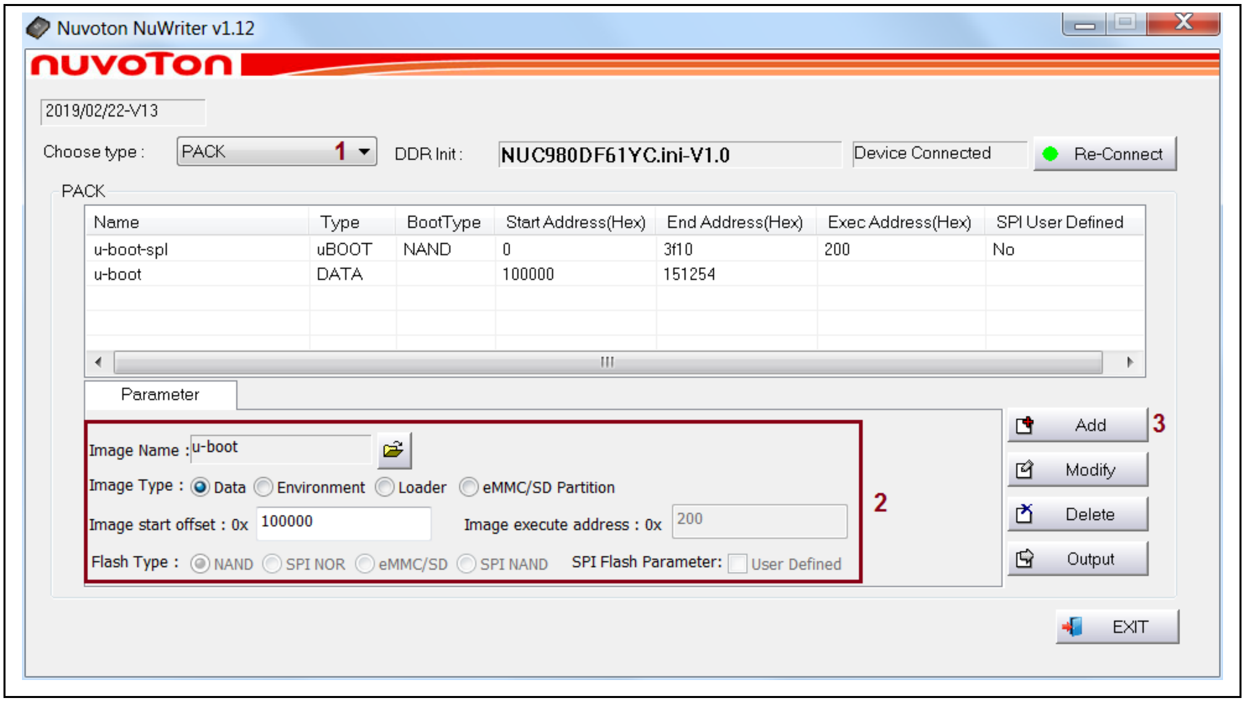
	0x0	**0x4**	**0x8**	**0xC**
0x00	Reserved	Partition Number	Reserved Space Size	Reserved
0x10	Partition1 Size	Partition1 Sector Size	Partition2 Size	Partition2 Sector Size
0x20	Partition3 Size	Partition3 Sector Size	Partition4 Size	Partition4 Sector Size

Figure 4-38 Pack Mode eMMC/SD Partition Header

4.7.2 Operation Steps

Pack mode can merge many images into a pack image, user can use the NuWriter to burn pack image into the device (NAND Flash, SPI Flash, eMMC/SD, SPI NAND Fash).

4.7.2.1 Add a New Image



2019/02/22-V13

Choose type : PACK

DDR Init : NUC980DF61YC.ini-V1.0

Device Connected

PACK

Name	Type	BootType	Start Address(Hex)	End Address(Hex)	Exec Address(Hex)	SPI User Defined
u-boot-spl	uBOOT	NAND	0	3f10	200	No
u-boot	DATA		100000	151254		

Image Name : u-boot

Image Type : ● Data ○ Environment ○ Loader ○ eMMC/SD Partition

Image start offset : 0x 100000

Image execute address : 0x 200

Flash Type : ● NAND ○ SPI NOR ○ eMMC/SD ○ SPI NAND

SPI Flash Parameter: ☐ User Defined

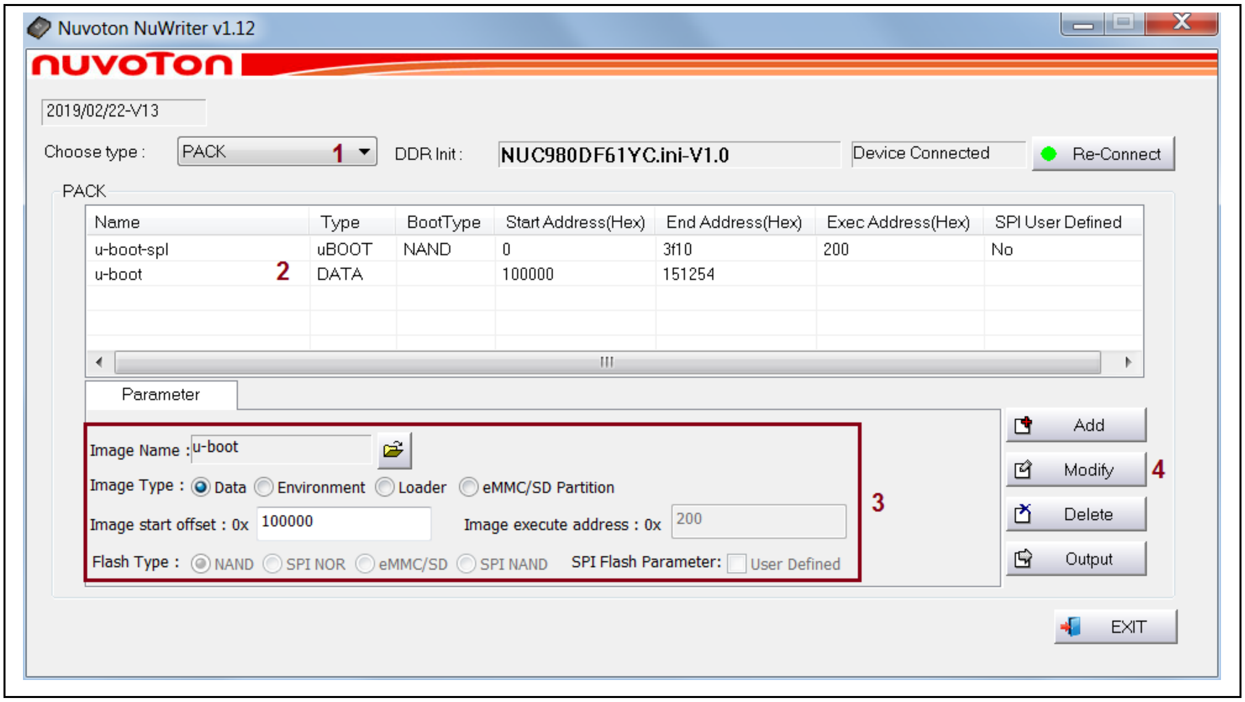
Figure 4-39 PACK – New Image

The steps below explain how to add a new image to pack list pane.

1. Select the “**Pack**”.
2. Fill in the image information：
   (1) Image Type: Data、Environment or Loader
   - Image Name: Browse the image file.
   - Image Type: Select the image type. (only one type can be selected)
   - Image execute address: Enter image execute address. Only is Loader Type is valid.
   - Image start offset: Enter image start offset.
   - Flash Type: Select the Flash type. (Only is Loader Type is valid)
   - The SPI Flash Parameter: User defined parameters. (Only for Flash Type is SPI NOR or SPI NAND and the Image Type is Loader).

   (2) Image Type: eMMC/SD Partition
   - Image Type: Select the eMMC/SD Partition type, and the eMMC/SD format window will pop up to provide users to configure the eMMC/SD partition size.. For the setting description, please refer to 4.5.2.3.
   - Image Name, Image execute address, Image start offset, Flash Type, and SPI Flash Parameter do not need to configure.
3. Click the “**Add**” button. If added image sucessfully, the image informations are shown in aboved pack list pane. The Name field is the Image file name. If the Image type is eMMC/SD Partition, this field will display "Partition_INFO". The Type field is Image type. If the Image type is eMMC/SD Partition, this field will display "FORMAT". The BootType field is Flash type. The Start Address (Hex) field is the burning start address and displayed by hexadecimal. The Exec Address (Hex) field is the execution address and displayed by hexadecimal. The SPI User Defined field displayed Yes or No to indicate the SPI parameter is user defined or not.

