ARM926EJ-S™

32-bit Microprocessor

NUC970/N9H30 NuWriter User Manual

Table of Contents

1 OVERVIEW

The Nuvoton NUC970/N9H30 Series MPU supports the following four system boot-up conditions:

1. Boot from eMMC device
2. Boot from SPI FLASH device
3. Boot from NAND-type Flash ROM device
4. USB ISP mode

All these four conditions are selected by power-setting (PA0 and PA1). This tool can help users to burn their image binary code into the on-board ROM device when the system enters USB ISP mode. On-Board ROM device includes eMMC device, SPI Flash device and NAND Flash.

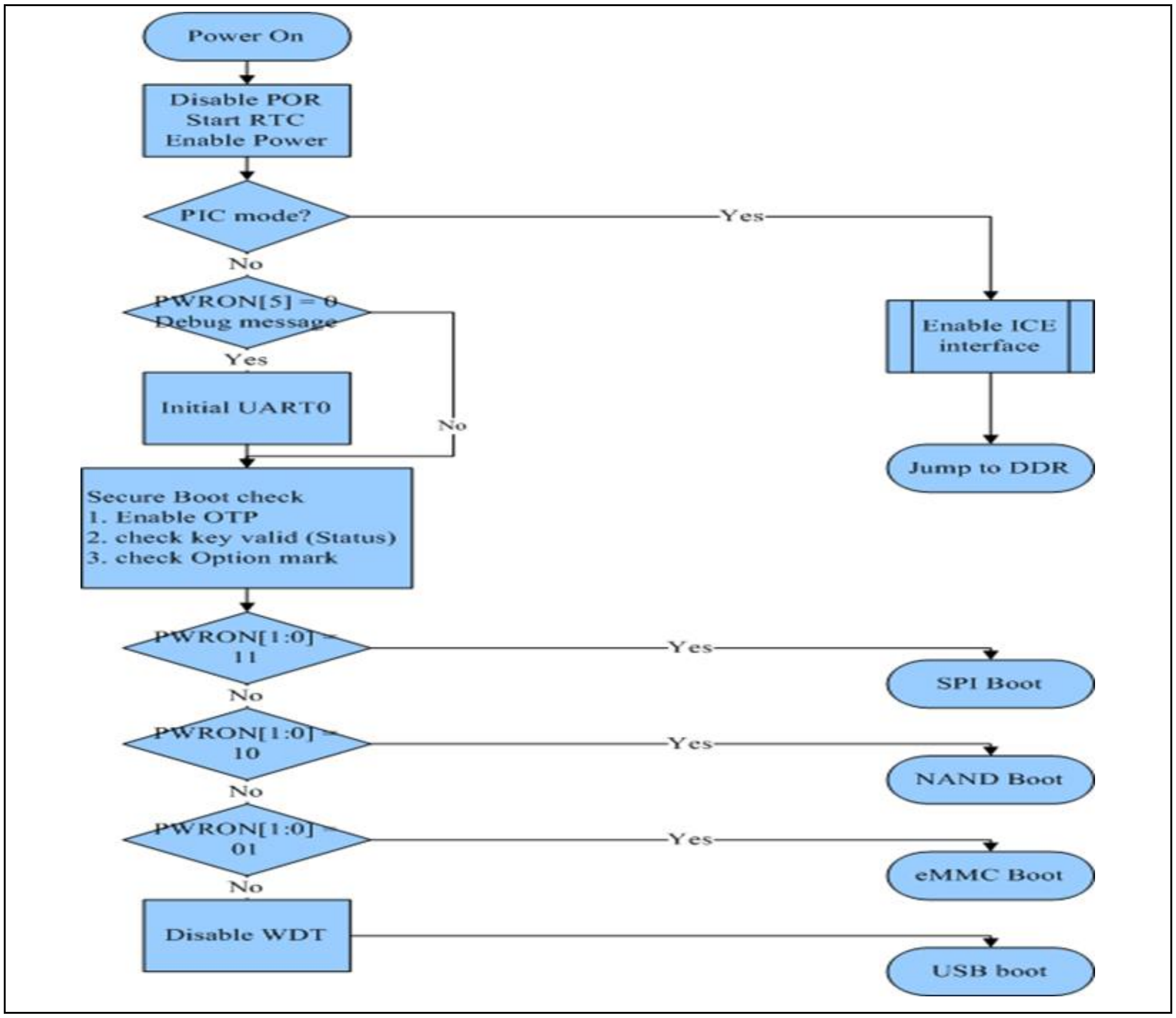


Figure 1-1 Startup Flow

The internal ROM boot flow is as above. The power-on setting is enabled or disabled to decide startup flow. For example, watchdog enable/disable, timer enable/disable, selection clock source and JTAG enable/disable. The table shown below describes the definition of each power-on setting bit.

Power-On Setting Pin	Description	Power-On Setting Register Bit
USB0_ID	**USB Port 0 Role Selection**	PWRON[16]

	0 = USB Port 0 act as a USB host. 1 = USB Port 0 act as a USB device.	
PA[1:0]	**Boot Source Selection** 00 = Boot from USB. 01 = Boor from eMMC. 10 = Boot from NAND Flash. 11 = Boot from SPI Flash.	PWRON[1:0]
PA.2	**System Clock Source Selection** 0 = System clock is from 12 MHz crystal. 1 = System clock is from UPLL output.	PWRON[2]
PA.3	**Watchdog Timer (WDT) Enabled/Disabled Selection** 0 = WDT Disabled after power-on. 1 = WDT Enabled after power-on.	PWRON[3]
PA.4	**JTAG Interface ON/OFF Selection** 0 = Pin PJ[4:0] used as GPIO pin. 1 = Pin PJ[4:0] used as JTAG interface.	PWRON[4]
PA.5	**UART 0 Debug Message Output ON/OFF Selection** 0 = UART 0 debug message output ON. 1 = UART 0 debug message output OFF.	PWRON[5]
PA[7:6]	**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.	PWRON[7:6]
PA[9:8]	**NAND Flash ECC Type Selection** 00 = NAND Flash ECC type is BCH T12. 01 = NAND Flash ECC type is BCH T15. 10 = NAND Flash ECC type is BCH T24. 11 = Ignore Power-On Setting.	PWRON[9:8]

Table 1-2 Power On Setting

1.1 Boot from NAND

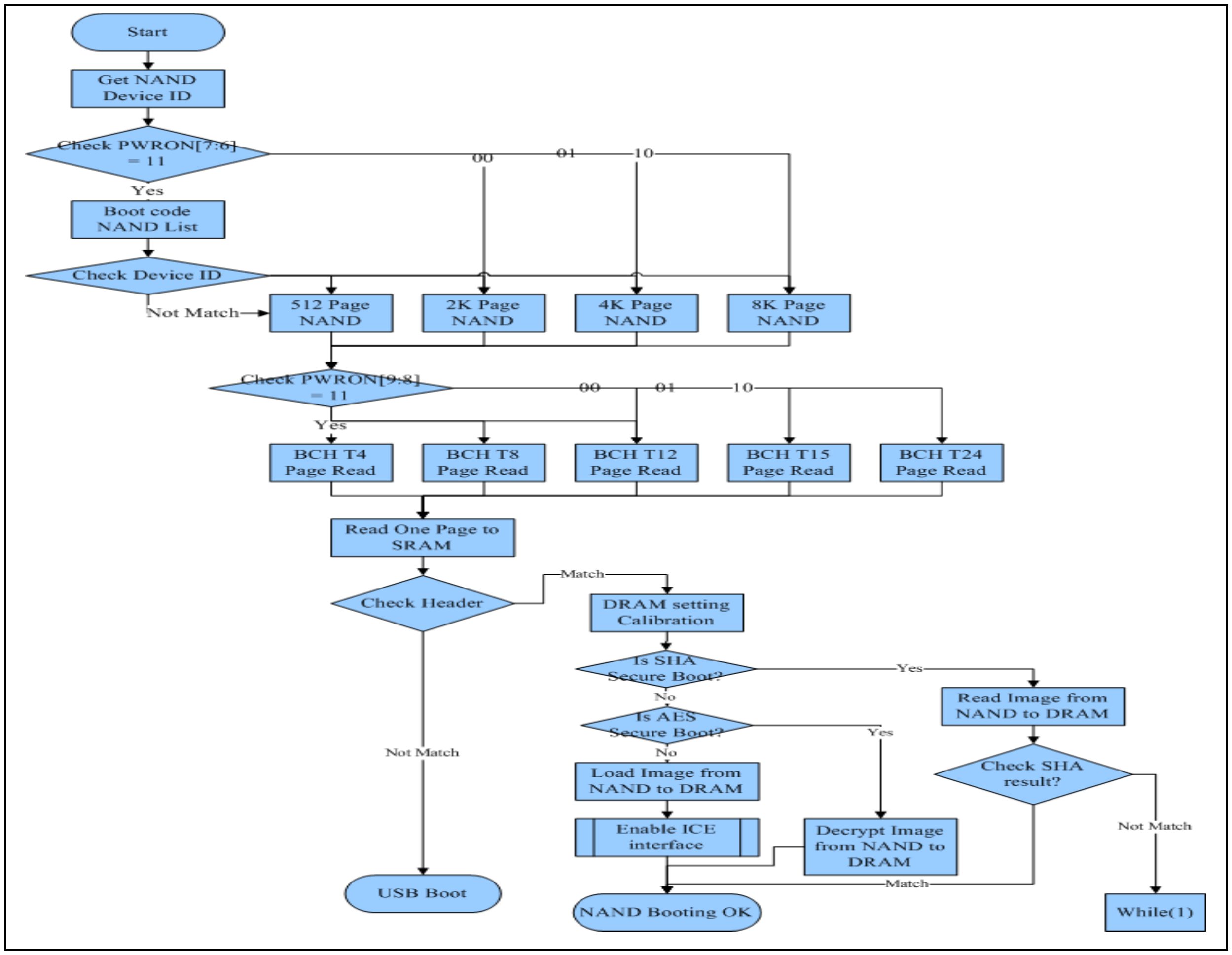


Figure 1-2 Nand Startup Flow

NAND boot start flow detect PWRON[7:6] to determine page size and PWRON[9:8] to determine NAND flash ECC type, then read the block 0 in NAND flash to determine boot code marker is correct. Boot code marker is refer to 4.2.1.1uBoot type. If boot code marker cannot be found in the block0, block1, block2 and block3, it is skipping from “Check Header” states to USB boot. When boot code marker detects correct, then do DDR initialization. Next step is to check SHA enabled, and then calculate SHA, if SHA is not enabled, next step is to check AES enabled, and then decode AES. It copy just uboot image from NAND to DDR, and execute uboot image.

Note: uBoot image of total size plus DDR parameter of total size is equal or lesser than one block size.

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

Note: N9H30 does not support secure boot.

1.2 Boot from SPI

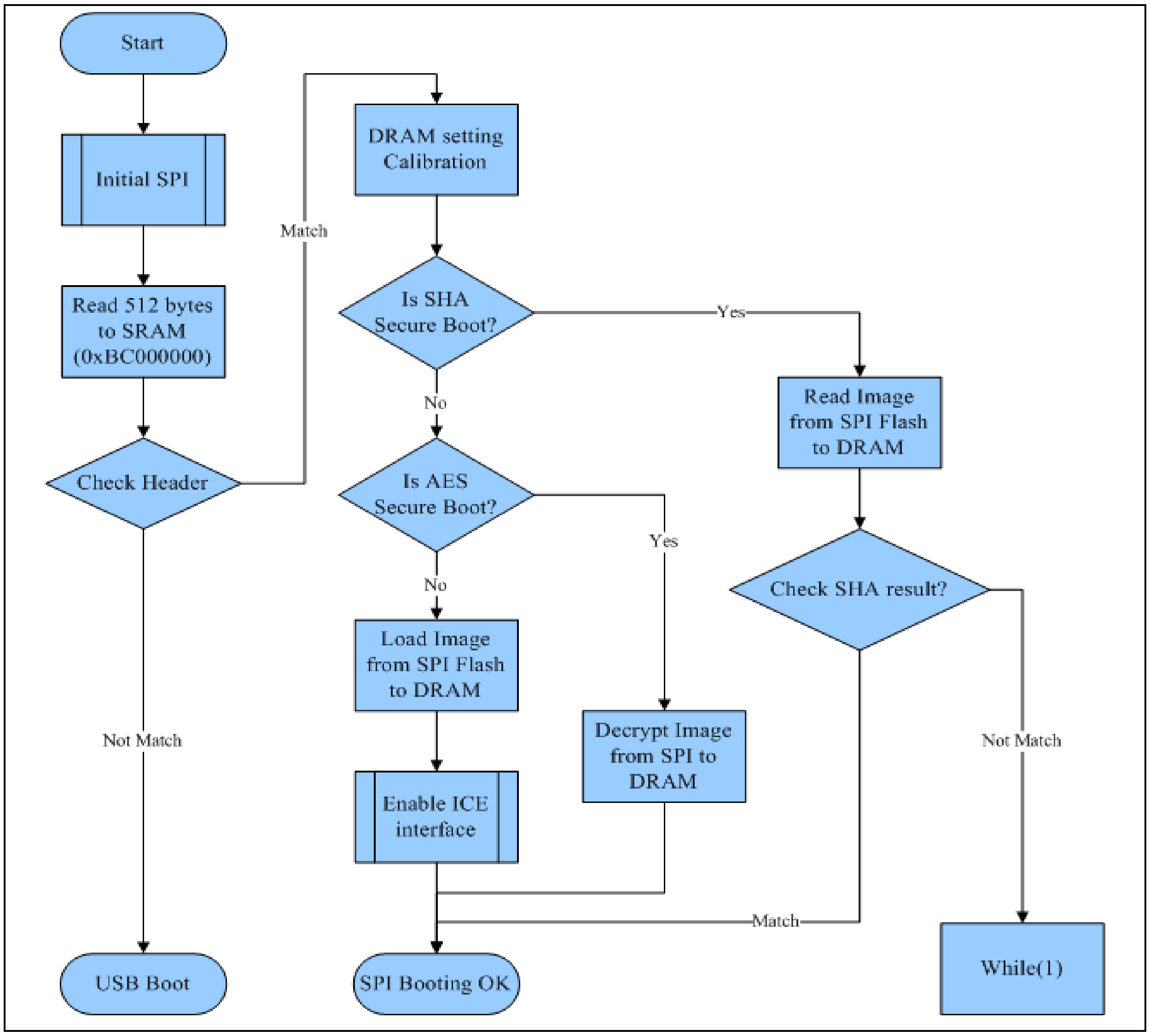


Figure 1-3 SPI Startup Flow

SPI boot flow read address 0x0000000 from SPI flash to check boot code marker is correct, Boot code marker is refer to 4.3.1.1uBoot type. If cannot be found boot code marker in the SPI flash, then it is skipping from “Check Header” states to USB boot. When boot code marker detects correct, then do DDR initialization. Next step is to check SHA enabled, and then calculate SHA. If SHA is not enabled, Net step is to check AES enabled, and then decode AES. It copy just uboot image from SPI to DDR, and execute uboot image.

Note: N9H30 does not support secure boot.

1.3 Boot from eMMC

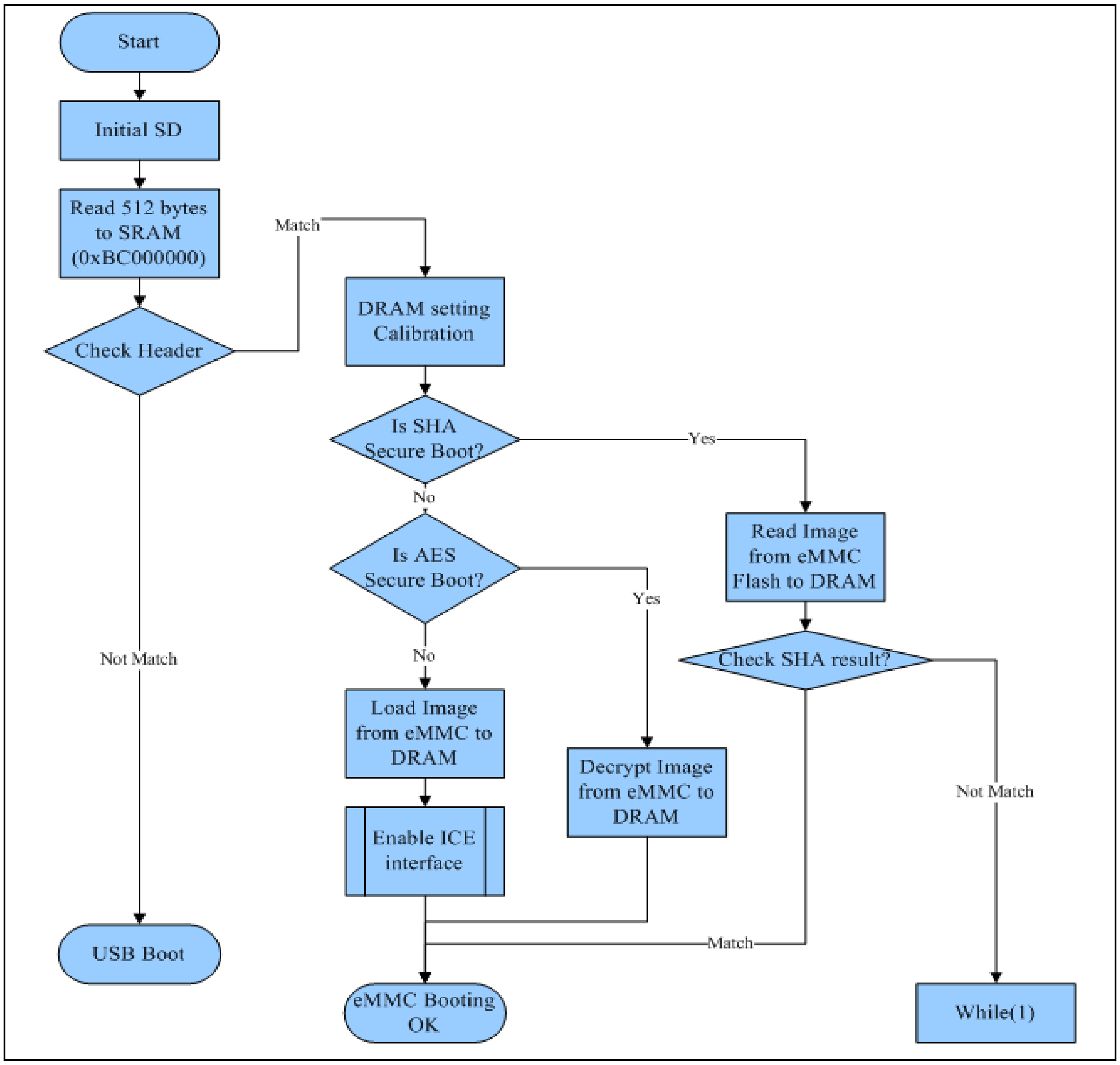


Figure 1-4 eMMC Startup Flow

eMMC boot flow read address 0x0000400 from eMMC to check boot code marker is correct, Boot code marker is refer to 4.4.1.1uBoot type. If boot code marker cannot be found in the eMMC, then it is skipping from “Check Header” states to USB boot. When boot code marker detects correct and then do DDR initialization, Next step is to check SHA enabled, and then calculate SHA. If SHA is not enabled, Net step is to check AES enabled, and then decode AES. It copy just uboot image from eMMC to DDR, and execute uboot image.

Note: N9H30 does not support secure boot.

