N9H20
SD Loader
Reference Guide

V1.0

Publication Release Date: May 2018

Support Chips:
N9H20 Series

Support Platforms:
Non-OS

The information in this document is subject to change without notice.

The Nuvoton Technology Corp. shall not be liable for technical or editorial errors or omissions contained herein; nor for incidental or consequential damages resulting from the furnishing, performance, or use of this material.

Table of Contents

1. General Description

N9H20 Non-OS library consists of a set of libraries. These libraries are built to access those on-chip functions such as VPOST, APU, SIC, USBH, USBD, GPIO, I2C, SPI and UART, as well as File System (NVTFAT), USB Mass Storage devices (UMAS) and Generic NAND library (GNAND). This document describes the basic function of SD Loader. With this introduction, user can quickly understand the SD Loader on N9H20 microprocessor.

2. Overview

2.1. IBR Overview

N9H20 built-in 16K bytes IBR (Internal Booting ROM) where stored the boot loader to initial chip basically when power on, and then try to find out the next stage boot loader from different type of storage. It could be SD card, NAND, SPI Flash, or USB storage. The search sequence by IBR is shown in the Figure 2-1.

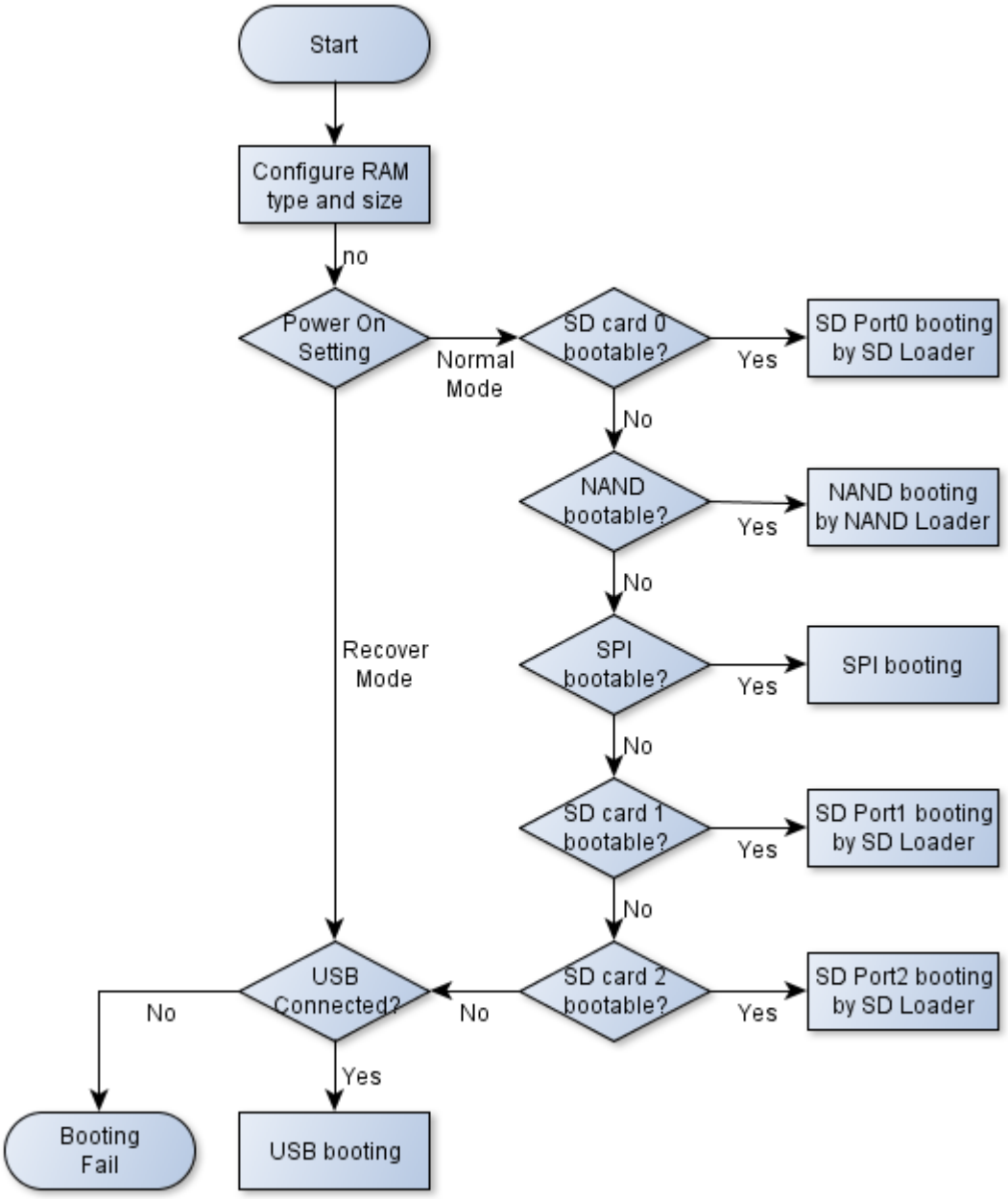

Figure 2-1 IBR Booting Flow

The boot loader in IBR will hand over the chip controlling to SD Loader on SD card 0 if SD card 0 has valid SD Loader. The IBR may also hand over the chip controlling to SD Loader on SD card 1 if all SD card 0, NAND, and SPI have no valid loader.

2.2. SD Loader Overview

SD Loader is a firmware stored at SD card sector 1 ~ 32 as Figure 2-5. It has the following features:

- Initial system clock. The default system clock is 192MHz.
- Check and load images according to the configuration in **Image Information Table**
  - Load Logo image with image type "Logo"
  - Load next firmware with image type "Execute"
- Initial more modules such as SPU, RTC, and so on if necessary.
