N9H26 NAND Writer User Guide

Document Information

Abstract	Introduce the steps to build and launch NAND Writer 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

NAND writer is the firmware stored in the SD card, which is used to write the necessary firmware to the NAND Flash chip for mass production. After the NAND writer writes the necessary firmware, the device should become a complete product.

The NAND writer can write following firmware to the NAND Flash chip.

- System reserved area
  - NAND loader
  - Logo image
  - NVT loader
- Data area
  - Create FAT file system NAND1-1 and copy files to it
  - Create FAT file system NAND1-2 and copy files to it

Figure 1-1 and Figure 1-2 describe the working environment and workflow of NAND Writer.

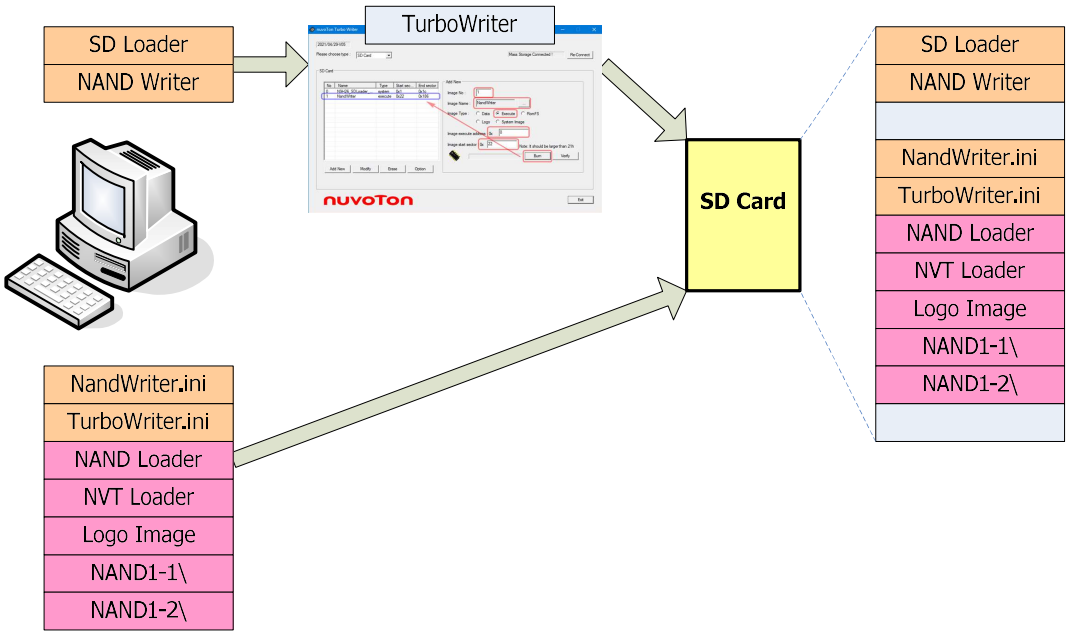

Figure 1-1 Prepare SD Card for NAND Writer

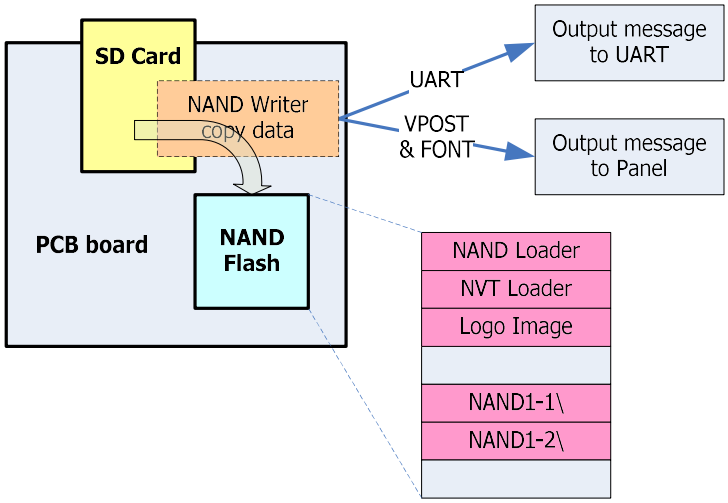

Figure 1-2 Execute NAND Writer to Copy Firmware to NAND Flash Chip

Nuvoton provides NAND writer source code within the N9H26 series microprocessor (MPU) BSP.

2 NAND Writer BSP Directory Structure

This chapter introduces the NAND writer related files and directories in the N9H26 BSP.

2.1 Tools\MassProduction_Tools\NandWriter

Doc	The document of the NAND writer.
Ini	The sample for the NAND writer initial file.
Src	The source code of the NAND writer.

2.2 Tools\MassProduction_Tools\NandWriter\Src

GCC	The GCC project files for the NAND writer.
KEIL	The KEIL project files for the NAND writer.
main.c	The main function for the NAND writer.
writer.h	The header file for the NAND writer.
processIni.c	The function to process *NandWriter.ini* initial file.
processOptionalIni.c	The function to process *TurboWriter.ini* initial file.
nvtNandDrv.c	Special NAND Flash driver for the NAND writer.

2.3 Tools\MassProduction_Tools\NandWriter\Ini

NandWriter.ini	The sample initial file for NAND writer.

2.4 Binary file

NandWriter.bin	The binary file of the NAND writer.

3 NAND Writer Source Code

Complete source codes are included in the *N9H26 BSP Tools\MassProduction_Tools\NandWriter* 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 NAND writer project includes one main function file and some sub-function files. It links some N9H26 library that includes GNAND, SIC, NVTFAT, SYS, FONT, VPOST, and GPIO.

