N9H26 NVT Loader Reference Guide

Document Information

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

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

www.nuvoton.com

Table of Contents

1 Introduction

The NVT Loader is a firmware stored at the SD Card or NAND Flash for booting purpose. It's loaded by SD Loader or NAND Loader and used to load the next firmware to DRAM to execute.

The NVT Loader supports the following features:

- Initialize more modules such as VPOST, KPI, and so on.
- Export USB Disk to update Kernel or usr data.
- Supports FAT file system for Kernel or usr data.
- Play the booting animation if it existed at SD Card / NAND Flash.
- Load next firmware to DRAM if it existed at SD Card / NAND Flash.
- Execute next firmware. Normally, it should be Kernel.

Nuvoton provides NVT Loader source code within the N9H26 series microprocessor (MPU) BSP. The sequence for loading the kernel from power on was showed as following figure.

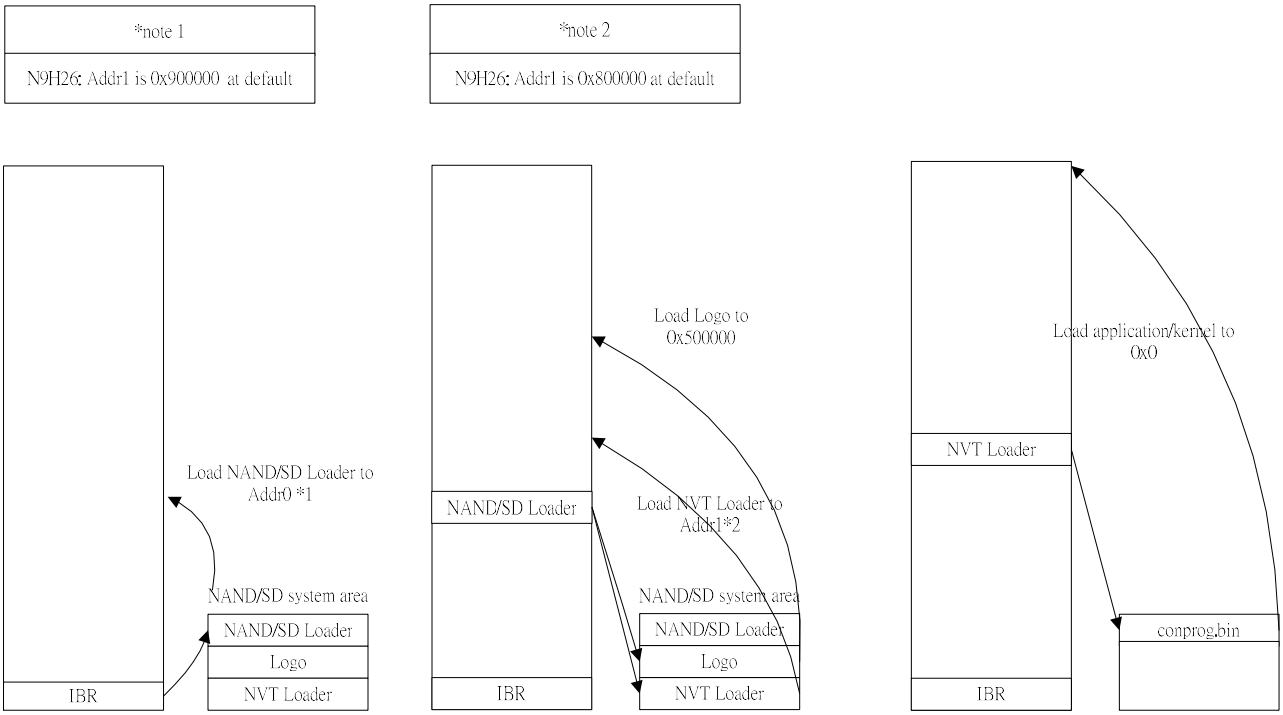

Figure 1 Sequence for loading Kernel

2 NVT Loader BSP Directory Structure

This chapter introduces the NVT Loader related files and directories in the N9H26 BSP.

2.1 Loaders\NVTLoader

GCC	The GCC project files for the NVT Loader.
KEIL	The KEIL project files for the NVT Loader.
NVT_main.c	The main function for the NVT Loader.
scat.scf	The scatter file for the NVT Loader.
NVT_avi.c	Functions for playing booting animation.
NVT_Keypad.c	Functions for Key pad control.
NVT_mass.c	Functions for USB Disk export.
NVT_vpost.c	Functions for Panel initialization.
NVT_boot_from_nand.c	Functions for loading Kernel from NAND Flash.
NVT_boot_from_sd.c	Functions for loading Kernel from SD Card.
Other files	System driver of N9H26.

2.2 Loaders\Binary

NVT_SD_Fast_FW050TFT_800x480_24B.bin ***NVT_NAND_Fast_FW050TFT_800x480_24B.bin***	The binary file of the NVT Loader for different project targets.

3 Execution Flow

There are two mainly branches to load kernel from on board NAND Flash or external SD Card. The following figure shows the control flow of NVT Loader.