1.4 USB ISP

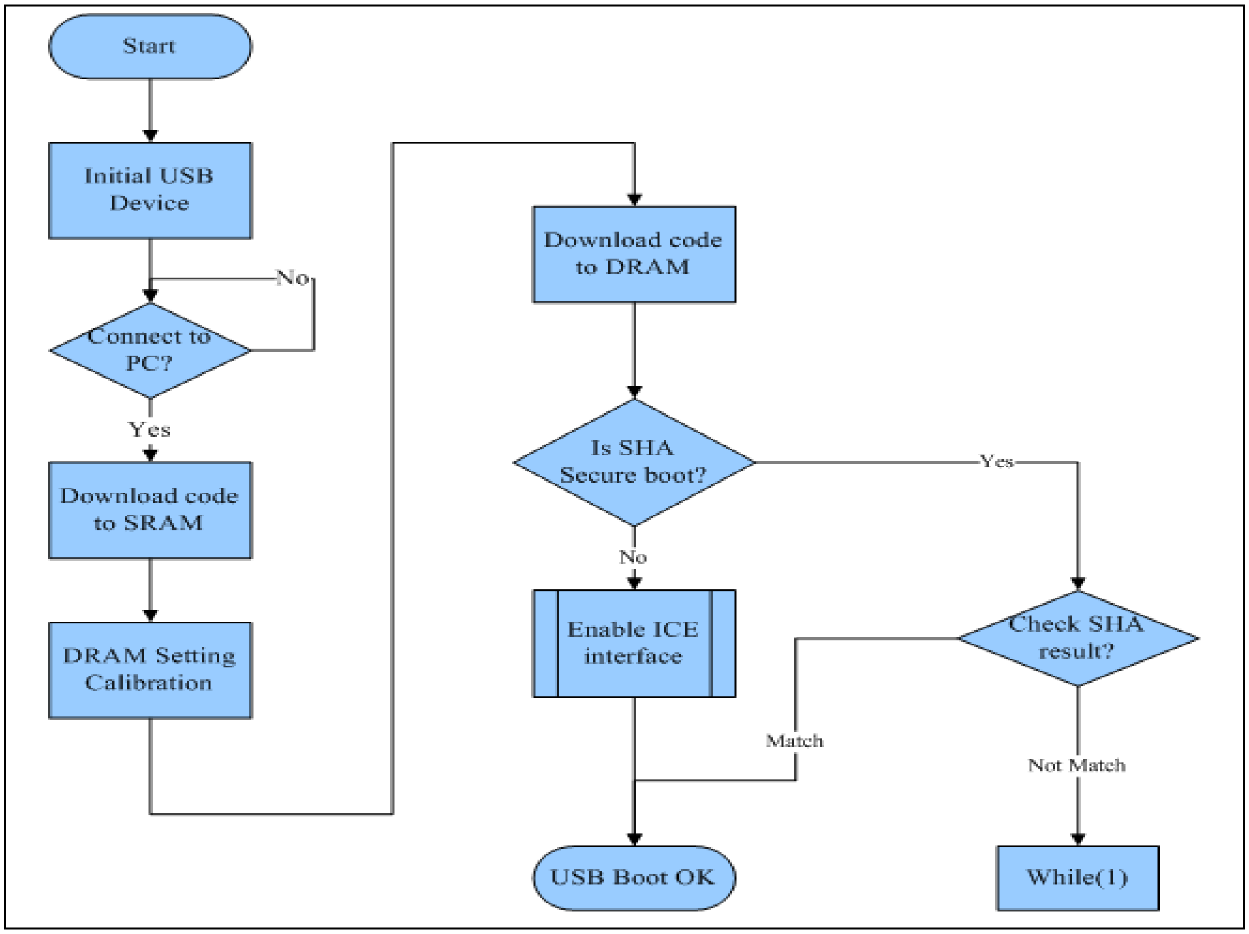


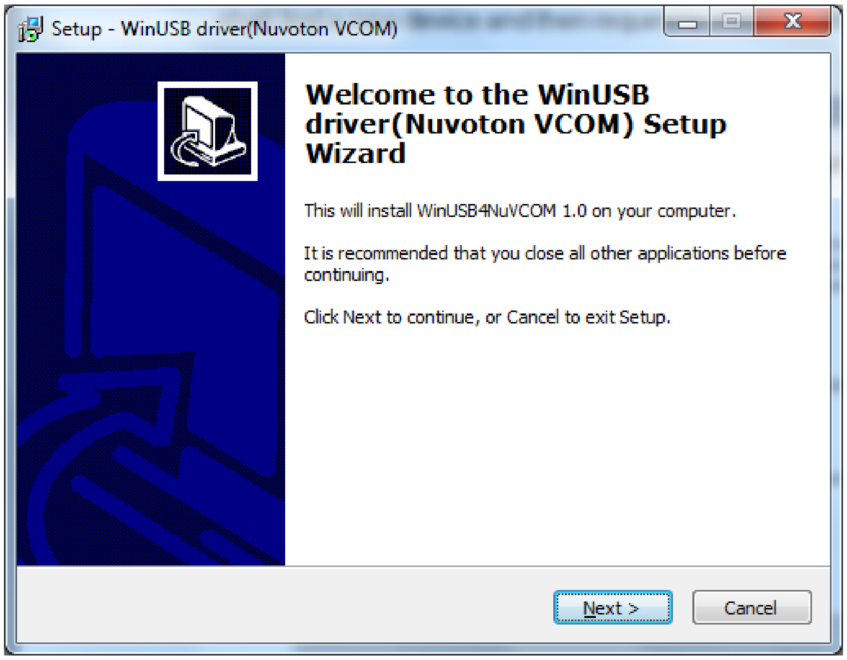
Figure 1-5 USB Startup Flow

USB boot flow first step is waiting DDR parameter to complete initialization. NuWriter connected NUC970/N9H30 series MPU and transmitted DDR parameter. After completion, NuWriter transmit xusb.bin to NUC970/N9H30 series MPU and execution.

2 INSTALLING THE NUWRITER DRIVER

The burning tool requires a NuWriter driver to be installed on PC first. Please follow the steps below to install the driver.

Run the "*WinUSB4NuVCOM.exe*" before the USB cable is plugged in. The "*WinUSB4NuVCOM.exe*" can be found in the "Tool" directory. Power on the NUC970/N9H30 Series MPU EVB and plug the USB cable into PC, the Windows shall find a new device and then request to install its driver.


Setup - WinUSB driver(Nuvoton VCOM)

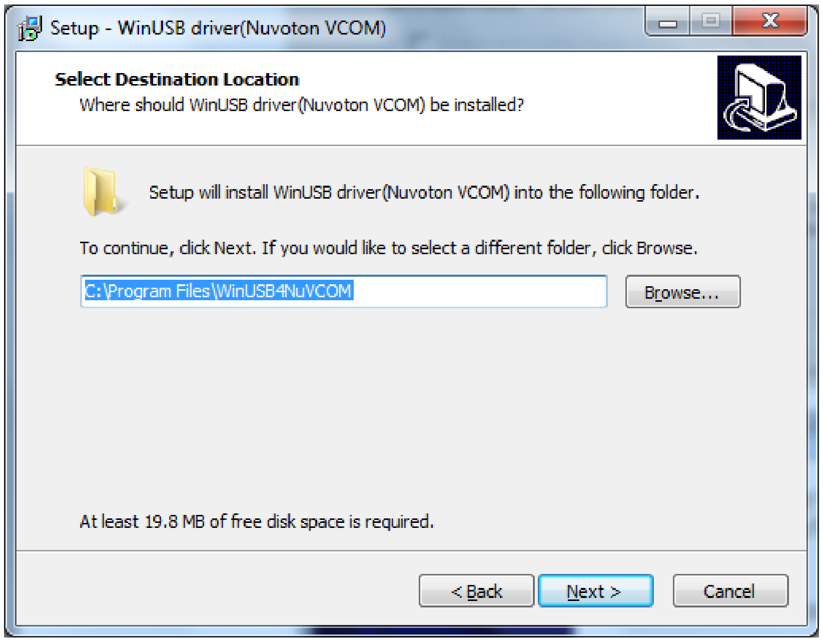
Welcome to the WinUSB driver(Nuvoton VCOM) Setup Wizard

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.

Click "**Next**". The software installation will ask you how to install the driver. As follows.


Setup - WinUSB driver(Nuvoton VCOM)

Select Destination Location

Where should WinUSB driver(Nuvoton VCOM) be installed?

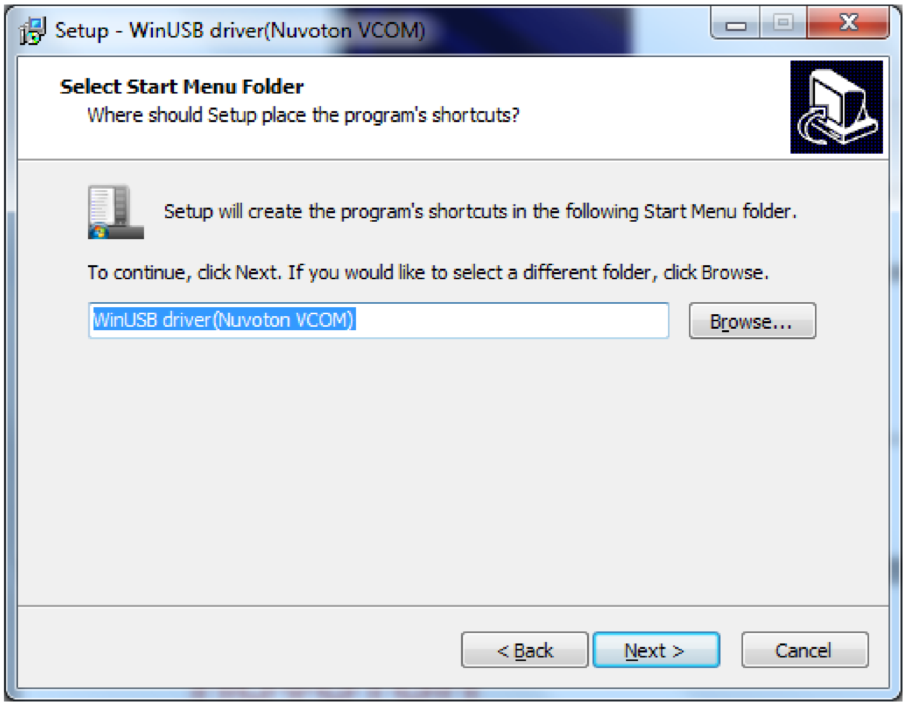
Setup will install WinUSB driver(Nuvoton VCOM) into the following folder.

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.

Select "setup path to specific location (Advanced), and then click "**Next**". The installation software will ask you the option as follows.


Setup - WinUSB driver(Nuvoton VCOM)

Select Start Menu Folder

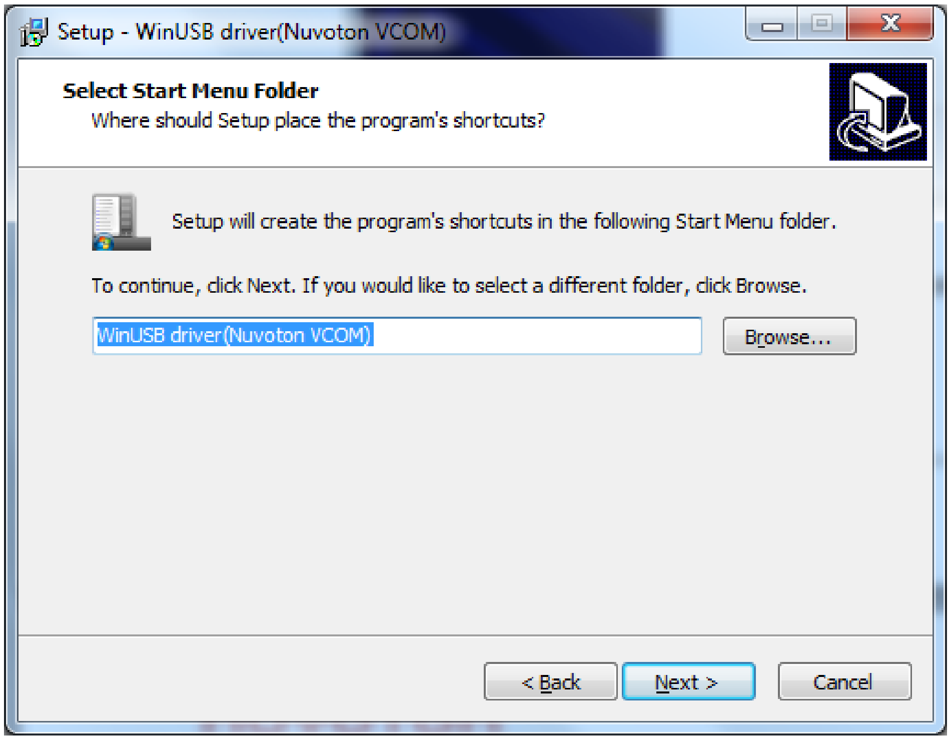
Where should Setup place the program's shortcuts?

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)

Click “**Next**”. As follows.


Setup - WinUSB driver(Nuvoton VCOM)

Select Start Menu Folder

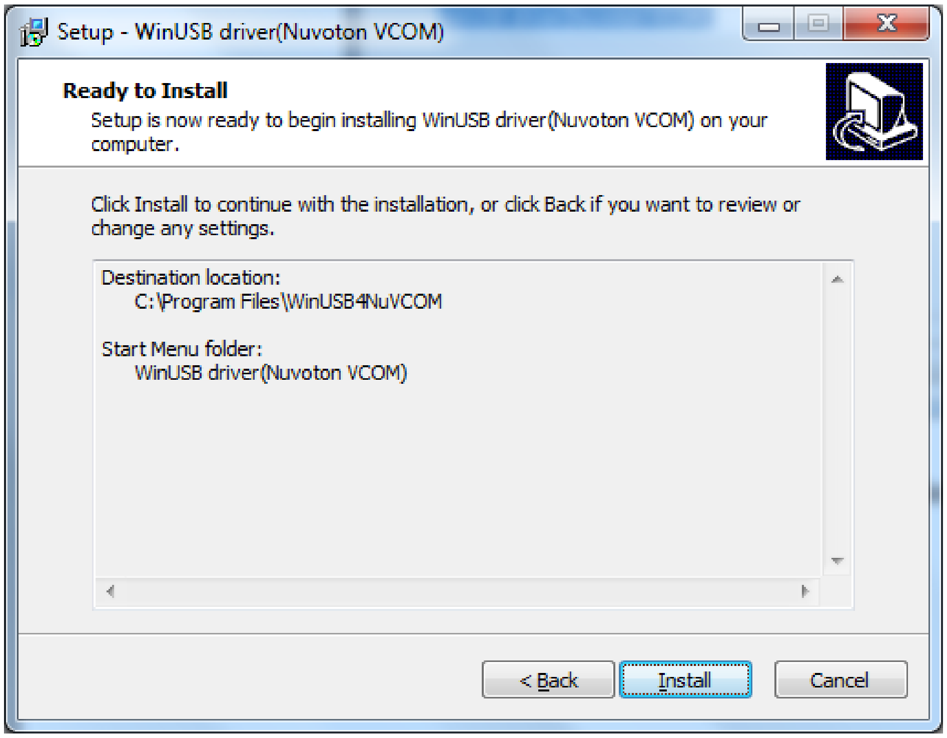
Where should Setup place the program's shortcuts?

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)

Click “**Install**”. As follows:



Ready to Install

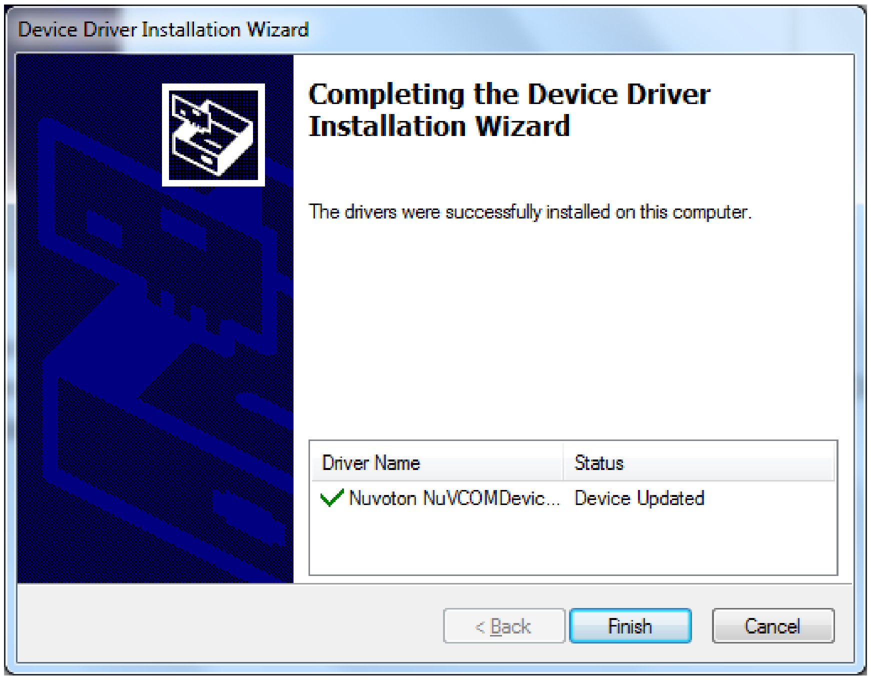
Setup is now ready to begin installing WinUSB driver(Nuvoton VCOM) on your computer.

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)

Click “**Finish**” to finish install driver. As follows

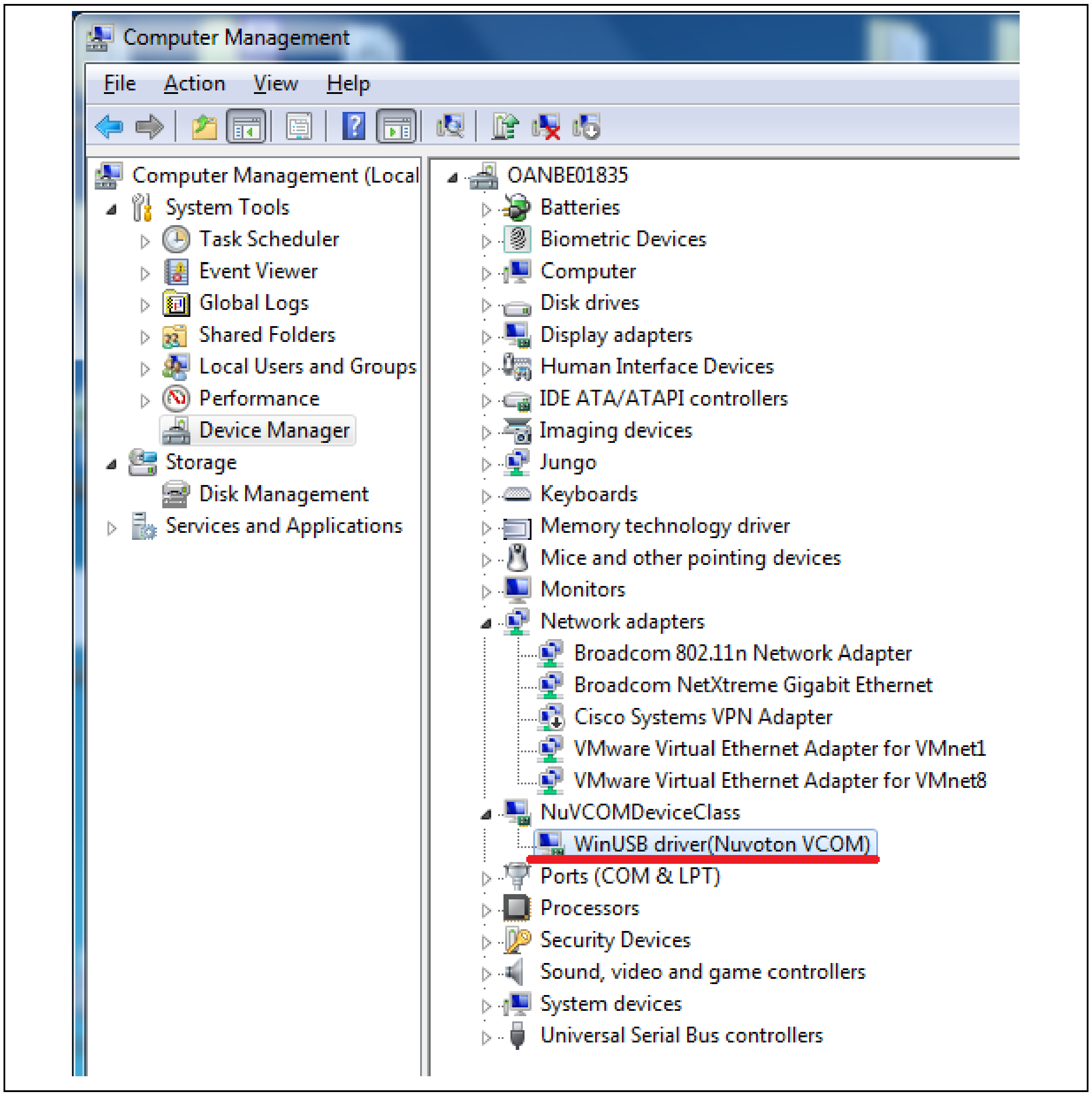


Completing the Device Driver Installation Wizard

The drivers were successfully installed on this computer.

Driver Name	Status
✓ Nuvoton NuVCOMDevic...	Device Updated

If the installation is successful, a virtual COM port named “**WinUSB driver (Nuvoton VCOM)**” can be found by using “Device Manager” to check the ports devices.


Computer Management
File Action View Help
Computer Management (Local
System Tools
Task Scheduler
Event Viewer
Global Logs
Shared Folders
Local Users and Groups
Performance
Device Manager
Storage
Disk Management
Services and Applications
OANBE01835
Batteries
Biometric Devices
Computer
Disk drives
Display adapters
Human Interface Devices
IDE ATA/ATAPI controllers
Imaging devices
Jungo
Keyboards
Memory technology driver
Mice and other pointing devices
Monitors
Network adapters
Broadcom 802.11n Network Adapter
Broadcom NetXtreme Gigabit Ethernet
Cisco Systems VPN Adapter
VMware Virtual Ethernet Adapter for VMnet1
VMware Virtual Ethernet Adapter for VMnet8
NuVCOMDeviceClass
WinUSB driver(Nuvoton VCOM)
Ports (COM & LPT)
Processors
Security Devices
Sound, video and game controllers
System devices
Universal Serial Bus controllers

3 USB ISP MODE

The NUC970/N9H30 Series MPU EVB provides jumpers to select boot-up conditions. To select USB ISP mode, PA1 must be set to low and PA1 must be set to low. Other boot select can refer to the following table:

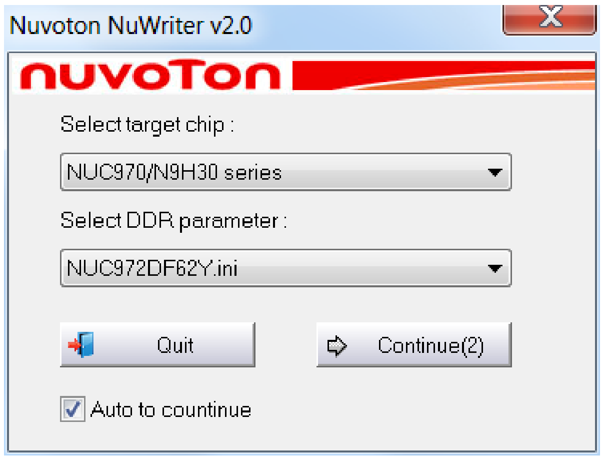
Power-on setting	PA1	PA0
USB ISP	Low	Low
Boot from eMMC	Low	High
Boot from NAND	High	Low
Boot from SPI	High	High

Power-on NUC970/N9H30 Series Micorprocessor EVB, and then open the burning tool, "*NuWriter*", on the PC. Note that the tool cannot work if the "WinUSB4NuVCOM" driver is not found.

4 NUWRITER TOOL

Double click “nuwriter.exe” on PC. NuWriter will start and a window appears. Select target chip to NUC970/N9H30 series and select DDR parameter to DDR initial files.

After select DDR parameter, click “**Continue**” to use NuWriter tool.