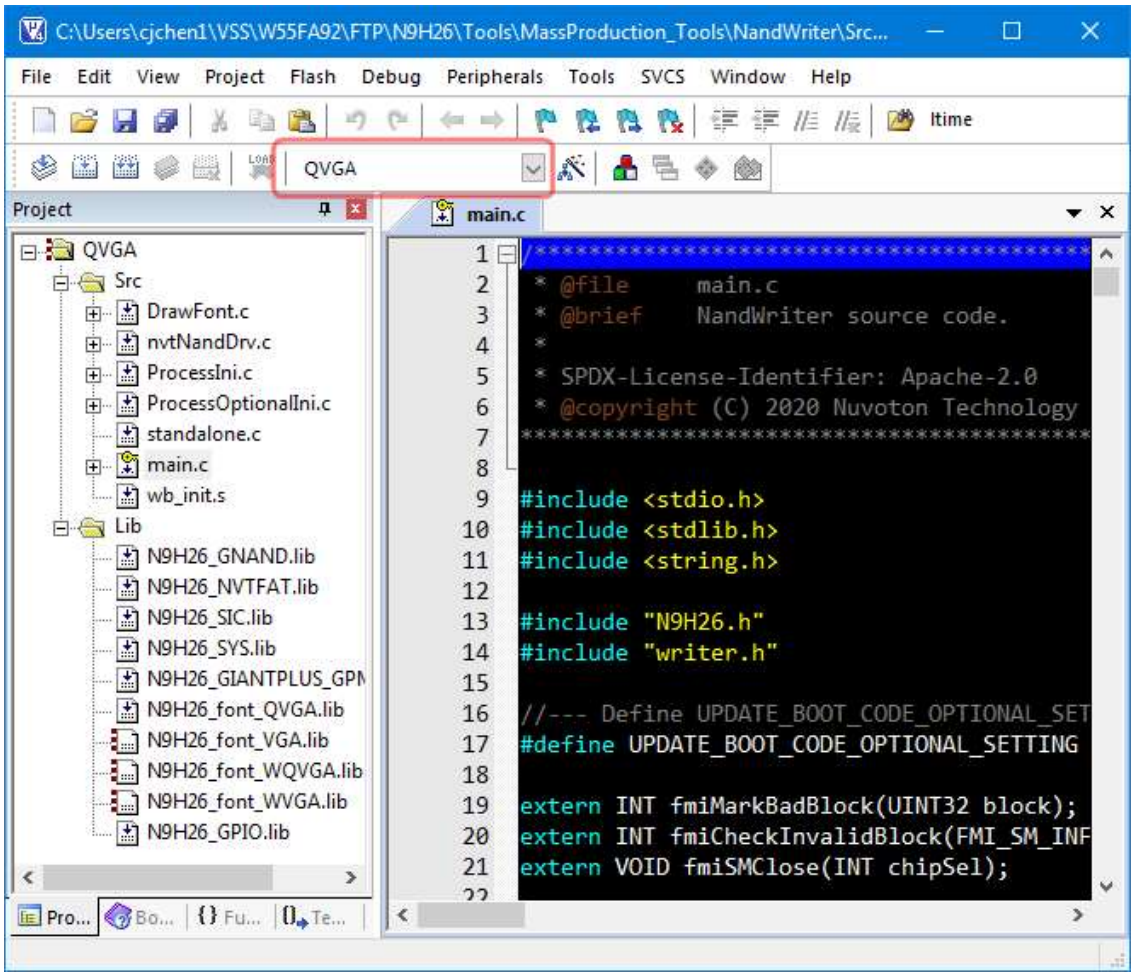

Figure 3-2 NAND Writer Project Tree on Keil MDK

The NAND writer project includes some targets that can be used in different LCM panel. Please select the project target according to the used panel.

- **QVGA**: For QVGA (320 x 240) LCM panel.
- **WVGA**: For WVGA (800 x 400) LCM panel.
- **WQVGA**: For QVGA (480 x 272) LCM panel.

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 initialization process.

```
int main()
{
    /* Omit some source code in document. */
```

```
    //--- Reset SIC engine to make sure it under normal status.
    outp32(REG_AHBCLK, inp32(REG_AHBCLK) | (SIC_CKE | NAND_CKE | SD_CKE));
    outp32(REG_AHBIPRST, inp32(REG_AHBIPRST) | SIC_RST);    // SIC engine reset is avtive
    outp32(REG_AHBIPRST, inp32(REG_AHBIPRST) & ~SIC_RST);   // SIC engine reset is no
active. Reset completed.

    /* enable cache */
    sysDisableCache();
    sysInvalidCache();
    sysEnableCache(CACHE_WRITE_BACK);

    //--- initial UART
    u32ExtFreq = sysGetExternalClock();
    sysUartPort(1);
    uart.uiFreq = u32ExtFreq;
    uart.uiBaudrate = 115200;
    uart.uiDataBits = WB_DATA_BITS_8;
    uart.uiStopBits = WB_STOP_BITS_1;
    uart.uiParity = WB_PARITY_NONE;
    uart.uiRxTriggerLevel = LEVEL_1_BYTE;
    uart.uart_no = WB_UART_1;
    sysInitializeUART(&uart);
    sysSetLocalInterrupt(ENABLE_FIQ_IRQ);

    sysSetDramClock(eSYS_MPLL, 360000000, 360000000);
    sysSetSystemClock(eSYS_UPLL,            //E_SYS_SRC_CLK eSrcClk
                      240000000,            //UINT32 u32PllKHz
                      240000000);           //UINT32 u32SysKHz
    sysSetCPUClock(240000000/2);

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

3.4 SD Card Initialization

The major tasks of the NAND writer is to copy the next firmware from SD card to the NAND Flash chip. To initialize the SD card driver, both sicOpen() and sicSdOpen0() must be called in source code.

```
int main()
{
```

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

    u32PllOutHz = sysGetPLLOutputHz(eSYS_UPLL, sysGetExternalClock());
    sicIoctl(SIC_SET_CLOCK, u32PllOutHz/1000, 0, 0);
    sicOpen();
    status = sicSdOpen0();
    if (status < 0)
    {
        Draw_Status(font_x+ u32SkipX*g_Font_Step, font_y, Fail);
        sysprintf("===> 1 (No SD Card)\n");
        bIsAbort = TRUE;
        goto _end_;
    }

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

3.5 NAND Flash Initialization

The major tasks of the NAND writer is to copy some firmware from SD card to the NAND Flash chip. To initialize the NAND Flash driver, the sub-function nvtSMInit() must be called in source code.