4.7.2.2 Modify an Image



2019/02/22-V13

Choose type : PACK

DDR Init : NUC980DF61YC.ini-V1.0

Device Connected

● Re-Connect

PACK

Name	Type	BootType	Start Address(Hex)	End Address(Hex)	Exec Address(Hex)	SPI User Defined
u-boot-spl	uBOOT	NAND	0	3f10	200	No
u-boot	DATA		100000	151254		

Image Name : u-boot

Image Type : ● Data ○ Environment ○ Loader ○ eMMC/SD Partition

Image start offset : 0x 100000

Image execute address : 0x 200

Flash Type : ● NAND ○ SPI NOR ○ eMMC/SD ○ SPI NAND

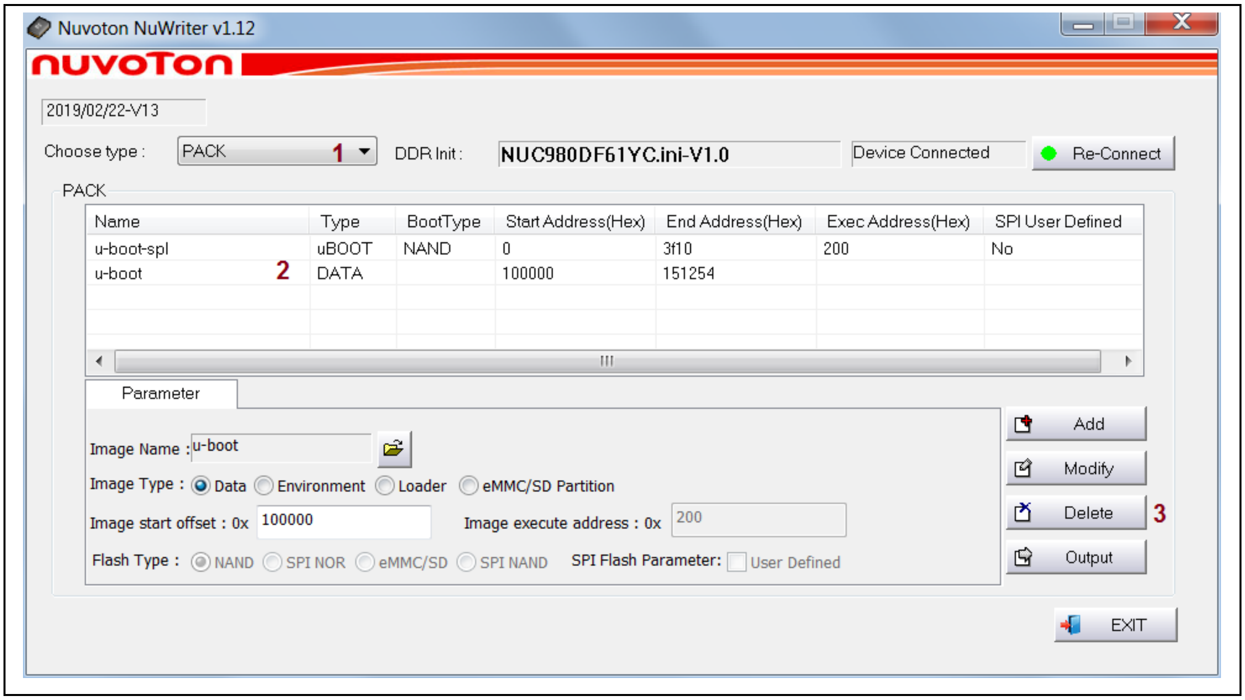
SPI Flash Parameter: ☐ User Defined

Figure 4-40 PACK – Modify Image

According to Figure 4-40 to modify image from pack list pane.

1. Select the “**PACK**”.
2. Click Image name in the pack list pane to modify the Image.
3. Fill in the image information:
   - Image Name: Browse the image file.
   - Image Type: Select the image type. (only one type can be selected)
   - Image execute address: Enter image execute address. Only Loader Type is valid.
   - Image start offset: Enter image start offset.
   - Flash Type: Select the Flash type. (Loader and Environment types are valid)
   - The SPI Flash Parameter: User defined parameters. (Only for Flash Type is SPI NOR or SPI NAND and the Image Type is Loader).
4. Click the “**Modify**” button.

4.7.2.3 Delete an Image



2019/02/22-V13

Choose type : PACK

DDR Init : NUC980DF61YC.ini-V1.0

Device Connected

Name	Type	BootType	Start Address(Hex)	End Address(Hex)	Exec Address(Hex)	SPI User Defined
u-boot-spl	uBOOT	NAND	0	3f10	200	No
u-boot	DATA		100000	151254		

Image Name : u-boot

Image Type : ● Data ○ Environment ○ Loader ○ eMMC/SD Partition

Image start offset : 0x 100000

Image execute address : 0x 200

Flash Type : ● NAND ○ SPI NOR ○ eMMC/SD ○ SPI NAND

SPI Flash Parameter: ☐ User Defined

Figure 4-41 PACK – Delete Image

According to Figure 4-41 to delete image from pack list pane:

1. Select “**PACK**”.
2. Click Image name on the pack list pane.
3. Click the “**Delete**” button.

4.7.2.4 Create a Pack Image

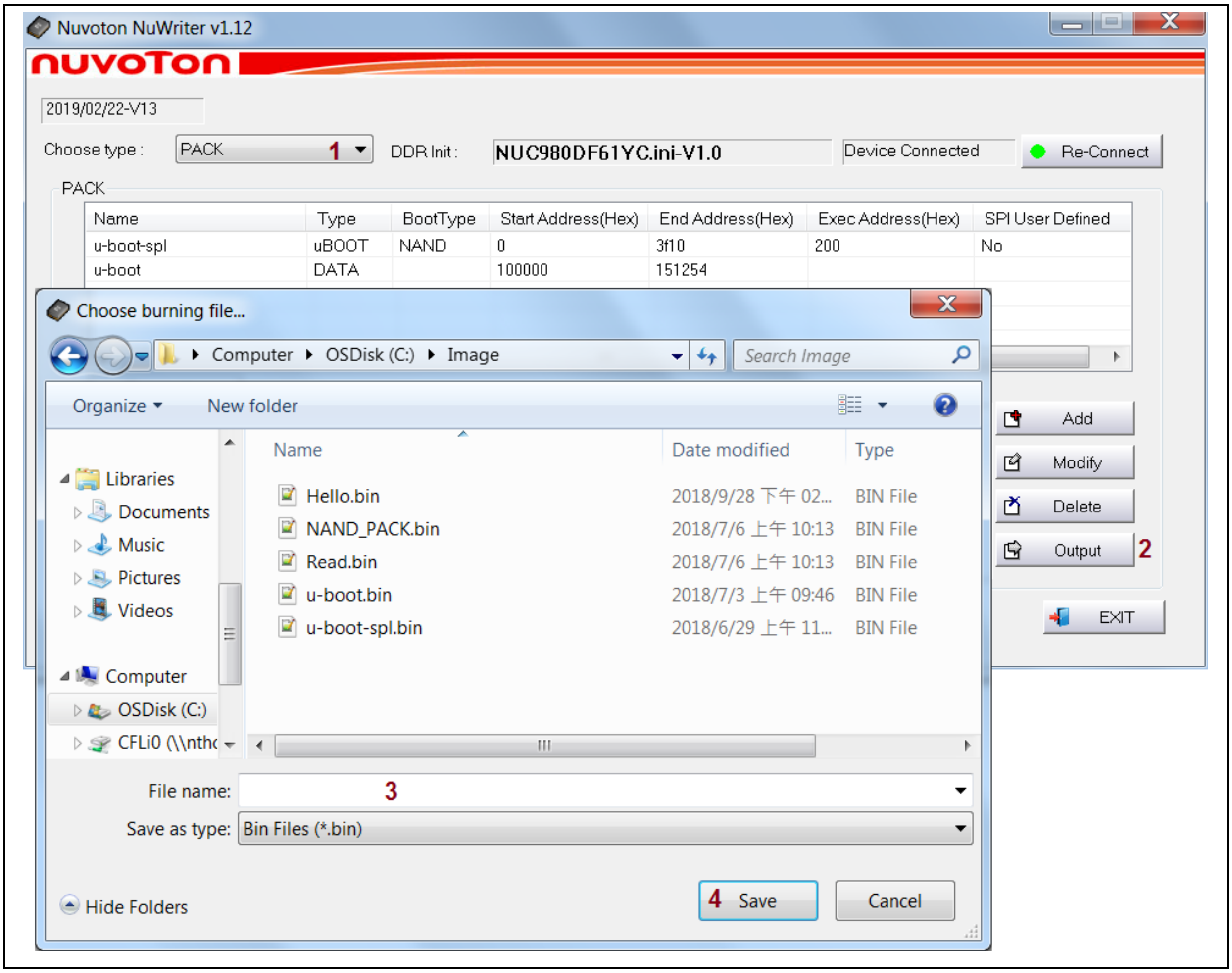
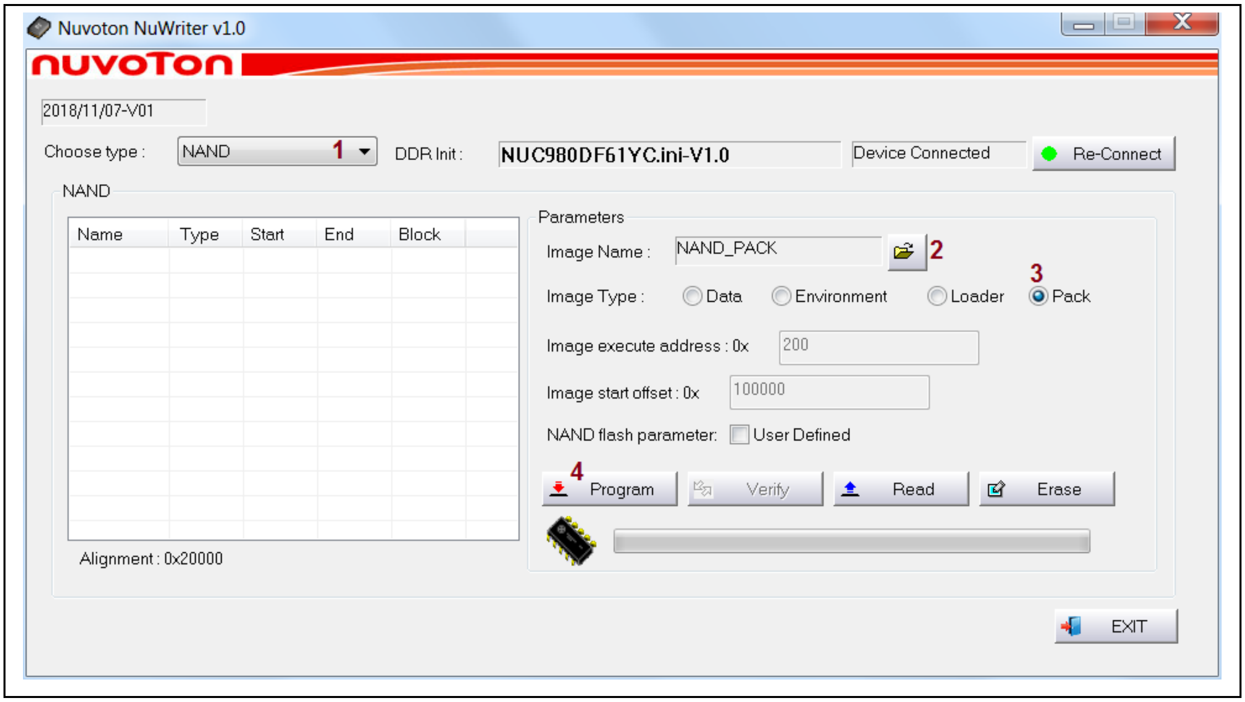


Figure 4-42 PACK – Output Pack Image

According to Figure 4-42 to output a pack image.

1. Select “**PACK**”.
2. Click the “**Output**” button.
3. Browse save file.
4. Click the ”**Save**” button to output a pack image.

4.7.2.5 *Burn a Pack Image*



2018/11/07-V01

Choose type : NAND

DDR Init : NUC980DF61YC.ini-V1.0

Device Connected

Image Name : NAND_PACK

Image Type : ○ Data ○ Environment ○ Loader ● Pack

Image execute address : 0x 200

Image start offset : 0x 100000

NAND flash parameter: ☐ User Defined

Alignment : 0x20000

Figure 4-43 PACK – Burn Pack Image

According to Figure 4-43 to burn a pack image into NAND Flash.

1. Select “**NAND**”.
2. Browse pack image.
3. Select image type to “**Pack**”.
4. Click the “**Program**” button.

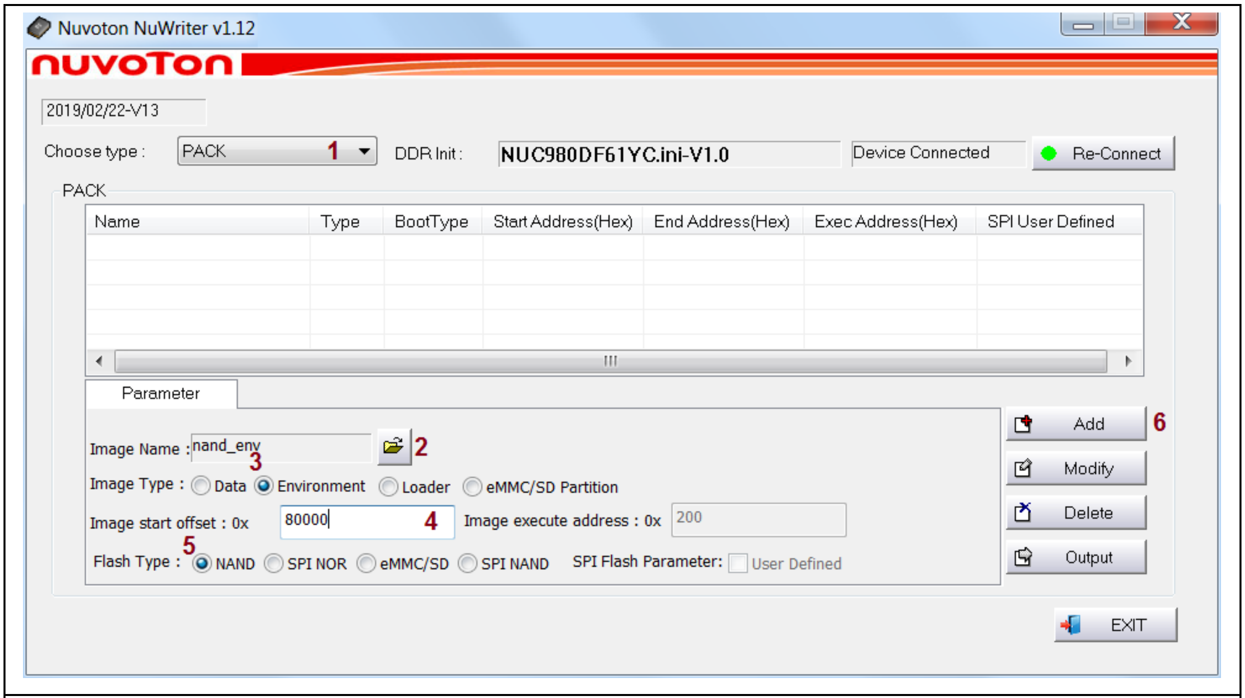
4.7.3 Create and Program a Pack Image

This section demonstrates how to make *nand_env.txt*, *u-boot.bin*, and *u-boot-spl.bin* into a Pack Image and burn it to NAND Flash. Pack Image will be merged from the following three files:

1. *Nand_env.txt* (Default : offset address is set to 0x80000) .
2. *u-boot.bin* (Default: offset address is set to 0x10000, execute address is set to 0xE00000) .
3. *u-boot-spl.bin* (Default:DDR execute address is set to 0x200).

Follow the steps below to generate a pack image:

1. Select “**PACK**”.
2. Selcet the “*nand_env.txt*” file path.
3. Select image type to “**Environment**”.
4. Offset address is set to 0x80000.
5. Select Flash Type to “**NAND**”.
6. Click the “**Add**” button.
7. After adding Image successfully, users can see the image information will be listed above. To repeat the step2~7 to add other images.
8. Finally, click the “**Output**” button to generate a package file.