- Hand over chip controlling to next firmware. Normally, it should be NVT Loader

The **Image Information Table** is a table that records the image type, image execute address, and image stored location in SD card for each image. You can add or modify them by **TurboWriter** utility. Please follow the configuration below to make sure the SD Loader work fine.

For N9H20K5

	SD Loader	**Logo Image**	**Next Firmware**
Image No.	0	1	2
Image Name	*File name for SD Loader that selected by TurboWriter*	*File name for Logo image that selected by TurboWriter*	*File name for next firmware that selected by TurboWriter*
Image Type	System Image	Logo	Execute
Image execute address	***0x900000*** *and cannot be modified*	**0x500000**	*Depend on firmware (**0x800000** for NVTLoader)*
Image start block	*Sector 1 and cannot by modified*	*Behind SD Loader and **MUST > 0x21***	*Behind Logo Image*

For N9H20K3

	SD Loader	**Logo Image**	**Next Firmware**
Image No.	0	1	2
Image Name	*File name for SD Loader that selected by TurboWriter*	*File name for Logo image that selected by TurboWriter*	*File name for next firmware that selected by TurboWriter*
Image Type	System Image	Logo	Execute
Image execute address	***0x700000*** *and cannot be modified*	**0x500000**	*Depend on firmware (**0x600000** for NVTLoader)*
Image start block	*Sector 1 and cannot by modified*	*Behind SD Loader and **MUST > 0x21***	*Behind Logo Image*

For N9H20K1

	SD Loader	**Logo Image**	**Next Firmware**
Image No.	0	*Don't support*	1
Image Name	*File name for SD Loader that selected*	*Don't support*	*File name for next firmware that*

	by TurboWriter		*selected by TurboWriter*
Image Type	System Image	*Don't support*	Execute
Image execute address	***0x180000*** *and cannot be modified*	*Don't support*	*Depend on firmware*
Image start block	*Sector 1 and cannot by modified*	*Don't support*	*Behind SD Loader and* ***MUST > 0x21***

Please operate TurboWriter as below.

First, burn SD Loader with image type "System Image" as Figure 2-2.

