N9H26 SD Loader Reference Guide

Document Information

Abstract	Introduce the steps to build and launch SD Loader for the N9H26 series microprocessor (MPU).
Apply to	N9H26 series

Table of Contents

1 Introduction

The SD loader is a firmware stored at the SD card for booting purpose. It will set the system clock, initialize the relevant modules, and then load the next firmware to DRAM to execute.

The SD loader supports the following features:

- Initialize more modules such as SPU, RTC, and so on.
- Load Logo image to DRAM if it existed at SD card.
- Load next firmware to DRAM if it existed at SD card.
- Execute next firmware. Normally, it should be NVT loader.

Nuvoton provides SD loader source code within the N9H26 series microprocessor (MPU) BSP.

2 SD Loader BSP Directory Structure

This chapter introduces the SD loader related files and directories in the N9H26 BSP.

2.1 Loaders\SDLoader

GCC\	The GCC project files for the SD loader.
KEIL\	The KEIL project files for the SD loader.
SdLoader.c	The main function for the SD loader.
SdLoader.scf	The scatter file for the SD loader
sd.c	SD card driver of N9H26.
Other files	System driver of N9H26.

2.2 Loaders\Binary

N9H26_SDLoader_xxx.bin	The binary file of the SD loader for different project targets.

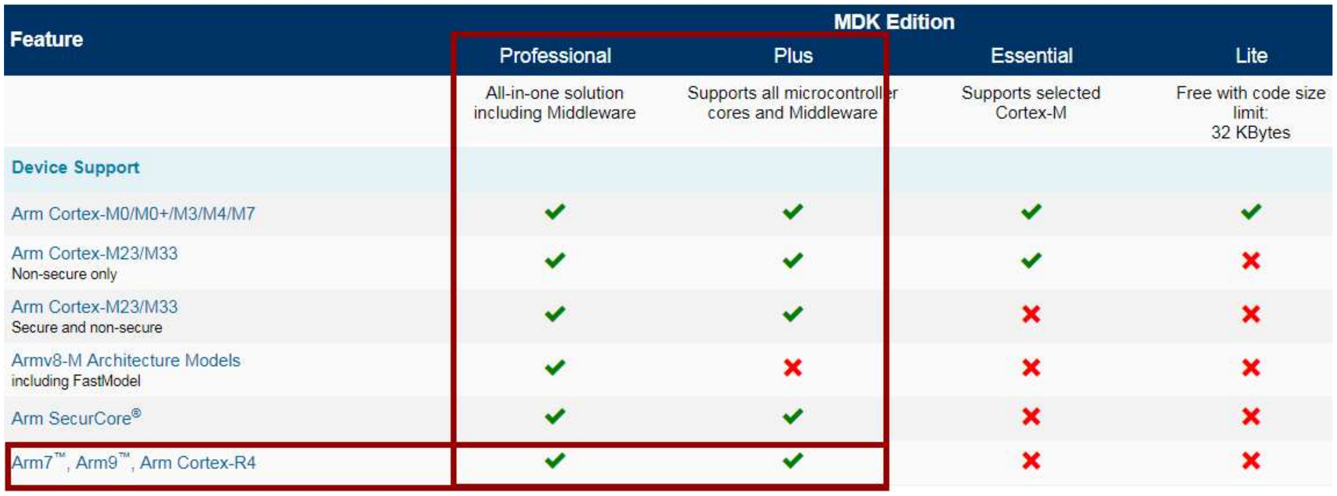
3 SD Loader Source Code

Complete source codes are included in the *N9H26 BSP Loaders\SDLoader* directory:

3.1 Development Environment

Keil IDE and Eclipse are used as Non-OS BSP development environment, which uses J-Link ICE or ULINK2 ICE (optional) for debugging. This document uses Keil IDE to describe the project structure. To support ARM9, MDK Plus or Professional edition shall be used.

Note that Keil IDE and ICE need to be purchased from vendor sources.