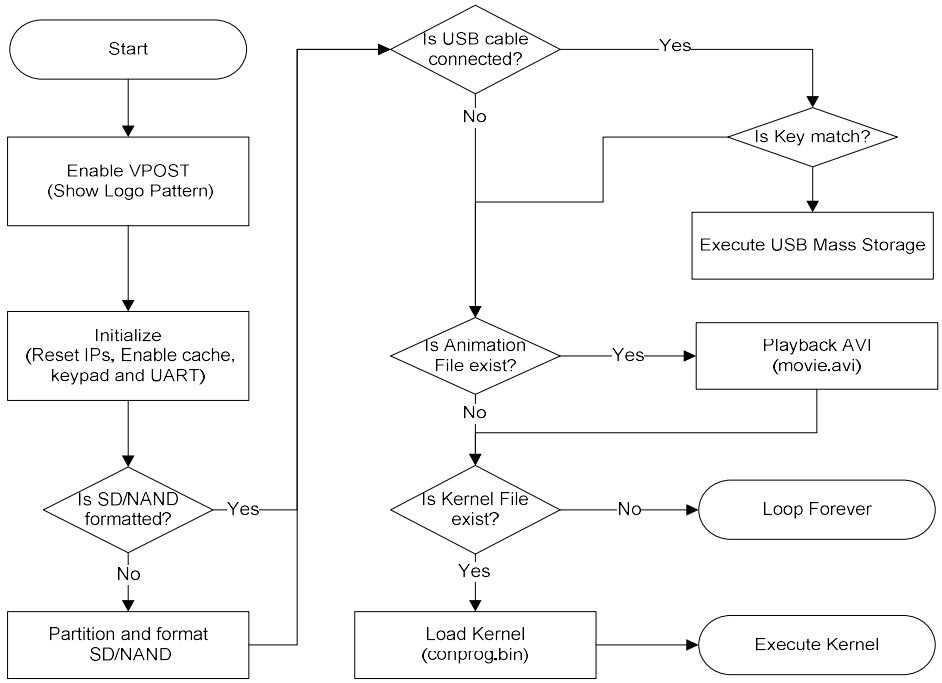

Figure 2 Execution Flow

NVT Loader follows following steps to update files or launch next image.

1. Record the port number of booting device and initialize system.

2. Initialize panel if necessary. The frame buffer address always fixed at address 0x500000. The address follows Linux driver's setting to avoid flash or un-continuous symptom as starting up Linux kernel.

3. Mount SD or NAND device from the port number of booting device. If it is unknown device, partition and format the device.

4. Detect Key pad and USB cable plug in or not. If USB cable plug in and Key pad match, run mass-storage function. Then user can update image or AVI file to the root directory of disk label "*SD1-1*" or "*NAND1-1*" through PC. After update image done, user can plug out the USB cable. NVT Loader will disable mass-storage function and enter step 6.

5. Play AVI file with file name "*movie.avi*" if the file exists.

6. Load next image with file name "*conprog.bin*" if the file exists. After this action done, pass the CPU control to next image.

4 Memory Map

Scatter description loading file describes the memory map in load and execution view. NVT Loader should be loaded to address 0x800000. The following table and figure show the execution view of Loader and NVT Loader.

Chip	SD/NAND Loader Address	NVT Loader Address	Logo Address
N9H26	0x900000	0x800000	0x500000

Table 1 Execution address for chip

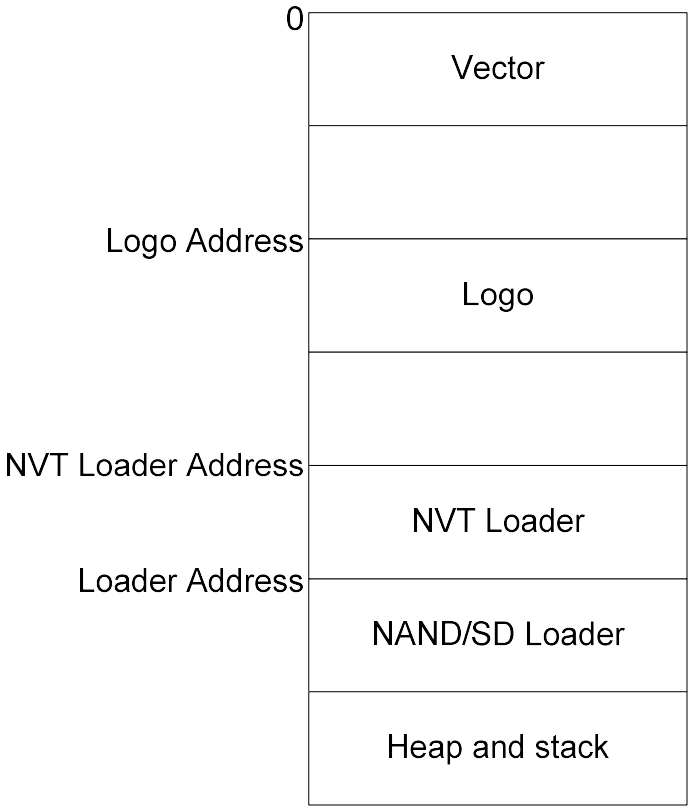

Figure 3 NVT-Loader Execution View

5 NVT Loader Source Code

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

5.1 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 4 Keil MDK License Chart

5.2 Project Structure

The NVT Loader project includes one main function file and some driver files of N9H26.