Nuvoton NuWriter v2.0

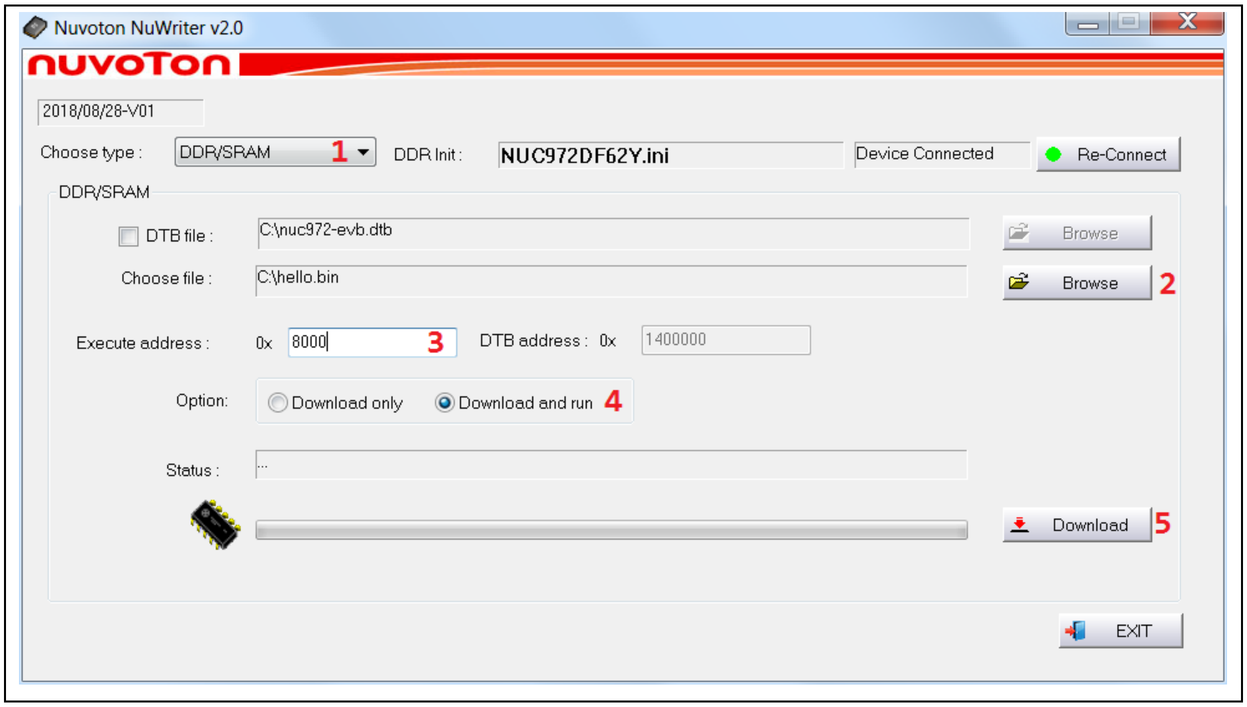
Select target chip : NUC970/N9H30 series

Select DDR parameter : NUC972DF62Y.ini

Figure 4-1 NuWriter – Set Chip/DDR

4.1 DDR/SRAM Mode

4.1.1 Operation Steps


Nuvoton NuWriter v2.0

2018/08/28-V01

Choose type : DDR/SRAM 1 DDR Init : NUC972DF62Y.ini Device Connected

DDR/SRAM

DTB file : C:\nuc972-evb.dtb

Choose file : C:\hello.bin

Execute address : 0x 8000 3 DTB address : 0x 1400000

Option: Download only [✓ Download and run] 4

Status : ...

Figure 4-2 NuWriter – DDR/SDRAM mode(1)

According to the figure above, The DDR/SRAM mode is used to download an image to DDR or SDRAM for debugging purpose. Follow the steps is listed below:

1. Select “**SDRAM**”.
2. Browse the image.
3. Enter the image execution address. Note: Execution address between 0x00000000 ~ 0x01F00000 (31MB).
4. Select **“Download only**” or ”**Download and run**”.
5. Click “**Download**”.

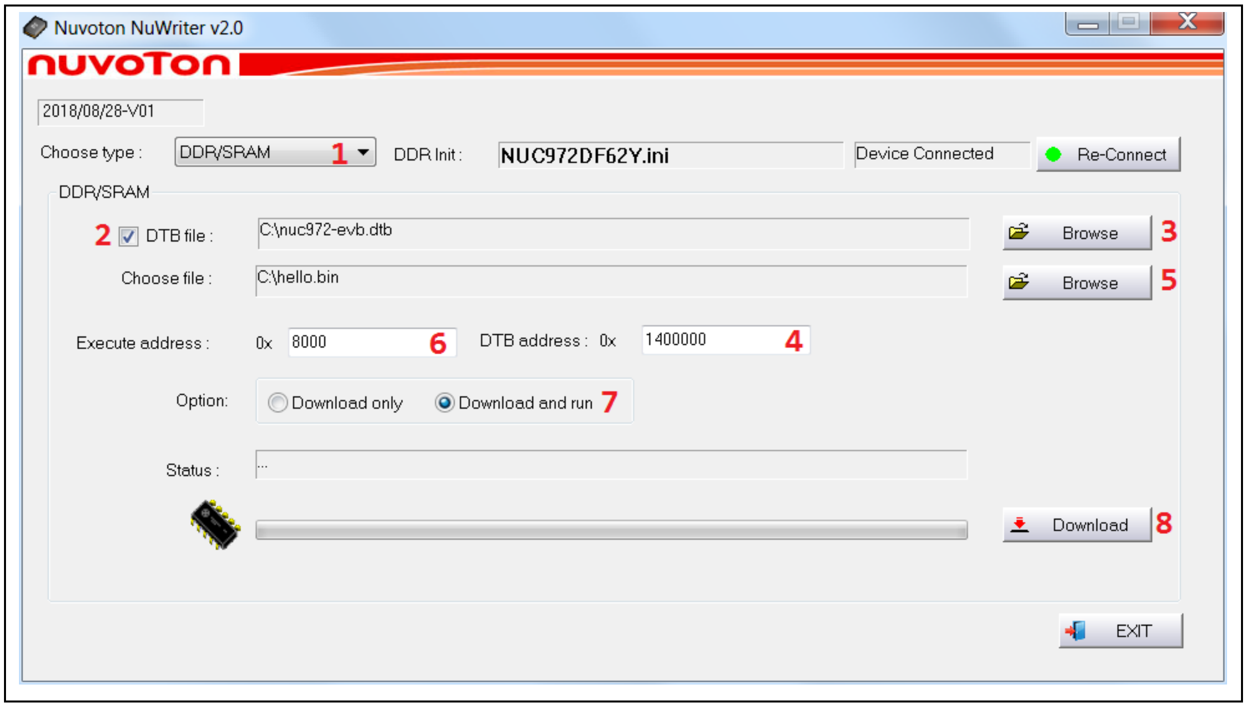

Nuvoton NuWriter v2.0
2018/08/28-V01
Choose type : DDR/SRAM 1
DDR Init : NUC972DF62Y.ini
Device Connected
Re-Connect
DDR/SRAM
2 DTB file : C:\nuc972-evb.dtb
Browse 3
Choose file : C:\hello.bin
Browse 5
Execute address : 0x 8000 6
DTB address : 0x 1400000 4
Option: Download only / Download and run 7
Status :
Download 8
EXIT

Figure 4-3 NuWriter – DDR/SDRAM mode(2)

According to the figure above, The DDR/SRAM mode is used to download an image and a device tree file(*.dtb) to DDR or SDRAM for debugging purpose. Follow the steps is listed below:

1. Select “**SDRAM**”.
2. Enable Device tree check box
3. Select 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.
5. Browse the image.
6. Enter the image execution address. Note: Execution address between 0x00000000 ~ 0x01F00000 (31MB).
7. Select **“Download only**” or ”**Download and run**”.
8. Click “**Download**”.

4.2 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 uboot or Linux. The Image type is set uBoot, Data, Environment or Pack.

4.2.1 **Image Type**

4.2.1.1 uBoot Type

When NUC970/N9H30 is powered on, Image of uboot type is a first execution program, users can use Keil compiler to generate binary file, this file can be used by uboot type. image of uboot type equals 16 bytes header plus DDR parameters and binary file, as follows：

	0x0	**0x0**	**0x0**	**0x0**
0x00	Boot Code Marker	Execute Address	Image Size	Decrypt Address
0x10	DDR - Initial Marker	DDR - Counter	DDR - Address 0	DDR – Value0

0x20	DDR - Address 1	DDR - Value 1	DDR - Address 2	DDR - Value 2
0x30	DDR - Address 3	DDR - Value 3	DDR - Address 4	DDR - Value 5
0x40	DDR - Address 5	DDR - Value 5	DDR - Dummy	DDR - Dummy
0x50	uBoot			

Figure 4-4 Boot code header

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

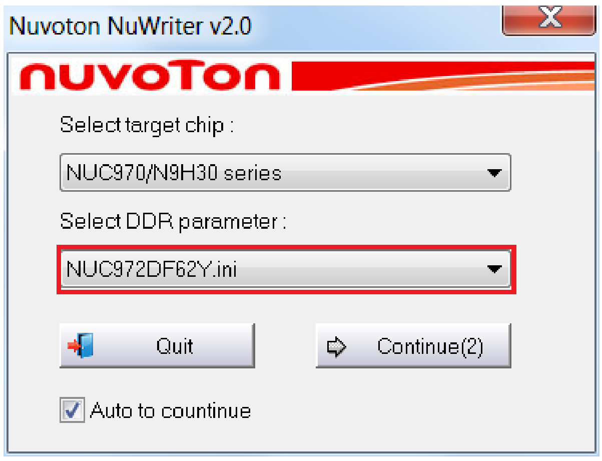
Execute Address = Copy uboot image to { Execute Address } location.

Image Size = {uBoot image length}.

Decrypt Address = {0xFFFFFFFF}.

DDR - Initial Marker = {0x55AA55AA}.

DDR - Counter = DDR parameter length, Calculate the figure below the selected parameter DDR NUC72DF61Y.ini,user can find NUC72DF61Y.ini file in “NuWriter\sys_cfg\” path.



Nuvoton NuWriter v2.0

Select target chip : NUC970/N9H30 series

Select DDR parameter : NUC972DF62Y.ini

Figure 4-5 NuWriter – Select DDR parameter

For example, NUC72DF61Y.ini follows

```
0xB0000220=0x01000000
0xB0000264=0xC0000018
0xB0000220=0x01000018
0x55AA55AA=0x1
0x55AA55AA=0x1
0xB0001828=0x53DCD84A
0xB0001808=0x00008014
0x55AA55AA=0x1
0x55AA55AA=0x1
0x55AA55AA=0x1
```

```
0xB0001800=0x00030476
0x55AA55AA=0x1
0xB0001804=0x00000021
0x55AA55AA=0x1
0xB0001804=0x00000023
0x55AA55AA=0x1
0x55AA55AA=0x1
0x55AA55AA=0x1
0xB0001804=0x00000027
0x55AA55AA=0x1
0x55AA55AA=0x1
0x55AA55AA=0x1
0xB0001820=0x00000000
0xB0001824=0x00000000
0xB000181C=0x00004000
0xB0001818=0x00000332
0xB0001810=0x00000006
0xB0001804=0x00000027
0x55AA55AA=0x1
0x55AA55AA=0x1
0x55AA55AA=0x1
0xB0001804=0x0000002B
0xB0001804=0x0000002B
0xB0001804=0x0000002B
0xB0001818=0x00000232
0xB000181C=0x00004781
0xB000181C=0x00004401
0xB0001804=0x00000020
0xB0001834=0x00888820
0x55AA55AA=0x1
0xB0000218=0x00000008
0xB8003160=0x00008000
0xB80031A0=0x00008000
0xB000022C=0x00000100
0xB000022C=0x00000100
0xB000022C=0x00000100
```

NUC72DF61Y.ini contents can be calculated DDR - Counter = 42.

DDR - Address 0 = {0xB0000220}.DDR - Value0 = {0x01000000}.

DDR - Address 1 = {0xB0000264}. DDR - Value1 = {0xC0000018}.

DDR - Address 2 = {0xB0000220}. DDR -Value2 = {0x01000018}.

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

DDR - Address 4 = {0x55AA55AA}. DDR - Value4 = {0x1}.

And so on · · ·

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

uBoot = uboot binary file.

Then burned image of uboot type into Nand flash in the block0, block1, Block2 and Block3, as follows:

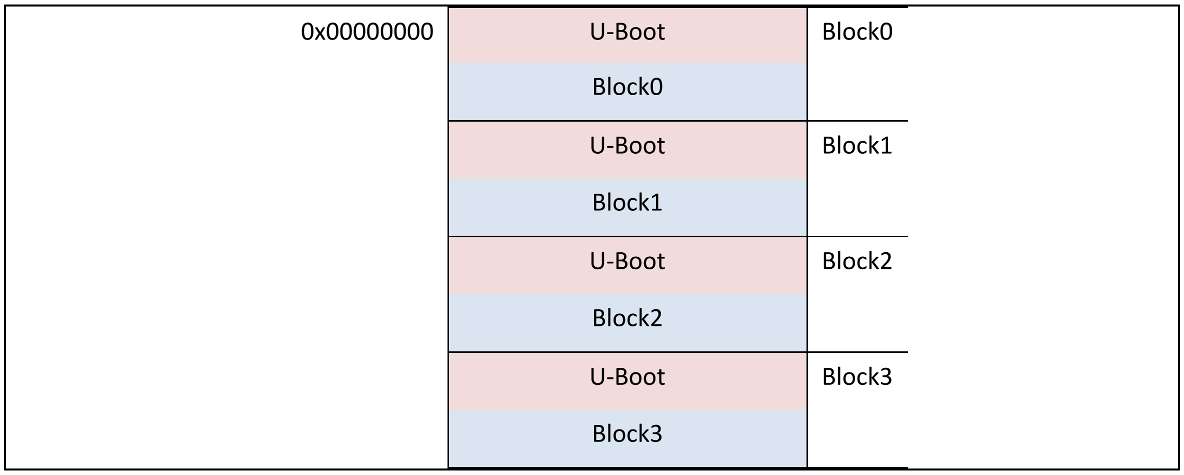


Figure 4-6 Block 0~3 of NAND Flash

Below is read out from the NAND Flash:

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

Execute Address = 0x00000200.

Image Size = 0x0000A628.

Decrypt Address = 0xFFFFFFFF.

DDR - Initial Marker = 0x55AA55AA.

DDR - Counter = 0x0000002A.

DDR - Address 0 = 0xB0000220. DDR - Value0 = 0x01000000.

And so on · · ·

DDR - Address 40 = 0xB8003160.DDR - Value40= 0x00008000.

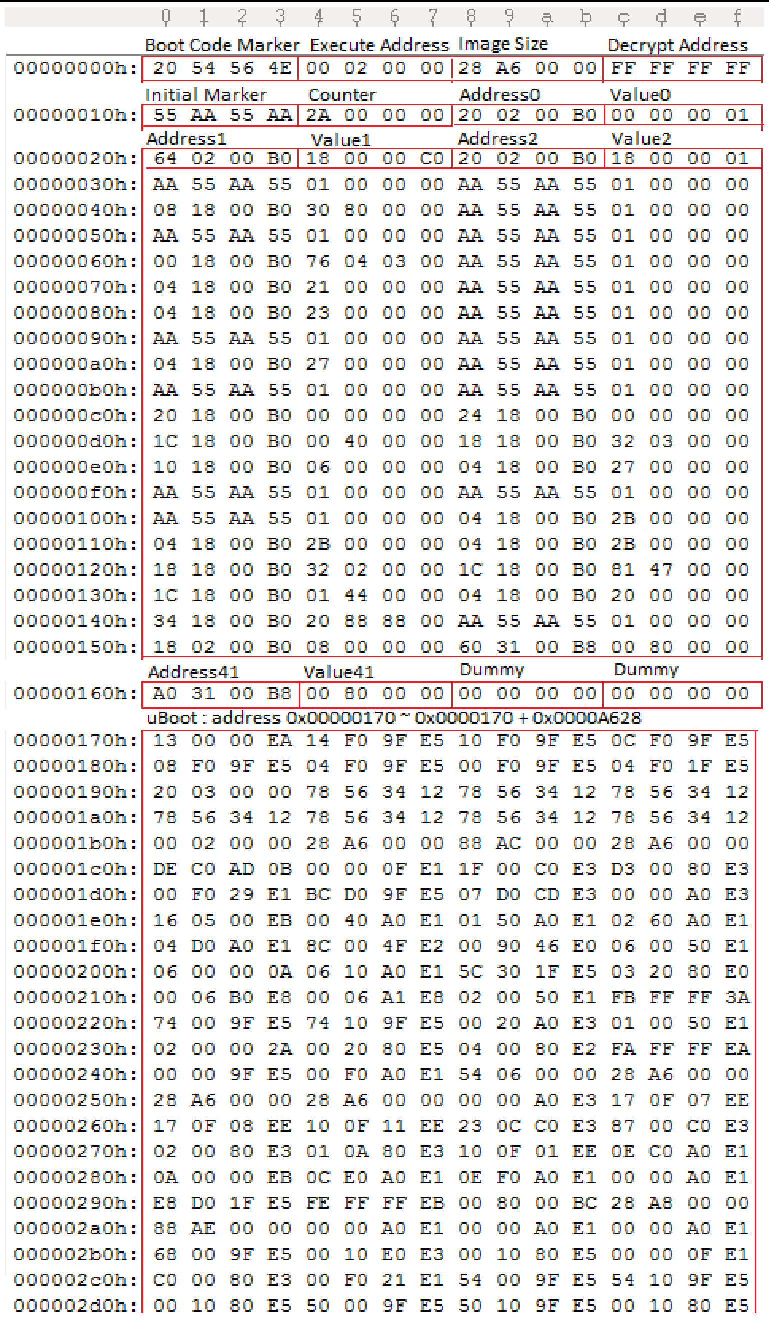
DDR - Address 41 = 0xB80031A0.DDR - Value41 = 0x00008000.

DDR – Dummy = 0x00000000. DDR – Dummy = 0x00000000.

uBoot = address 0x00000170 ~ 0x00000170 + 0x0000A628.

Image of uBoot type have limitations, as follows:

uBoot Image + Header + DDR parameter ≤ Block Size



```
              0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
             Boot Code Marker Execute Address Image Size  Decrypt Address
00000000h:  20 54 56 4E 00 02 00 00 28 A6 00 00 FF FF FF FF
             Initial Marker  Counter      Address0     Value0
00000010h:  55 AA 55 AA 2A 00 00 00 20 02 00 B0 00 00 00 01
             Address1        Value1       Address2     Value2
00000020h:  64 02 00 B0 18 00 00 C0 20 02 00 B0 18 00 00 01
00000030h:  AA 55 AA 55 01 00 00 00 AA 55 AA 55 01 00 00 00
00000040h:  08 18 00 B0 30 80 00 00 AA 55 AA 55 01 00 00 00
00000050h:  AA 55 AA 55 01 00 00 00 AA 55 AA 55 01 00 00 00
00000060h:  00 18 00 B0 76 04 03 00 AA 55 AA 55 01 00 00 00
00000070h:  04 18 00 B0 21 00 00 00 AA 55 AA 55 01 00 00 00
00000080h:  04 18 00 B0 23 00 00 00 AA 55 AA 55 01 00 00 00
00000090h:  AA 55 AA 55 01 00 00 00 AA 55 AA 55 01 00 00 00
000000a0h:  04 18 00 B0 27 00 00 00 AA 55 AA 55 01 00 00 00
000000b0h:  AA 55 AA 55 01 00 00 00 AA 55 AA 55 01 00 00 00
000000c0h:  20 18 00 B0 00 00 00 00 24 18 00 B0 00 00 00 00
000000d0h:  1C 18 00 B0 00 40 00 00 18 18 00 B0 32 03 00 00
000000e0h:  10 18 00 B0 06 00 00 00 04 18 00 B0 27 00 00 00
000000f0h:  AA 55 AA 55 01 00 00 00 AA 55 AA 55 01 00 00 00
00000100h:  AA 55 AA 55 01 00 00 00 04 18 00 B0 2B 00 00 00
00000110h:  04 18 00 B0 2B 00 00 00 04 18 00 B0 2B 00 00 00
00000120h:  18 18 00 B0 32 02 00 00 1C 18 00 B0 81 47 00 00
00000130h:  1C 18 00 B0 01 44 00 00 04 18 00 B0 20 00 00 00
00000140h:  34 18 00 B0 20 88 88 00 AA 55 AA 55 01 00 00 00
00000150h:  18 02 00 B0 08 00 00 00 60 31 00 B8 00 80 00 00
             Address41       Value41      Dummy        Dummy
00000160h:  A0 31 00 B8 00 80 00 00 00 00 00 00 00 00 00 00
             uBoot : address 0x00000170 ~ 0x0000170 + 0x0000A628
00000170h:  13 00 00 EA 14 F0 9F E5 10 F0 9F E5 0C F0 9F E5
00000180h:  08 F0 9F E5 04 F0 9F E5 00 F0 9F E5 04 F0 1F E5
00000190h:  20 03 00 00 78 56 34 12 78 56 34 12 78 56 34 12
000001a0h:  78 56 34 12 78 56 34 12 78 56 34 12 78 56 34 12
000001b0h:  00 02 00 00 28 A6 00 00 88 AC 00 00 28 A6 00 00
000001c0h:  DE C0 AD 0B 00 00 0F E1 1F 00 C0 E3 D3 00 80 E3
000001d0h:  00 F0 29 E1 BC D0 9F E5 07 D0 CD E3 00 00 A0 E3
000001e0h:  16 05 00 EB 00 40 A0 E1 01 50 A0 E1 02 60 A0 E1
000001f0h:  04 D0 A0 E1 8C 00 4F E2 00 90 46 E0 06 00 50 E1
00000200h:  06 00 00 0A 06 10 A0 E1 5C 30 1F E5 03 20 80 E0
00000210h:  00 06 B0 E8 00 06 A1 E8 02 00 50 E1 FB FF FF 3A
00000220h:  74 00 9F E5 74 10 9F E5 00 20 A0 E3 01 00 50 E1
00000230h:  02 00 00 2A 00 20 80 E5 04 00 80 E2 FA FF FF EA
00000240h:  00 00 9F E5 00 F0 A0 E1 54 06 00 00 28 A6 00 00
00000250h:  28 A6 00 00 28 A6 00 00 00 00 A0 E3 17 0F 07 EE
00000260h:  17 0F 08 EE 10 0F 11 EE 23 0C C0 E3 87 00 C0 E3
00000270h:  02 00 80 E3 01 0A 80 E3 10 0F 01 EE 0E C0 A0 E1
00000280h:  0A 00 00 EB 0C E0 A0 E1 0E F0 A0 E1 00 00 A0 E1
00000290h:  E8 D0 1F E5 FE FF FF EB 00 80 00 BC 28 A8 00 00
000002a0h:  88 AE 00 00 00 00 A0 E1 00 00 A0 E1 00 00 A0 E1
000002b0h:  68 00 9F E5 00 10 E0 E3 00 10 80 E5 00 00 0F E1
000002c0h:  C0 00 80 E3 00 F0 21 E1 54 00 9F E5 54 10 9F E5
000002d0h:  00 10 80 E5 50 00 9F E5 50 10 9F E5 00 10 80 E5
```

Figure 4-7 Address 0x00000000~0x000002e0 of NAND Flash

4.2.1.2 Data Type

Mainly the image of data type into NAND flash in the specified address. Depending on the value of image start offset (aligned on 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.