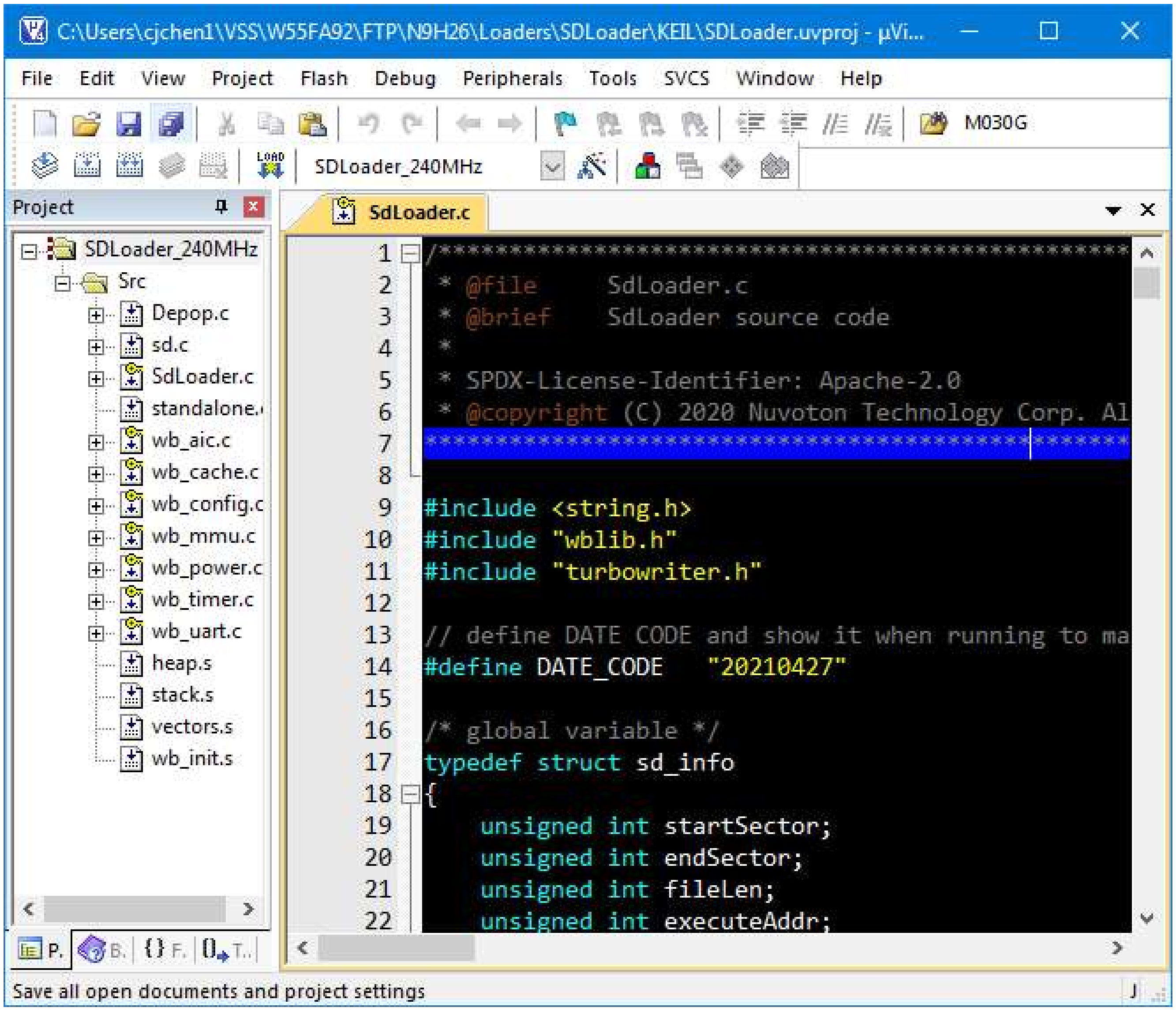
Feature	MDK Edition			
	Professional	Plus	Essential	Lite
	All-in-one solution including Middleware	Supports all microcontroller cores and Middleware	Supports selected Cortex-M	Free with code size limit: 32 KBytes
Device Support				
Arm Cortex-M0/M0+/M3/M4/M7	✓	✓	✓	✓
Arm Cortex-M23/M33 Non-secure only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

Figure 3-1 Keil MDK License Chart

3.2 Project Structure

The SD loader project includes one main function file and some driver files of N9H26. It doesn't link any driver library in order to shrink the binary code size.