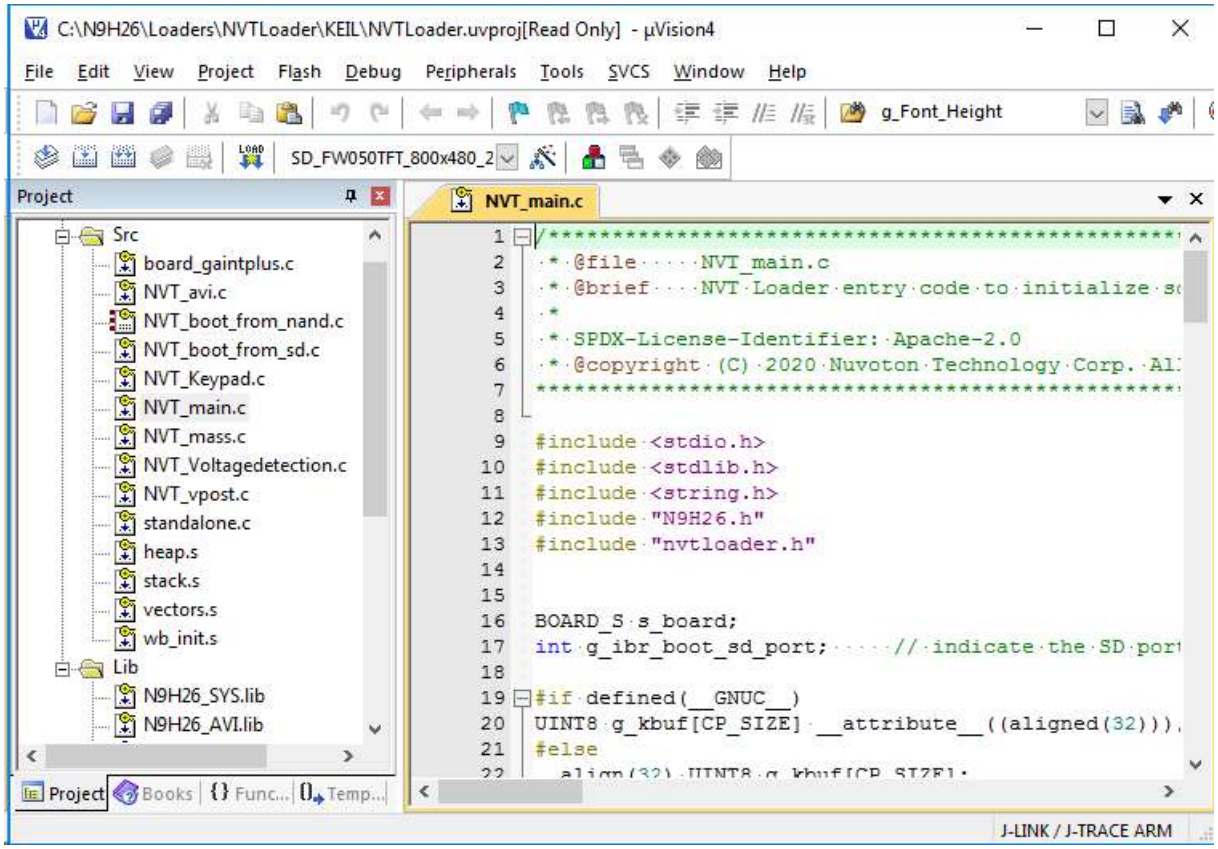

Figure 5 NVT Loader Project Tree on Keil MDK

The NVT Loader project includes some targets that can be used in different situations.

- **SD_GIANTPLUS_QVGA**: Initialize VPOST for QVGA and load firmware from SD.
- **NAND_GAINTPLUS_QVGA**: Initialize VPOST for QVGA and load firmware from NAND.
- **SD_FW050TFT_800x480_24B**: Initialize VPOST for WVGA and load firmware from SD.
- **NAND_FW050TFT_800x480_24B**: Initialize VPOST for WVGA and load firmware from NAND.

5.3 Initialization

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

```
void init(BOARD_S* ps_board)
{
    WB_UART_T   uart;
    UINT32      u32ExtFreq;
    UINT32 u32Cke = inp32(REG_AHBCLK);
    u32ExtFreq = sysGetExternalClock();
    /* enable UART */
    sysUartPort(1);
    /* Omit some source code in document. */
    sysInitializeUART(&uart);

    /* Omit some source code in document. */

    /* RTC has been open in NAND/SD Loader */
    /* RTC_Ioctl(0,RTC_IOC_SET_POWER_KEY_DELAY, 7, 0); */
    if(ps_board ->spkpower_init != NULL)
        ps_board ->spkpower_init();
    if(ps_board ->earphone_init != NULL)
        ps_board ->earphone_init();

    /* Reset SIC engine to fix USB update kernel and movie file */
    outp32(REG_AHBCLK, u32Cke  | (SIC_CKE | NAND_CKE | SD_CKE));
    outp32(REG_AHBIPRST, inp32(REG_AHBIPRST )|SIC_RST );
    outp32(REG_AHBIPRST, 0);
    outp32(REG_APBCLK, inp32(REG_APBCLK) | RTC_CKE);
    outp32(REG_APBIPRST, TMR0RST | TMR1RST);
    outp32(REG_APBIPRST, 0);

    sysprintf("PWRON =  0x%x\n", inp32(PWRON));
    outp32(REG_AHBCLK,u32Cke);

    /* init timer */
    sysSetTimerReferenceClock (TIMER0,
                               u32ExtFreq);    /* Hz unit */
    sysStartTimer(TIMER0,
                  100,
```

```
                    PERIODIC_MODE);

    sysSetLocalInterrupt(ENABLE_IRQ);

    /* Omit some source code in document. */

    initVPostShowLogo(ps_board);

    DrvADC_Open();
}

INT32 main(void)
{
    // IBR and SDLoader keep the booting SD port number on register SDCR.
    // NVTLoader should load image from same SD port.
    outpw(REG_AHBCLK, inpw(REG_AHBCLK) | SD_CKE);   // SDLoader disable SIC/SD clock
before jump to NVTLoader.
    outpw(REG_AHBCLK, inpw(REG_AHBCLK) | SIC_CKE);  // Now, enable clock to read SD
register.
    g_ibr_boot_sd_port = (inpw(REG_SDCR) & SDCR_SDPORT) >> 29;
    outpw(REG_AHBCLK, inpw(REG_AHBCLK) & ~SD_CKE);
    outpw(REG_AHBCLK, inpw(REG_AHBCLK) & ~SIC_CKE);
    sysprintf("NVT Loader Start\n");

    sysDisableCache();
    sysFlushCache(I_D_CACHE);
    sysEnableCache(CACHE_WRITE_BACK);

    register_board(&s_board);
    init(&s_board);
    initVPostShowLogo(&s_board);

    sysprintf("NVT Loader: g_ibr_boot_sd_port = %d\n", g_ibr_boot_sd_port);

    fsInitFileSystem();
#ifdef __ENABLE_SD_CARD_0__
    NVT_LoadKernelFromSD(&s_board,                  /* Board */
                         g_ibr_boot_sd_port,        /* Boot port */
                         g_kbuf);                   /* Temp buffer */
#endif
#if defined(__ENABLE_NAND_0__) || defined(__ENABLE_NAND_1__) || defined(__ENABLE_NAND_2__)
```

```
    NVT_LoadKernelFromNAND(&s_board,                /* Board */
                           g_ibr_boot_sd_port,      /* Boot port */
                           g_kbuf);                 /* Temp buffer */
#endif
    return Successful;
}
```

5.4 Check Disk File system

Before loading kernel, NVT Loader checks disk information.