Nuvoton NuWriter v1.12

2019/02/22-V13

Choose type : PACK 1

DDR Init : NUC980DF61YC.ini-V1.0

Device Connected

Name	Type	BootType	Start Address(Hex)	End Address(Hex)	Exec Address(Hex)	SPI User Defined

Parameter

Image Name : nand_env 2 3

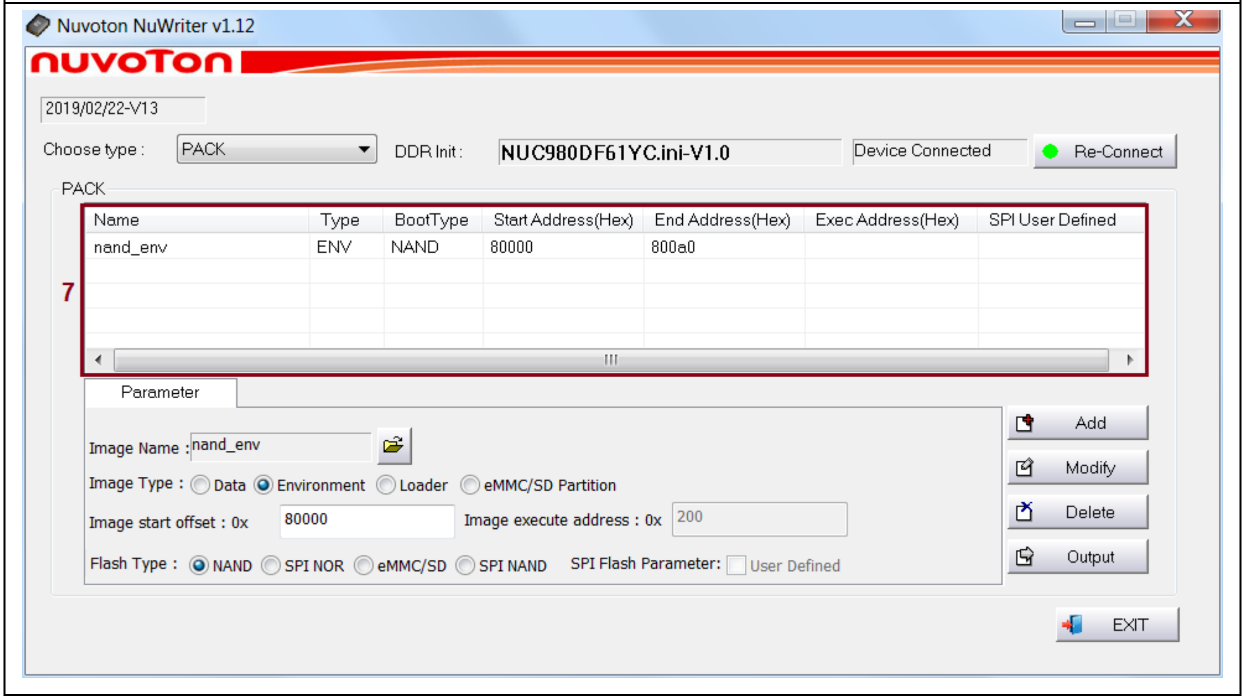
Image Type : ○ Data ● Environment ○ Loader ○ eMMC/SD Partition

Image start offset : 0x 80000 4

Image execute address : 0x 200

Flash Type : 5 ● NAND ○ SPI NOR ○ eMMC/SD ○ SPI NAND

SPI Flash Parameter: ☐ User Defined



Nuvoton NuWriter v1.12

2019/02/22-V13

Choose type : PACK

DDR Init : NUC980DF61YC.ini-V1.0

Device Connected

Name	Type	BootType	Start Address(Hex)	End Address(Hex)	Exec Address(Hex)	SPI User Defined
nand_env	ENV	NAND	80000	800a0		

7

Parameter

Image Name : nand_env

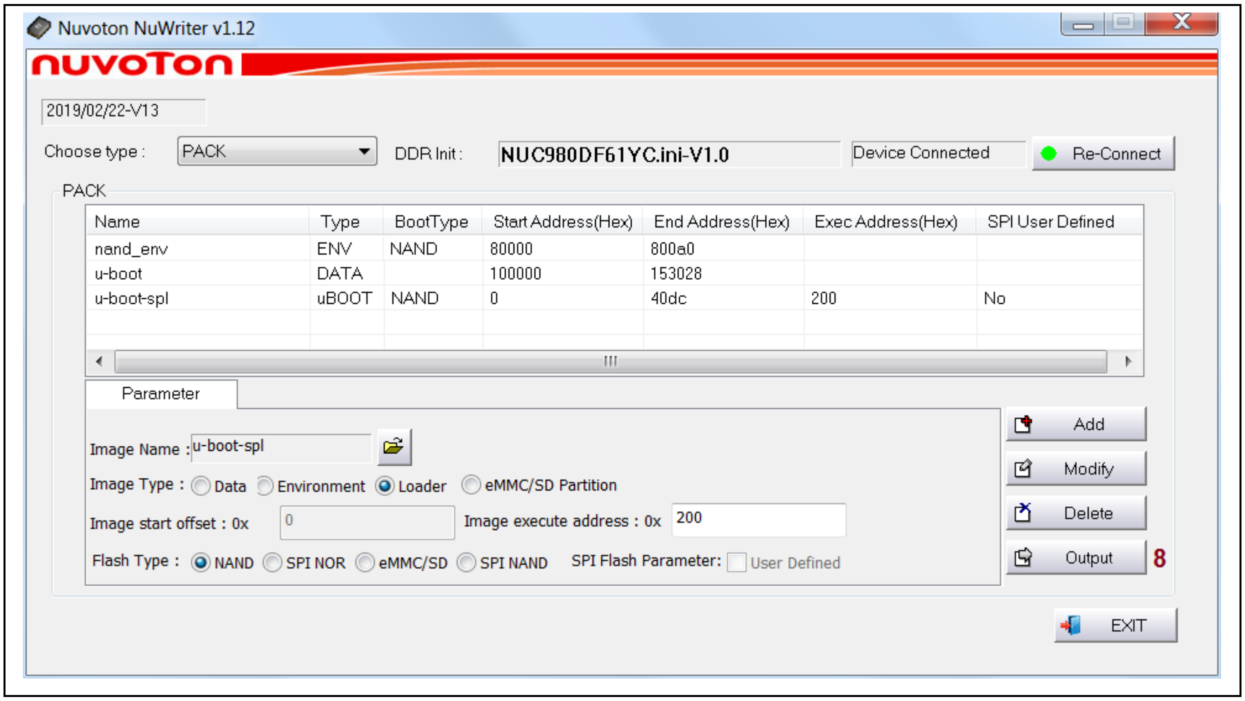
Image Type : ○ Data ● Environment ○ Loader ○ eMMC/SD Partition

Image start offset : 0x 80000

Image execute address : 0x 200

Flash Type : ● NAND ○ SPI NOR ○ eMMC/SD ○ SPI NAND

SPI Flash Parameter: ☐ User Defined



2019/02/22-V13

Choose type : PACK

DDR Init : NUC980DF61YC.ini-V1.0

Device Connected

Name	Type	BootType	Start Address(Hex)	End Address(Hex)	Exec Address(Hex)	SPI User Defined
nand_env	ENV	NAND	80000	800a0		
u-boot	DATA		100000	153028		
u-boot-spl	uBOOT	NAND	0	40dc	200	No