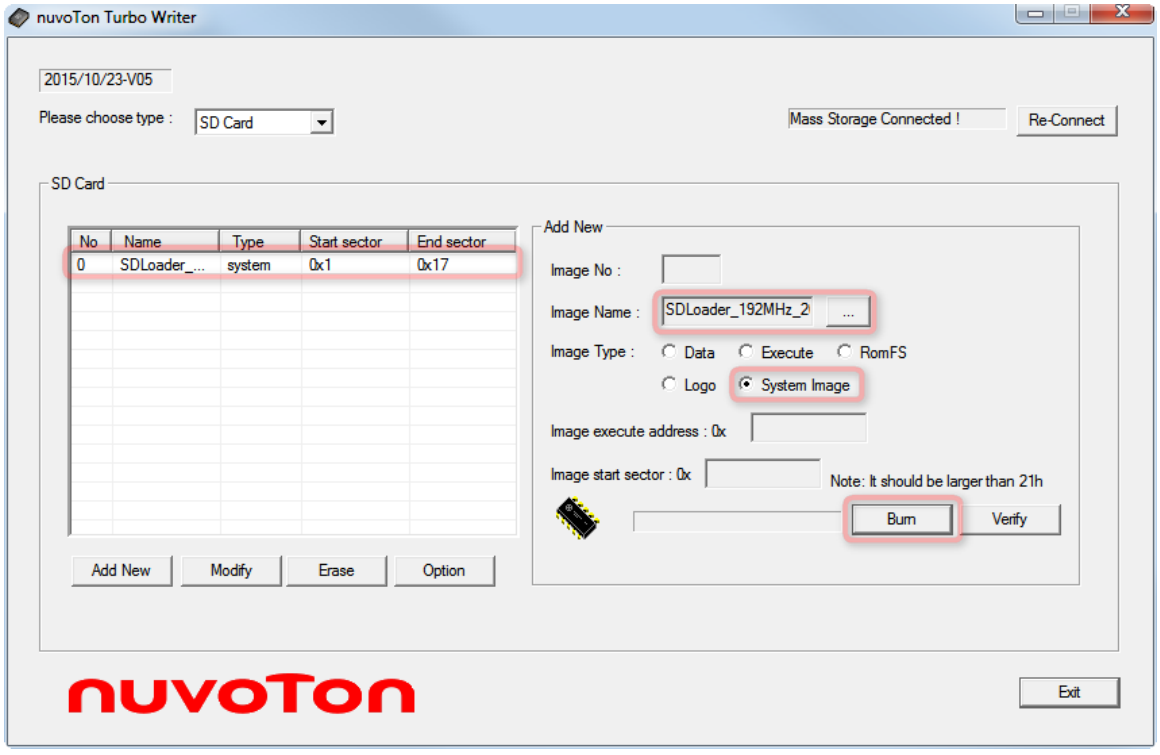

Figure 2-2 Burn SD Loader by TurboWriter

Second, burn Logo image with image type “Logo” as Figure 2-3. Please note that the “Image start sector” must behind the SD Loader and greater than **0x21**. For example, the end sector of SD Loader is sector 0x17, the image start sector of Logo should be 0x22.