5.4.1 SD

If the disk is not formatted, NVT Loader formats the disk to "*SD1-1*" & "*SD1-2*" for SD Card.

```
UINT32 NVT_LoadKernelFromSD(BOARD_S* ps_board,
                            UINT32 g_ibr_boot_sd_port,
                            unsigned char* pkBuf)
{
     /* Omit some source code in document. */

    DBG_PRINTF("Loader will load conprog.bin from SD card.\n");

    /* Omit some source code in document. */

    fsAssignDriveNumber('X', DISK_TYPE_SD_MMC, 0, 1);
    fsAssignDriveNumber('Y', DISK_TYPE_SD_MMC, 0, 2);
    /*---------------------------------------------------------------------*/
    /*  Init SD card                                                       */
    /*---------------------------------------------------------------------*/
    u32ExtFreq = sysGetExternalClock();
    u32PllOutHz = sysGetPLLOutputHz(eSYS_UPLL, u32ExtFreq);
    sicIoctl(SIC_SET_CLOCK, u32PllOutHz/1000, 0, 0);
    sicOpen();

    if (g_ibr_boot_sd_port == 0)
    {
        sysprintf("Load code from SD0\n");
        i32BootSD0TotalSector = sicSdOpen0();   /* Total sector or error code */
        if(i32BootSD0TotalSector < 0)
            sicSdClose0();
        sysprintf("total SD0 sectors number (%x)\n", i32BootSD0TotalSector);
```

```
    }
    else if (g_ibr_boot_sd_port == 1)
    {
        sysprintf("Load code from SD1\n");
        i32BootSD1TotalSector = sicSdOpen1();   /* Total sector or error code */
        if(i32BootSD1TotalSector < 0)
            sicSdClose1();
        sysprintf("total SD1 sectors (%x)\n", i32BootSD1TotalSector);
    }
    else if (g_ibr_boot_sd_port == 2)
    {
        sysprintf("Load code from SD2\n");
        i32BootSD2TotalSector = sicSdOpen2();   /* Total sector or error code */
        if(i32BootSD2TotalSector < 0)
            sicSdClose2();
        sysprintf("total SD2 sectors (%x)\n", i32BootSD2TotalSector);
    }

    /* Get SD disk information*/
    pDiskList = fsGetFullDiskInfomation();
    sysprintf("Total Disk Size = %dMB\n", pDiskList->uDiskSize/1024);
    /* Format disk if necessery */
    if ((fsDiskFreeSpace('X', &block_size, &free_size, &disk_size) < 0) ||
            (fsDiskFreeSpace('Y', &block_size, &free_size, &disk_size) < 0))
    {
        UINT32 u32Reserved;
        u32Reserved = ParsingReservedArea(g_ibr_boot_sd_port);
        sysprintf("unknow disk type, format device ...Reserved Area= %dKB \n",
u32Reserved/2);
        fsSetReservedArea(u32Reserved);
#if 1
        if (g_ibr_boot_sd_port == 0)
            u32TotalSize = (i32BootSD0TotalSector-u32Reserved)*512;
        else if (g_ibr_boot_sd_port == 1)
            u32TotalSize = (i32BootSD1TotalSector-u32Reserved)*512;
        else if (g_ibr_boot_sd_port == 2)
            u32TotalSize = (i32BootSD2TotalSector-u32Reserved)*512;
#endif
        if (fsTwoPartAndFormatAll((PDISK_T *)pDiskList->ptSelf, SD1_1_SIZE*1024,
(u32TotalSize- SD1_1_SIZE*1024)) < 0)
        {
```

```
            sysprintf("Format failed\n");
            goto sd_halt;
        }
        fsSetVolumeLabel('X', "SD1-1\n", strlen("SD1-1"));
        fsSetVolumeLabel('Y', "SD1-2\n", strlen("SD1-2"));

    }

    /* Read volume config file */
    VolumeConfigFile();
    /* Omit some source code in document. */
}
```

5.4.2 NAND

If the disk is not formatted, NVT Loader formats the disk to "*NAND1-1*" & "*NAND1-2*" for NAND Flash.

```
UINT32 NVT_LoadKernelFromNAND(BOARD_S* ps_board,
                              UINT32 g_ibr_boot_sd_port,
                              unsigned char* pkBuf)
{
    /* Omit some source code in document. */

    DBG_PRINTF("Loader will load conprog.bin from NAND device.\n");

   /* Omit some source code in document. */

    fsAssignDriveNumber('X', DISK_TYPE_SD_MMC, 0, 1);
    fsAssignDriveNumber('Y', DISK_TYPE_SD_MMC, 0, 2);
    /*-----------------------------------------------------------------------*/
    /*  Init NAND device                                                     */
    /*-----------------------------------------------------------------------*/
    u32ExtFreq = sysGetExternalClock();
    u32PllOutHz = sysGetPLLOutputHz(eSYS_UPLL, u32ExtFreq);
    sicIoctl(SIC_SET_CLOCK, u32PllOutHz/1000, 0, 0);
    sicOpen();

    /* Initialize GNAND */
    if(GNAND_InitNAND(&_nandDiskDriver0, &g_sNDisk0, TRUE) < 0)
    {
```

```
        sysprintf("GNAND_InitNAND error\n");
        goto nandboot_halt;
    }

    if(GNAND_MountNandDisk(&g_sNDisk0) < 0)
    {
        sysprintf("GNAND_MountNandDisk error\n");
        goto nandboot_halt;
    }

    /* Get NAND disk information*/
    u32TotalSize = (UINT32)((UINT64)g_sNDisk0.nZone*
g_sNDisk0.nLBPerZone*g_sNDisk0.nPagePerBlock*g_sNDisk0.nPageSize/1024);
    sysprintf("Total Disk Size %u KB\n", u32TotalSize);
    /* Format NAND if necessery */
    if ((fsDiskFreeSpace('C', &block_size, &free_size, &disk_size) < 0) ||
            (fsDiskFreeSpace('D', &block_size, &free_size, &disk_size) < 0))
    {
        sysprintf("unknow disk type, format device .....\n");
        if (fsTwoPartAndFormatAll((PDISK_T *)g_sNDisk0.pDisk, NAND1_1_SIZE*1024,
(u32TotalSize- NAND1_1_SIZE*1024)) < 0)
        {
            sysprintf("Format failed\n");
            goto nandboot_halt;
        }
        fsSetVolumeLabel('C', "NAND1-1\n", strlen("NAND1-1"));
        fsSetVolumeLabel('D', "NAND1-2\n", strlen("NAND1-2"));
    }

    /* Read volume config file */
    VolumeConfigFile();
    /* Omit some source code in document. */
}
```

5.5 USB Disk Export

KPI is initialized for USB Disk Export condition. NVT Loader check USB cable connect status and KPI input. If meets the following conditions, NVT Loader will export USB Disk.

