N9H26 SD Writer User Guide

Document Information

Abstract	Introduce the steps to build and launch SD 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 3

2 SD WRITER BSP DIRECTORY STRUCTURE 5

2.1 Tools\MassProduction_Tools\SDWriter 5

2.2 Tools\MassProduction_Tools\SDWriter\Src 5

2.3 Tools\MassProduction_Tools\SDWriter\Ini 5

2.4 Binary file 5

3 SD WRITER SOURCE CODE 6

3.1 Development Environment 6

3.2 Project Structure 6

3.3 System Initialization 7

3.4 SD Card Initialization 8

3.5 NVTFAT Initialization 10

3.6 VPOST and FONT Initialization 10

3.7 Build SD Writer Project 12

4 DOWNLOAD AND RUN 13

4.1 Prepare SD Card for SD Writer 13

4.1.1 Burn SD Loader and SD Writer to SD Card 13

4.1.2 Copy Product Firmware to SD Card 16

4.1.3 SD Writer Initialization File 17

4.1.4 Turbo Writer Initialization File 19

4.2 Run SD Writer 19

5 CHANGE DISPLAY PANEL AND FONT 23

5.1 Display Panel and Font Configuration 23

5.2 Display Panel and Font Driver 24

6 SUPPORTING RESOURCES 26

1 Introduction

SD writer is the firmware stored in the SD card on port 0, which is used to write the necessary firmware to the SD card on target port for mass production. The target port could be SD port 1, SD port 2, SDIO port 0, or SDIO port 1. After the SD writer writes the necessary firmware, the SD card on target port should become the part of the complete product.

The SD writer can write following firmware to the SD card on target port.

- System reserved area
  - SD loader
  - Logo image
  - NVT loader
- Data area
  - Create FAT file system SD1-1 and copy files to it
  - Create FAT file system SD1-2 and copy files to it