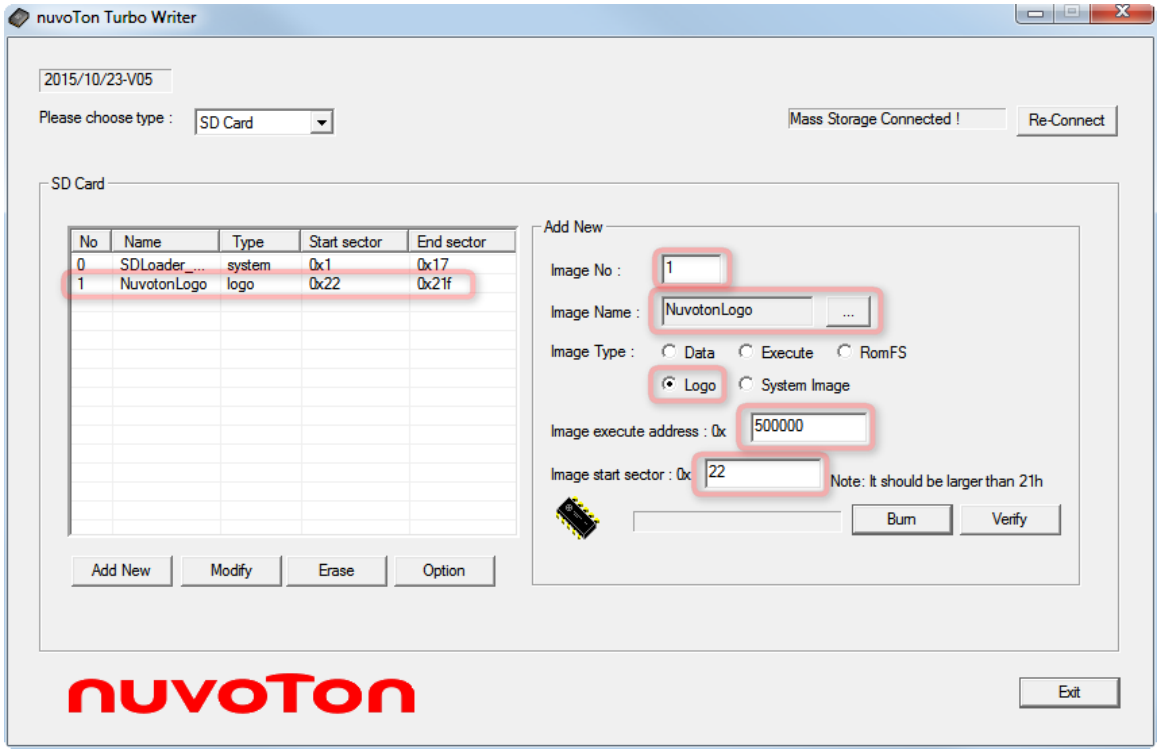

Figure 2-3 Burn Logo image by TurboWriter

Finally, burn next firmware with image type “Execute” as Figure 2-4. Please note that the “Image start sector” must behind the last image and greater than 0x21. For example, the end sector of Logo image is sector 0x21f, the image start sector of next firmware should be 0x220.