- USB Cable is connected
- KPI input match **MASS_STORAGE** defined in *nvtloader.h*
  - #define **MASS_STORAGE** (**UP_KEY**+**DOWN_KEY**)

```
/* NVT_LoadKernelFromSD or NVT_LoadKernelFromNAND */
{
    /* Omit some source code in document. */

    kpi_init();
    kpi_open(3); // use nIRQ0 as external interrupt source

    /* Omit some source code in document. */

    /* In here for USB VBus stable. Othwise, USB library can not judge VBus correct */
    udcOpen();

    /* Omit some source code in document. */

    /* Detect USB */
    //u32KpiReport = kpi_read(KEY_ADC_CHANNEL) & MASS_STORAGE;
    u32KpiReport = kpi_read(KPI_NONBLOCK) & MASS_STORAGE;
    sysprintf("KPI  Key Code = 0x%x\n", u32KpiReport);

    if(inp32(0xFF001804) == 0x6D617373)  //AutoWriter
    {
        outp32(0xFF001804, 0);
        u32KpiReport = MASS_STORAGE;
    }
    if(u32KpiReport==(MASS_STORAGE))     //Demo board = "Up"+"Down" Key
    {
        sysprintf("Enter USB\n");
        if(udcIsAttached())
        {
            //for mass's issue. sicSdClose();
            sysprintf("Detect USB plug in\n");
            mass(NULL, NULL, NULL, i32BootSD0TotalSector, i32BootSD1TotalSector,
i32BootSD2TotalSector, 0);                  /* ptNDisk is useless for SD mass-storage*/
```

```
            sysprintf("USB plug out\n");

            outp32(REG_MISCR, inp32(REG_MISCR) | CPURST);

        }
    }
    outp32(PHY_CTL, inp32(PHY_CTL) & (~Phy_suspend));

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

5.6 Play Animation

NVT Loader will play animation file if it exists. The code for playing animation is the same in NVT_LoadKernelFromSD and NVT_LoadKernelFromNAND (The difference is the file path). The file path is defined in *nvtloader.h.*

- SD
  - #define MOVIE_PATH_SD "*x:\\movie.avi*"
- NAND
  - #define MOVIE_PATH "*c:\\movie.avi*"

The following is the source code for SD case.

```
UINT32 NVT_LoadKernelFromSD(BOARD_S* ps_board,
                            UINT32 g_ibr_boot_sd_port,
                            unsigned char* pkBuf)
{
    /* Omit some source code in document. */

#ifdef __AVI_PLAYBACK__
    fsAsciiToUnicode(MOVIE_PATH_SD, path, TRUE);
    g_mfd = fsOpenFile(path, 0, O_RDONLY);
    if(g_mfd > 0)
    {
        found_avi = 1;
        fsCloseFile(g_mfd);
        sysprintf("animation file found\n");
    }
#endif
    fsAsciiToUnicode(KERNEL_PATH_SD, path, TRUE);
    g_kfd = fsOpenFile(path, 0, O_RDONLY);
```

```
    if(g_kfd > 0)
    {
        found_kernel = 1;
        sysprintf("kernel found\n");
    }
    /* Omit some source code in document. */

    /* Initial SPU in advance for linux set volume issue */
#define OPT_DEPOP_20140311
#ifdef OPT_DEPOP_20140311
    spuOpen(eDRVSPU_FREQ_8000);
    if(found_avi)
    {
#ifdef __AVI_PLAYBACK__
        char ucString[64]= MOVIE_PATH_SD;

//      spuDacPrechargeEnable();
        s_delay_10ms(70);   // delay 700 ms
        spuADCVmidEnable();
        s_delay_10ms(100);   // delay 1000 ms
        spuDacEnable(u16Volume);

        playAnimation(ps_board, g_kfd, ucString);
#endif
    }
    else
    {
       /* Omit some source code in document. */
    }
#else
    /* Omit some source code in document. */
#endif
    /* Omit some source code in document. */
}
```

5.7 Load Kernel

NVT Loader will load Kernel if it exists. In order to reduce booting time, it load Kernel during time playing animation if animation file exists. The code for load Kernel is the same in NVT_LoadKernelFromSD and NVT_LoadKernelFromNAND (The difference is the file path). The file path is defined in *nvtloader.h.*

- SD
  - #define KERNEL_PATH_SD "*x:\\conprog.bin*"
- NAND
  - #define KERNEL_PATH "*c:\\conprog.bin*"

The following is the source code for SD case.

```
void playAnimation(BOARD_S* ps_board, int kfd, char* pcString)
{
    /* Omit some source code in document. */

    if(kfd > 0 && _complete == 0)
    {
        loadKernelCont(kfd, _offset, g_kbuf);
    }

    if(kfd > 0)
    {
        fsCloseFile(_fd);
    }
    lcmFill2Dark((unsigned char *)FB_ADDR);
    return;

}

UINT32 NVT_LoadKernelFromSD(BOARD_S* ps_board,
                            UINT32 g_ibr_boot_sd_port,
                            unsigned char* pkBuf)
{

    /* Omit some source code in document. */

    fsAsciiToUnicode(KERNEL_PATH_SD, path, TRUE);
    g_kfd = fsOpenFile(path, 0, O_RDONLY);
    if(g_kfd > 0)
    {
```