Image Name : u-boot-spl

Image Type : ○ Data ○ Environment ● Loader ○ eMMC/SD Partition

Image start offset : 0x 0

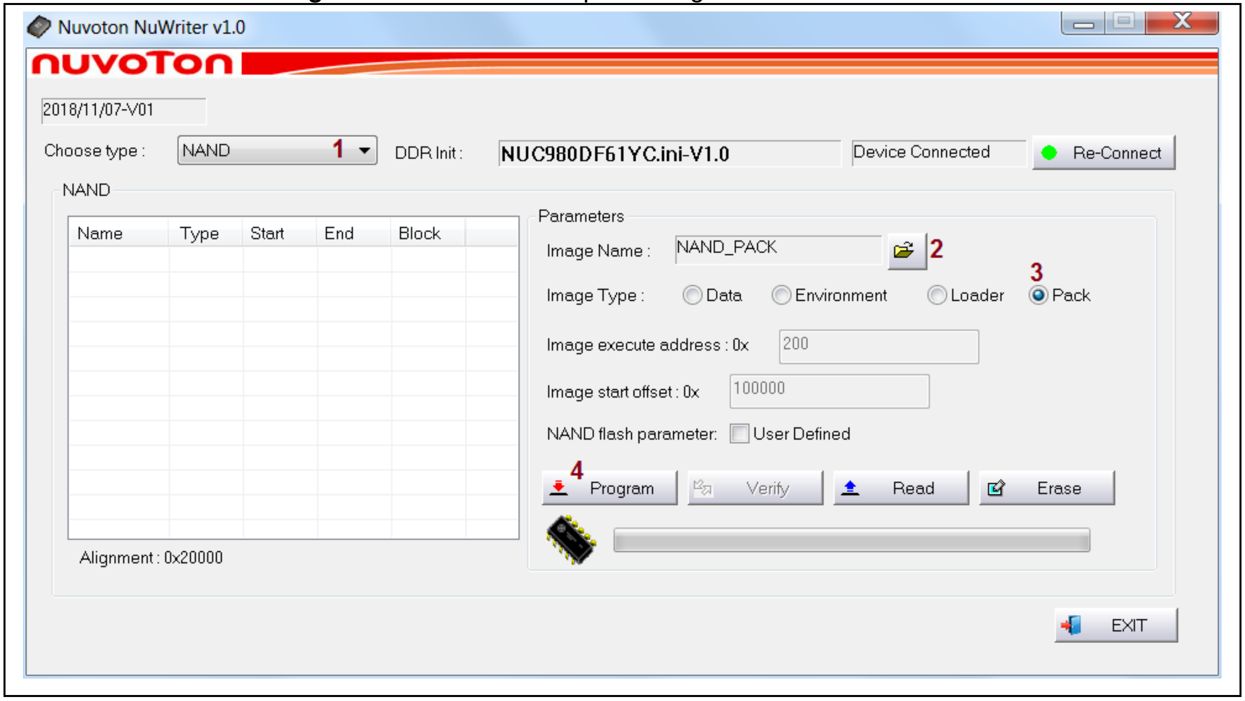
Image execute address : 0x 200

Flash Type : ● NAND ○ SPI NOR ○ eMMC/SD ○ SPI NAND

SPI Flash Parameter: ☐ User Defined

The following steps can be used to burn a pack image into NAND Flash.

1. Select “**NAND**”.
2. Browse pack image.
3. Select image type to “**Pack**”.
4. Click the “**Program**” button to burn a pack image into NAND Flash.



2018/11/07-V01

Choose type : NAND

DDR Init : NUC980DF61YC.ini-V1.0

Device Connected

Alignment : 0x20000

Image Name : NAND_PACK

Image Type : ○ Data ○ Environment ○ Loader ● Pack

Image execute address : 0x 200

Image start offset : 0x 100000

NAND flash parameter: ☐ User Defined

4.8 Mass Production Mode

Mass Production mode provides the multiple chip programming functionality, which is designed for mass-production programming. This function automatically detects how many devices are connected to the PC, and simultaneously support up to eight devices to perform the programming operation on the same storage device. The file used in this mode must be created by the Pack mode. Mass production mode operation steps includes erase, program and verify.

4.8.1 Operation Steps

Because the Pack mode has merged all the images into one file, the user only needs to select programming file and storage type, finally click the “**Start**” button to program multiple devices at the same time. If the storage device is NAND, users can check the box to configure parameters.

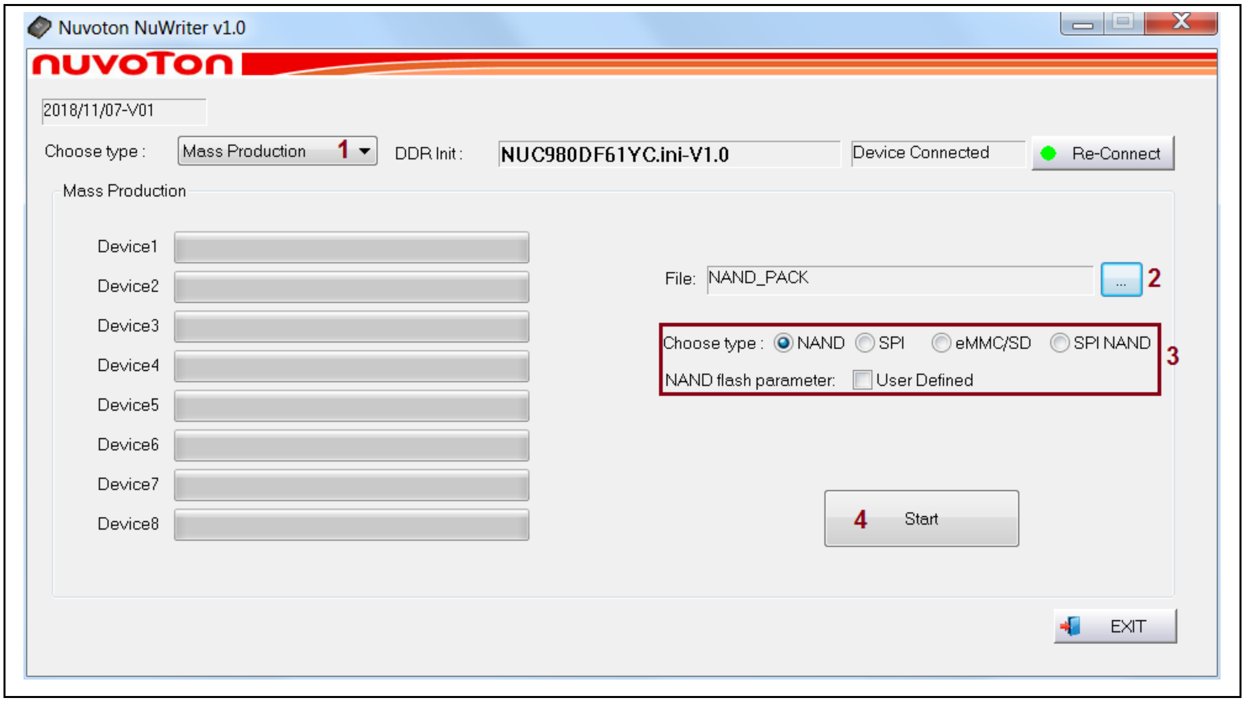
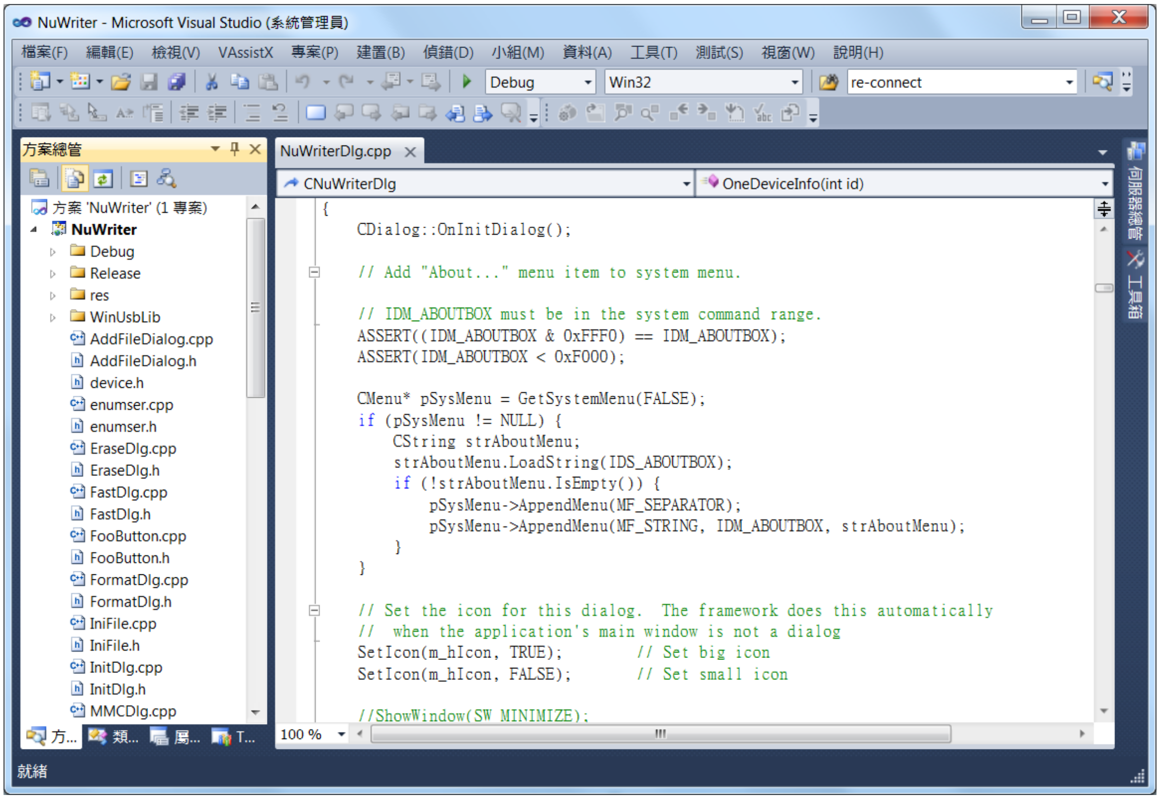