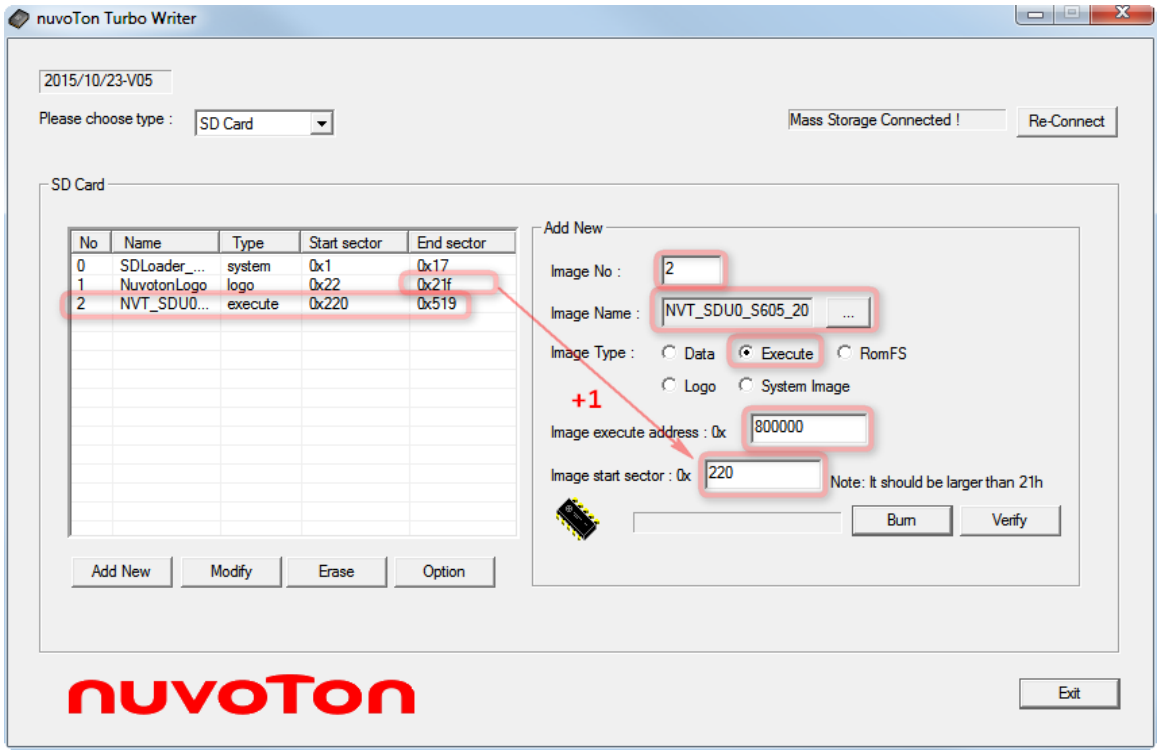

Figure 2-4 Burn next firmware by TurboWriter

After the above operation in TurboWriter, the SD card layout should be look like the Figure 2-5.

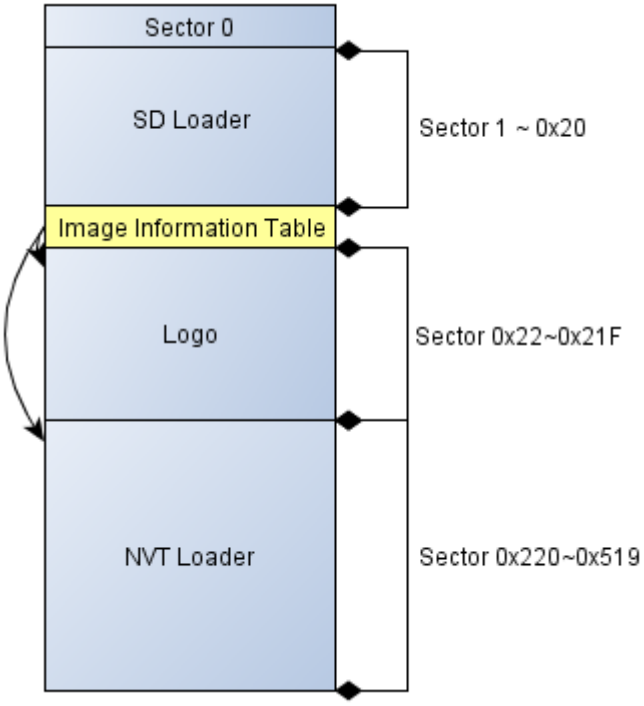

Figure 2-5 SD card layout after burn booting image

Note that the location of Image Information Table always is the sector 33 (0x21).

3. Source Code Review

3.1. Build SD Loader Image

SD Loader project support Keil uVision IDE. Each project file provides several targets for different system clock or configuration. Please select "**N9H20K5_SDLoader_192MHz**" as the standard target for N9H20 demo board.