```
        found_kernel = 1;
        sysprintf("kernel found\n");
    }
    /* Omit some source code in document. */
#define OPT_DEPOP_20140311
#ifdef OPT_DEPOP_20140311
    spuOpen(eDRVSPU_FREQ_8000);
    if(found_avi)
    {
       /* Omit some source code in document. */
    }
    else
    {
       /* Omit some source code in document. */
       if(found_kernel)
            loadKernelCont(g_kfd, 0, pkBuf);
    }
#else
    /* Omit some source code in document. */
#endif
    if(g_kfd > 0)
    {
        fsCloseFile(g_kfd); //Close kernel file
        if (g_ibr_boot_sd_port == 0)
            sicSdClose0();
        else if (g_ibr_boot_sd_port == 1)
            sicSdClose1();
        else if (g_ibr_boot_sd_port == 2)
            sicSdClose2();
        sicClose();
        sysSetGlobalInterrupt(DISABLE_ALL_INTERRUPTS);
        sysSetLocalInterrupt(DISABLE_FIQ_IRQ);

        /* Omit some source code in document. */

        sysprintf("Jump to kernel\n");
#ifdef OPT_DEPOP_20140311
        if(!found_avi)
        {
            sysprintf("Jump to kernel aaaaaa\n");
```

```
            spuADCVmidEnable();
        }
#endif
        /* Omit some source code in document. */

        _jump = (void(*)(void))(0x0); // Jump to 0x0 and execute kernel
        _jump();