Figure 4-44 Mass Production

5 NUWRITER SOURCE CODE

5.1 Software Code

The NuWriter uses the VS2010 development environment, as shown in Figure 5-1.



```
{
    CDialog::OnInitDialog();

    // Add "About..." menu item to system menu.

    // IDM_ABOUTBOX must be in the system command range.
    ASSERT((IDM_ABOUTBOX & 0xFFF0) == IDM_ABOUTBOX);
    ASSERT(IDM_ABOUTBOX < 0xF000);

    CMenu* pSysMenu = GetSystemMenu(FALSE);
    if (pSysMenu != NULL) {
        CString strAboutMenu;
        strAboutMenu.LoadString(IDS_ABOUTBOX);
        if (!strAboutMenu.IsEmpty()) {
            pSysMenu->AppendMenu(MF_SEPARATOR);
            pSysMenu->AppendMenu(MF_STRING, IDM_ABOUTBOX, strAboutMenu);
        }
    }

    // Set the icon for this dialog.  The framework does this automatically
    //  when the application's main window is not a dialog
    SetIcon(m_hIcon, TRUE);         // Set big icon
    SetIcon(m_hIcon, FALSE);        // Set small icon

    //ShowWindow(SW_MINIMIZE);
```

Figure 5-1 NuWriter – SW Project

User can modify dialog on the resource view to meet all your needs.

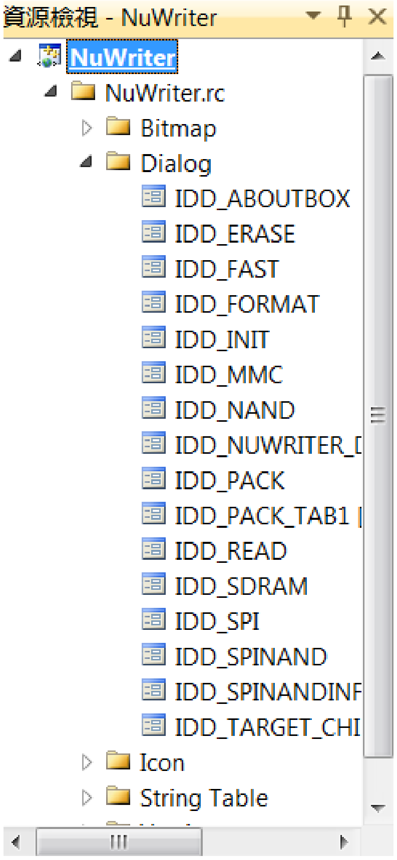


Figure 5-2 NuWriter – Resource View

After double click IDD_SRAM, user can see the following screen.

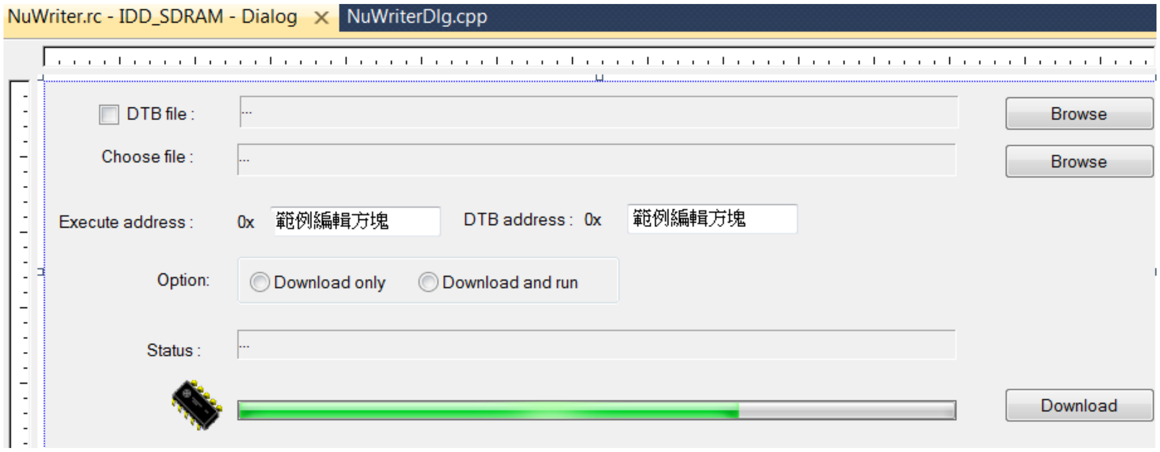
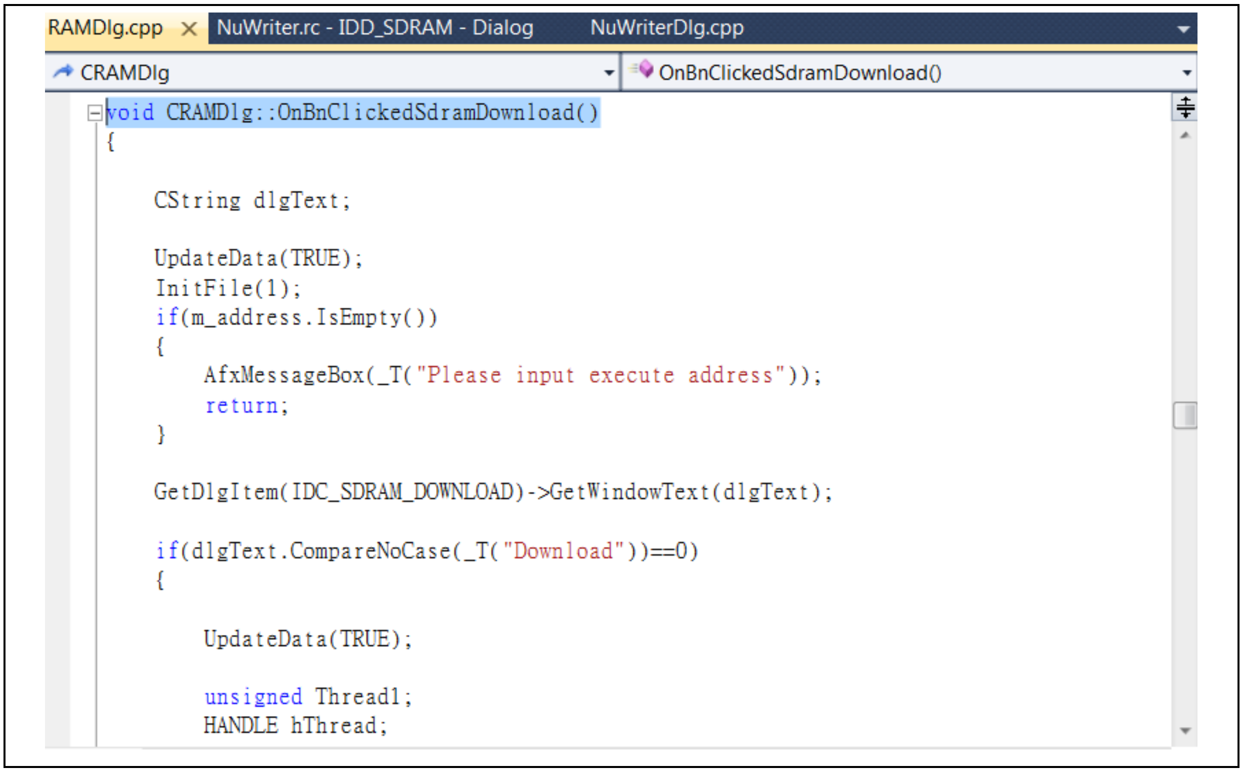


Figure 5-3 IDD_SDRAM Dialog

And then double click Download button, user can see source code of download button.



```
void CRAMDlg::OnBnClickedSdramDownload()
{

    CString dlgText;

    UpdateData(TRUE);
    InitFile(1);
    if(m_address.IsEmpty())
    {
        AfxMessageBox(_T("Please input execute address"));
        return;
    }

    GetDlgItem(IDC_SDRAM_DOWNLOAD)->GetWindowText(dlgText);

    if(dlgText.CompareNoCase(_T("Download"))==0)
    {

        UpdateData(TRUE);

        unsigned Thread1;
        HANDLE hThread;
```

Figure 5-4 OnBnClockedSdramDownload Function

There are many buttons to complete all functions in the NuWriter tool. User can view the source code of button to understand how communication between the NuWriter and NCU980 series MPU. The main source code is NucWinUsb.h and NucWinUsb.cpp.