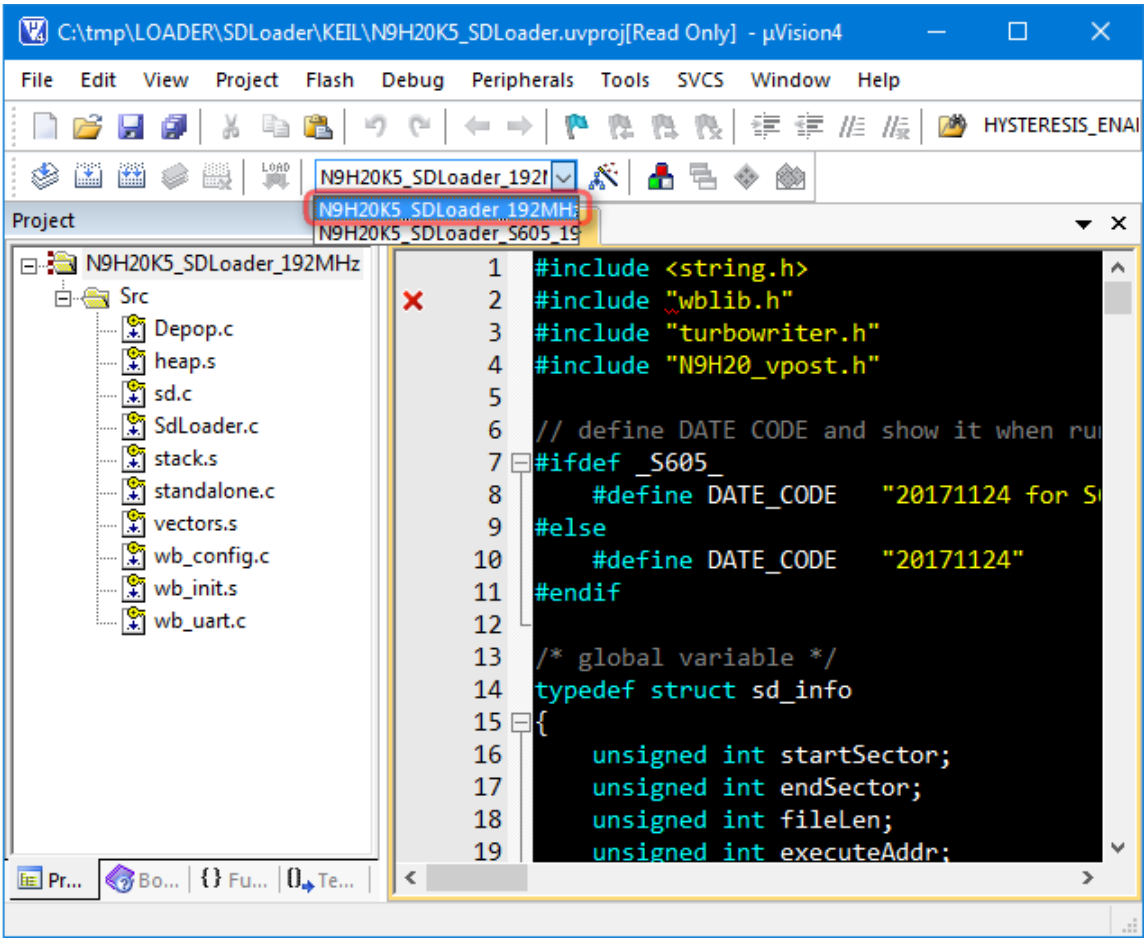

Figure 3-1 SD Loader project in Keil

3.2. Source Code Review

If you want to modify SD Loader by yourself, following description about SD Loader source code could be helpful for you.

Since the Image Information Table always located at the sector 33 (0x21) of SD card, the SD Loader binary file size has upper limitation to **512 * 32 = 16K bytes.** Please don't add too much code to meet the binary file size limitation.

3.2.1. Set System Clock

The first job of SD Loader is to set system clock. It is implemented by function **initClock** (). The standard system clock is 192MHz because the predefined constant "**__UPLL_192__"** is defined in target "**N9H20K5_SDLoader_192MHz**" in the project file. Select different target will build SD Loader image with different system clock.