        while(1);
//      return(0); // avoid compilation warning
    }
    else
    {
        sysprintf("Cannot find conprog.bin in SD card.(err=0x%x)\n",g_kfd);
        goto sd_halt;
    }
//  return Successful;
sd_halt:
    outp32(REG_AHBCLK, inp32(REG_AHBCLK) &
~(SPU_CKE|SD_CKE|NAND_CKE|USBD_CKE|I2S_CKE|VIN0_CKE|SEN0_CKE));
    outp32(REG_APBCLK, inp32(REG_APBCLK) &
~(KPI_CKE|WDCLK_CKE|TOUCH_CKE|TMR1_CKE|RTC_CKE|I2C_CKE|ADC_CKE));
    outp32(REG_AHBCLK2, inp32(REG_AHBCLK2) & ~(VIN1_CKE|SEN1_CKE));
    sysprintf("systen exit\n");
    while(1); // never return
}
```

5.8 Build NVT Loader Project

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

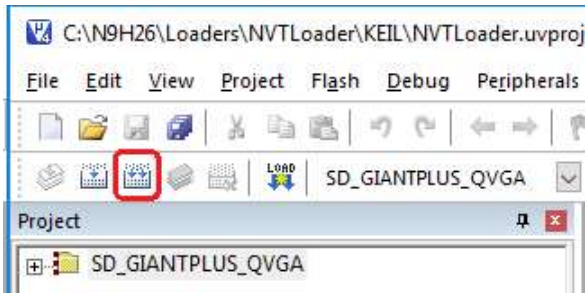

Figure 6 Shortcut Icon to Rebuild the NVT Loader on Keil MDK

The binary file of NVT Loader will be copied to the *Loaders\Binary* folder with the file name *NVT_ xxx.bin*. The "*xxx*" is depend on the project target. For the **SD_FW050TFT_800x480_24B** project target, the binay file name is *NVT_SDU0_Fast_FW050TFT_800x480_24B.bin*.

5.9 Download NVT Loader Binary to SD Card / NAND Flash

The NVT Loader binary on SD Card / NAND Flash can be programmed by the tool *TurboWriter* and here are the steps. Further information about *TurboWriter* for NAND can be found at BSP *Tools/PC_Tools/TurboWriter Tool User Guide.pdf*. The following lines describe how to use SD mode to burn images for NVT Loader.

- SD Loader
  1. Choose the type "**SD**"
  2. Press the button "**Add New**"
  3. Browse the file "*N9H26_SDLoader_240MHz_Fast.bin*"
  4. Set Image type "**System Image**"
  5. Press the button "**Burn**"

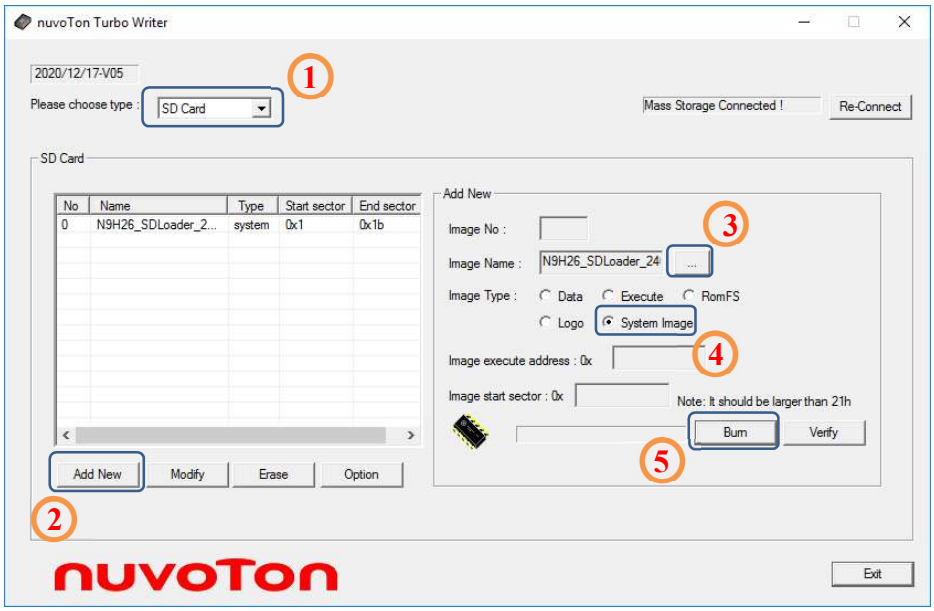

Figure 7 System image – SD Loader

- Logo