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

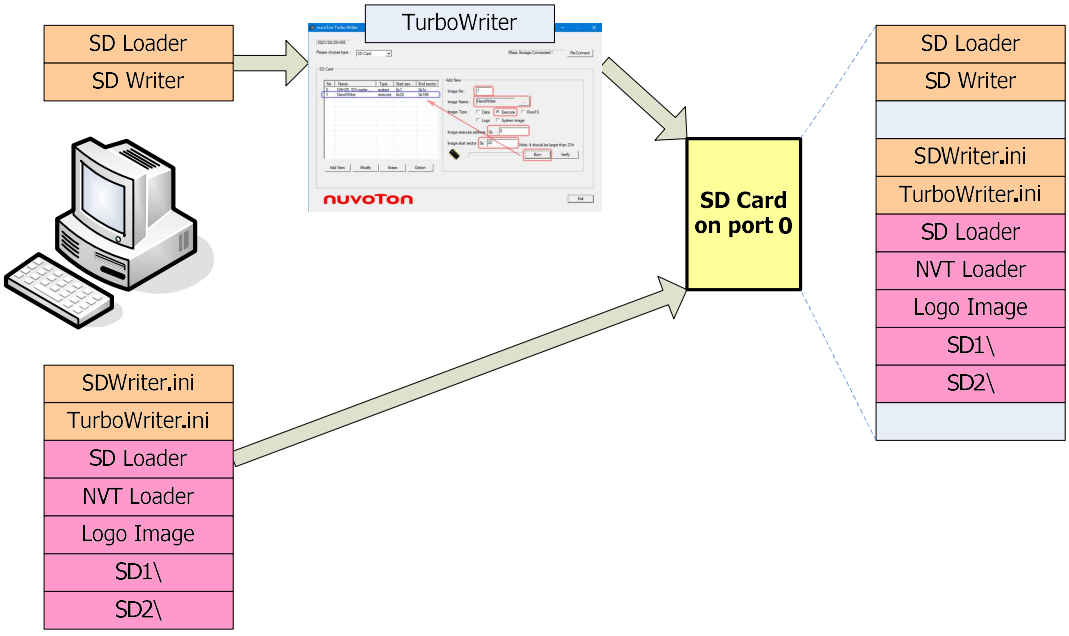

Figure 1-1 Prepare SD Card for SD Writer

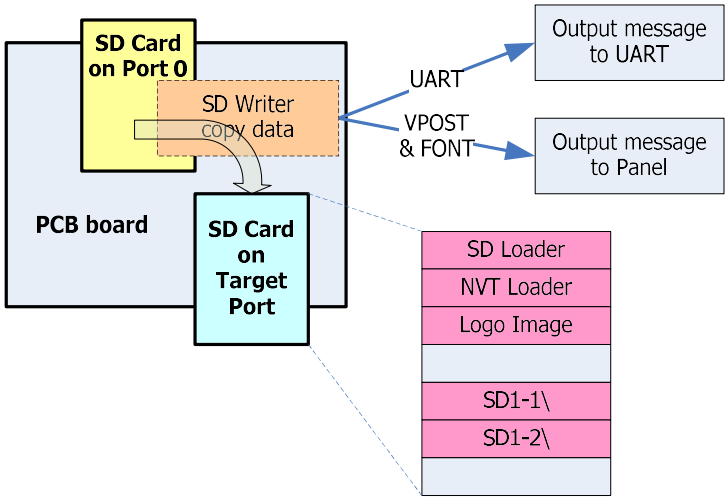

Figure 1-2 Execute SD Writer to Copy Firmware to SD Card on Target Port

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

2 SD Writer BSP Directory Structure

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

2.1 Tools\MassProduction_Tools\SDWriter

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

2.2 Tools\MassProduction_Tools\SDWriter\Src

GCC	The GCC project files for the SD writer.
KEIL	The KEIL project files for the SD writer.
main.c	The main function for the SD writer.
writer.h	The header file for the SD writer.
processIni.c	The function to process *SDWriter.ini* initial file.
processOptionalIni.c	The function to process *TurboWriter.ini* initial file.

2.3 Tools\MassProduction_Tools\SDWriter\Ini

SDWriter.ini	The sample initial file for SD writer.

2.4 Binary file

SDWriter.bin	The binary file of the SD writer.