Please note that the binary code size of the SD loader MUST less than or equal to 15360 bytes (512 bytes * 30 sectors).


C:\Users\cjchen1\VSS\W55FA92\FTP\N9H26\Loaders\SDLoader\KEIL\SDLoader.uvproj - µVi...
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
M030G
SDLoader_240MHz
Project
SDLoader_240MHz
Src
Depop.c
sd.c
SdLoader.c
standalone.
wb_aic.c
wb_cache.c
wb_config.c
wb_mmu.c
wb_power.c
wb_timer.c
wb_uart.c
heap.s
stack.s
vectors.s
wb_init.s
SdLoader.c
Save all open documents and project settings

```
/*********************************************************
 * @file     SdLoader.c
 * @brief    SdLoader source code
 *
 * SPDX-License-Identifier: Apache-2.0
 * @copyright (C) 2020 Nuvoton Technology Corp. Al
*********************************************************

#include <string.h>
#include "wblib.h"
#include "turbowriter.h"

// define DATE CODE and show it when running to ma
#define DATE_CODE    "20210427"

/* global variable */
typedef struct sd_info
{
    unsigned int startSector;
    unsigned int endSector;
    unsigned int fileLen;
    unsigned int executeAddr;
```

Figure 3-2 SD Loader Project Tree on Keil MDK

The SD loader project includes some targets that can be used in different situations.

- **SDLoader_240MHz**: Set system core clock to 240MHz. This is the official standard target.
- **SDLoader_192MHz**: Set system core clock to 192MHz.

3.3 System Initialization

The system initialization code is located in main function, including system code clock setting, enable cache feature, and UART debug port setting. It also initializes some necessary peripherals during the boot process.

```
int main()
{
    NVT_SD_INFO_T image;
    int volatile count, i;
    int ibr_boot_sd_port;

    /* Clear Boot Code Header in SRAM to avoid booting fail issue */
    outp32(0xFF000000, 0);

    //--- enable cache to speed up booting
    sysDisableCache();
    sysFlushCache(I_D_CACHE);
    sysEnableCache(CACHE_WRITE_BACK);

    sysprintf("N9H26 SD Boot Loader entry (%s).\n", DATE_CODE);

    //--- initial SPU
    DrvSPU_Open();
    spuDacPLLEnable();

    /* PLL clock setting */
    initClock();

    spuDacPrechargeEnable();

    sysprintf("System clock = %dHz\nDRAM clock = %dHz\nREG_SDTIME = 0x%08X\n",
               sysGetSystemClock(), sysGetDramClock(), inp32(REG_SDTIME));

     /* Omit some source code in document. */
}
```

3.4 SD Card Initialization

One of the major tasks of the SD loader is to copy the next firmware on the SD card to DRAM for execution. To initialize the SD card driver, both fmiInitDevice() and fmiInitSDDevice () must be called in source code.

```
    // IBR keep the booting SD port number on register SDCR.
    // SDLoader should load image from same SD port.
    ibr_boot_sd_port = (inpw(REG_SDCR) & SDCR_SDPORT) >> 29;
    /* Initial DMAC and FMI interface */
    fmiInitDevice();

    sysprintf("Boot from SD%d ...\n", ibr_boot_sd_port);
    i = fmiInitSDDevice(ibr_boot_sd_port);
    if (i < 0)
        sysprintf("SD init fail <%d>\n", i);
```

3.5 Build SD Loader Project

Normally, the SD loader doesn't need to modify. If the SD loader is modified, clicking the **Rebuild** icon as shown below or press **F7** function key to rebuilt it in Keil MDK.