```
int main()
{
    /* Omit some source code in document. */

    nvtSMInit();    // initial NAND controller and pNvtSM0

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

INT nvtSMInit(void)
{
    if (!bIsNandInit)
    {
        //--- Initial DMAC module in FMI engine
        outpw(REG_DMACCSR, inpw(REG_DMACCSR) | DMAC_EN);
        outpw(REG_DMACCSR, inpw(REG_DMACCSR) | DMAC_SWRST);
        outpw(REG_DMACCSR, inpw(REG_DMACCSR) & ~DMAC_SWRST);
```

```
        //--- Initial SM module in FMI engine
        outpw(REG_FMICR, FMI_SM_EN);                // enable SM module in FMI engine
        outpw(REG_SMCSR, inpw(REG_SMCSR) | SMCR_SM_SWRST);  // software reset SM module
        while(inpw(REG_SMCSR)&SMCR_SM_SWRST);       // waiting for SM module reset done

        outpw(REG_SMTCR, 0x20305);
        outpw(REG_SMCSR, (inpw(REG_SMCSR) & ~SMCR_PSIZE) | PSIZE_512);  // default is 512B
page size
        outpw(REG_SMCSR, inpw(REG_SMCSR) | SMCR_ECC_EN | SMCR_ECC_CHK |
SMCR_ECC_3B_PROTECT | SMCR_REDUN_AUTO_WEN); // enable ECC

        outpw(REG_GPDFUN0, (inpw(REG_GPDFUN0) & (~0xF0F00000)) | 0x20200000);   // enable
NRE/RB0 pins
        outpw(REG_GPDFUN1, (inpw(REG_GPDFUN1) & (~0x0000000F)) | 0x00000002);   // enable
NWR pins
        outpw(REG_GPEFUN1, (inpw(REG_GPEFUN1) & (~0x000FFF0F)) | 0x00022202);   // enable
CS0/ALE/CLE/ND3 pins
        outpw(REG_SMCSR, inpw(REG_SMCSR) & ~SMCR_CS0);                          // enable
CS0
        outpw(REG_SMCSR, inpw(REG_SMCSR) |  SMCR_CS1);                          // disable
CS1

        pNvtSMInfo = (FMI_SM_INFO_T *)malloc(sizeof(NVT_SM_INFO_T));
        memset((char *)pNvtSMInfo, 0, sizeof(NVT_SM_INFO_T));
        pSM0= pNvtSMInfo;
        pNvtSM0 = pSM0;

        if (nvtSM_ReadID(pNvtSM0) < 0)
            return -1;
        fmiSM_Initial(pNvtSM0);

        pNvtSM0->uIBRBlock = fmiSM_GetIBRAreaSize(pNvtSM0);

        bIsNandInit = TRUE;
    }
    return 0;
}
```

3.6 NVTFAT Initialization

The NAND writer will create two partitions for FAT file system on the NAND Flash chip. To initialize the NVTFAT file system, both fsInitFileSystem() and fsAssignDriveNumber() must be called in source code. The size of the first partition depends on the setting of "**[NAND1-1 DISK**

SIZE]" in *NandWriter.ini*. The second partition size is the remaining size of the NAND Flash chip.

```
int main()
{
    /* Omit some source code in document. */

    fsInitFileSystem();
    fsAssignDriveNumber('X', DISK_TYPE_SD_MMC, 0, 1);           // SD0, single partition
    fsAssignDriveNumber('D', DISK_TYPE_SMART_MEDIA, 0, 1);      // NAND1-1, 2 partitions
    fsAssignDriveNumber('E', DISK_TYPE_SMART_MEDIA, 0, 2);      // NAND1-2, 2 partitions

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

3.7 VPOST and FONT Initialization

The NAND writer can display the writing progress on the LCM panel through the VPOST and FONT libraries. To initialize the VPOST and FONT driver, the sub-function Draw_Init() must be called in source code.

```
int main()
{
    /* Omit some source code in document. */

    Draw_Init();

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

//*************************************************************************
// Draw_Init :
//          Initalize the VPOST, Font and prepared the related rectangle for display
//*************************************************************************
void Draw_Init(void)
{
    S_DEMO_RECT s_sRect;
    char Array1[64];

    initVPost(FrameBuffer);
```

```
    InitFont(&s_sDemo_Font,(UINT32)FrameBuffer);

    // Font Width = 16, Height = 22
    g_Font_Width = s_sDemo_Font.u32FontRectWidth;
    g_Font_Height = s_sDemo_Font.u32FontRectHeight;
    g_Font_Step = s_sDemo_Font.u32FontStep;

    // Draw the outside boarder
    DemoFont_ChangeFontColor(&s_sDemo_Font, COLOR_RGB16_WHITE);
    Draw_InitialBorder(&s_sDemo_Font);

    // Draw the Boarder for "N9H26 NandWriter (..)"
    s_sRect.u32StartX =0;
    s_sRect.u32StartY = 0;
    s_sRect.u32EndX = _LCM_WIDTH_,
    s_sRect.u32EndY = g_Font_Height+1;
    DemoFont_Border(&s_sDemo_Font,
                    &s_sRect,
                    2);

    sprintf(Array1, "N9H26 NandWriter (v%d.%d)",MAJOR_VERSION_NUM, MINOR_VERSION_NUM);
    Draw_Font(COLOR_RGB16_WHITE, &s_sDemo_Font,
                    0,
                    0,
                    Array1);

    // Only draw the Boarder for "Nand: Block * Page * PageSize"
    // But, Nand information is delayed to Nand initialization
    s_sRect.u32StartX =0;
    s_sRect.u32StartY = _LCM_HEIGHT_-1-g_Font_Height;
    s_sRect.u32EndX = _LCM_WIDTH_-1,
    s_sRect.u32EndY =  _LCM_HEIGHT_-1;
    DemoFont_Border(&s_sDemo_Font,
                    &s_sRect,
                    2);
}
```

3.8 Build NAND Writer Project

Normally, the NAND writer doesn't need to modify. If the NAND writer 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 NAND Writer on Keil MDK

The binary file of the NAND writer is named *NandWriter.bin*, and it is placed in the folder *NandWriter\Src\KEIL\NandWriter_Data\<target>*.

3.9 Prepare SD Card for NAND Writer

There are two steps to execute the NAND writer for mass production. First, prepare an SD card and write NAND Writer binary and other necessary firmware to it. Next, insert the SD card into the device and power-on device to execute the NAND writer. Because the SD loader will be executed before the NAND loader when booting, the NAND writer loaded by the SD loader will be executed first.

This section will describe how to prepare the SD card for NAND writer. Next section will describe how to execute the NAND writer on device.

Please refer to Figure 1-1 for the SD card preparing for NAND writer.

3.9.1 Burn SD Loader and NAND Writer to SD Card

The SD loader firmware and NAND writer firmware can be programmed to SD card by the tool *TurboWriter* and here is the step. 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. Insert 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. Press "**Option**" button to switch to the option page.