NucWinUsb.h is listed below.

```
static const GUID OSR_DEVICE_INTERFACE[] = { 0xD696BFEB, 0x1734, 0x417d,
{ 0x8A, 0x04, 0x86,0xD0,0x10,0x71,0xC5,0x12 } };
typedef struct PIPE_ID
{
     UCHAR  PipeInId;
     UCHAR  PipeOutId;
}Pipe_Id,*PPipe_Id;

typedef struct WINUSBHANDLE
{
     HANDLE hDeviceHandle;
     WINUSB_INTERFACE_HANDLE hUSBHandle;
     Pipe_Id pipeid;
     CString DevicePath;
}SWinUsbHandle,*PSWinUsbHandle;

#define MAX_DEVICE_NUM 32
class CNucWinUsb
```

```
{
     public:
          CNucWinUsb();
          SWinUsbHandle WinUsbHandle[MAX_DEVICE_NUM];
          int WinUsbNumber;
          UCHAR DeviceSpeed;
          /* Enable WinUSB driver */
          BOOL EnableWinUsbDevice(void);
          /* Set mode */
          BOOL NUC_SetType(int id,USHORT type,UCHAR * ack,ULONG len);
          /* Write data to deivce */
          BOOL NUC_WritePipe(int id,UCHAR *buf,ULONG len);
          /* Read data from deivce */
          BOOL NUC_ReadPipe(int id,UCHAR *buf,ULONG len);
          /* Disable WinUSB driver */
          void NUC_CloseHandle(void);
          /* Check whether ID is connected or not */
          BOOL NUC_CheckFw(int id);
     protected:
          BOOL GetDeviceHandle(void);
          BOOL GetWinUSBHandle(void);
          BOOL GetUSBDeviceSpeed(void);
          BOOL QueryDeviceEndpoints(void);
};
static CNucWinUsb NucUsb;
```

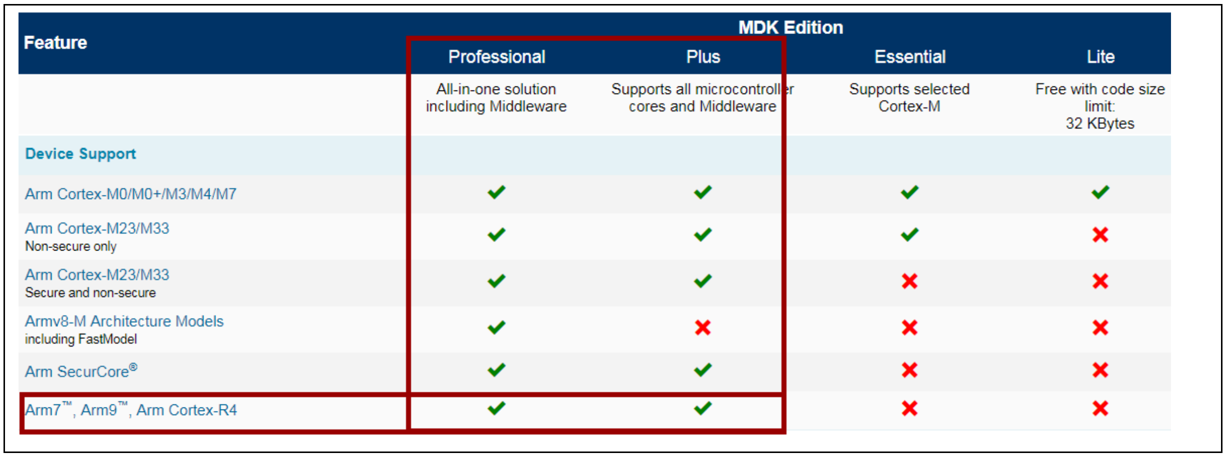
Plug the NUC980 series into the PC, they can communicate via the following code.

```
bResult=NucUsb.EnableWinUsbDevice(); //Enable WinUsb driver
bResult=NucUsb.NUC_CheckFw(0); //Check device0 connect to PC
if(bResult==FALSE)
{
     AfxMessageBox(_T("Please reset device and Re-connect now !!!\n"));
return FALSE;
}
USHORT typeack=0x0;
/* Set SDRAM mode,others mode can refer to Serial.h */
bResult=NucUsb.NUC_SetType(0,SDRAM,(UCHAR *)&typeack,sizeof(typeack));
if(bResult==FALSE)
{
AfxMessageBox(_T("typeack failed !!!\n"));
NucUsb.NUC_CloseHandle();
return FALSE;
```

```
}

........skips........

/* Write SDRAM deader to device  */

bResult=NucUsb.NUC_WritePipe(0,(UCHAR *)lpBuffer,
sizeof(SDRAM_RAW_TYPEHEAD));
if(bResult==FALSE)
{
     delete []lpBuffer;
     NucUsb.NUC_CloseHandle();
     fclose(fp);
     AfxMessageBox(_T("Write SDRAM head error\n"));
     return FALSE;
}

/* waiting for devcie return ack, and check to receive sdram header */

bResult=NucUsb.NUC_ReadPipe(0,(UCHAR *)&ack,4);
if(bResult==FALSE)
{
     delete []lpBuffer;
     NucUsb.NUC_CloseHandle();
     fclose(fp);
     AfxMessageBox(_T("SDRAM head ack error\n"));
     return FALSE;
}
........skips........
while(scnt>0)
{
fread(lpBuffer,BUF_SIZE,1,fp);
/* Write data to device,length is 4096bytes */

bResult=NucUsb.NUC_WritePipe(0,(UCHAR *)lpBuffer,BUF_SIZE);
if(WaitForSingleObject(m_ExitEvent, 0) != WAIT_TIMEOUT)
     {
               delete []lpBuffer;
               fclose(fp);
               return FALSE;
     }
     if(bResult==TRUE)
     {
          total+=BUF_SIZE;
          pos=(int)(((float)(((float)total/(float)file_len))*100));
          posstr.Format(_T("%d%%"),pos);
          /*Waiting for device return ack */
          bResult=NucUsb.NUC_ReadPipe(0,(UCHAR *)&ack,4);
          if(bResult==FALSE || ack!=BUF_SIZE)
          {
               delete []lpBuffer;
               NucUsb.NUC_CloseHandle();
               fclose(fp);
               AfxMessageBox(_T("ACK error !"));
               return FALSE;
          }
........skips........
}
```

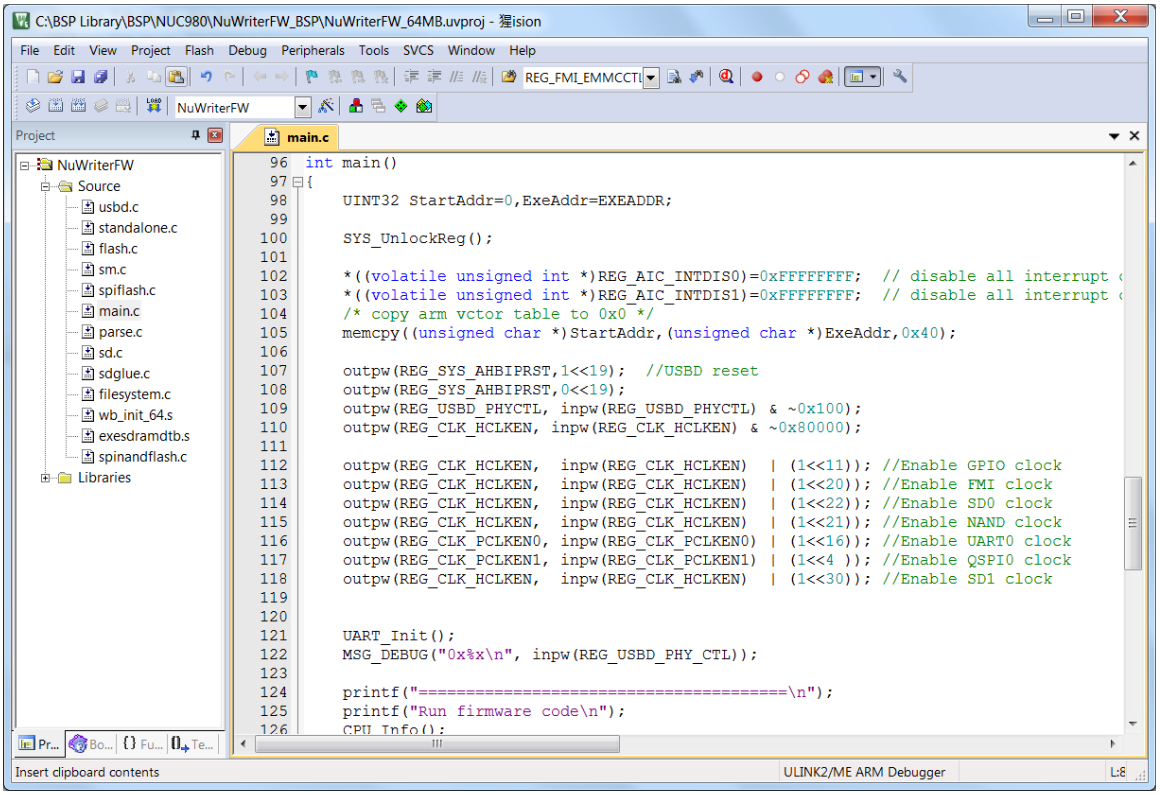
5.2 Firmware Code (xusb.bin)

Keil IDE is used as NuWriter Firmware (*xusb.bin*) development environment. To support ARM9, MDK Plus or Professional edition shall be used.