3 SD Writer Source Code

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

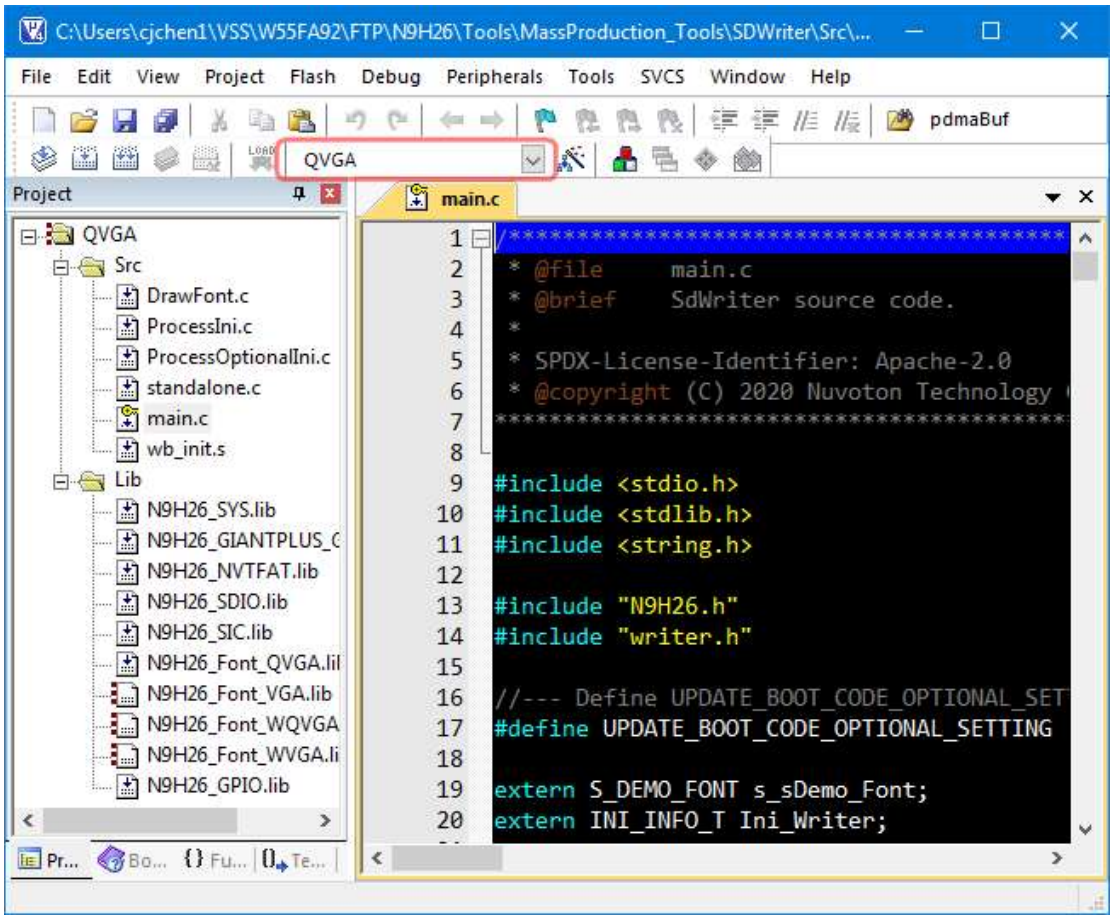

Figure 3-2 SD Writer Project Tree on Keil MDK

The SD 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. */

    /* enable cache */
```

```
    sysDisableCache();
    sysInvalidCache();
    sysEnableCache(CACHE_WRITE_BACK);

    init_UART();

    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 task of the SD writer is to copy the next firmware from SD card on port 0 to the SD card on target port. To initialize the SD card driver for SD card port 0, 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. */
}
```

To initialize the SD card driver for SD port 1, SD port 2, SDIO port 0, or SDIO port 1, the sicSdOpen1(), sicSdOpen2(), sdioSdOpen0(), or sdioSdOpen1() must be called in source code.

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

    //--- mount SD card on port 1 or 2
    switch (Ini_Writer.Target_SD_Port)
    {
        case 1:     // SD 1
            Draw_Font(COLOR_RGB16_WHITE,  &s_sDemo_Font, font_x, font_y, "Mount SD Card
1:");
            u32SkipX = 16;
            status = sicSdOpen1();
            pDisk_target = pDisk_SD1;
            break;
        case 2:     // SD 2
            Draw_Font(COLOR_RGB16_WHITE,  &s_sDemo_Font, font_x, font_y, "Mount SD Card
2:");
            u32SkipX = 16;
            status = sicSdOpen2();
            pDisk_target = pDisk_SD2;
            break;
        case 3:     // SDIO 0
            sdioIoctl(SDIO_SET_CLOCK, u32PllOutHz/1000, 0, 0);  // clock from PLL
            sdioOpen();
            Draw_Font(COLOR_RGB16_WHITE,  &s_sDemo_Font, font_x, font_y, "Mount SDIO Card
0:");
            u32SkipX = 18;
            status = sdioSdOpen0();
            pDisk_target = pDisk_SDIO0;
            break;