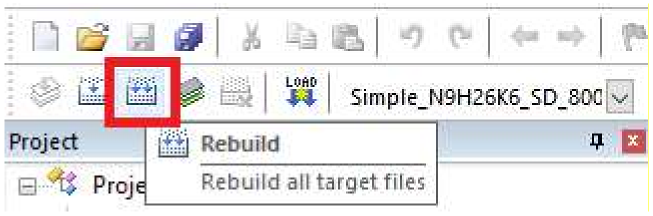


Figure 3-3 Shortcut Icon to Rebuild the SD Loader on Keil MDK

The binary file of SD loader will be copied to the *Loaders\Binary* folder with the file name *N9H26_SDLoader_xxx.bin*. The "*xxx*" is depend on the project target. For the **SDLoader_240MHz** project target, the binay file name is *N9H26_SDLoader_240MHz_Fast.bin*.

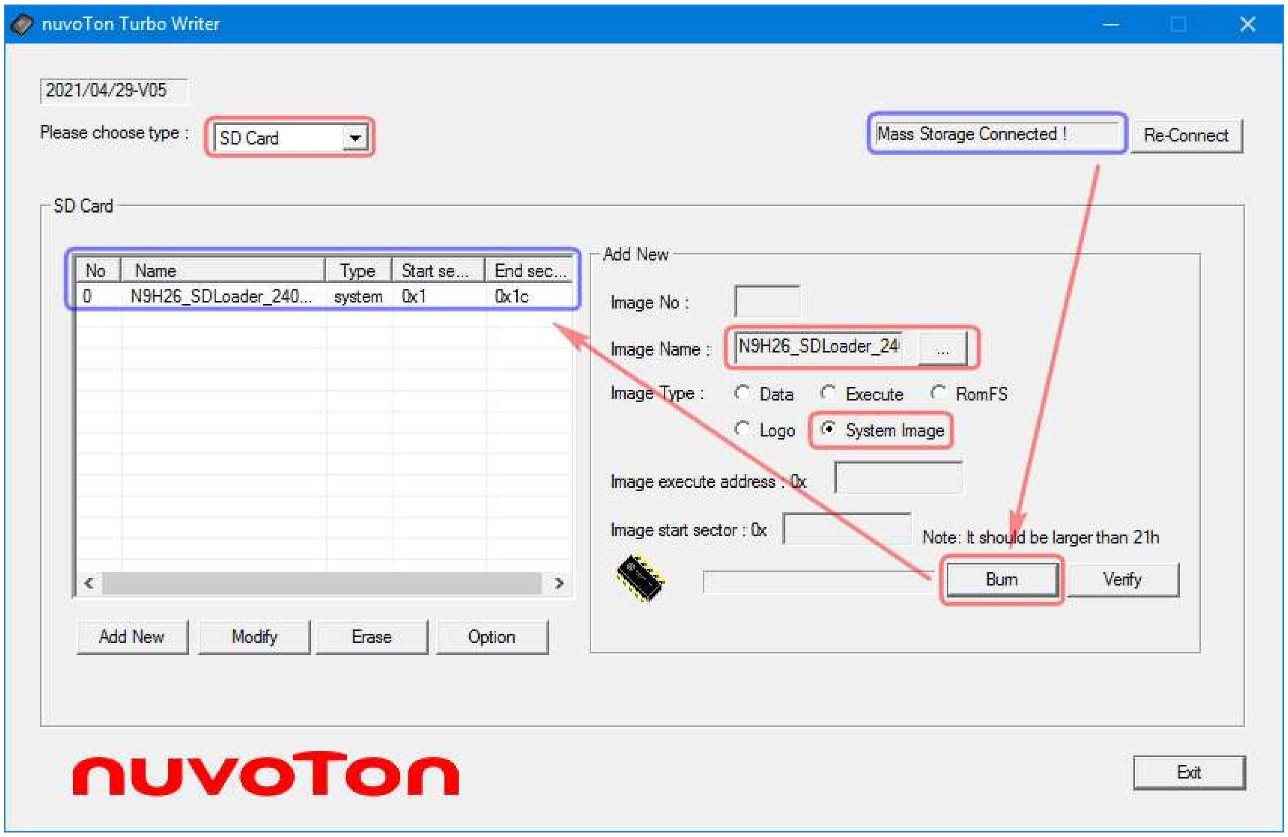
Please note that the binary code size of the SD loader MUST less than or equal to 15360 bytes (512 bytes * 30 sectors)

4 Download and Run

4.1 Download SD Loader Binary to SD Card

The SD loader binary on SD card can be programmed by the tool *TurboWriter* and here are the steps. Further information about *TurboWriter* can be found at BSP *Tools/PC_Tools/TurboWriter Tool User Guide.pdf*.