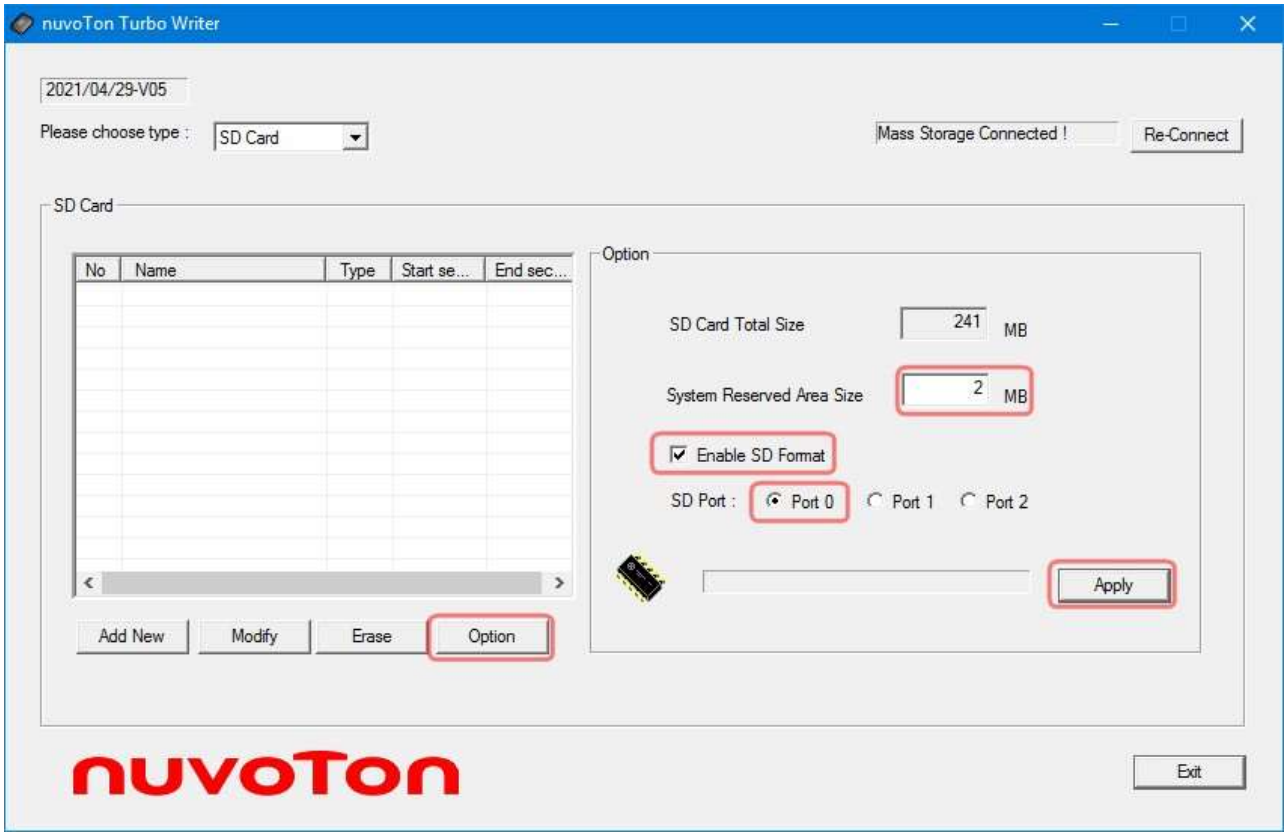

Figure 3-4 Reverse Area for SD Loader and NAND Writer by TurboWriter

9. Fill in a number in the field "**System Reserved Area Size**". 1 or 2 MB is enough for normal case.
10. Check the "**Enable SD Format**"
11. Select the "**Port 0**" on the option "**SD Port**".
12. Press "**Apply**" button to reserve some sectors for SD loader and NAND writer and then format the SD card.
    Please note that the original content in the SD card will be lost.
13. Press "**Add New**" button to switch back to the Add New page to burn SD loader binary

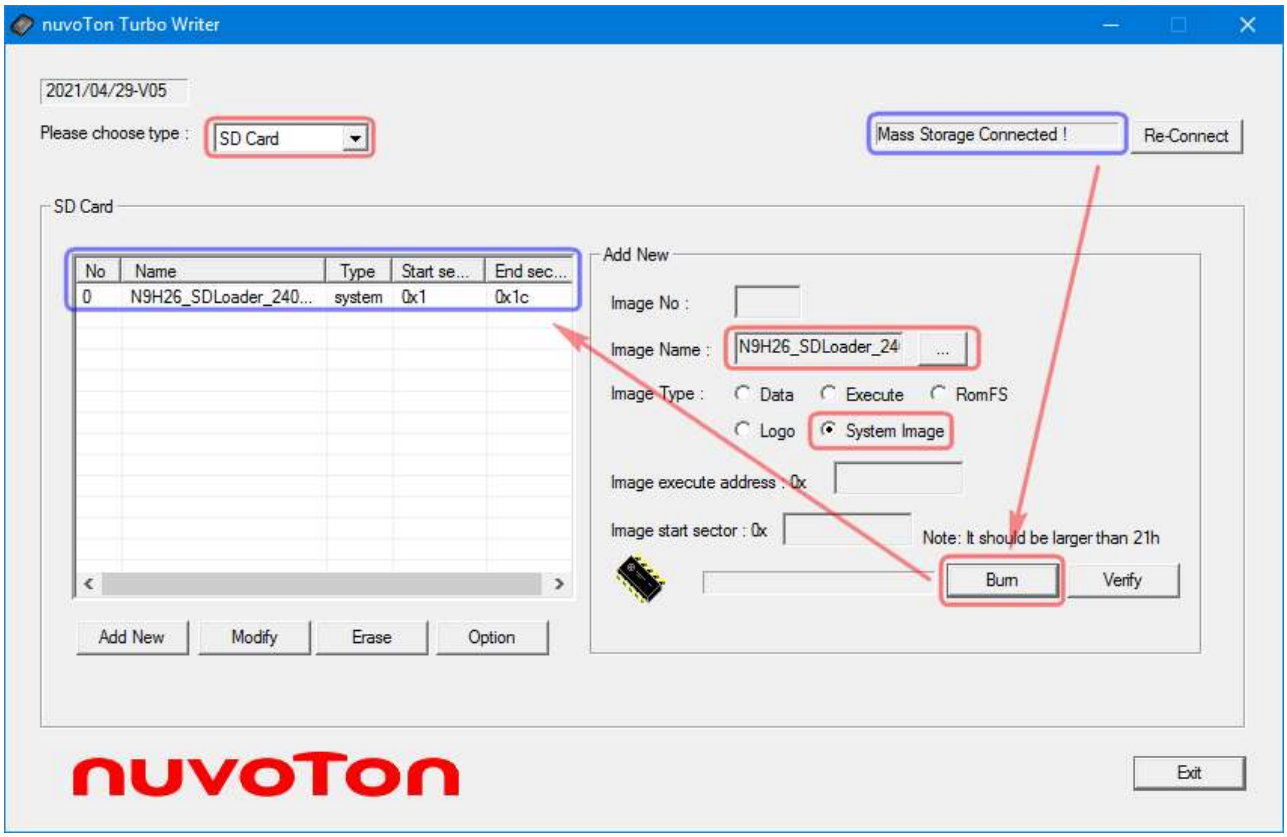

Figure 3-5 Programmed SD Loader by TurboWriter

14. Select SD loader binary file on the option "**Image Name**".
15. Select "**System Image**" on the option "**Image Type**".
16. Press "**Burn**" button to burn the SD loader binary into SD card.
17. After burning completed, check the SD loader information in the left table.