  1. Image number "1"
  2. Browse the file "*Logo.bin*"
  3. Set Image type "**Logo**"
  4. Set the image execute address: 0x500000
  5. Set the start block number: 0x22

6. Press the button “**Burn**”

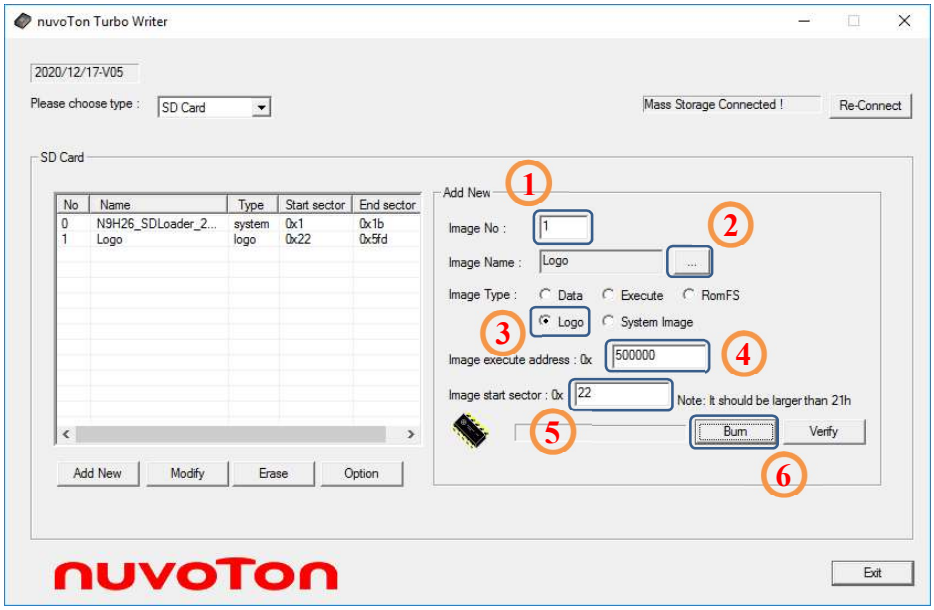

Figure 8 Logo image

- NVT Loader
  1. Image number “2”
  2. Browse the file “*NVT_SDU0_Fast_FW050TFT_800x480_24B.bin*”
  3. Set Image type “**Execute**”
  4. Set the executed address: 0x800000
  5. Set the start block number: 0x5FE.
  6. Press the button “**Burn**”

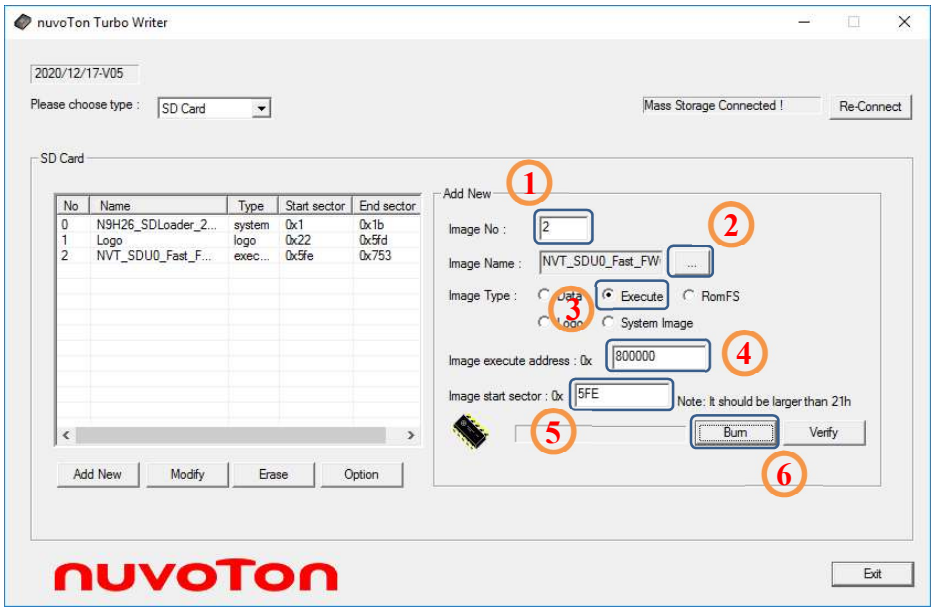

Figure 9 Execute image – NVT Loader

6 Run NVT Loader

Set N9H26 to normal mode to run NVT Loader.

6.1 Format Disk

When the disk is not formatted, NVT Loader formats the disk to “SD1-1” & “SD1-2” for SD Card and it will show the message “Cannot fin *conprog.bin*” if the file doesn’t exist.

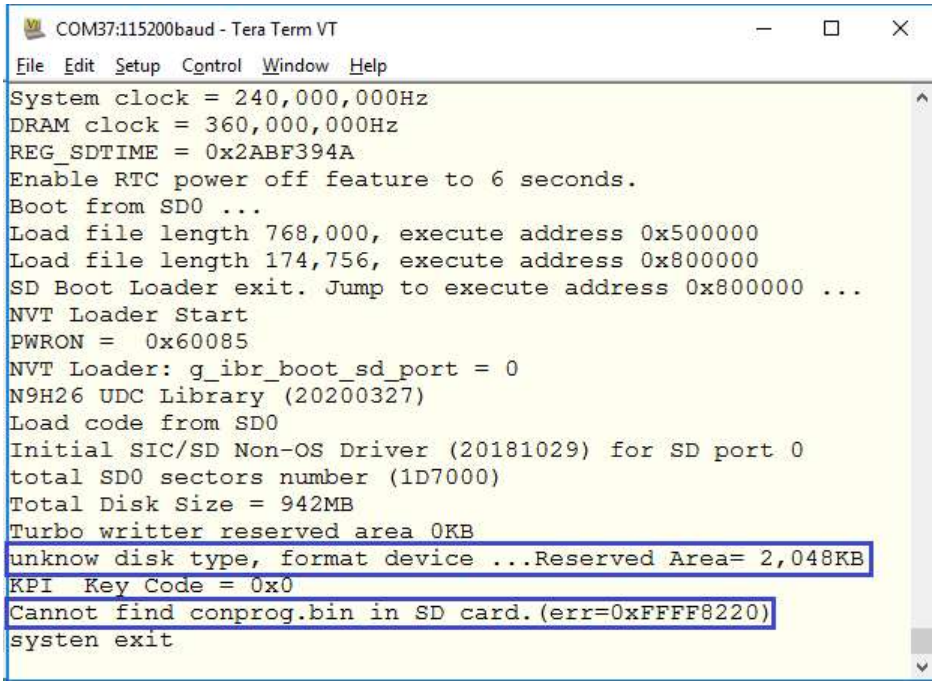

Figure 10 The disk is not formatted

6.2 USB Export

When USB cable is connected to PC / NB and KPI input matched, NVT Loader will export USB Disk.

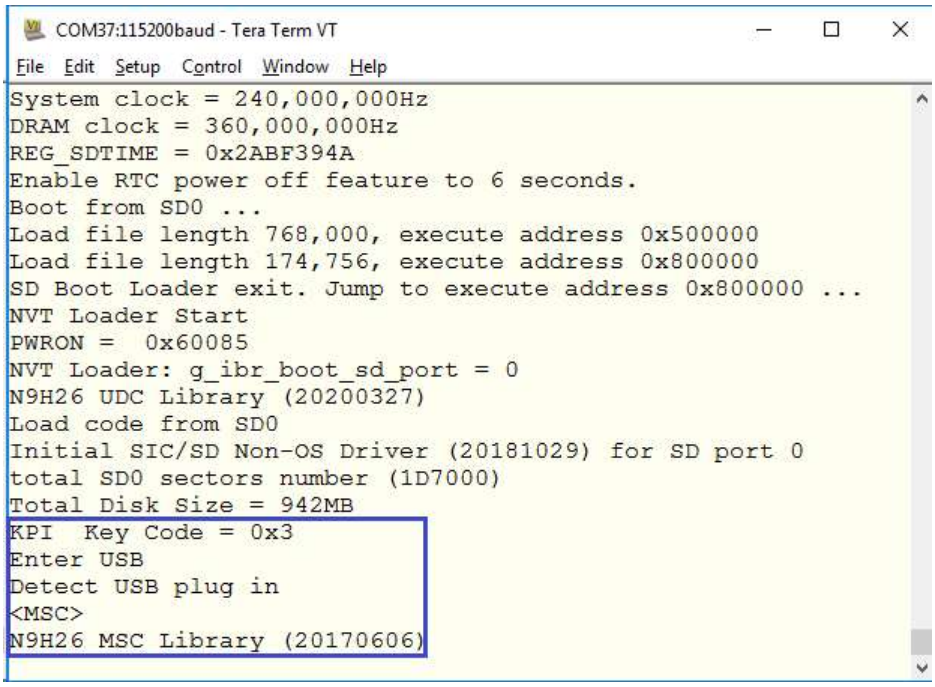

Figure 11 Export USB Disk

User can copy Kernel & Animation file to SD1-1 and other user files to SD1-2.

Figure 12 USB Disk exported by NVT Loader

6.3 Load Kernel

NVT Loader will play animation file, load kernel, and execute kernel.

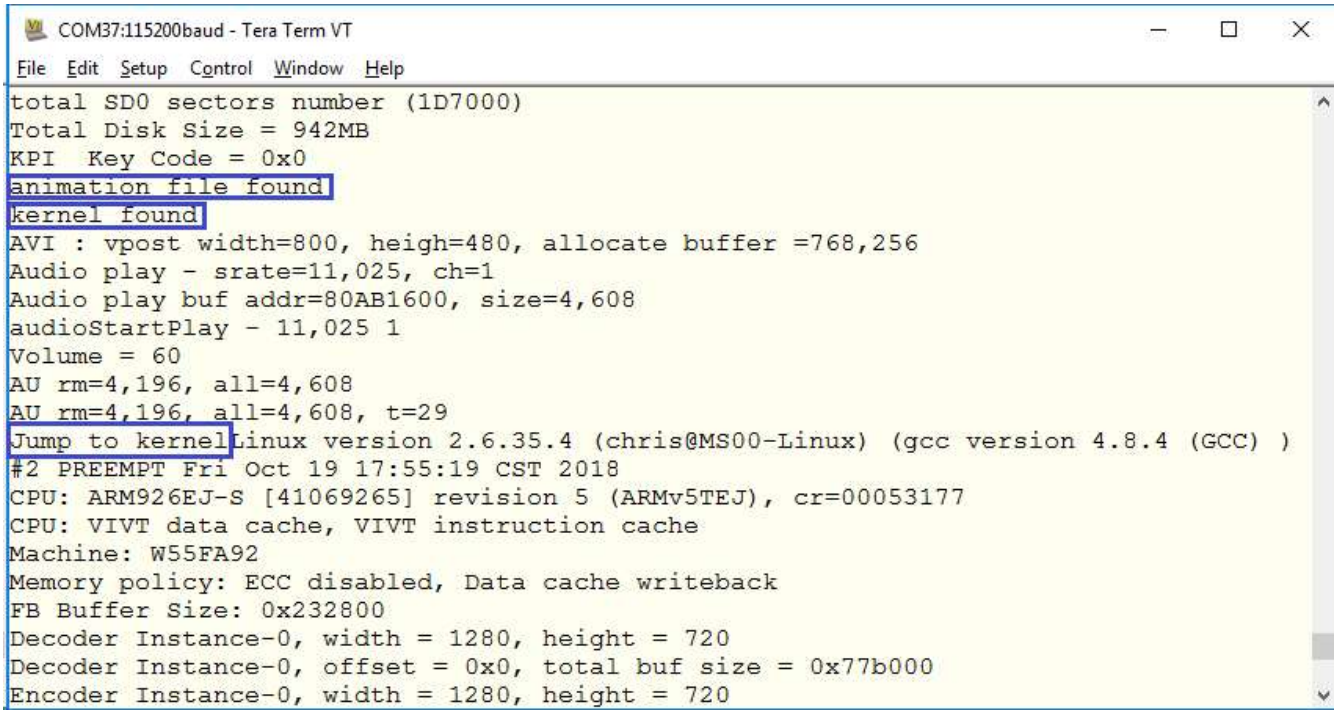

Figure 13 Play animation & Load Kernel

7 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
2021.6.4	1.01	1. Modify document structure.
2018.4.5	1.00	1. Initially issued.

Important Notice

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

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

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