Feature	MDK Edition			
	Professional	Plus	Essential	Lite
	All-in-one solution including Middleware	Supports all microcontroller cores and Middleware	Supports selected Cortex-M	Free with code size limit: 32 KBytes
Device Support				
Arm Cortex-M0/M0+/M3/M4/M7	✓	✓	✓	✓
Arm Cortex-M23/M33 Non-secure only	✓	✓	✓	✗
Arm Cortex-M23/M33 Secure and non-secure	✓	✓	✗	✗
Armv8-M Architecture Models including FastModel	✓	✗	✗	✗
Arm SecurCore®	✓	✓	✗	✗
Arm7™, Arm9™, Arm Cortex-R4	✓	✓	✗	✗

xusb.bin uses the Keil development environment as shown in Figure 5-5.



```
int main()
{
    UINT32 StartAddr=0,ExeAddr=EXEADDR;

    SYS_UnlockReg();

    *((volatile unsigned int *)REG_AIC_INTDIS0)=0xFFFFFFFF;  // disable all interrupt
    *((volatile unsigned int *)REG_AIC_INTDIS1)=0xFFFFFFFF;  // disable all interrupt
    /* copy arm vctor table to 0x0 */
    memcpy((unsigned char *)StartAddr,(unsigned char *)ExeAddr,0x40);

    outpw(REG_SYS_AHBIPRST,1<<19);  //USBD reset
    outpw(REG_SYS_AHBIPRST,0<<19);
    outpw(REG_USBD_PHYCTL, inpw(REG_USBD_PHYCTL) & ~0x100);
    outpw(REG_CLK_HCLKEN, inpw(REG_CLK_HCLKEN) & ~0x80000);

    outpw(REG_CLK_HCLKEN,  inpw(REG_CLK_HCLKEN)  | (1<<11)); //Enable GPIO clock
    outpw(REG_CLK_HCLKEN,  inpw(REG_CLK_HCLKEN)  | (1<<20)); //Enable FMI clock
    outpw(REG_CLK_HCLKEN,  inpw(REG_CLK_HCLKEN)  | (1<<22)); //Enable SD0 clock
    outpw(REG_CLK_HCLKEN,  inpw(REG_CLK_HCLKEN)  | (1<<21)); //Enable NAND clock
    outpw(REG_CLK_PCLKEN0, inpw(REG_CLK_PCLKEN0) | (1<<16)); //Enable UART0 clock
    outpw(REG_CLK_PCLKEN1, inpw(REG_CLK_PCLKEN1) | (1<<4 )); //Enable QSPI0 clock
    outpw(REG_CLK_HCLKEN,  inpw(REG_CLK_HCLKEN)  | (1<<30)); //Enable SD1 clock


    UART_Init();
    MSG_DEBUG("0x%x\n", inpw(REG_USBD_PHY_CTL));

    printf("=======================================\n");
    printf("Run firmware code\n");
    CPU_Info();
```

Figure 5-5 NuWriterFW – Project

Firmware code (*xusb.bin*) is a communication between the NuWriter and NUC980 series. First, the NuWriter transmits xusb.bin to NUC980 series and executed. The NUC980 series waiting for the PC transmit command on ParseFlashType(void) functions. If users want to add new commands to NuWriter, user can add value of _usbd_flash_type, and modify ParseFlashType(void) and usbd_control_packet(void) from parse.c file. The source code is listed below.

```
INT ParseFlashType(void){
switch (_usbd_flash_type){
    /* Add new commands from nuwriter transmit */
    /*
    case yyyyy:
      TODO: Add your control notification handler code here
    break;
    */
    case USBD_FLASH_SDRAM: {
        UXmodem_SDRAM();
        _usbd_flash_type = -1;
    }
    break;
    case USBD_FLASH_MMC: {
        UXmodem_MMC();
        _usbd_flash_type = -1;
    }
    break;
    case USBD_FLASH_NAND: {
        UXmodem_NAND();
        _usbd_flash_type = -1;
    }
    break;
    case USBD_FLASH_SPI: {
        UXmodem_SPI();
        _usbd_flash_type = -1;
    }
    break;
    case USBD_FLASH_SPINAND: {
        UXmodem_SPINAND();
        _usbd_flash_type = -1;
    }
    break;
    case USBD_INFO: {
        UXmodem_INFO();
        _usbd_flash_type = -1;
    }
```

usbd_control_packet() function in usbd.c, as follows:

```
void usbd_control_packet()
{
     ........skips........
     if (_usb_cmd_pkt.bRequest == 0xb0){
     /* Add new commands from nuwriter transmit */
     /*
     if (_usb_cmd_pkt.wValue == (xxxxx)){
          _usbd_flash_type = yyyyy;
          outpw(REG_USBD_CEP_CTRL_STAT, CEP_NAK_CLEAR);
          // clear nak so that sts stage is complete
     }
     xxxxx : NuWriter transmit value
     yyyyy : ParseFlashType(void) Parse value
     */

     if (_usb_cmd_pkt.wValue == (USBD_BURN_TYPE+USBD_FLASH_SDRAM)){
          _usbd_flash_type = USBD_FLASH_SDRAM;
          outpw(REG_USBD_CEP_CTRL_STAT, CEP_NAK_CLEAR);
```

```
                // clear nak so that sts stage is complete
        }
        else if (_usb_cmd_pkt.wValue == (USBD_BURN_TYPE+USBD_FLASH_NAND)){
                _usbd_flash_type = USBD_FLASH_NAND;
                // clear nak so that sts stage is complete
                outpw(REG_USBD_CEP_CTRL_STAT, CEP_NAK_CLEAR);
        }
        else if (_usb_cmd_pkt.wValue == (USBD_BURN_TYPE+USBD_FLASH_NAND_RAW))
        {       _usbd_flash_type = USBD_FLASH_NAND_RAW;
                // clear nak so that sts stage is complete
                outpw(REG_USBD_CEP_CTRL_STAT, CEP_NAK_CLEAR);
        }
        else if (_usb_cmd_pkt.wValue == (USBD_BURN_TYPE+USBD_FLASH_MMC)){
                _usbd_flash_type = USBD_FLASH_MMC;
                // clear nak so that sts stage is complete
                outpw(REG_USBD_CEP_CTRL_STAT, CEP_NAK_CLEAR);
        }
        ........skips........
}
```

After editing done, you can compile a *xusb.bin*.

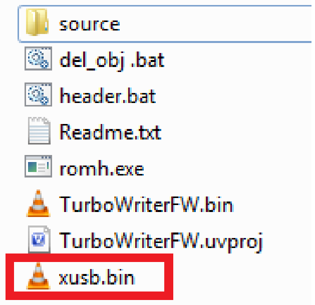


Figure 5-6 NuWriterFW – Folder

And cover xusb.bin in the NuWriter folder.

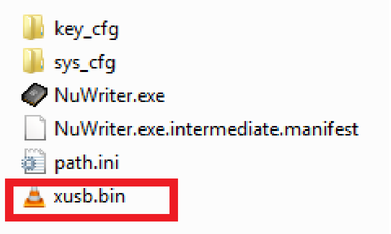


Figure 5-7 NuWriter – Folder

6 REVISION HISTORY

Date	Revision	Description
2018.09.17	1.00	Initially issued.
2019.01.11	1.01	Modified the format and enhance readability.
2019.01.24	1.10	eMMC/SD Format: Supported up to 4 partitions.
2019.02.22	1.11	Pack Mode: Supported eMMC/SD Partition image type.
2019.04.22	1.12	Editorial change

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.