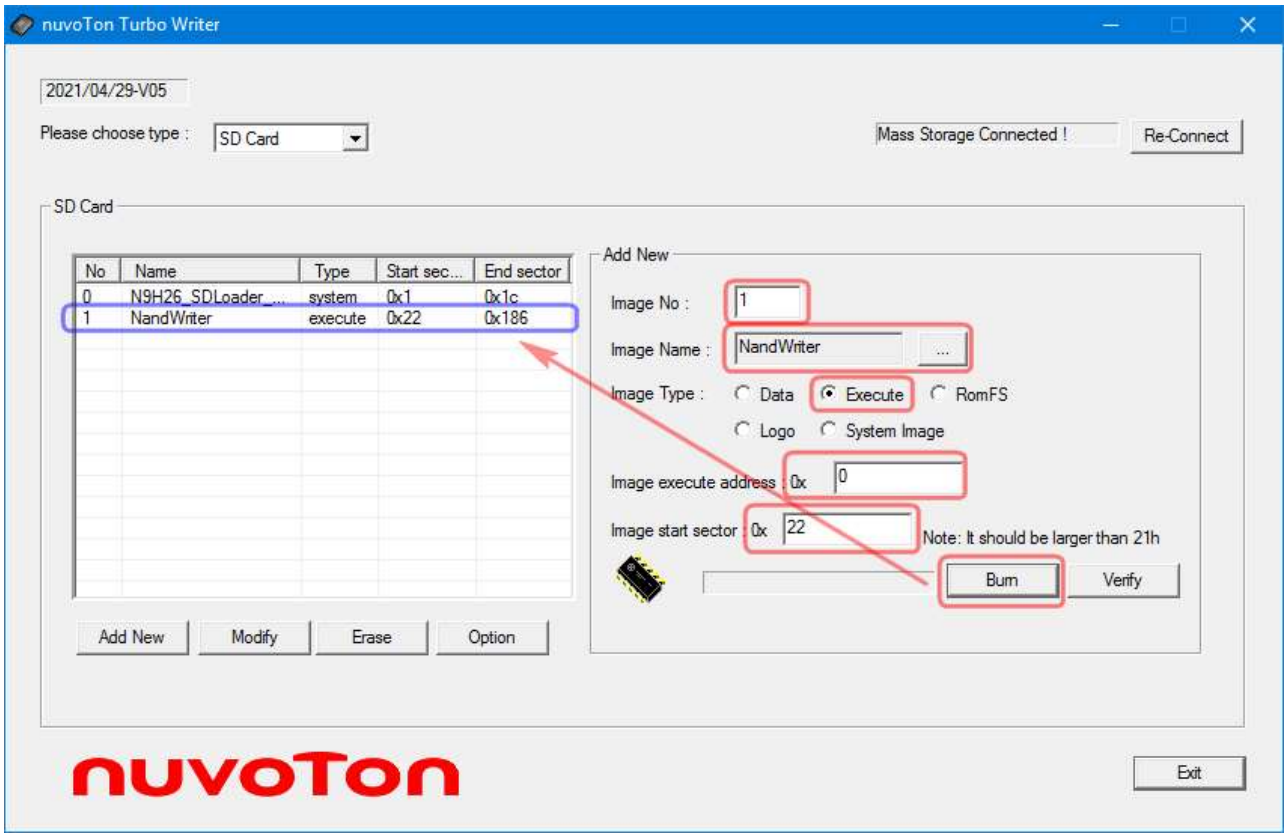

Figure 3-6 Programmed NAND Writer by TurboWriter

18. Select "**Execute**" on the option "**Image Type**".
19. Fill "**1"** in the field "**Image No**"
20. Select NAND writer binary file on the option "**Image Name**"
21. Fill "**0"** in the field "**Image execute address**"
22. Fill "**22"** in the field "**Image start sector**"
23. Press "**Burn**" button to burn the NAND writer binary into SD card.
24. After burning completed, check the NAND writer information in the left table.
25. Remove USB device safely.
26. Plug out USB cable from PC/NB.
27. Reset the device under Normal mode.

3.9.2 Copy Product Firmware to SD Card

The product firmware files that want to burn to NAND Flash chip also have to copy to the SD card through the SD card reader.

The SD card content structure is as below figure. The root directory contains the *NandWriter.ini* (must), *TurboWriter.ini* (must), *NandLoader.bin* (must), *Logo.dat* (option), *NvtLoader.bin* (option), *NAND1-1* folder, *NAND1-2* folder, and *NANDCARD* folder (option). The files in *NAND1-1* folder are copied to root folder of partition *NAND1-1* on NAND Flash and files in *NAND1-2* folder are copied to root folder of partition *NAND1-2* on NAND Flash. It also provides some option in *NandWriter.ini* for user. Please check the INI File section for detail.

Please note that the disk volume label of SD card cannot be the same as any folder name in SD card. Fox example, "*NAND1-1*" or "*NAND1-2*"

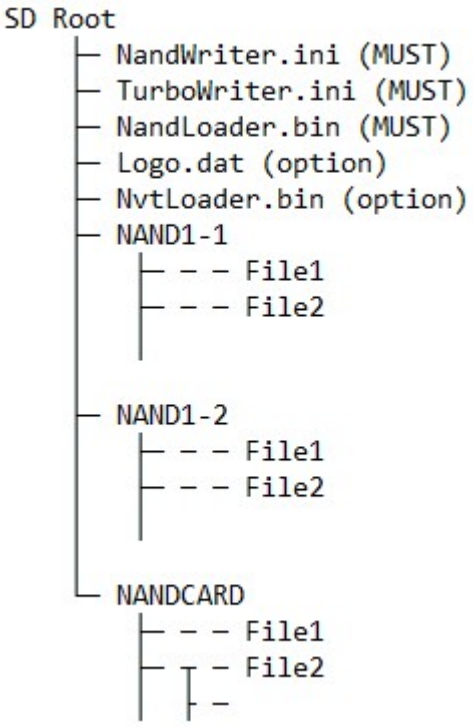

Figure 3-7 The SD Card Content Structure for NAND Writer

3.9.3 NAND Writer Initialization File

NAND writer uses an initialization file, *NandWriter.ini*, to provides the user a flexible way to do a restricted modification without modifying the source code.