Make a new yaffs2 image with inband tags, as follow (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 NuWriter tool to burn rootfs_yafffs2.img into NAND flash.

Enter the following command will mount yaffs2 file system on the Linux.

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

yaffs2 command can be found in yaffs2utils.tar.gz

Make a new ubifs image, as follows:

```
# 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 description parameter, as follows:

-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 description parameter, as follows:

-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

rootfs_ubinize.cfg content as follows：

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

rootfs folder can be compressed into ubi.img, user can use NuWriter tool to brun ubi.img into NAND flash.

Enter the following command will mount UBIFS file system on the Linux.

Refer to “/sys/class/misc/ubi_ctrl/dev” content, assuming that the content is "10:56",user can set as follows.

```
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

Linux kernel must also be configure, as follows:

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

4.2.1.3 Environment Type

uboot type is set uboot environment variables, the image of environment type into NAND flash in the specified address. Let uboot read environment variables 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”, as follows:

```
baudrate=115200
bootdelay=3
ethact=emac
ethaddr=00:00:00:11:66:88
stderr=serial
stdin=serial
stdout=serial
```

4.2.1.4 Pack Type

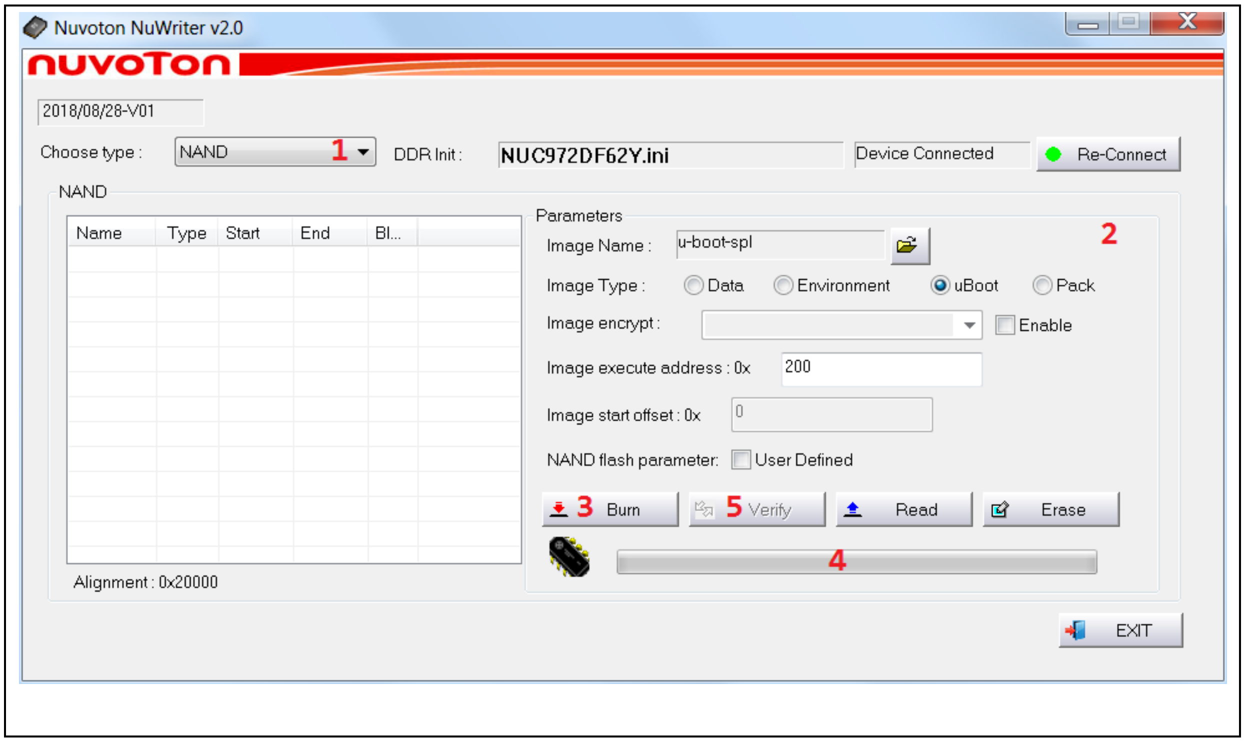
Pack type can burn pack image to NAND flash. The image of pack type into NAND flash in the specified address. Depending on the value of image start offset (aligned on block size boundary, block size is based on SPI specifications). User can refer to pack mode section. Let you know How to make a pack image.

4.2.1.5 Bad Block Process

NAND Flash supports four types to burn into the NAND Flash. We need to detect and skip bad blocks, it is assumed that NAND is written a data to page 10, page 11, page 12, page 13, but page 13 is bad page. Then write page 10, page 11, page 12 and page 14 instead.

4.2.2 **Operation Steps**

4.2.2.1 Add a New Image



NUC972DF62Y.ini

Image Name : u-boot-spl

Image Type : ( ) Data ( ) Environment (● uBoot) ( ) Pack

Image encrypt : ☐ Enable

Image execute address : 0x 200

Image start offset : 0x 0

NAND flash parameter: ☐ User Defined

Figure 4-8 NAND – New Image

According to the figure above, follow the steps below to add a new image to NAND flash:

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 encrypt :Select encrypt file and Set enable or disable.It can not used by file system type and environment type. (N9H30 does not support this option).
   - Image execute address: Enter image execute address. Only is uboot type is vaild.
   - Image start offset: Enter image start offset.
   - NAND flash parameter: The checkbox User Defined to decide whether to customize NAND parameters
3. Click “Burn”.
4. Waiting for finishing progress bar.
5. After “Burn” the image, click the “**Verify**” button to read back the image data to make sure the burning status.

4.2.2.2 Configure NAND flash parameters

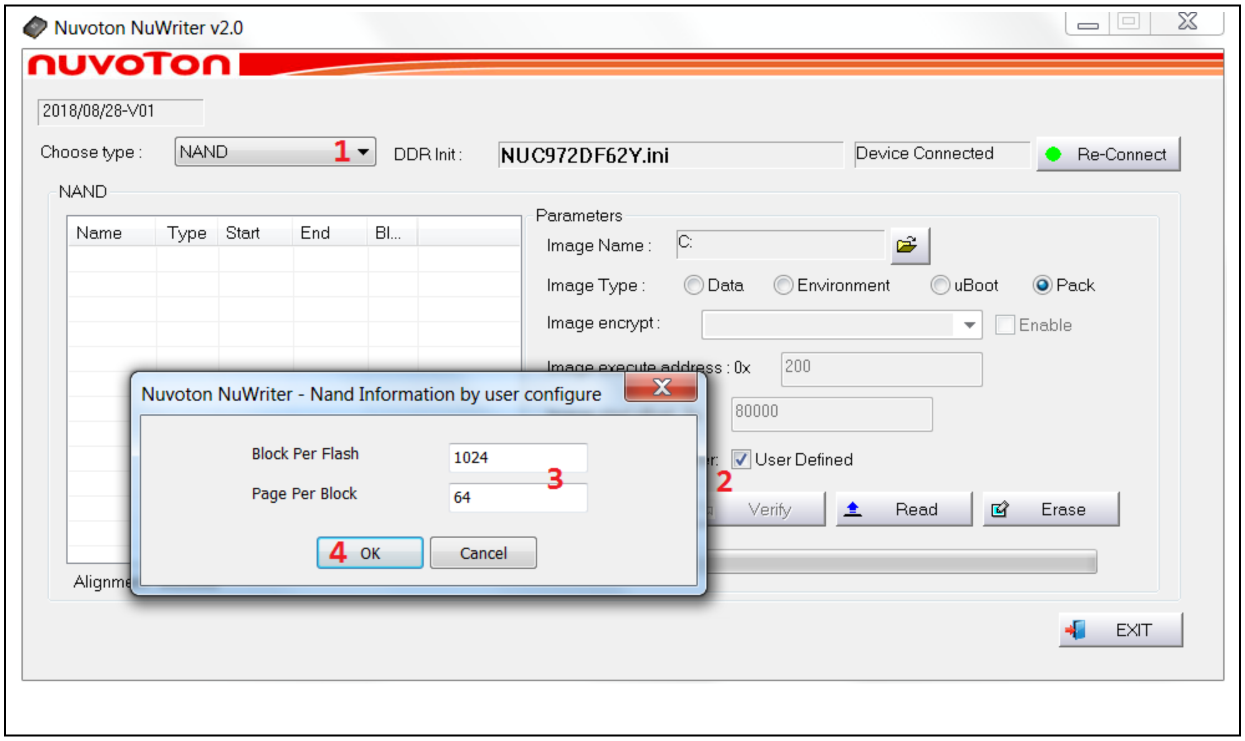


Figure 4-9 NAND – Configure Parameters

NuWriter will automatically detect the NAND ID to determine its parameters, and the user can set the NAND parameters is based on NAND specifications. According to the figure above. Follow the steps below to set NAND flash parameters:

1. Select the “**NAND**”.
2. Check “User Defined” to pop up the window for setting NAND parameter.
3. Fill the following parameter to set Device size and block unit.
   - Block Per Flash: Block size per NAND Flash.
   - Page Per Block: Page size per Block.
4. Click “OK”.

4.2.2.3 Read Raw Data

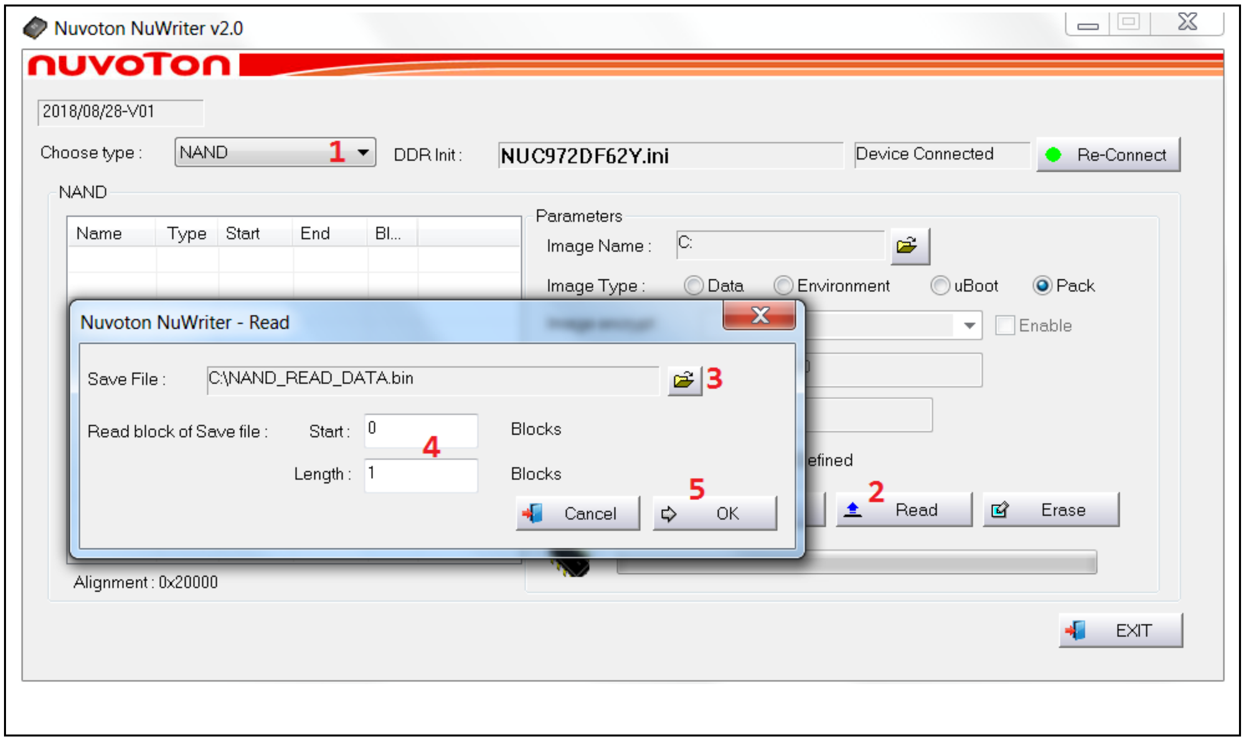


Figure 4-10 NAND – Read

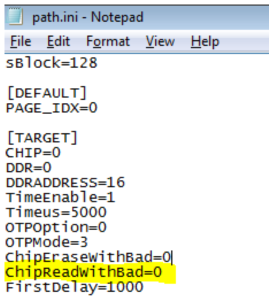
According to the figure above. Follow the steps below to read data from NAND flash:

5. Select the “**NAND**”.
6. Click “Read”.
7. Browse save file.
8. Enter the read back of blocks , Aligned on block size boundary, Block size is based on NAND specifications.
   - Start: Start addres of blocks
   - Length: Length of blocks
9. Click “OK”.

4.2.2.3.1 Read Data Mode

According to ChipReadWithBad in NuWriter/path.ini, It can change read data function as 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.



```
sBlock=128

[DEFAULT]
PAGE_IDX=0

[TARGET]
CHIP=0
DDR=0
DDRADDRESS=16
TimeEnable=1
Timeus=5000
OTPOption=0
OTPMode=3
ChipEraseWithBad=0
ChipReadWithBad=0
FirstDelay=1000
```

Figure 4-11 NAND – path.ini(1)

4.2.2.4 Erase NAND Flash

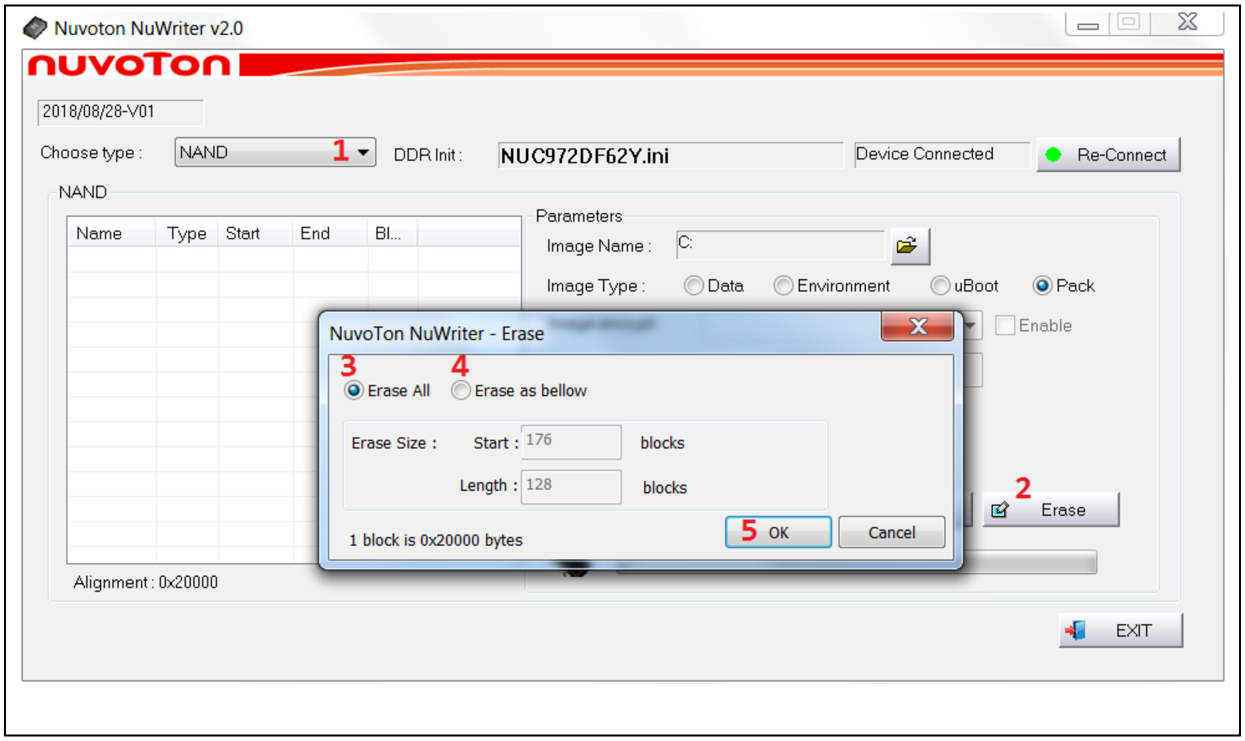


Figure 4-12 NAND – Erase

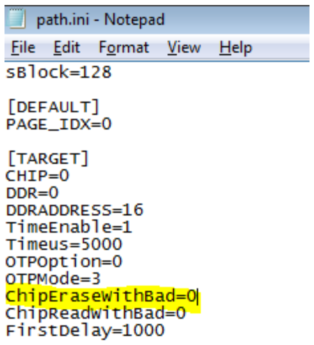
According to the figure above, Follow the steps below to erase NAND flash:

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

4.2.2.4.1 Erase Flash Mode

According to ChipEraseWithBad in Nuwriter/path.ini, It can change erase flash function as bellow.

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



```
sBlock=128

[DEFAULT]
PAGE_IDX=0

[TARGET]
CHIP=0
DDR=0
DDRADDRESS=16
TimeEnable=1
Timeus=5000
OTPOption=0
OTPMode=3
ChipEraseWithBad=0
ChipReadWithBad=0
FirstDelay=1000
```

Figure 4-13 NAND – path.ini(2)

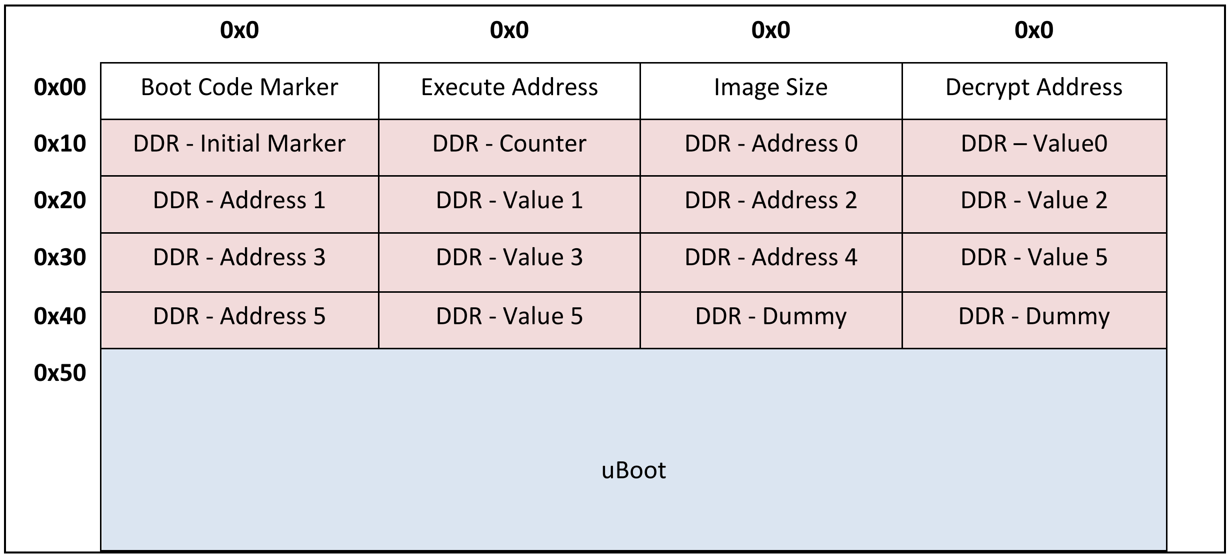
4.3 SPI Mode

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 uBoot, Data, Environment or Pack.