1. Power off device.
2. Plug in USB cable to PC/NB.
3. Plug in SD card to SD card socket on device.
4. Power on device under Recovery mode.
5. Run *TurboWriter* for N9H26 version on PC/NB.
6. Wait for the *TurboWriter* message to change to "**Mass Storage Connected !**". If not, press the "**Re-Connect**" button to reconnect the device.
7. Select "**SD Card**" on the option "**Please choose type**".
8. Select SD loader binary file on the option "**Image Name**".
9. Select "**System Image**" on the option "**Image Type**".
10. Press "**Burn**" button to burn the SD loader binary into SD card.
11. After burning completed, check the SD loader information in the left table.

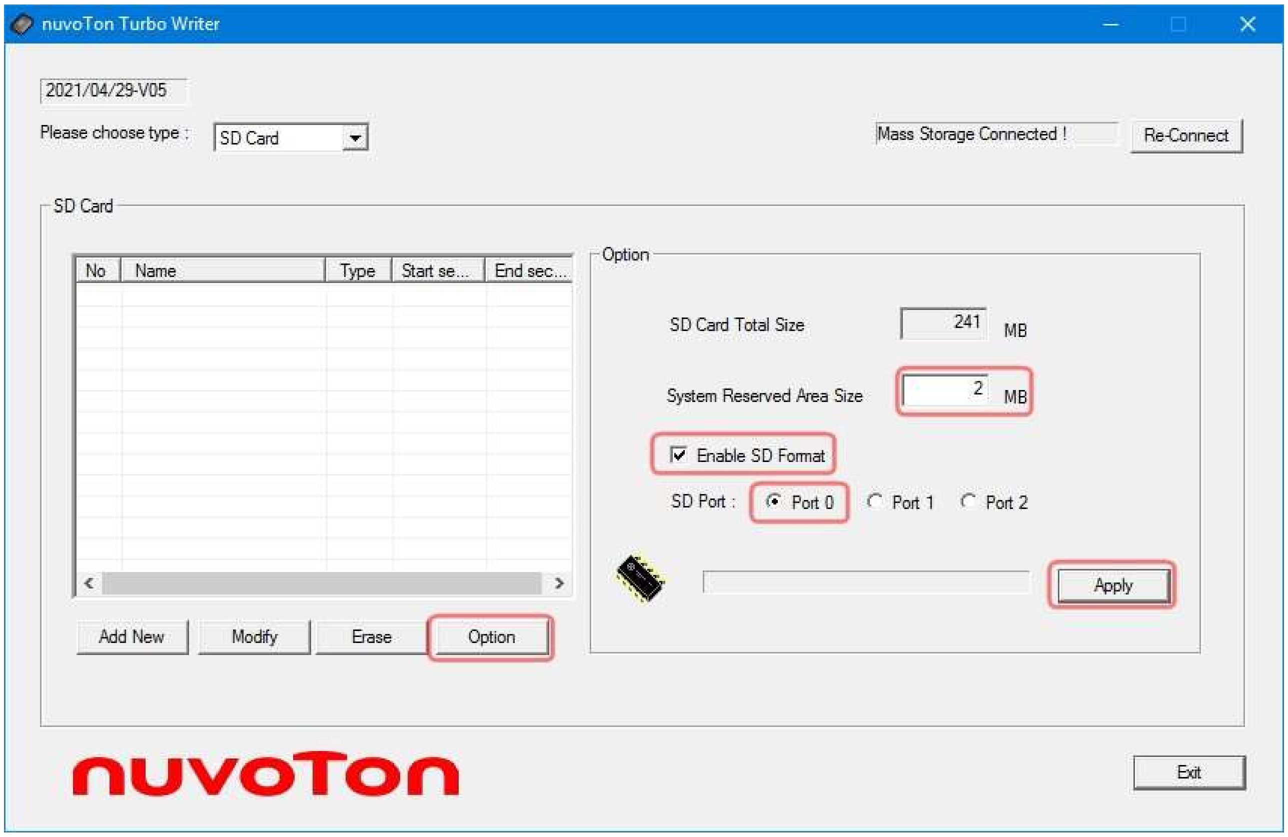


No	Name	Type	Start se...	End sec...
0	N9H26_SDLoader_240...	system	0x1	0x1c

Figure 4-1 Programmed SD Loader by TurboWriter

12. Remove USB device safely.
13. Plug out USB cable from PC/NB.
14. Reset the device under Normal mode.

If the SD card is first time to use on N9H26 system, maybe it need an extra step to reserve some sectors for SD loader used as below.


nuvoTon Turbo Writer

2021/04/29-V05

Please choose type : SD Card

Mass Storage Connected !

SD Card

No	Name	Type	Start se...	End sec...

Option

SD Card Total Size: 241 MB

System Reserved Area Size: 2 MB

[✓] Enable SD Format

SD Port : [✓ Port 0] Port 1 Port 2

Figure 4-2 Reverse Area for SD Loader by TurboWriter

1. Run *TurboWriter* and connect to device.
2. Select “**SD Card**” on the option “**Please choose type**”.
3. Press “**Option**” button to switch to the Option page.
4. Fill in a number in the field “**System Reserved Area Size**”. 1 or 2 MB is enough for normal case.
5. Check the “**Enable SD Format**”
6. Select the “**SD Port**”.
7. Press “**Apply**” button to reserve some sectors for SD loader.
8. Press “**Add New**” button to switch back to the Add New page to burn SD loader binary.

4.2 Run SD Loader

N9H26 has built-in 16K bytes IBR (Internal Booting ROM) where is the boot loader to initialize chip basically when power on, and then try to find out the next stage loader from different type of storage. It could be SD card, NAND Flash, SPI Flash, or USB storage. The booting sequence by the IBR as Figure 4-3.