The *NandWriter.ini* file provides some features as below:

```
[NAND1-1 FAT FILE]
// -1 to skip NAND1-1 copy, 0 to use DiskImage without MBR,
// 1 to Use FAT file, 2 to use DiskImage with MBR
1

[NandLoader File Name]
```

```
// All file name length MUST <= 511 bytes
// Unavailable if [NAND1-1 FAT FILE] is -1
NANDLoader.bin

[Logo File Name]
// Unavailable if [NAND1-1 FAT FILE] is -1
Logo.bin

[NVTLoader File Name]
// Unavailable if [NAND1-1 FAT FILE] is -1
NVTLoader_NAND.bin

[System Reserved MegaB]
// Unit: Mega Byte
2

[NAND1-1 DISK SIZE]
// Unit: Mega Byte (default: 16MB)
// This specify Nand1-1 partition size, total capacity - Nand1-1 = Nand1-2 partition size
// Unavailable if [NAND1-1 FAT FILE] is 2
32

[NAND1-2 FAT FILE]
// Unavailable if [NAND1-1 FAT FILE] is 2
// -1 to skip NAND1-2 copy, 0 to use DiskImage without MBR, 1 to Use FAT file
1

[NANDCARD FAT FILE]
// -1 to skip NANDCARD copy, 0 to use DiskImage without MBR,
// 1 to Use FAT file, 2 to use DiskImage with MBR
-1
```

Due to the limited parsing ability of NAND writer, there are some constraints in *NandWriter.ini* as below:

- No space is allowed to precede the option for each line.
- Only "*//*" comment is allowed at the beginning of each line
- String in "**[ ]**" is not allowed to changed.
- Only "**[Logo File Name]**", "**[NVTLoader File Name]**" and "**[System Reserved MegaB]**" sections are option for its setting. The others are must.

If the "**[System Reserved MegaB]**" section is not provided, the default reserved size is 4 megabytes for it.

If the logo file is not necessary for the NAND writer, below two methods are all to skip burning *Logo.dat* into the NAND Flash.

```
[Logo File Name]
//Logo.dat
```

or

```
[Logo File Name]

```

It also allows changing the file name for burning. Below sample changes the file name from *NandLoader.bin* to *Nuvoton.bin* for "**[NandLoader File Name]**" section. Please note that the file name length MUST less than or equal to 511 bytes.

```
[NandLoader File Name]
Nuvoton.bin
```

There are 4 options for section **[NAND1-1 FAT FILE], [NAND1-2 FAT FILE], and [NANDCARD FAT FILE]**:

- Option "**-1**": Skip to check the *NAND1-x* folder.
- Option "**0**": The NAND writer copies disk image file *content.bin* on *NAND1-x* folder in the SD card through the GNAND to *NAND1-x* partition. It gets the best performance but it need to prepare the disk image file by *NRomMaker* tool or Linux
- Option "**1**": The NAND writer copy those files on *NAND1-x* folder in the SD card through FAT to *NAND1-x* partition.
- Option "**2**": Like option "0" but the disk image must include partition table (MBR, Master Boot Record).

Please note that the value **[NAND1-1 FAT FILE]** could influence the action of other options.

- If NAND1-1 is "**-1**" (skip), NAND writer do nothing at NAND Flash chip. It includes all files in System Reserved Area and *NAND1-2*.
- If NAND1-1 is "**2**" (disk image with MBR), the options **[NAND1-1 DISK SIZE]** and **[NAND1-2 FAT FILE]** are unavailable since they are decided by MBR within disk image, not by NAND writer.

3.9.4 Turbo Writer Initialization File

NAND writer need another initialization file, *TurboWriter.ini*, to provides the user a flexible way to configure system before NAND loader running on device.

Please obtain the *TurboWriter.ini* file from the *TurboWriter* folder for the target N9H26 platform.

3.10 Run NAND Writer

After preparing the SD card, insert it into the SD card socket on the device and power on the device to execute the NAND writer. The NAND writer will display the burning progress on the UART port and the panel.

Please refer to Figure 1-2 for the NAND writer workflow.