        case 4:     // SDIO 1
            sdioIoctl(SDIO_SET_CLOCK, u32PllOutHz/1000, 0, 0);  // clock from PLL
            sdioOpen();
            Draw_Font(COLOR_RGB16_WHITE,  &s_sDemo_Font, font_x, font_y, "Mount SDIO Card
1:");
            u32SkipX = 18;
            status = sdioSdOpen1();
            pDisk_target = pDisk_SDIO1;
            break;
```

```
        default:
            ERR_PRINTF("===> 1.3 (Wrong target SD port %d that defined in
SDWriter.ini)\n", Ini_Writer.Target_SD_Port);
            bIsAbort = TRUE;
            goto _end_;
    }

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

3.5 NVTFAT Initialization

The SD writer will create two partitions for FAT file system on the SD card on port 1 or port 2. To initialize the NVTFAT file system, the fsInitFileSystem() must be called in source code. The size of the first partition depends on the setting of "**[SD1-1 DISK SIZE]**" in *SDWriter.ini*. The second partition size is the remaining size of the SD card.

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

    fsInitFileSystem();

    // The NVTFAT limitation, don't need to assign drive number to SD.
    //      The NVTFAT will auto assign 'C' to SD card partition 1 on SD port 0

    // set volumn lable for dest disk
    fsSetVolumeLabel(dstDrive1, "SD1-1\n", strlen("SD1-1"));
    fsSetVolumeLabel(dstDrive2, "SD1-2\n", strlen("SD1-2"));

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

3.6 VPOST and FONT Initialization

The SD 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 SDWriter
    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 SDWriter (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 SD information
    // But, SD information is delayed to SD 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);
}
```