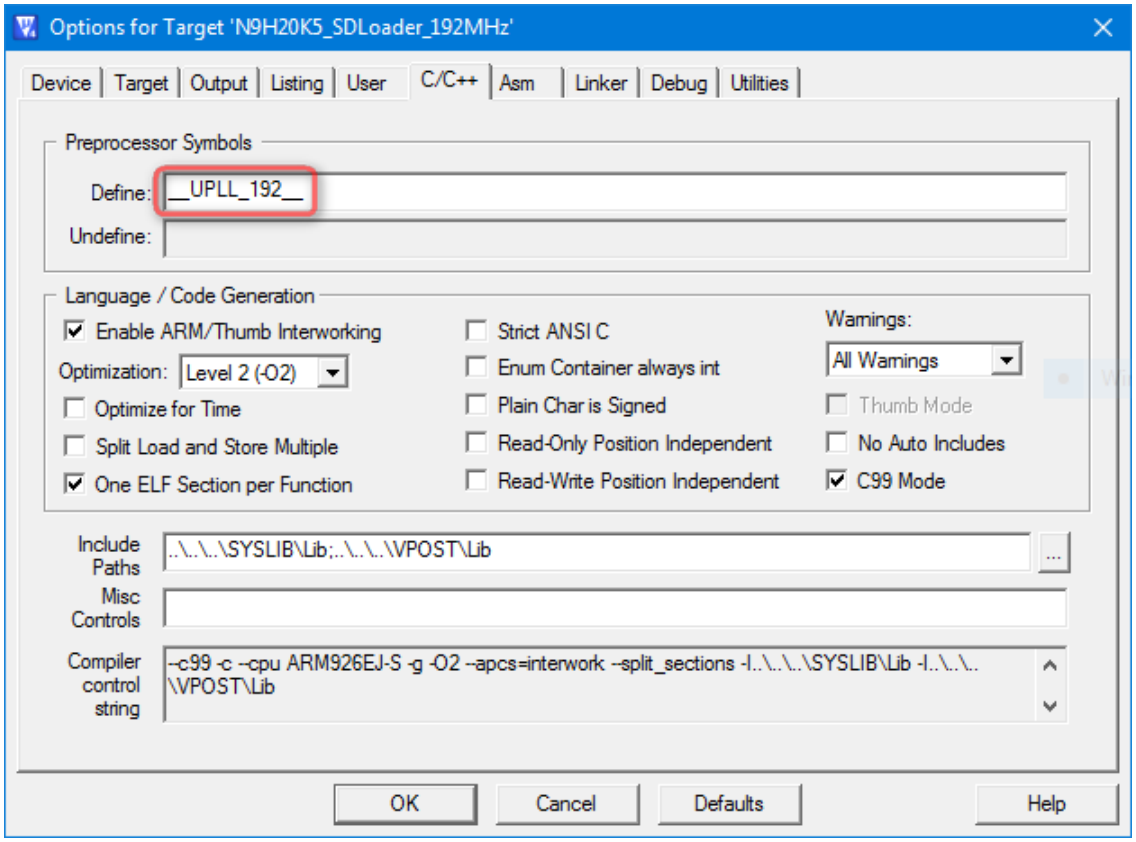

Figure 3-2 Predefined constant for system clock in project file

```
void initClock(void)
{
    UINT32 u32ExtFreq;

    u32ExtFreq = sysGetExternalClock();     /* KHz unit */
    outp32(REG_DQSODS, 0x1010);
    outp32(REG_CKDQSDS, E_CLKSKEW);
    if(u32ExtFreq==12000)
    {
        outp32(REG_SDREF, 0x805A);
    }
    else
    {
        outp32(REG_SDREF, 0x80C0);
    }

#ifdef __UPLL_192__
    if((inp32(REG_CHIPCFG) & SDRAMSEL) == 0x20)  // Power On Setting SDRAM type is DDR2
    {
        outp32(REG_SDMR, 0x432);
        outp32(REG_DQSODS, 0x00001010);
        outp32(REG_MISCPCR,0x00000001);     // Driving strength
        outp32(REG_SDTIME, 0x21667525);     // for Winbond 32MB DDR2 under 96MHz HCLK
    }
    else if((inp32(REG_CHIPCFG) & SDRAMSEL)==0x30) //Power On Setting SDRAM type is DDR
    {
    #ifdef __DDR_75__
        outp32(REG_SDTIME, 0x098E7549); // DDR Speed grade-75
    #endif
    #ifdef __DDR_6__
        outp32(REG_SDTIME, 0x094E7425); // DDR Speed grade-6, for DDR-6 under 96MHz HCLK
    #endif
    #ifdef __DDR_5__
        outp32(REG_SDTIME, 0x094E6425); // DDR Speed grade-5
    #endif
        outp32(REG_SDMR, 0x22);         // Cas Latency = 2
    }

    sysSetSystemClock(eSYS_UPLL,    //E_SYS_SRC_CLK eSrcClk,
                      192000,       //UINT32 u32PllKHz,
                      192000,       //UINT32 u32SysKHz,
                      192000,       //UINT32 u32CpuKHz,
                      192000/2,     //UINT32 u32HclkKHz,
                      192000/4);    //UINT32 u32ApbKHz
#endif  // end of __UPLL_192__
}
```

Different system clock need different setting value about SDTIME. If you don't know how to get correct SDTIME, please don't modify it.