4.3.1 Image Type

4.3.1.1 uBoot Type

When NUC970/N9H30 series microprocessor is powered on, Image of uboot type is a first execution program, users can use Keil compiler to generate binary file, this file can be used by uboot type. image of uboot type equals 16 bytes header plus DDR parameters and binary file, as follows：



	0x0	**0x0**	**0x0**	**0x0**
0x00	Boot Code Marker	Execute Address	Image Size	Decrypt Address
0x10	DDR - Initial Marker	DDR - Counter	DDR - Address 0	DDR – Value0
0x20	DDR - Address 1	DDR - Value 1	DDR - Address 2	DDR - Value 2
0x30	DDR - Address 3	DDR - Value 3	DDR - Address 4	DDR - Value 5
0x40	DDR - Address 5	DDR - Value 5	DDR - Dummy	DDR - Dummy
0x50	uBoot			

Figure 4-14 Boot code header

Detailed contents can refer NAND mode uBoot type section. Then burned image of uboot type into Nand flash in the address 0x00000000, as follows:

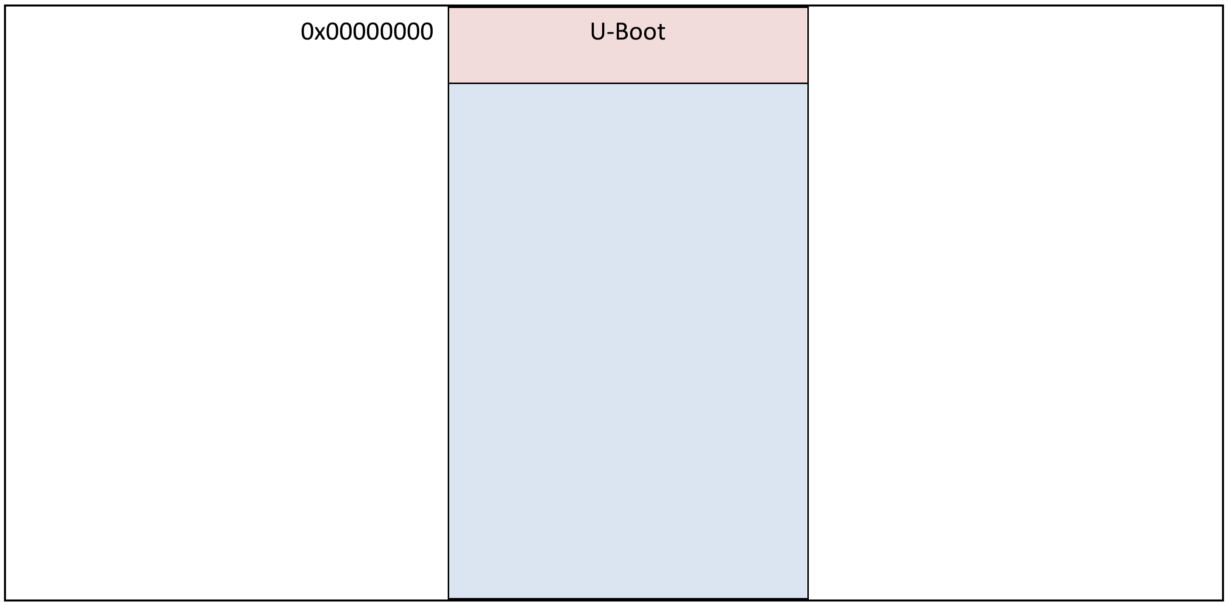


Figure 4-15 SPI Flash

4.3.1.2 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.

4.3.1.3 Environment Type

uboot type is set uboot environment variables, the image of environment type into SPI flash in the specified address. Let uboot read environment variables 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.

4.3.1.4 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 pack mode section. Let you know How to make a pack image.

4.3.2 **Operation Steps**

4.3.2.1 Add a New Image

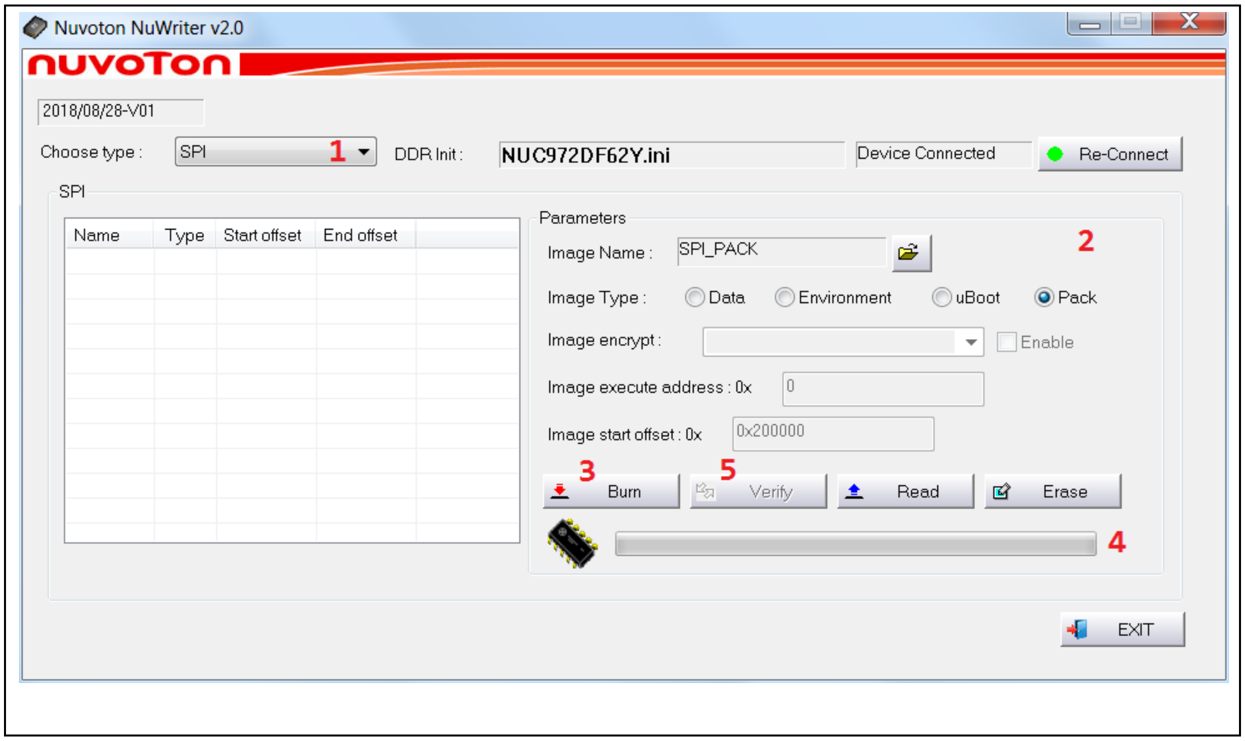

Nuvoton NuWriter v2.0
2018/08/28-V01
Choose type : SPI
DDR Init : NUC972DF62Y.ini
Device Connected
Image Name : SPI_PACK
Image Type : ( ) Data ( ) Environment ( ) uBoot [✓ Pack]
Image encrypt : [ ] Enable
Image execute address : 0x 0
Image start offset : 0x 0x200000

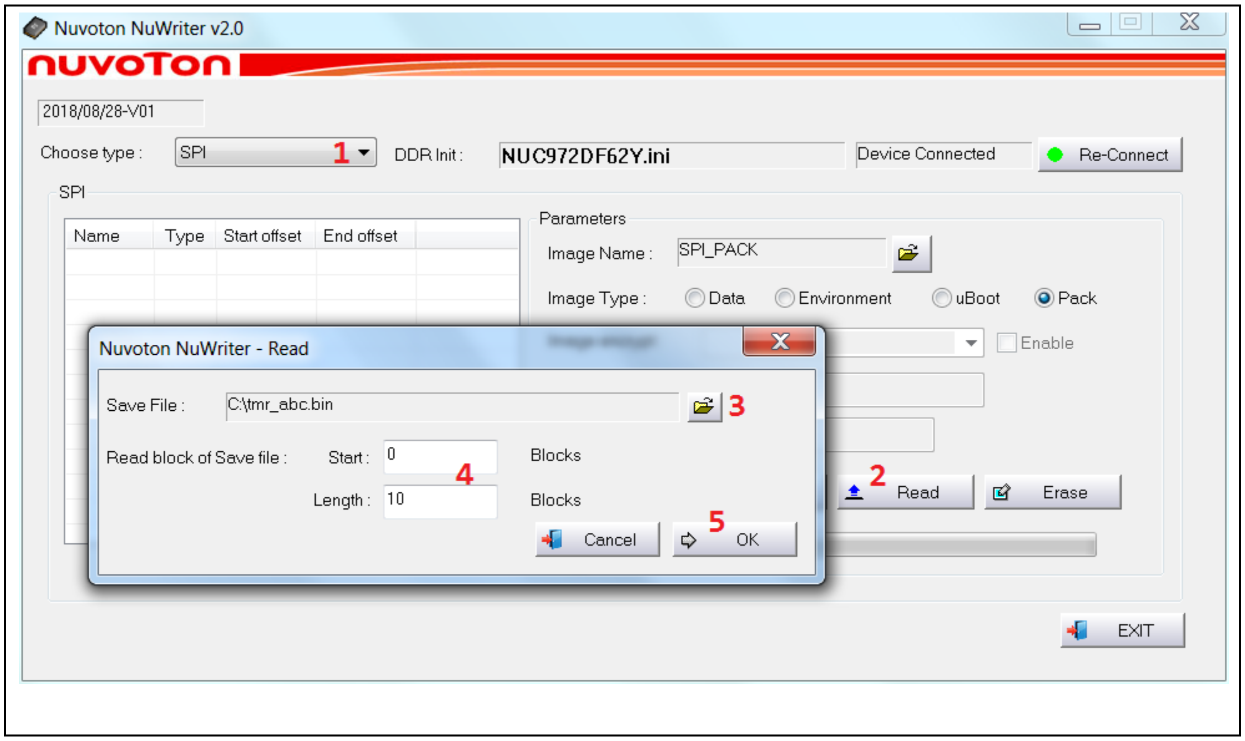
Figure 4-16 SPI – New Image

According to the figure above, follow the steps below 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 encrypt :Select encrypt file and Set enable or disable. (N9H30 does not support this option)
   - Image execute address: Enter image execute address. Only is uboot type is vaild.
   - Image start offset: Enter image start offset.
3. Click “Burn”.
4. Waiting for finishing progress bar.
5. After “Burn” the image, click the “Verify” button to read back the image data to make sure the

burning status.

4.3.2.2 Read Raw Data



Choose type : SPI
DDR Init : NUC972DF62Y.ini
Device Connected
Image Name : SPI_PACK
Image Type : ( ) Data ( ) Environment ( ) uBoot [✓ Pack]
Save File : C:\tmr_abc.bin
Read block of Save file : Start : 0 Blocks
Length : 10 Blocks

Figure 4-17 SPI – Read Image

According to the figure above. Follow the steps below to read data from SPI flash:

1. Select the “**SPI**”.
2. Click “Read”.
3. Browse save 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”.

4.3.2.3 Erase SPI Flash

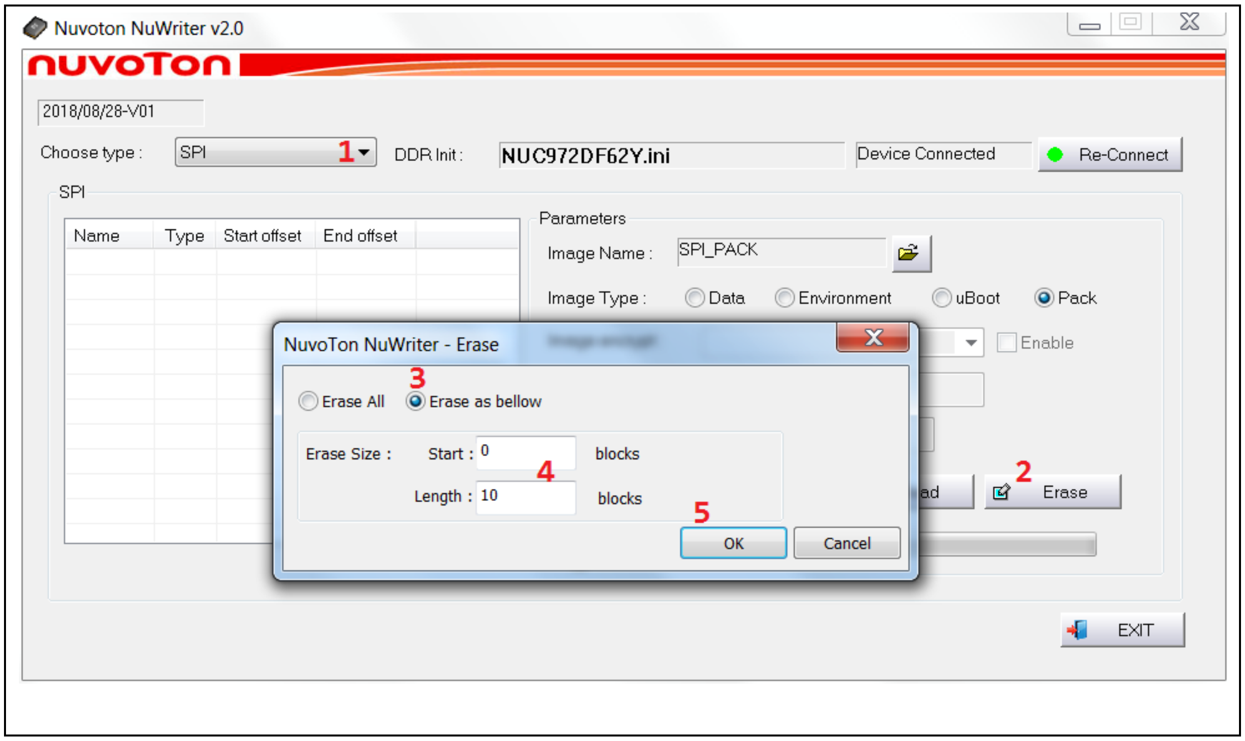


Figure 4-18 NuWriter – Erase

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

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

4.4 eMMC Mode

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

4.4.1 Image Type

4.4.1.1 uBoot Type

When NUC970/N9H30 is powered on, Image of uboot type is a first execution program, users can use Keil compiler to generate binary file, this file can be used by uboot type. image of uboot type equals 16 bytes header plus DDR parameters and binary file, as follows：

	0x0	**0x0**	**0x0**	**0x0**
0x00	Boot Code Marker	Execute Address	Image Size	Decrypt Address
0x10	DDR - Initial Marker	DDR - Counter	DDR - Address 0	DDR – Value0
0x20	DDR - Address 1	DDR - Value 1	DDR - Address 2	DDR - Value 2
0x30	DDR - Address 3	DDR - Value 3	DDR - Address 4	DDR - Value 5

0x40	DDR - Address 5	DDR - Value 5	DDR - Dummy	DDR - Dummy
0x50	uBoot			

Figure 4-19 Boot code header

Detailed contents can refer NAND mode uBoot type section. Then burned image of uboot type into Nand flash in the address 0x400, as follows:

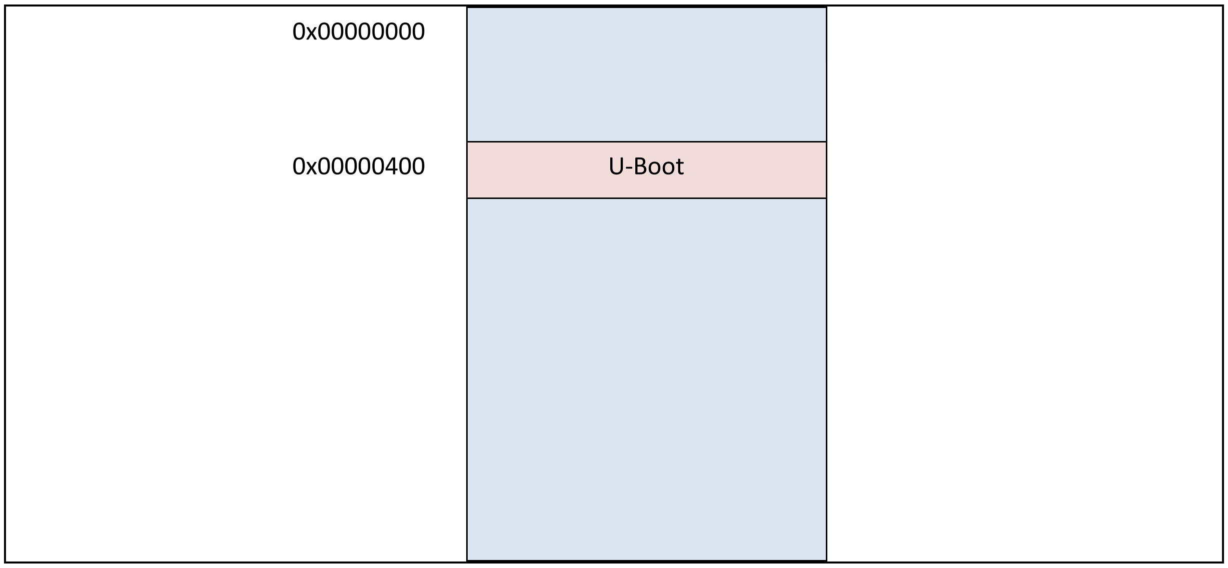


Figure 4-20 eMMC

4.4.1.2 *Data Type*

Mainly the image of data type into eMMC 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 flash in the 0x10000 address, it can help user to configure eMMC flash.

4.4.1.3 *Environment Type*

uboot type is set uboot environment variables, the image of environment type into eMMC in the specified address. Let uboot read environment variables to set the environment. If image start offset equal then 0x10000, image of data into eMMC flash in the 0x10000 address, it can help user to configure eMMC flash.

4.4.1.4 *Pack Type*

Pack type can burn pack image to eMMC. The image of pack type into eMMC in the specified address. Depending on the value of image start offset (aligned on 512bytes boundary). User can refer to pack mode section. Let you know How to make a pack image.

4.4.1.5 *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 can according to myself demand to decide reserved space.

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

4.4.2 Operation Steps

4.4.2.1 Add a New Image

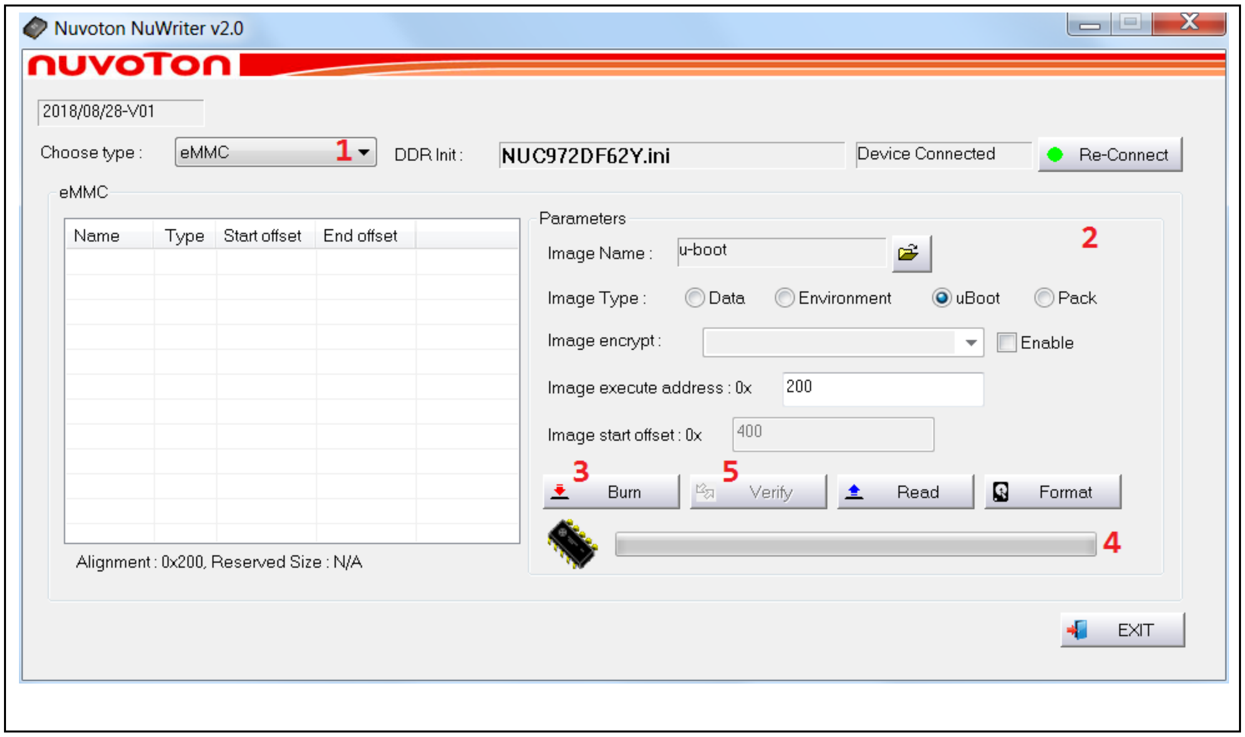


Figure 4-21 eMMC – New Image

According to the figure above, follow the steps below to add a new image to SPI flash:

1. Select the "**eMMC**" type, which will not list the pre-burned images in the eMMC.
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 encrypt: Select encrypt file and Set enable or disable. (N9H30 does not support this option).
   - Image execute address: Enter image execute address. Only is uboot type is valid.
   - Image start offset: Enter image start offset.
3. Click "Burn".
4. Waiting for finishing progress bar.
5. After "Burn" the image, click the "Verify" button to read back the image data to make sure the burning status.