The output messages on UART port as below.

```
=====> N9H26 NandWriter (v1.10) Begin [0] <=====
Initial SIC/SD Non-OS Driver (20210331) for SD port 0
Process X:\NandWriter.ini file ...
  NnandLoader = NANDLoader.bin
  Logo        = Logo.bin
  NvtLoader   = NVTLoader_NAND.bin
  SystemReservedMegaByte = 2
  NAND1_1_SIZE = 32
  NAND1_1_FAT  = 1
  NAND1_2_FAT  = 1
  NANDCARD_FAT = -1
Process X:\TurboWriter.ini file ...
  OptionalMarker = 0xAA55AA55
  Counter        = 39
  Pair 0: Address = 0xB0000204, Value = 0xFFFFFFFF
  Pair 1: Address = 0xB0000208, Value = 0xFFFFFFFF
  Pair 2: Address = 0xB0003008, Value = 0x0000805A
  Pair 3: Address = 0xB0003028, Value = 0x2AFF3B4A
  Pair 4: Address = 0xB0003004, Value = 0x00000021
  Pair 5: Address = 0x55AA55AA, Value = 0x00000100
  Pair 6: Address = 0xB0003004, Value = 0x00000023
  Pair 7: Address = 0xB0003004, Value = 0x00000027
  Pair 8: Address = 0x55AA55AA, Value = 0x00000100
  Pair 9: Address = 0xB000301C, Value = 0x00002402
  Pair 10: Address = 0x55AA55AA, Value = 0x00000100
  Pair 11: Address = 0xB0003018, Value = 0x00000532
  Pair 12: Address = 0x55AA55AA, Value = 0x00000100
  Pair 13: Address = 0xB0003004, Value = 0x00000027
  Pair 14: Address = 0x55AA55AA, Value = 0x00000100
  Pair 15: Address = 0xB0003004, Value = 0x0000002B
  Pair 16: Address = 0x55AA55AA, Value = 0x00000100
  Pair 17: Address = 0xB0003004, Value = 0x0000002B
  Pair 18: Address = 0x55AA55AA, Value = 0x00000100
  Pair 19: Address = 0xB0003018, Value = 0x00000432
  Pair 20: Address = 0x55AA55AA, Value = 0x00000100
  Pair 21: Address = 0xB000301C, Value = 0x00002782
  Pair 22: Address = 0x55AA55AA, Value = 0x00000100
  Pair 23: Address = 0xB000301C, Value = 0x00002402
  Pair 24: Address = 0x55AA55AA, Value = 0x00000100
  Pair 25: Address = 0xB0003004, Value = 0x00000020
  Pair 26: Address = 0x55AA55AA, Value = 0x00000100
  Pair 27: Address = 0xB0003054, Value = 0x00000013
  Pair 28: Address = 0x55AA55AA, Value = 0x00001000
  Pair 29: Address = 0xB0003054, Value = 0x0000001E
```

```
  Pair 30: Address = 0x55AA55AA, Value = 0x00005000
  Pair 31: Address = 0x5A5A5A5A, Value = 0x00000000
  Pair 32: Address = 0x55AA55AA, Value = 0x00001000
  Pair 33: Address = 0x5A5A5A5A, Value = 0x00000002
  Pair 34: Address = 0x5A5A5A5A, Value = 0x00000002
  Pair 35: Address = 0xB0003054, Value = 0x0000001A
  Pair 36: Address = 0x55AA55AA, Value = 0x00002000
  Pair 37: Address = 0xB0000208, Value = 0x00008354
  Pair 38: Address = 0xB0000204, Value = 0x00E5011F
nvtSM_ReadID: Found SLC NAND, ID [AD][F1][00][1D], page size 2,048, BCH T8
uIBRAreaSize = 4 blocks
=====> Erase NAND chip on CS0 [26] <=====
=====> copy and verify nandloader [110] <=====
=====> copy and verify logo [118] <=====
=====> copy and verify nvtloader [142] <=====
System Reserved Size = 2 MByte

=====> Create GNAND for NAND on CS0 [164] <=====
[GNAND] GNAND Library Version: V1.02.5
Initial NAND NonOS Driver (20210331) for NAND port 0
NAND: Found SLC NAND, ID [AD][F1][00][1D][AD], page size 2,048, BCH T8
uIBRAreaSize = 4 blocks
[GNAND] NAND flash chip detected, chip ID is 0xAD-0xF1 !
Nand1-1 Partition Size = 32
=====> partition and format [284] <=====

Disk X Size: 244,928 Kbytes, Free Space: 239,712 KBytes
Disk D Size: 32,752 Kbytes, Free Space: 32,752 KBytes
Disk E Size: 90,048 Kbytes, Free Space: 90,048 KBytes
Driver D -- Start sector : 4,096, Total sector : 65,536
Driver E -- Start sector : 69,695, Total sector : 180,161

=====> copy First Partition Content [301] <=====
Copying file CONPROG.BIN
Copying file POINTE~1
Copying file TIMEST~1
=====> Copy Second Partition Content [888] <=====
Copying file A2B.SH

=====> Finish [904] <=====
```

Figure 3-8 The NAND Writer Output on UART Port

The output on panel can divide into several parts:

- Version Number: Display the NAND writer version number.

- Final Status: Display the final operation status. If there is any fail in the operation sequence, the final status will be "**FAIL**".
- Operation Sequence: Display the current operation progress.
- Current operation: Display more detailed information about the current operation. For example, if it fails at a certain step, the "**Code**" field will display its return code.
- The NAND Flash Information: Display current NAND Flash information in the format "Nand: *Total_Block_Number* (Blk) * *Page_Number_Per_Block* (Pg) * *Page_Size* (Size).