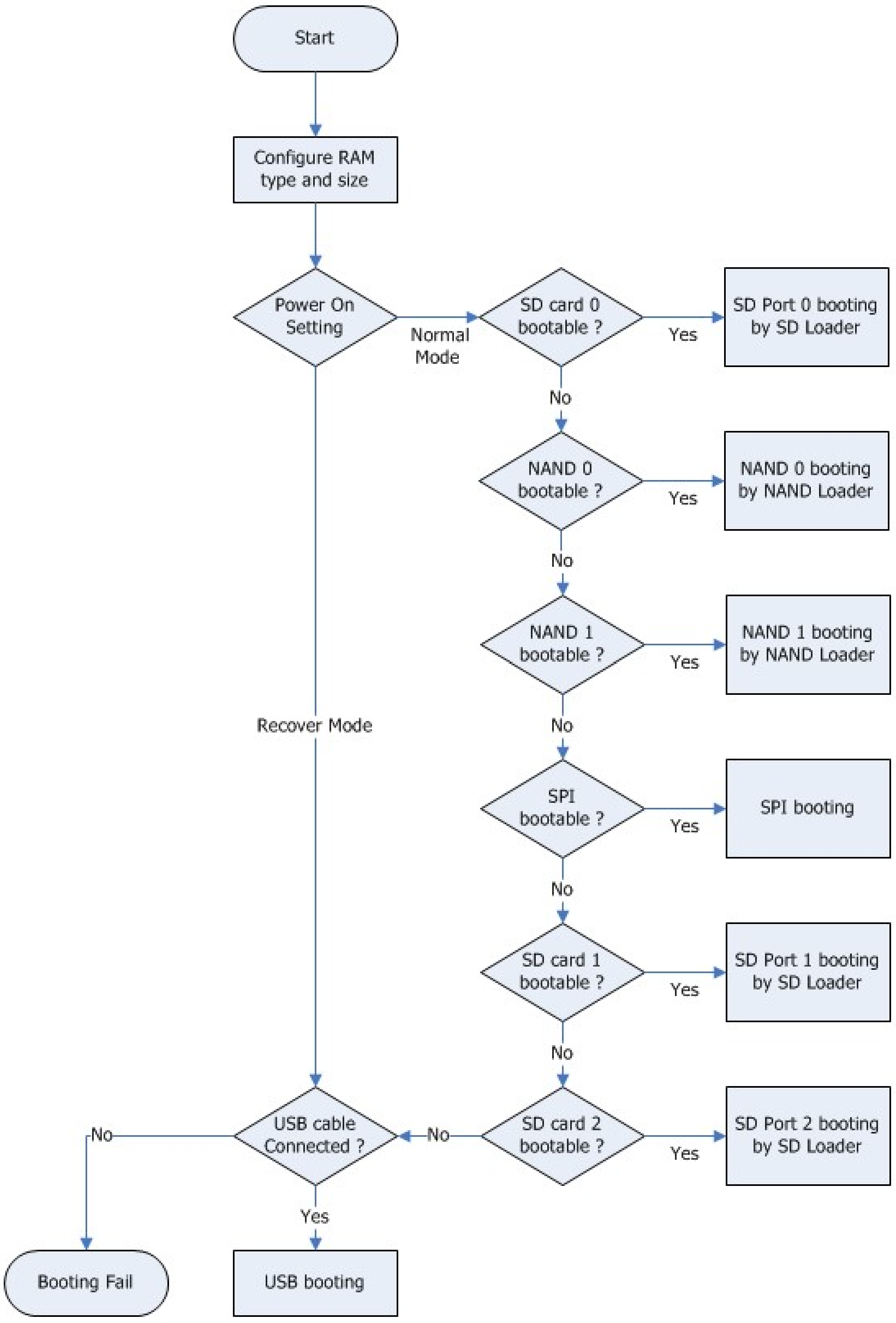
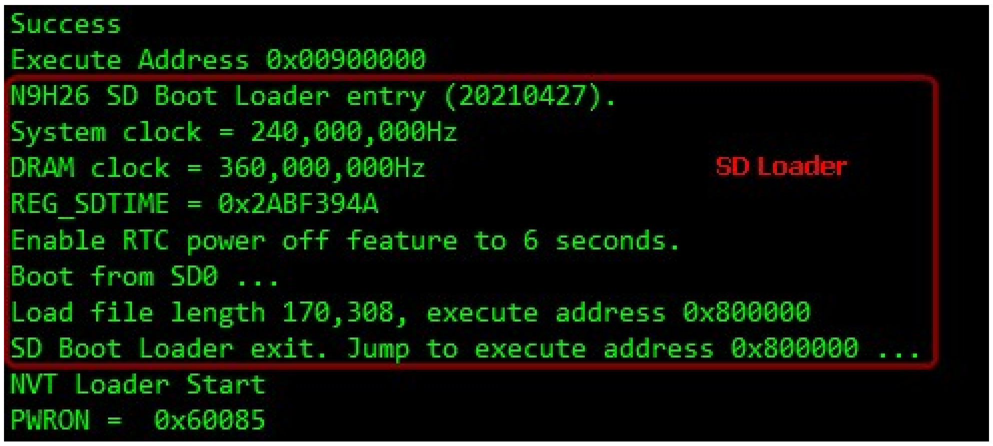


Figure 4-3 IBR Booting Sequence after Power On

The IBR will execute the SD loader if SD loader on SD card 0 is valid.



```
Success
Execute Address 0x00900000
N9H26 SD Boot Loader entry (20210427).
System clock = 240,000,000Hz
DRAM clock = 360,000,000Hz
REG_SDTIME = 0x2ABF394A
Enable RTC power off feature to 6 seconds.
Boot from SD0 ...
Load file length 170,308, execute address 0x800000
SD Boot Loader exit. Jump to execute address 0x800000 ...
NVT Loader Start
PWRON =  0x60085
```

Figure 4-4 The SD Loader Runs on N9H26

5 Supporting Resources

The N9H26 system related issues can be posted in Nuvoton's forum:

- ARM7/9 forum at: http://forum.nuvoton.com/viewforum.php?f=12.

Revision History

Date	Revision	Description
2021.6.15	1.01	1. Modify document structure.
2018.5.4	1.00	1. Initially issued.