4.4.2.2 Read Raw Data

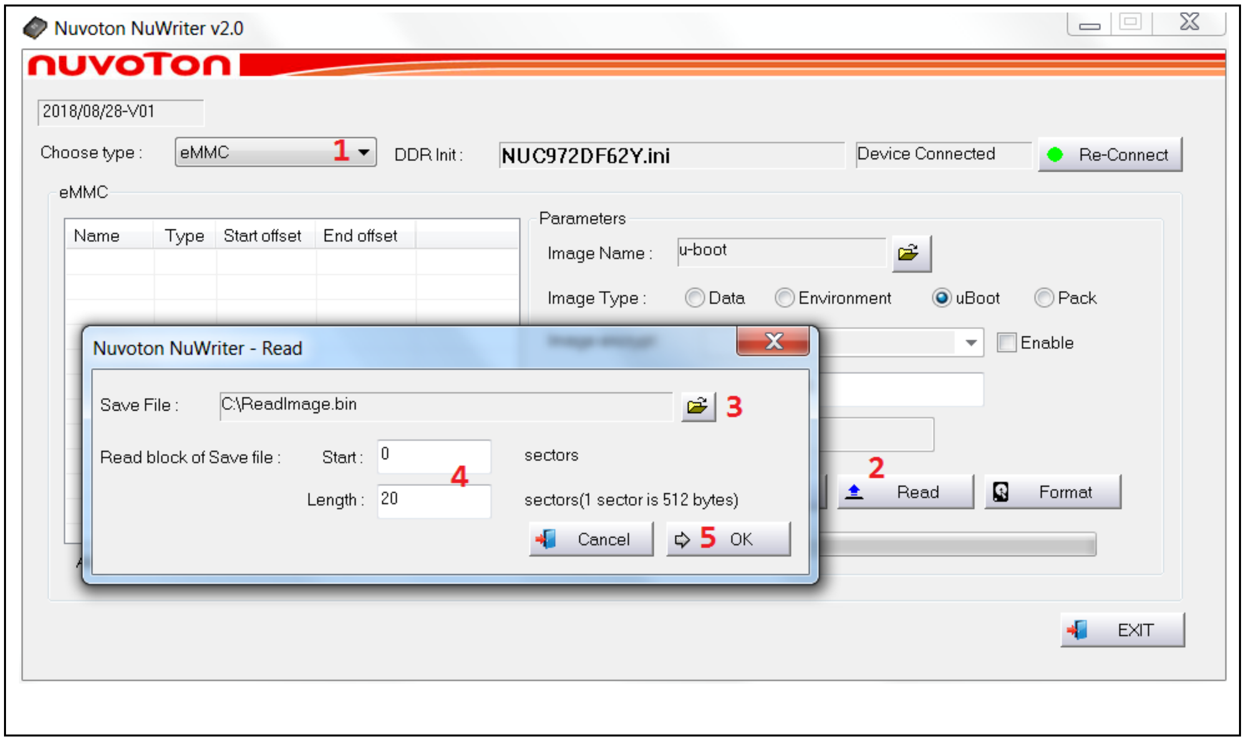

Nuvoton NuWriter v2.0
2018/08/28-V01
Choose type : eMMC 1
DDR Init : NUC972DF62Y.ini
Device Connected
Re-Connect
eMMC
Name Type Start offset End offset
Parameters
Image Name : u-boot
Image Type : Data Environment [✓ uBoot] Pack
Enable
Nuvoton NuWriter - Read
Save File : C:\ReadImage.bin 3
Read block of Save file : Start : 0 sectors
Length : 20 sectors(1 sector is 512 bytes)
4
Cancel
5 OK
2
Read
Format
EXIT

Figure 4-22 eMMC – Read

According to the figure above. Follow the steps below to read data from eMMC:

1. Select the "**eMMC**".
2. Click "Read".
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 "OK".

4.4.2.3 Format (FAT32)

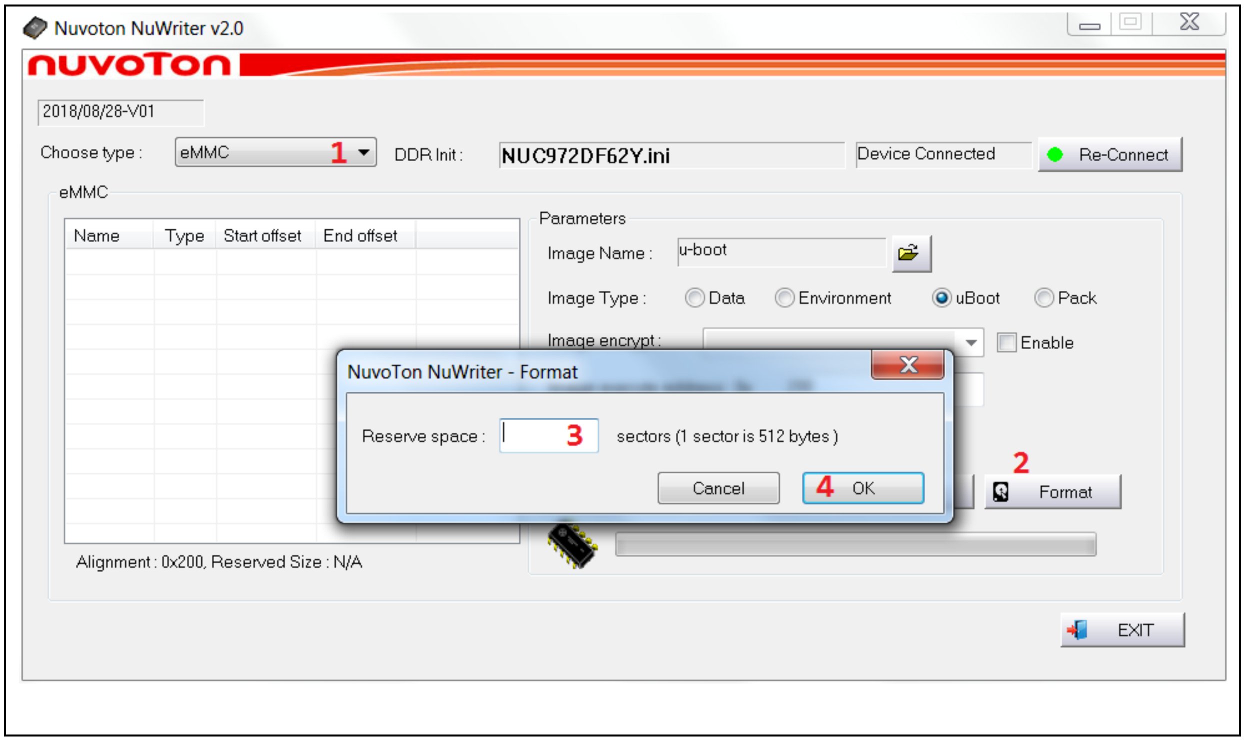


Figure 4-23 eMMC – Format

According to the figure above, follow the steps below to format eMMC:

1. Select "**eMMC**".
2. Click "Format".
3. Enter Reserve space(1 sector is 512bytes). Note : This parameter may cause existing image or file system is being destroyed.
4. Click "OK".

4.5 Pack Mode

Let users know how to merge many image into a pack image, as follows:

	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

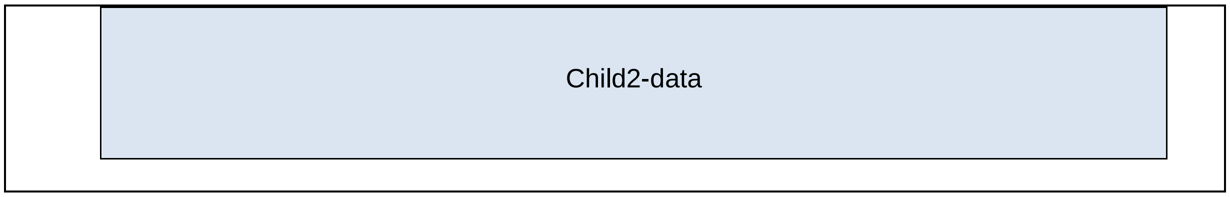


Figure 4-24 Pack header/Pack child header

Initial Marker = {0x00000005}.

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

File Number = { Number of image }.

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

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

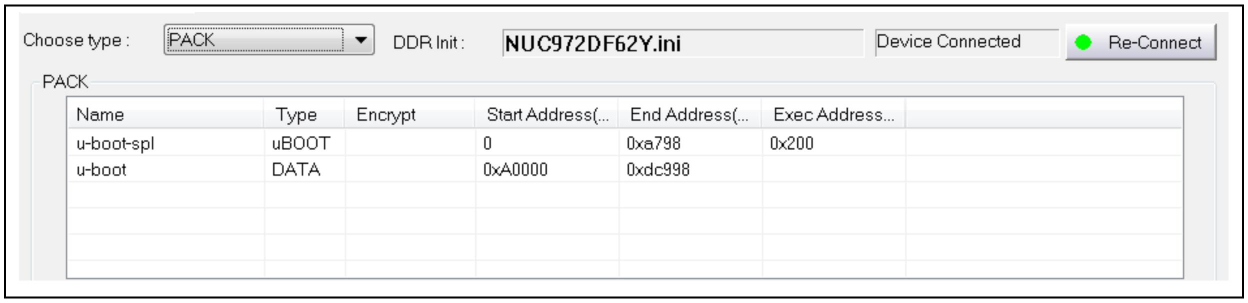
Child0-data = { Content of image 0 }.

Child0-File Length = { Length of image 1 }.

Child0-File Address = { Address of image 1 }.

Child0-data = { Content of image 1 }.

Assuming user merge "u-boot-spl.bin" and "u-boot.bin" into pack image as follows:


Choose type : PACK DDR Init : NUC972DF62Y.ini Device Connected Re-Connect

PACK

Name	Type	Encrypt	Start Address(...	End Address(...	Exec Address...
u-boot-spl	uBOOT		0	0xa798	0x200
u-boot	DATA		0xA0000	0xdc998	

Figure 4-25 Pack Image

Pack image content as follows:

Initial Marker = 0x00000005.

File Length = (0x0000a798+0x0003c998) ≅ 0x00050000.

File Number = 2.

Child0-File Length = 0x0000A798.

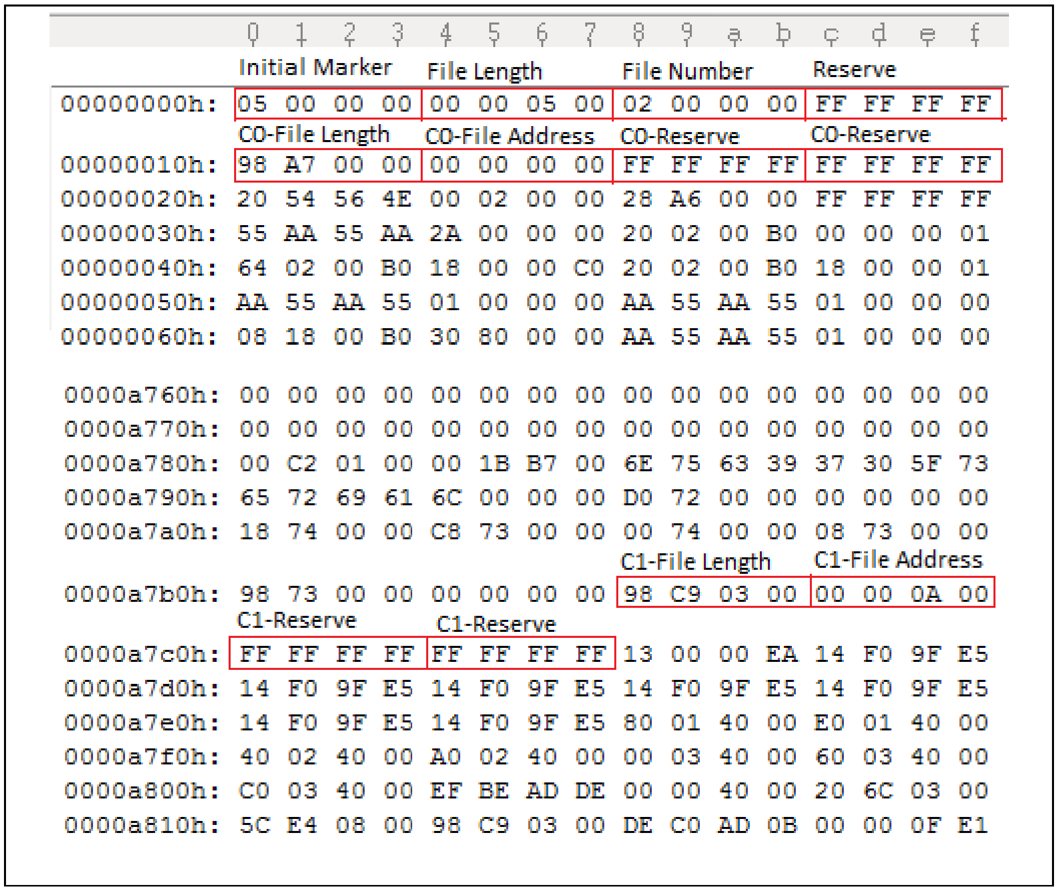
Child0-File Address = 0.

Child0-data = = { address 0x00000000 + 0x00000020 ~ 0x00000020 + 0x0000A798}.

Child1-File Length =0x3c998.

Child1-File Address =0xA0000.

Child1-data = = { address 0x0000A7B8 + 0x0000010 ~ 0x000A7C8 + 0x0003C998 }.



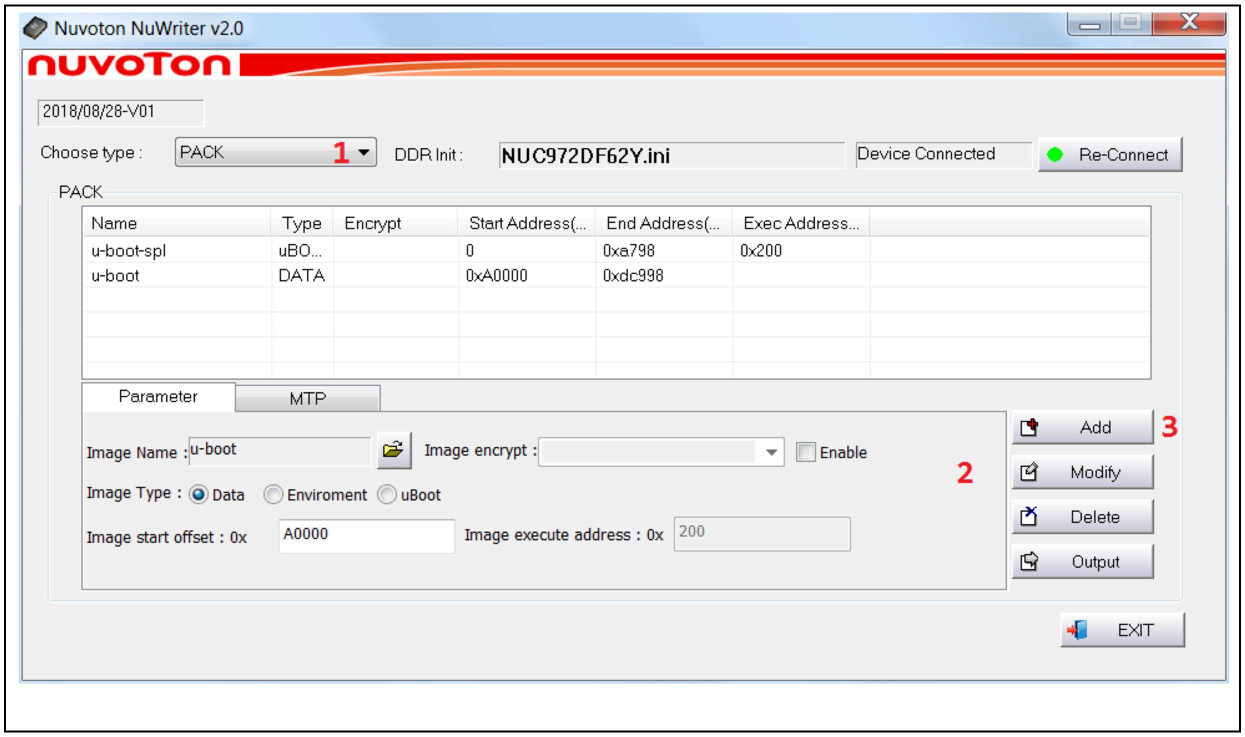
```
              0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
             Initial Marker File Length File Number Reserve
00000000h:  05 00 00 00 00 00 05 00 02 00 00 00 FF FF FF FF
             C0-File Length C0-File Address C0-Reserve C0-Reserve
00000010h:  98 A7 00 00 00 00 00 00 FF FF FF FF FF FF FF FF
00000020h:  20 54 56 4E 00 02 00 00 28 A6 00 00 FF FF FF FF
00000030h:  55 AA 55 AA 2A 00 00 00 20 02 00 B0 00 00 00 01
00000040h:  64 02 00 B0 18 00 00 C0 20 02 00 B0 18 00 00 01
00000050h:  AA 55 AA 55 01 00 00 00 AA 55 AA 55 01 00 00 00
00000060h:  08 18 00 B0 30 80 00 00 AA 55 AA 55 01 00 00 00

0000a760h:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000a770h:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000a780h:  00 C2 01 00 00 1B B7 00 6E 75 63 39 37 30 5F 73
0000a790h:  65 72 69 61 6C 00 00 00 D0 72 00 00 00 00 00 00
0000a7a0h:  18 74 00 00 C8 73 00 00 00 74 00 00 08 73 00 00
                                     C1-File Length C1-File Address
0000a7b0h:  98 73 00 00 00 00 00 00 98 C9 03 00 00 00 0A 00
             C1-Reserve    C1-Reserve
0000a7c0h:  FF FF FF FF FF FF FF FF 13 00 00 EA 14 F0 9F E5
0000a7d0h:  14 F0 9F E5 14 F0 9F E5 14 F0 9F E5 14 F0 9F E5
0000a7e0h:  14 F0 9F E5 14 F0 9F E5 80 01 40 00 E0 01 40 00
0000a7f0h:  40 02 40 00 A0 02 40 00 00 03 40 00 60 03 40 00
0000a800h:  C0 03 40 00 EF BE AD DE 00 00 40 00 20 6C 03 00
0000a810h:  5C E4 08 00 98 C9 03 00 DE C0 AD 0B 00 00 0F E1
```

Figure 4-26 Output Pack.bin

4.5.1 Operation Steps

Pack mode can merge many image into a pack image, user can use NuWriter to burn pack image into the device (NAND flash, SPI flash, eMMC).

4.5.1.1 *Add a New Image*



Name	Type	Encrypt	Start Address(...	End Address(...	Exec Address...
u-boot-spl	uBO...		0	0xa798	0x200
u-boot	DATA		0xA0000	0xdc998	

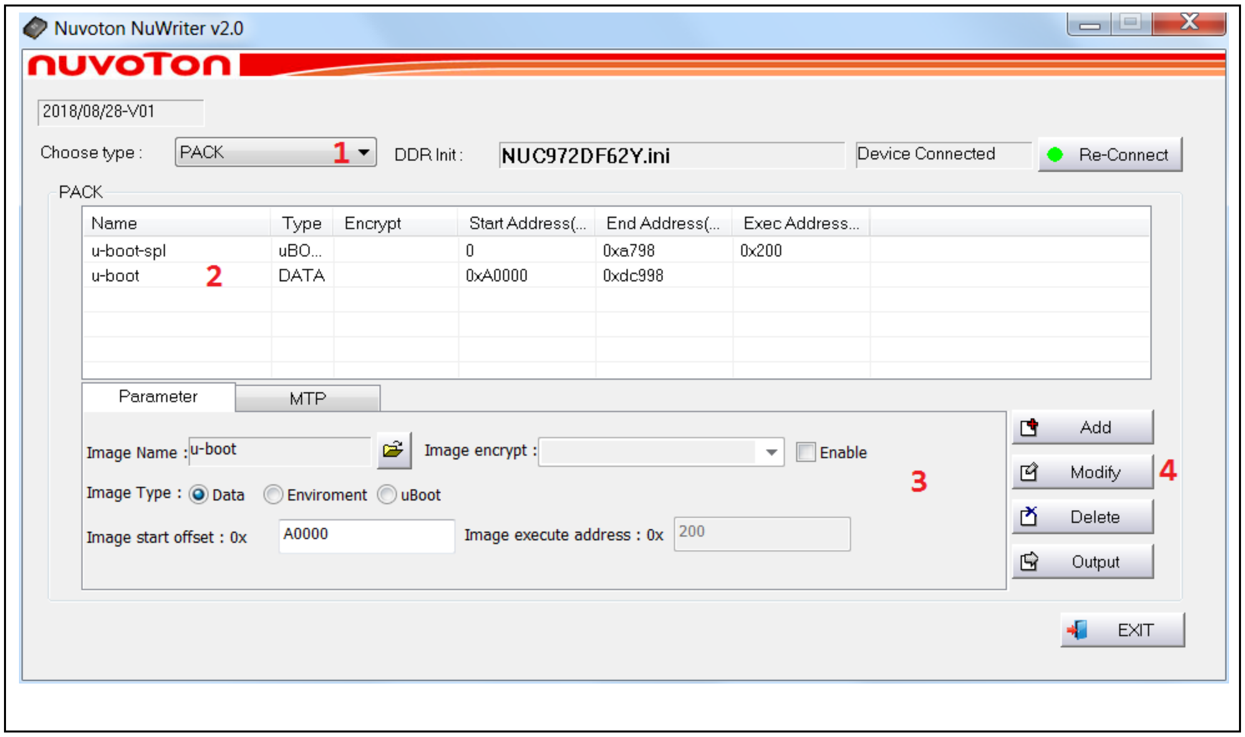
Parameter
MTP
Image Name : u-boot
Image encrypt :
Enable
2
Image Type : Data Enviroment uBoot
Image start offset : 0x A0000
Image execute address : 0x 200
Add
3
Modify
Delete
Output
EXIT

Figure 4-27 PACK – New Image

According to the figure above, follow the steps below to add a new image to pack list:

1. Select the “**Pack**”.
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 encrypt: Select encrypt file and Set enable or disable. (N9H30 does not support this option)
   - Image execute address: Enter image execute address. Only is uboot type is valid.
   - Image start offset: Enter image start offset.
3. Click “Add”.
4. Waiting for finishing progress bar.
5. After “Burn” the image, click the “Verify” button to read back the image data to make sure the burning status.