3.7 Build SD Writer Project

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

The binary file of the SD writer is named *SDWriter.bin*, and it is placed in the folder *SDWriter\Src\KEIL\SDWriter_Data\<target>*.

4 Download and Run

4.1 Prepare SD Card for SD Writer

There are two steps to execute the SD writer for mass production. First, prepare an SD card and write SD Writer binary and other necessary firmware to it. Next, insert the SD card into the SD socket 0 on device and power-on device to execute the SD writer. Because the SD loader on SD port 0 will be executed before the SD loader on SD port 1 or port 2 when booting, the SD writer loaded by the SD loader on SD port 0 will be executed first.

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

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

4.1.1 Burn SD Loader and SD Writer to SD Card

The SD loader firmware and SD 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.

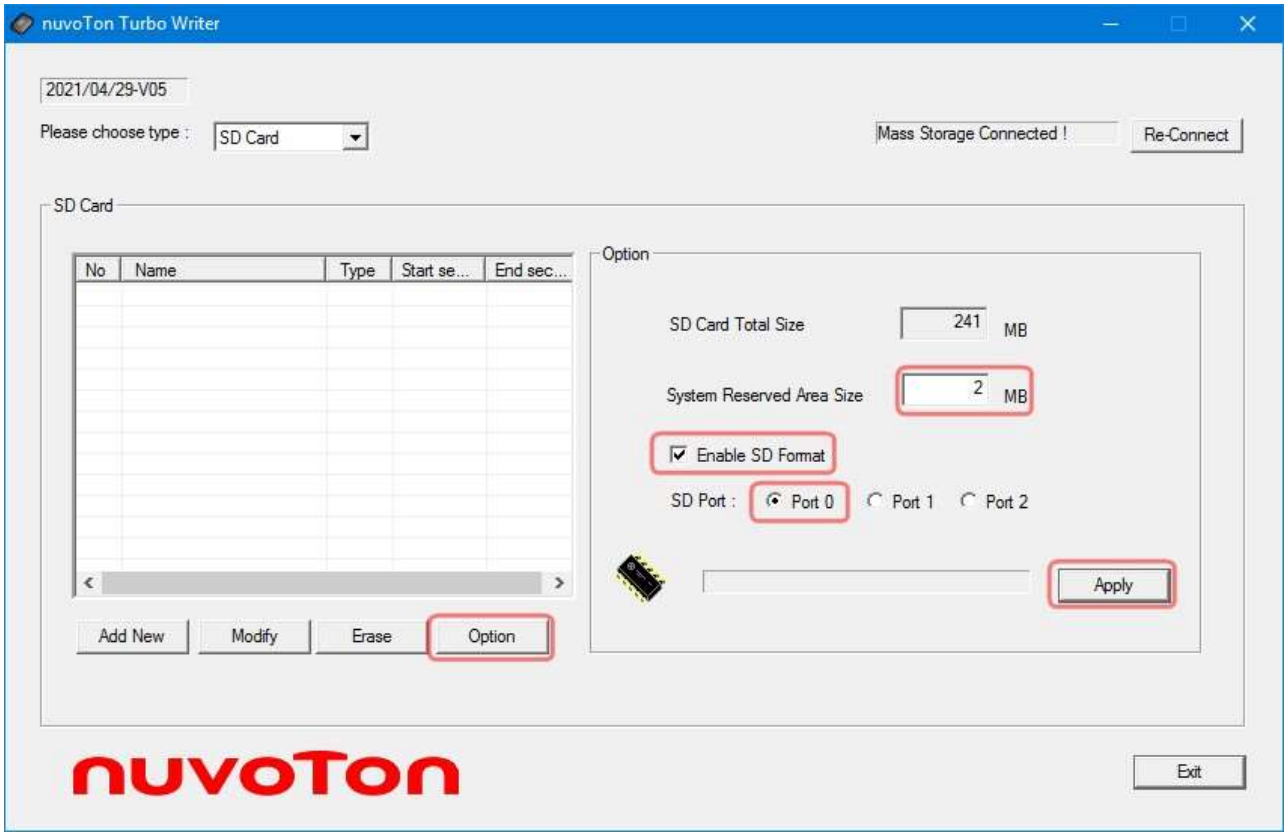

Figure 4-1 Reverse Area for SD Loader and SD Writer by TurboWriter

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 SD 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