The proposed system clock for N9H20 is **192MHz**. It can be divided to 48MHz for USBD engine and make N9H20 run stably. If you want to run N9H20 at higher system clock, you have to take risks by yourself.

3.2.2. Detect the Booting SD Port

The SD Loader could be loaded by IBR from different SD port such as SD port 0 or SD port 1. Therefore, the SD Loader has to know which booting SD port it be loaded and then be able to load other images from it. The IBR will keep the booting SD port number on register SDCR. The SD Loader can detect the booting SD port number from SDCR.

```
// 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/SD interface */
fmiInitDevice();

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

3.2.3. Got Image Information Table

The location of Image Information Table always is **the sector 33 (0x21)** in the SD card. SD Loader reads and parses it to found out all images that TurboWriter write into.

```
/* read image information */
fmiSD_Read(33, 1, (UINT32)buf);
```

3.2.4. Load Image from SD Card to RAM

After got Image Information Table, SD Loader will found out the Logo image first and then copy it from SD card to RAM. SD Loader doesn't process the Logo image on RAM since it will be used by next firmware. The second parameter of **MoveData** () is **FALSE** that means SD Loader doesn't execute this image after copy it to RAM.

Finally, SD Loader will found out first image with image type "Execute", copy it from SD card to RAM, and then hand over chip controlling to it. The second parameter of **MoveData** () is **TRUE** that means SD Loader will execute this image after copy it to RAM.

```
if (((*(pImageList+0)) == 0x574255aa) && ((*(pImageList+3)) == 0x57425963))
{
    count = *(pImageList+1);

    /* logo */
    pImageList = pImageList+4;
    for (i=0; i<count; i++)
    {
        if (((*(pImageList) >> 16) & 0xffff) == 4)  // logo
        {
            image.startSector = *(pImageList + 1) & 0xffff;
            image.endSector = (*(pImageList + 1) & 0xffff0000) >> 16;
            image.executeAddr = *(pImageList + 2);
            image.fileLen = *(pImageList + 3);
```

```
            MoveData(&image, FALSE);
            break;
        }
        /* pointer to next image */
        pImageList = pImageList+12;
    }

    pImageList = ((unsigned int*)(((unsigned int)dummy_buffer)|0x80000000));
    memset((char *)&image, 0, sizeof(NVT_SD_INFO_T));
    /* execution file */
    pImageList = pImageList+4;
    for (i=0; i<count; i++)
    {
        if (((*(pImageList) >> 16) & 0xffff) == 1)  // execute
        {
            image.startSector = *(pImageList + 1) & 0xffff;
            image.endSector = (*(pImageList + 1) & 0xffff0000) >> 16;
            image.executeAddr = *(pImageList + 2);
            image.fileLen = *(pImageList + 3);
            MoveData(&image, TRUE);
            break;
        }
        /* pointer to next image */
        pImageList = pImageList+12;
    }
    while(1);
}
```

3.2.5. Other Jobs

Section 3.2.1, 3.2.3, 3.2.4, and 3.2.4 are the necessary jobs of SD Loader. You MUST keep them work fine in any case. Except them, you can do something in SD Loader if you wanted such as enable SPU, disable RTC, initial Timer, and so on.

4. Revision History

Version	Date	Description
V1.0	May, 2018	● Created

Important Notice

Nuvoton products are not designed, intended, authorized or warranted for use as components in equipment or systems intended for surgical implantation, atomic energy control instruments, aircraft or spacecraft instruments, transportation instruments, traffic signal instruments, combustion control instruments, or for any other applications intended to support or sustain life. Furthermore, Nuvoton products are not intended for applications whereby failure could result or lead to personal injury, death or severe property or environmental damage.

Nuvoton customers using or selling these products for such applications do so at their own risk and agree to fully indemnify Nuvoton for any damages resulting from their improper use or sales.