4.5.1.2 Modify an Image


Nuvoton NuWriter v2.0

2018/08/28-V01

Choose type : PACK

DDR Init : NUC972DF62Y.ini

Device Connected

Name	Type	Encrypt	Start Address(...	End Address(...	Exec Address...
u-boot-spl	uBO...		0	0xa798	0x200
u-boot	DATA		0xA0000	0xdc998	

Image Name : u-boot

Image encrypt : [ ] Enable

Image Type : [✓ Data] Enviroment uBoot

Image start offset : 0x A0000

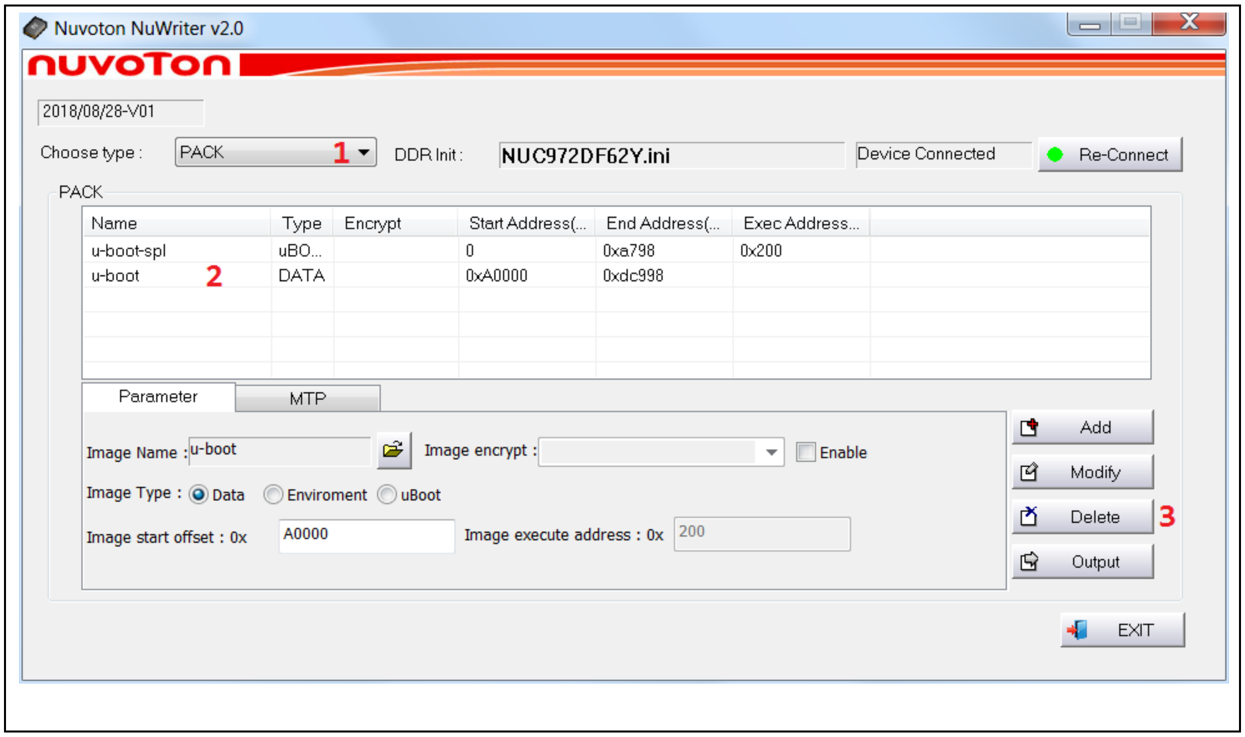
Image execute address : 0x 200

Figure 4-28 PACK – Modify Image

According to the figure above. Follow the steps below to modify image from pack list:

1. Select the “**PACK**”.
2. Double click Image name in the pack list 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 encrypt: Select encrypt file and Set enable or disable. (N9H30 does not support this option)
   - Image execute address: Enter image execute address. Only is uboot type is valid.
   - Image start offset: Enter image start offset.
4. Click “Modify”.

4.5.1.3 Delete an Image



Name	Type	Encrypt	Start Address(...	End Address(...	Exec Address...
u-boot-spl	uBO...		0	0xa798	0x200
u-boot	DATA		0xA0000	0xdc998	

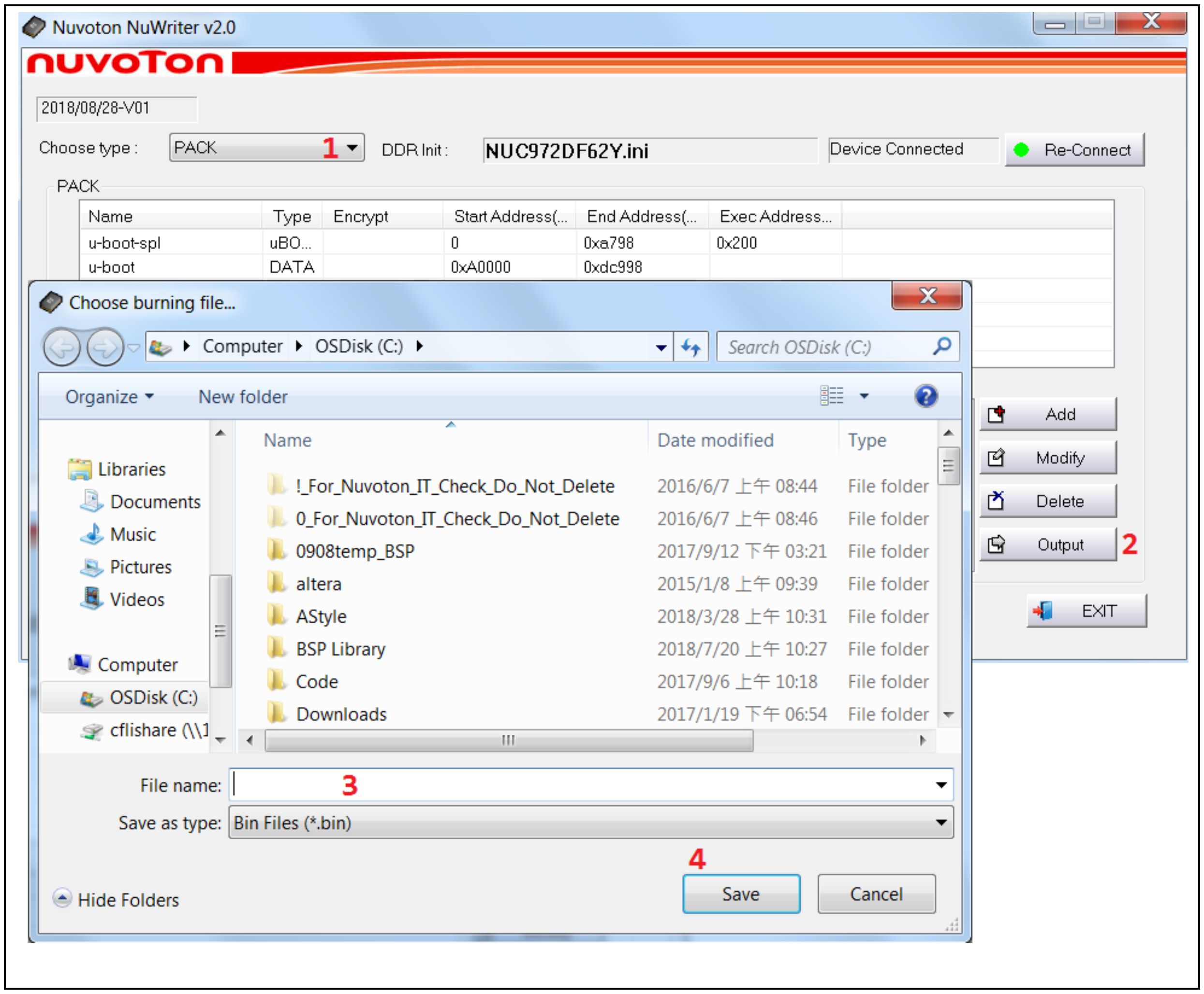
Image Name : u-boot
Image encrypt :
Enable
Image Type : (● Data) Enviroment uBoot
Image start offset : 0x A0000
Image execute address : 0x 200

Figure 4-29 PACK – Delete Image

According to the figure above. Follow the steps below to delete image from pack list:

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

4.5.1.4 *Create a Pack Image*



Name	Type	Encrypt	Start Address(...	End Address(...	Exec Address...
u-boot-spl	uBO...		0	0xa798	0x200
u-boot	DATA		0xA0000	0xdc998	

Name	Date modified	Type
!_For_Nuvoton_IT_Check_Do_Not_Delete	2016/6/7 上午 08:44	File folder
0_For_Nuvoton_IT_Check_Do_Not_Delete	2016/6/7 上午 08:46	File folder
0908temp_BSP	2017/9/12 下午 03:21	File folder
altera	2015/1/8 上午 09:39	File folder
AStyle	2018/3/28 上午 10:31	File folder
BSP Library	2018/7/20 上午 10:27	File folder
Code	2017/9/6 上午 10:18	File folder
Downloads	2017/1/19 下午 06:54	File folder

Figure 4-30 PACK – Output Pack Image

According to figure above. Follow the steps below to output pack image:

1. Select “**PACK**”.
2. Click “Output”.
3. Browse save file.
4. Click ”Open” to output pack image.

4.5.1.5 Burn a Pack Image

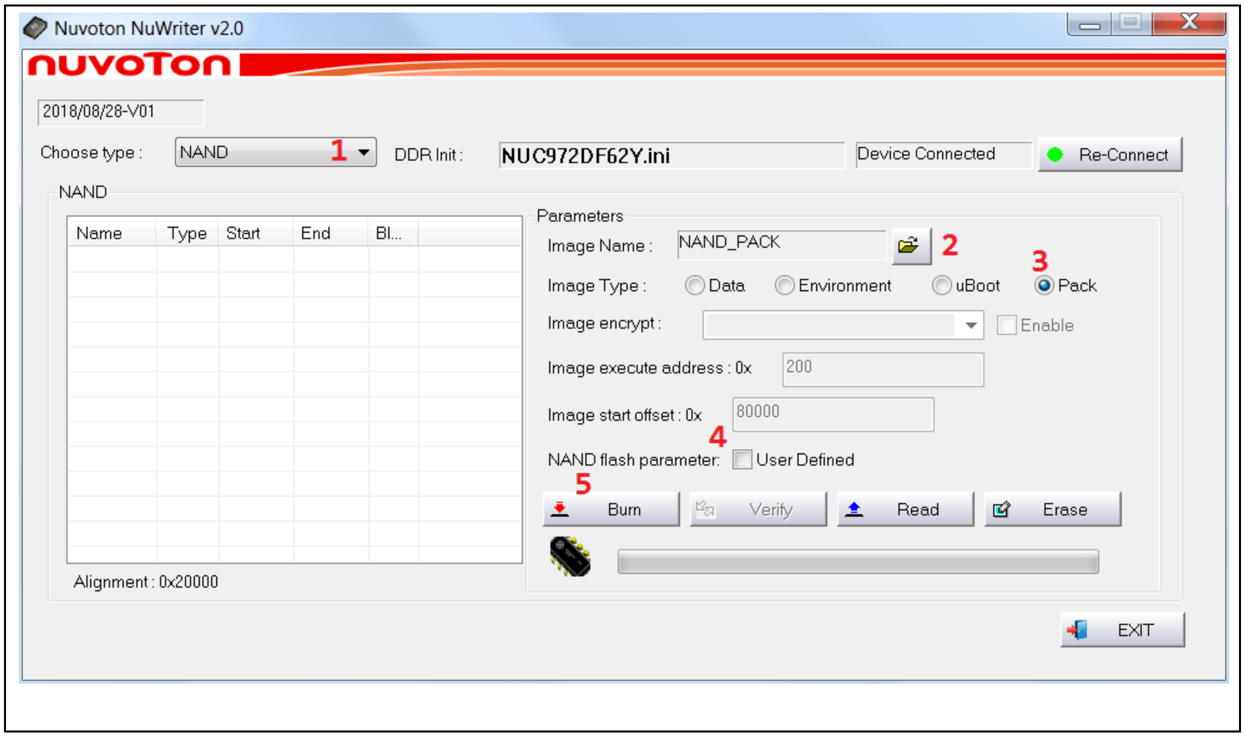


Figure 4-31 PACK – Burn Pack Image

According to figure above. Follow the steps below to burn pack image into NAND flash:

1. Select "**NAND**".
2. Browse pack image.
3. Select image type to "Pack".
4. Check the box if you want to use User-Defined NAND parameters.
5. Click "Burn".

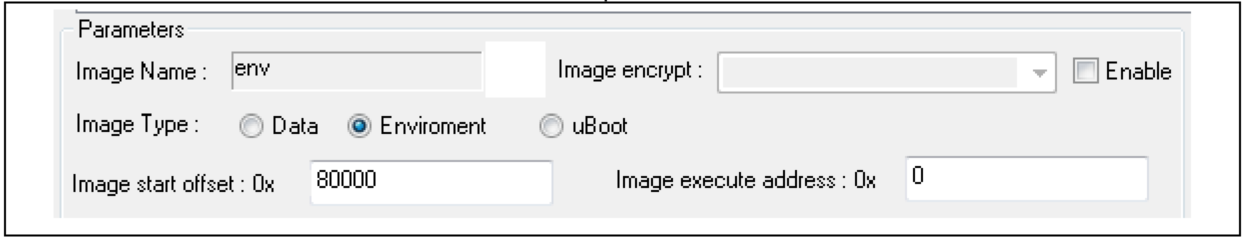
4.5.2 Create and Burn a Pack Image

Preparation of related files:

1. u-boot.bin (Default: offset address is set to 0xA0000, execute address is set to 0xE00000) .
2. u-boot-spl.bin (Default:DDR execute address is set to 0x200) .
3. env.txt (Default : offset address is set to 0x80000) .

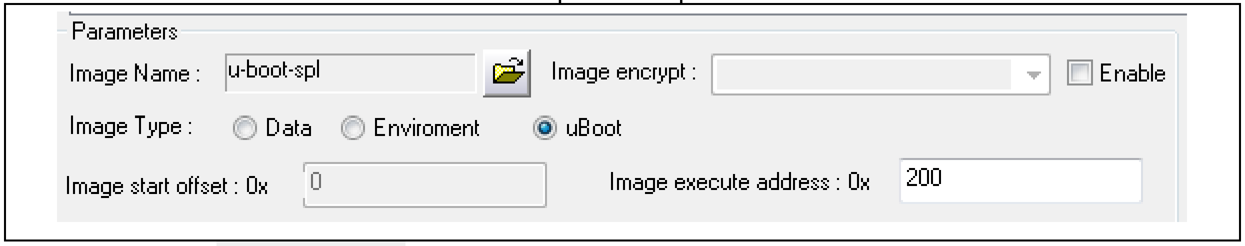
Currently, Suppose user wants to build the pack image, This pack image includes env.txt, u-boot.bin and u-boot-spl.bin. Then it burn to NAND flash. Follow the steps below to build pack image:

1. Select "PACK".
2. Click and selcet the "env.txt" file path:



3. Click.

4. Click and select the “u-boot-spl.bin” file path:



Parameters

Image Name : u-boot-spl

Image encrypt : ☐ Enable

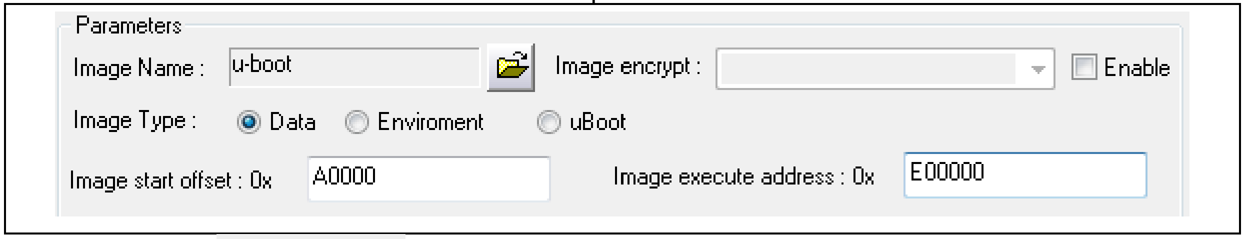
Image Type : Data Enviroment [✓ uBoot]

Image start offset : 0x 0

Image execute address : 0x 200

5. Click Add.

6. Click and select “u-boot.bin” file path:



Parameters

Image Name : u-boot

Image encrypt : ☐ Enable

Image Type : [✓ Data] Enviroment uBoot

Image start offset : 0x A0000

Image execute address : 0x E00000

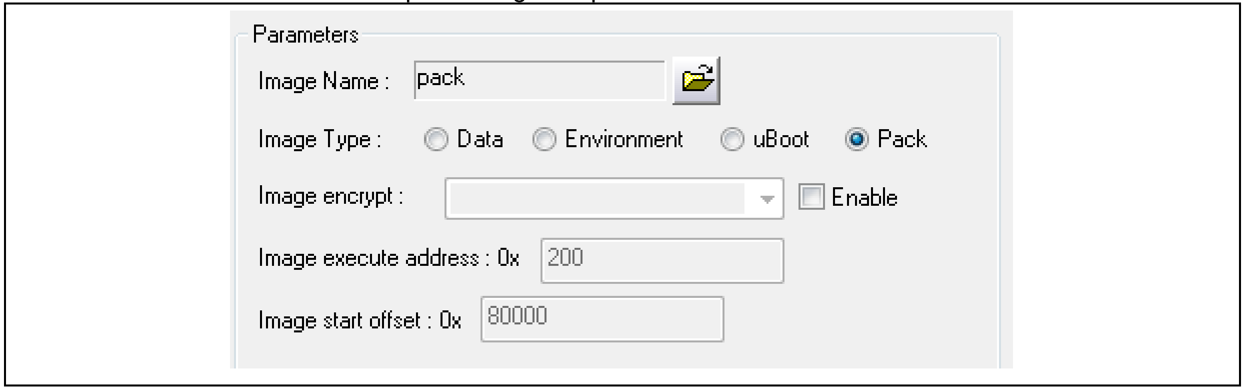
7. Click Add.

8. Click Output, and save the pack image.

Follow the steps below to burn pack image into NAND flash:

1. Select “NAND”.

2. Click and select pack image file path:



Parameters

Image Name : pack

Image Type : Data Environment uBoot [✓ Pack]

Image encrypt : ☐ Enable

Image execute address : 0x 200

Image start offset : 0x 80000

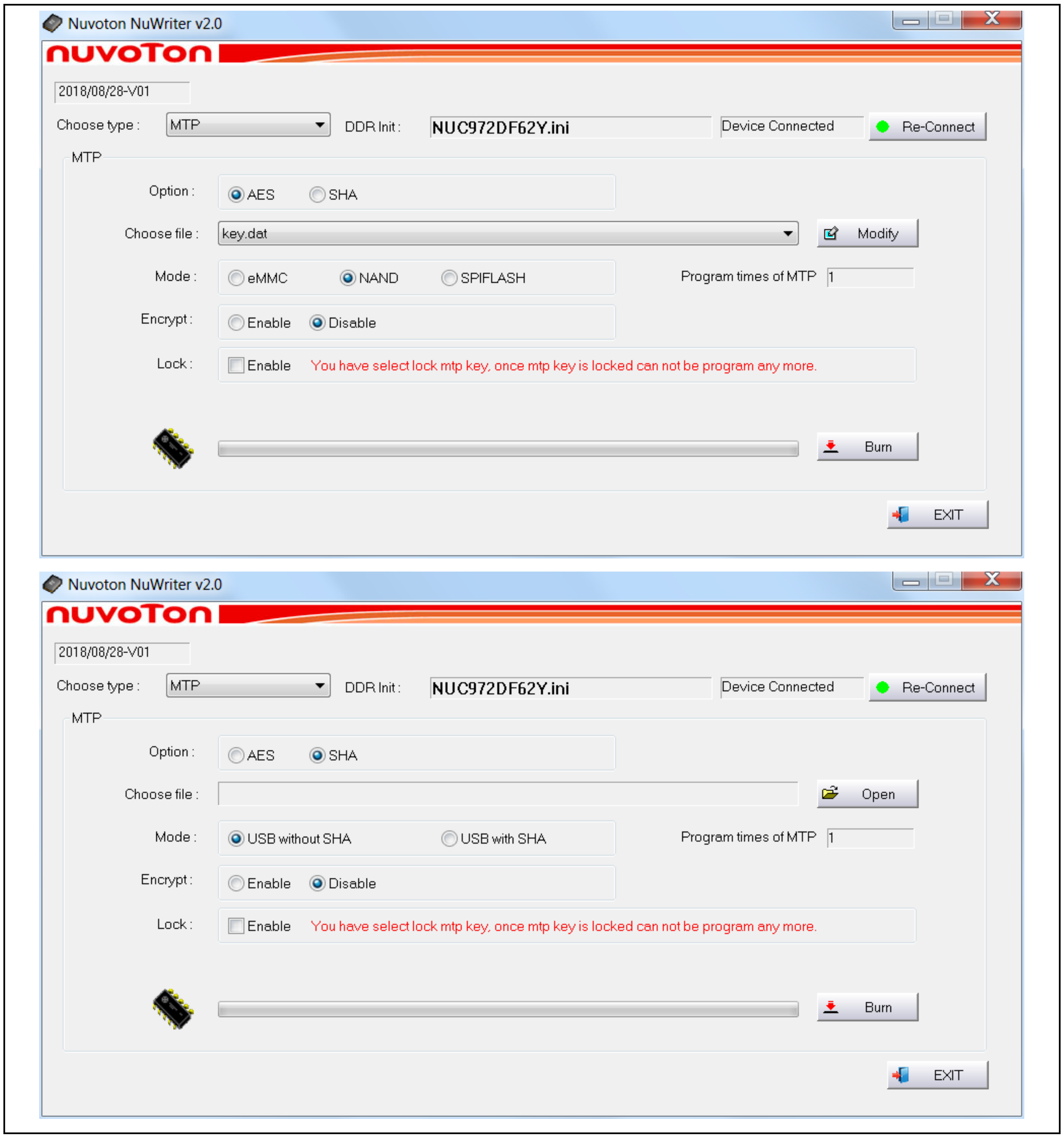
1. Click Burn.

4.6 MTP Mode (N9H30 does not support this mode)

On the MTP form, user can select MTP keys file to burn into NUC970 Series MPU.

This MTP keys can protect user’s binary code in device (eMMC, NAND Flash, SPI Flash).

4.6.1 Operation Steps



Nuvoton NuWriter v2.0

2018/08/28-V01

Choose type : MTP — DDR Init : NUC972DF62Y.ini — Device Connected — Re-Connect

MTP

Option : [✓ AES] SHA

Choose file : key.dat — Modify

Mode : eMMC [✓ NAND] SPIFLASH — Program times of MTP 1

Encrypt : Enable [✓ Disable]

Lock : ☐ Enable — You have select lock mtp key, once mtp key is locked can not be program any more.

Burn

EXIT

Nuvoton NuWriter v2.0

2018/08/28-V01

Choose type : MTP — DDR Init : NUC972DF62Y.ini — Device Connected — Re-Connect

MTP

Option : AES [✓ SHA]

Choose file : — Open

Mode : [✓ USB without SHA] USB with SHA — Program times of MTP 1

Encrypt : Enable [✓ Disable]

Lock : ☐ Enable — You have select lock mtp key, once mtp key is locked can not be program any more.