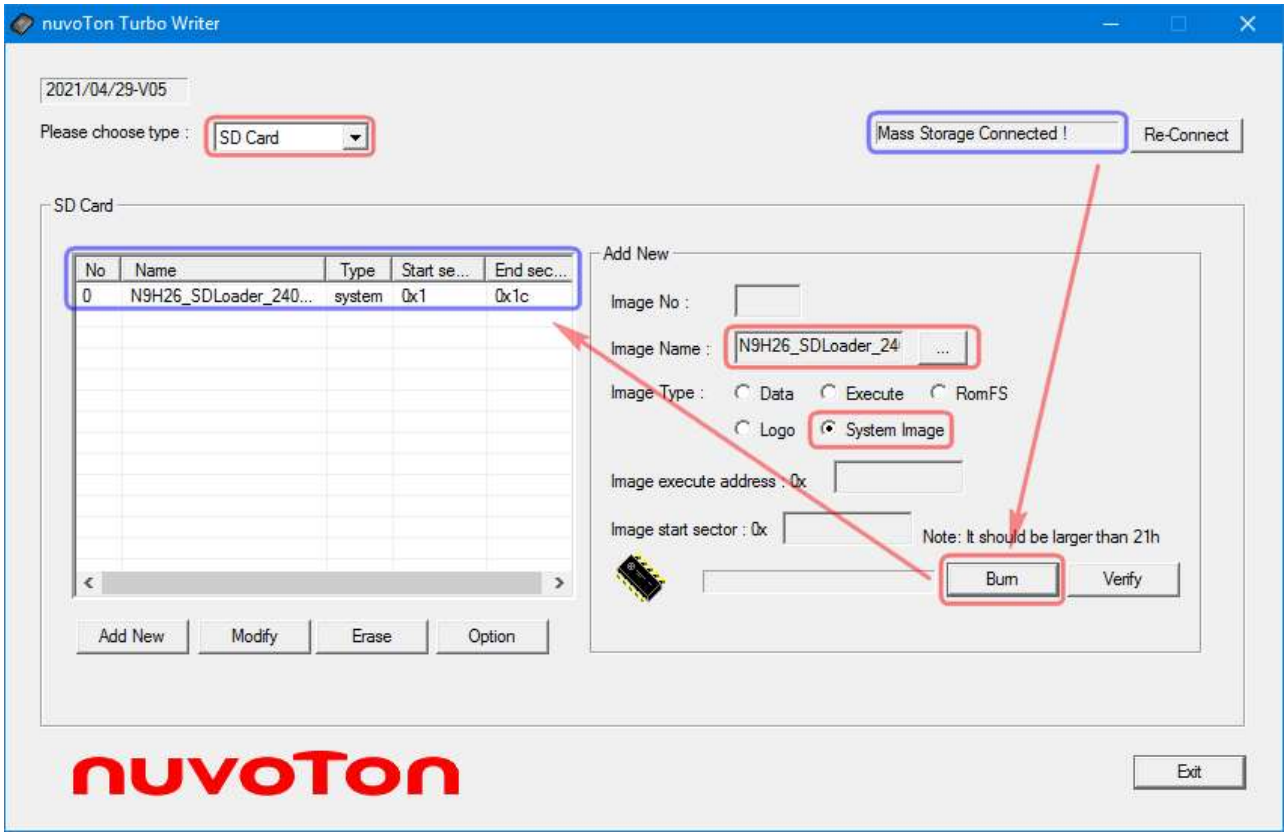

Figure 4-2 Programmed SD Loader by TurboWriter

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.