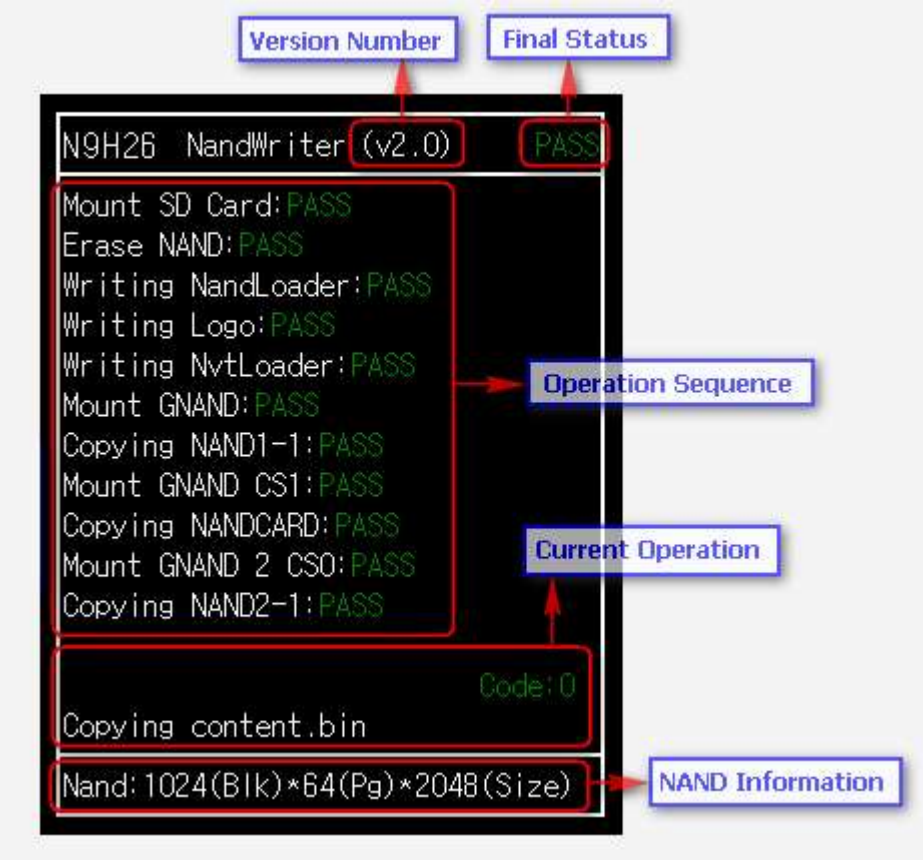

Figure 3-9 The NAND Writer Output on Panel

4 Change Display Panel and Font

4.1 Display Panel and Font Configuration

The NAND writer declares its font size and display panel resolution in *N9H26_Font.h*. This file can be found at *BSP\Driver\Include* directory.

```
#ifdef _DEMO_WQVGA_
    #define _FONT_STRIDE_ 480
    #define _FONT_LINE_       47          //480/10 = 48,
    #define _LCM_WIDTH_       480
    #define _LCM_HEIGHT_  272
#endif
#ifdef _DEMO_QVGA_
    #define _FONT_STRIDE_ 320
    #define _FONT_LINE_       31          //320/10 = 32,
    #define _LCM_WIDTH_       320
    #define _LCM_HEIGHT_  240
#endif
#ifdef _DEMO_VGA_
    #define _FONT_STRIDE_ 640
    #define _FONT_LINE_       63          //640/10 = 64,
    #define _LCM_WIDTH_       640
    #define _LCM_HEIGHT_  480
#endif
#ifdef _DEMO_WVGA_
    #define _FONT_STRIDE_ 800
    #define _FONT_LINE_       79          //800/10 = 80,
    #define _LCM_WIDTH_       800
    #define _LCM_HEIGHT_  480
#endif
#ifdef _DEMO_SVGA_
    #define _FONT_STRIDE_ 800
    #define _FONT_LINE_       79          //800/10 = 80,
    #define _LCM_WIDTH_       800
    #define _LCM_HEIGHT_  600
#endif
```

The macro definition about font and panel resolution is defined within project file.

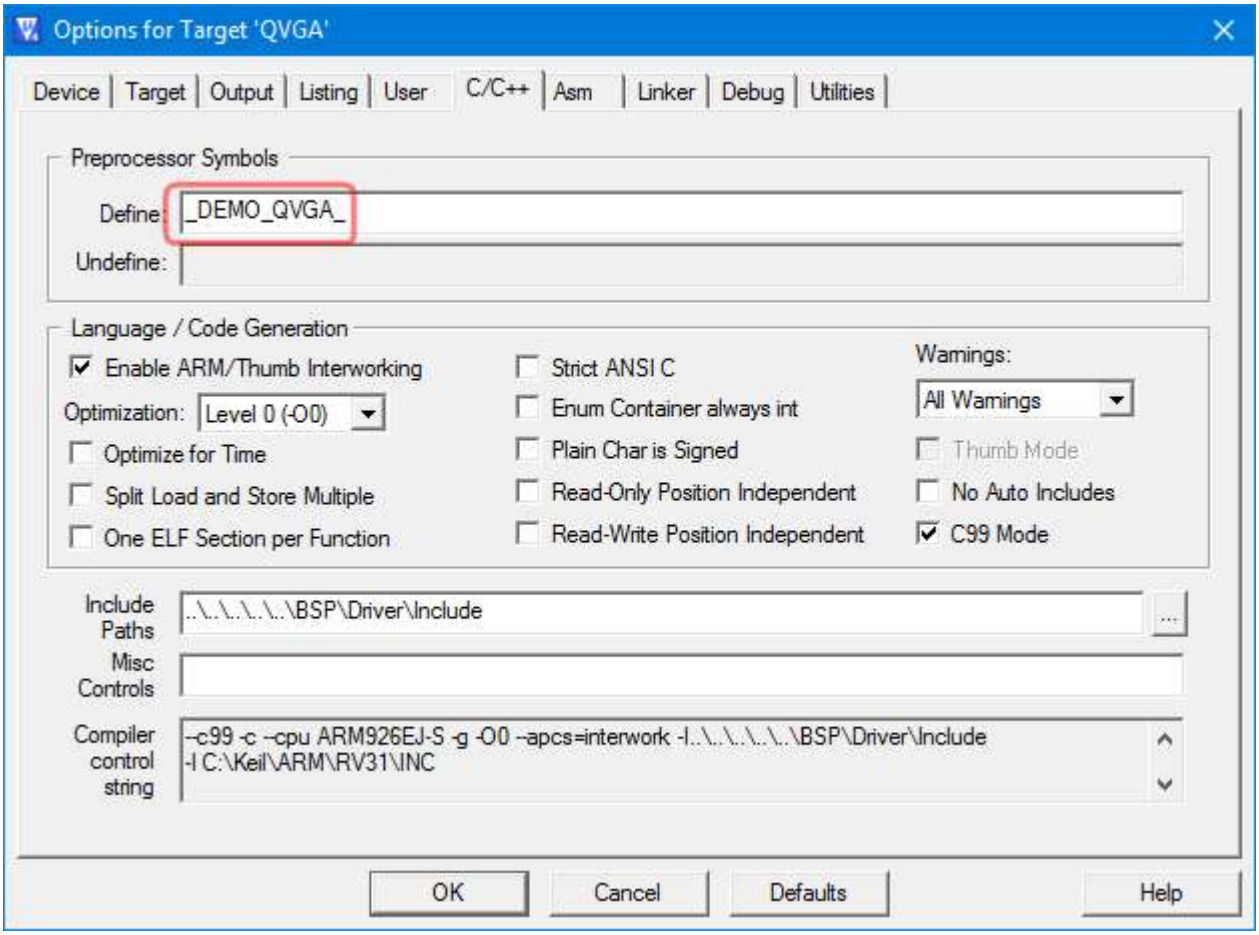

Figure 4-1 The Macro Definition about Font and Panel Resolution

4.2 Display Panel and Font Driver

The NAND writer project includes the Font library and the VPOST library to support different LCD resolution. For system connected to other panel that NAND writer supported, the Font library and the VPOST library have to be enabled by selecting the correct project target. For system connected to other panel that NAND writer does not supported, a new Font library and a new VPOST library have to be added to the project. Both the Font library and the VPOST library can be found under folder *Library\IPLib* in N9H26 BSP.

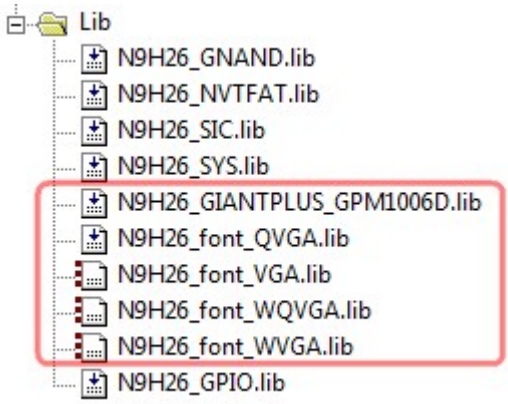

Figure 4-2 The Display Panel and Font Library in Project

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
2021.6.10	1.01	1. Modify document structure.
2018.5.4	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.