Burn

EXIT

Figure 4-32 NuWriter – MTP Mode

4.6.1.1 Add a New Key

1. Under the NuWrite folder(For example, C:\NuWriter). Enter the key_cfg folder.

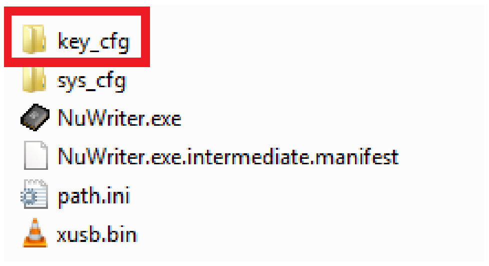


Figure 4-33 NuWriter – Folder

2. Create a text file and enter password. The password in the following format, The first line must be 256, and eight consecutive big-endian mode key. (For example, C:\NuWriter key_cfg\key.dat).

```
256
0x12345678
0x23456789
0x3456789a
0x456789ab
0x56789abc
0x6789abcd
0x789abcde
0x89abcdef
```

3. Re-open "Nu-Writer" tool, and select "MTP".
4. Select "key.dat".
5. Select burning option:
   - Protection mode selction:
     - AES
       - ➢ Boot mode selection: eMMC, NAND, or SPIFLASH.
     - SHA.
       - ➢ Boot mode selection: USB without SHA or USB with SHA.
         USB without SHA: NuWriter use USB interface to communicate NUC970 without SHA.
         USB with SHA: NuWriter use USB interface to communicate NUC970 with SHA.
6. Click "Burn".

4.7 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.

4.7.1 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 press the Start button to program multiple devices at the same time. If the storage device is NAND, users can check the box to set NAND parameters.

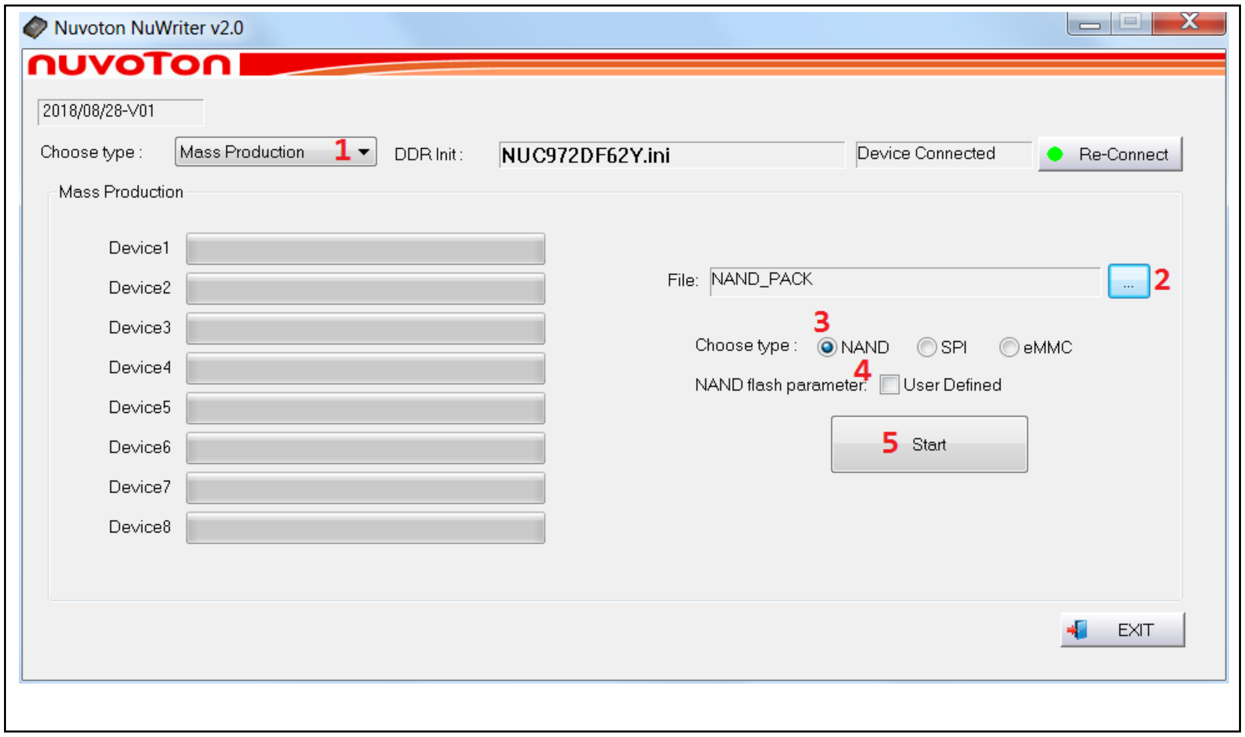
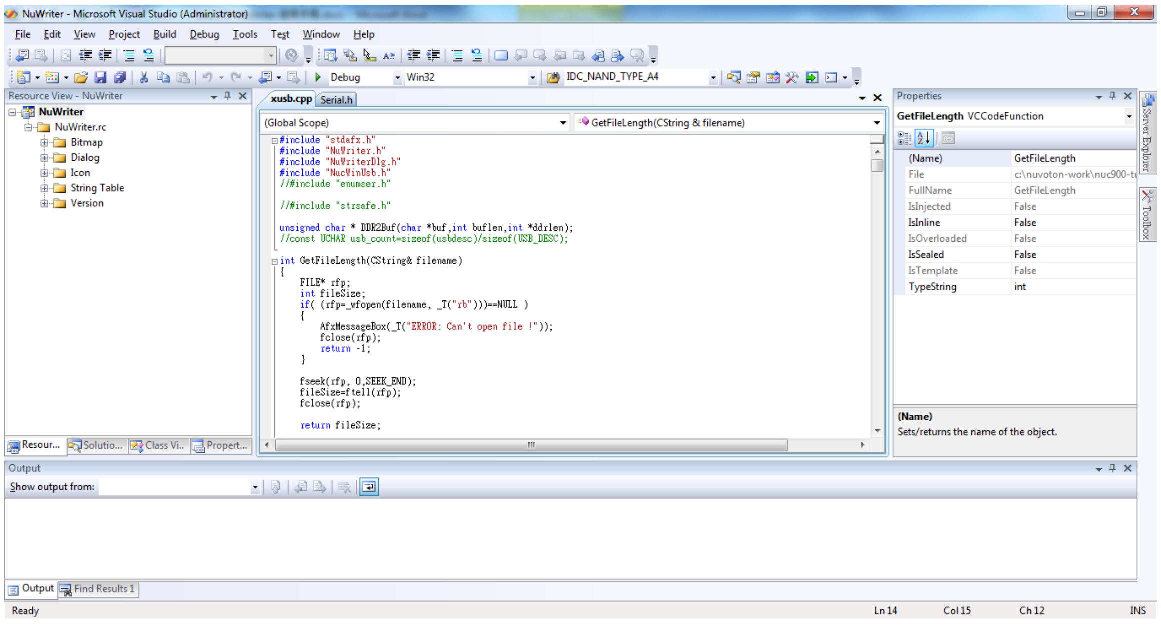


Figure 4-34 Mass Production

5 NUWRITER SOURCE CODE

NuWriter is using the VS2008 development environment, as follows.



```
#include "stdafx.h"
#include "NuWriter.h"
#include "NuWriterDlg.h"
#include "NucWinUsb.h"
//#include "enumser.h"

//#include "strsafe.h"

unsigned char * DDR2Buf(char *buf,int buflen,int *ddrlen);
//const UCHAR usb_count=sizeof(usbdesc)/sizeof(USB_DESC);

int GetFileLength(CString& filename)
{
    FILE* rfp;
    int fileSize;
    if( (rfp=_wfopen(filename, _T("rb")))==NULL )
    {
        AfxMessageBox(_T("ERROR: Can't open file !"));
        fclose(rfp);
        return -1;
    }

    fseek(rfp, 0,SEEK_END);
    fileSize=ftell(rfp);
    fclose(rfp);

    return fileSize;
```

Properties: GetFileLength VCCodeFunction	
(Name)	GetFileLength
File	c:\nuvoton-work\nuc900-t
FullName	GetFileLength
IsInjected	False
IsInline	False
IsOverloaded	False
IsSealed	False
IsTemplate	False
TypeString	int

(Name)
Sets/returns the name of the object.

Figure 5-1 NuWriter – Project

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

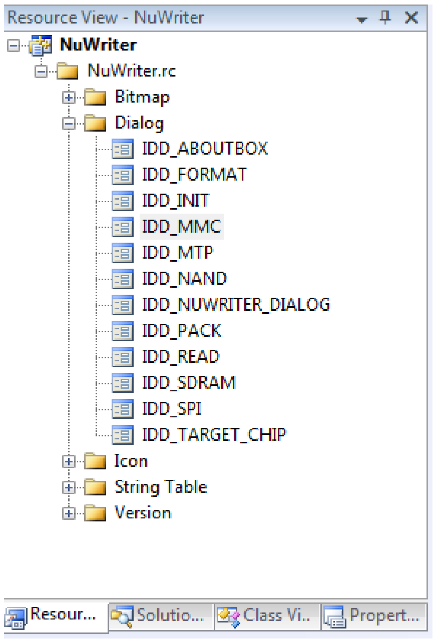


Figure 5-2 NuWriter – Resource View

After double click IDD_SRAM, user can see following screen:

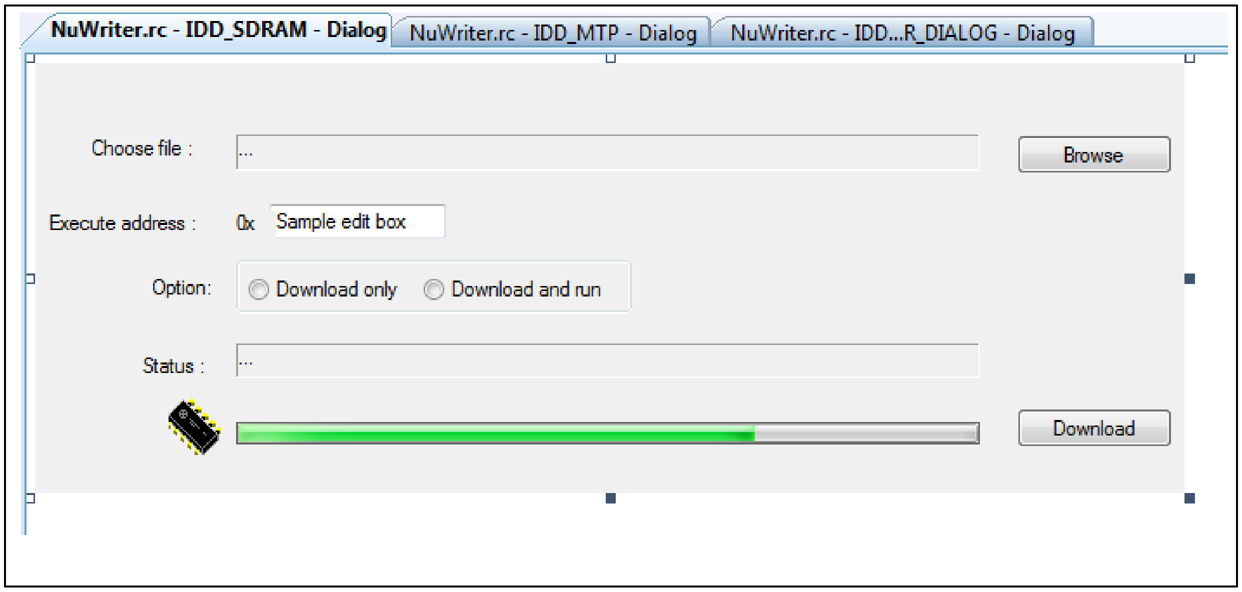
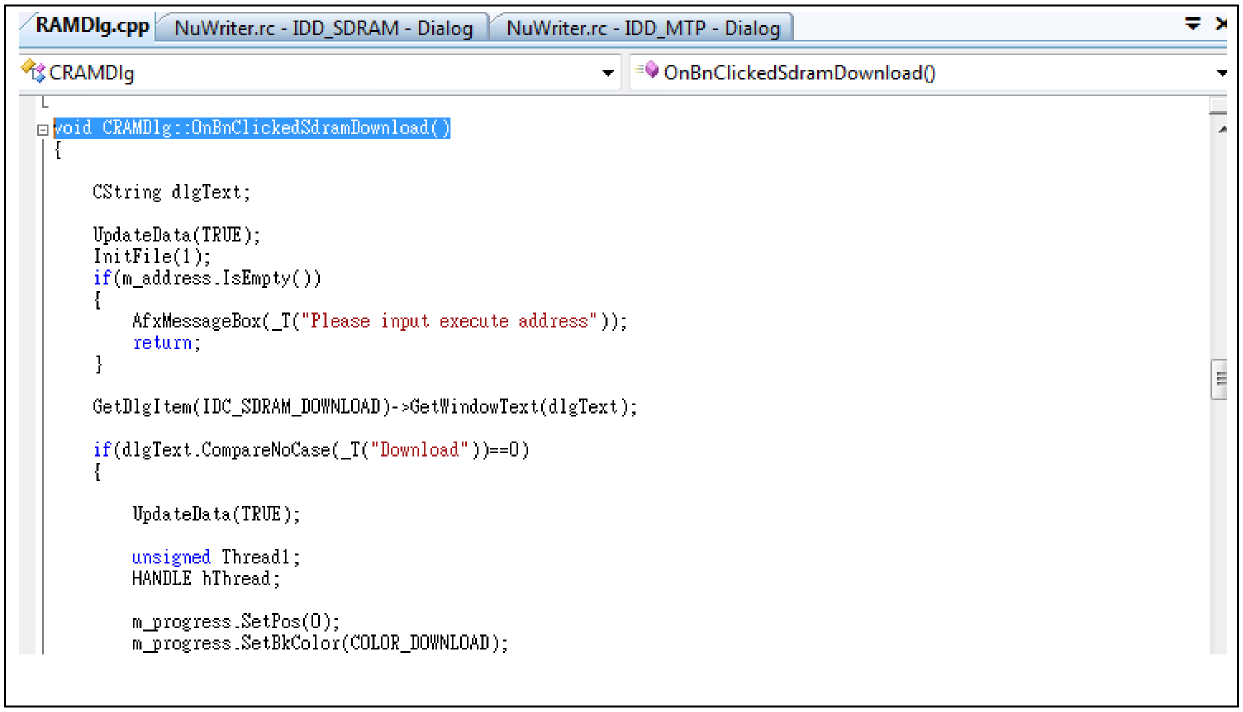


Figure 5-3 IID_SDRAM Dialog

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


RAMDlg.cpp
NuWriter.rc - IDD_SDRAM - Dialog
NuWriter.rc - IDD_MTP - Dialog
CRAMDlg
OnBnClickedSdramDownload()

```
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;

        m_progress.SetPos(0);
        m_progress.SetBkColor(COLOR_DOWNLOAD);
```

Figure 5-4 OnBnClockedSdramDownload function

Many buttons to complete all functions in NuWriter tools. User can view the source code of button to understand how communication between NuWriter and NUC970/N9H30. The main source code is NucWinUsb.h and NucWinUsb.cpp.

NucWinUsb.h ：

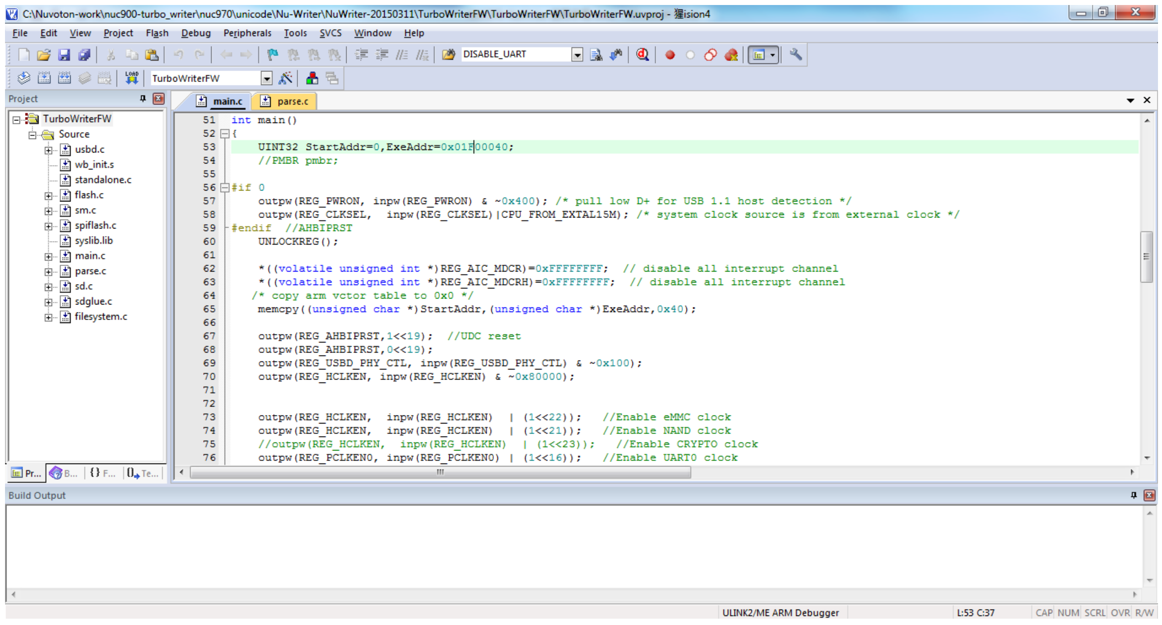
```
static const GUID OSR_DEVICE_INTERFACE[] = { 0xD696BFEB, 0x1734, 0x417d,
{ 0x8A, 0x04, 0x86,0xD0,0x10,0x71,0xC5,0x12 } };
typedef struct PIPE_ID
{
     UCHAR  PipeInId;
     UCHAR  PipeOutId;
}Pipe_Id,*PPipe_Id;

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;
```

If you have NUC970/N9H30 series microprocessor connect with the computer, they can communicate through 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........
}
```

6 FIRMWARE CODE(XUSB.BIN)

xusb.bin is using the Keil development environment, as follows.



```
int main()
{
    UINT32 StartAddr=0,ExeAddr=0x01F00040;
    //PMBR pmbr;

#if 0
    outpw(REG_PWRON, inpw(REG_PWRON) & ~0x400); /* pull low D+ for USB 1.1 host detection */
    outpw(REG_CLKSEL,  inpw(REG_CLKSEL)|CPU_FROM_EXTAL15M); /* system clock source is from external clock */
#endif  //AHBIPRST
    UNLOCKREG();

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

    outpw(REG_AHBIPRST,1<<19);  //UDC reset
    outpw(REG_AHBIPRST,0<<19);
    outpw(REG_USBD_PHY_CTL, inpw(REG_USBD_PHY_CTL) & ~0x100);
    outpw(REG_HCLKEN, inpw(REG_HCLKEN) & ~0x80000);


    outpw(REG_HCLKEN,  inpw(REG_HCLKEN)  | (1<<22));   //Enable eMMC clock
    outpw(REG_HCLKEN,  inpw(REG_HCLKEN)  | (1<<21));   //Enable NAND clock
    //outpw(REG_HCLKEN,  inpw(REG_HCLKEN)  | (1<<23));   //Enable CRYPTO clock
    outpw(REG_PCLKEN0, inpw(REG_PCLKEN0) | (1<<16));   //Enable UART0 clock
```

Figure 6-1 NuWriterFW – Project

Firmware code (xusb.bin) is a communication between NuWriter and NUC970/N9H30 series microprocessor. First, NuWriter transmit xusb.bin to NUC970/N9H30 and executed. NUC970/N9H30 waits for pc transmit command on ParseFlashType(void) functions. If user wants to increase command from NuWriter tools, user can add value of _usbd_flash_type, and modify ParseFlashType(void) and usbd_control_packet(void). You can refer to the following source code.

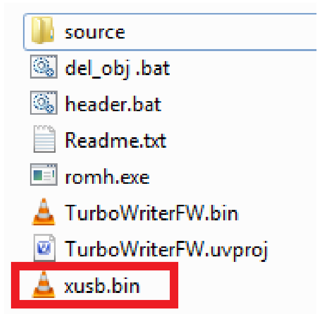
```
INT ParseFlashType(void){
switch (_usbd_flash_type){
     /* Add new commands from nuwriter transmit */
     /*
     case yyyyy:
     TODO: Add your control notification handler code here
     break;
     */
     case USBD_FLASH_SDRAM:
          UXmodem_SDRAM();
          _usbd_flash_type = -1;
     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_MTP:
            UXmodem_MTP();
            _usbd_flash_type = -1;
      break;
      default:
      break;
}
```

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........
}
```

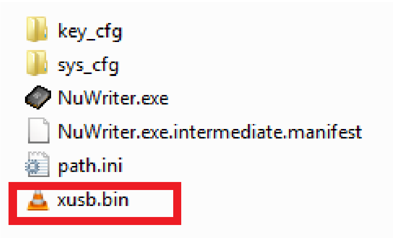
After editing, you can compile a xusb.bin as follows:



source
del_obj .bat
header.bat
Readme.txt
romh.exe
TurboWriterFW.bin
TurboWriterFW.uvproj
xusb.bin

Figure 6-2 NuWriterFW – Folder

And cover xusb.bin in the NuWriter folder.



key_cfg
sys_cfg
NuWriter.exe
NuWriter.exe.intermediate.manifest
path.ini
xusb.bin

Figure 6-3 NuWriter – Folder

7 REVISION HISTORY

Date	Revision	Description
2015.04.23	1.00	Initially issued.
2015.06.02	1.00	Burning Nand Image use block units.
2015.06.05	1.00	Add AES on the Pack mode
2015.06.15	1.00	Disable FS type from NAND mode page. User can use data mode instead of fs type.
2015.07.28	1.00	Modified eMMC clock to 1MHz when Nuwriter run on USB mode. eMMC clock is set to 6Mhz for boot.
2015.08.25	1.00	NuWriver add xusb16.bin, xusb.bin and xusb64.bin for DDR size(16MB, 32MB and 64MB).
2015.08.31	1.00	Add ChipReadWithBad in path.ini; 1: NAND read data and OOB , 0: NAND read data. Add ChipEraseWithBad in path.ini; 1: NAND erase data and OOB , 0: NAND erase data.
2015.09.24	1.00	Modified burning image failed when image size greater 31MB. Add burning MTP functions to pack mode.
2015.10.21	1.00	Add retry function to connect nuc970 for the first time.
2018.08.14	1.01	Add N9H30. Add mass production mode.
2018.08.30	2.00	Add User-Defined NAND parameter on the NAND mode and mass production mode,.