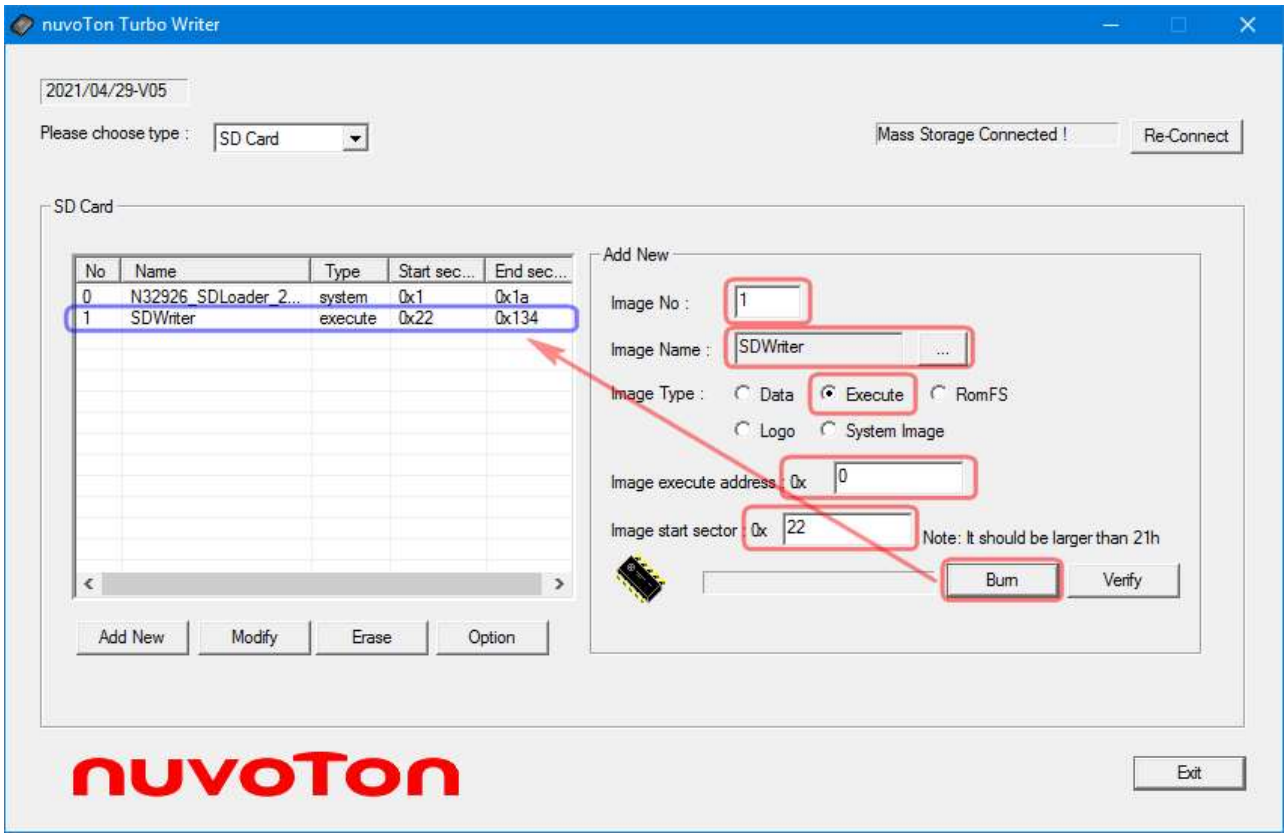

Figure 4-3 Programmed SD Writer by TurboWriter

18. Select "**Execute**" on the option "**Image Type**".
19. Fill "**1"** in the field "**Image No**"
20. Select SD 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 SD writer binary into SD card.
24. After burning completed, check the SD 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.

4.1.2 Copy Product Firmware to SD Card

The product firmware files that want to burn to SD card for product 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 *SDWriter.ini* (must), *TurboWriter.ini* (must), *SDLoader.bin* (must), *Logo.dat* (option), *NvtLoader.bin* (option), *SD1* folder and *SD2* folder. The files in *SD1* folder are copied to root folder of partition *SD1-1* on SD card and files in *SD2* folder are copied to root folder of partition *SD1-2* on SD card. It also provides some option in *SDWriter.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, "*SD1*" or "*SD2*"

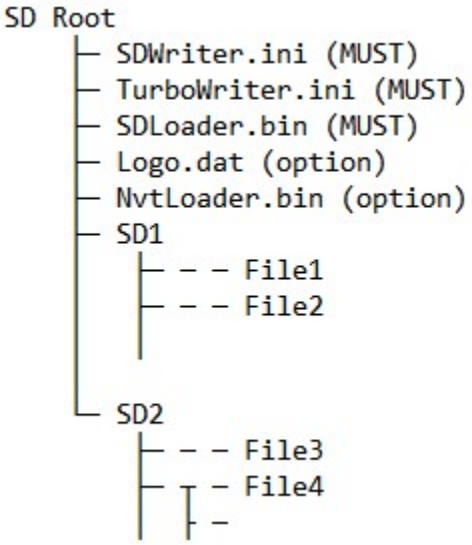

Figure 4-4 The SD Card Content Structure for SD Writer

4.1.3 SD Writer Initialization File

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

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

```
[SDLoader File Name]
SDLoader.bin

[Logo File Name]
Logo.bin

[NVTLoader File Name]
NvtLoader.bin

[System Reserved MegaB]
//Unit: Mega Byte
4
```

```
[SD1-1 DISK SIZE]
//Unit: Mega Byte (default: 16MB)
//This specify SD1-1 partition size, total capacity - Reserved - SD1-1 = SD1-2 partition size
200


[SD1-1 FAT FILE]
// 1 to Use FAT file , 0 to use DiskImage, -1 to skip SD1-1 copy
1


[SD1-2 FAT FILE]
// 1 to Use FAT file , 0 to use DiskImage, -1 to skip SD1-2 copy
1


[Target SD Port]
// [1, 2, 3, 4] 1 to write files to SD port 1,   2 to write files to SD port 2
//              3 to write files to SDIO port 0, 4 to write files to SDIO port 1
1
```

Due to the limited parsing ability of SD writer, there are some constraints in *SDWriter.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 SD writer, below two methods are all to skip burning *Logo.dat* into the SD card on target port.

```
[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 *SDLoader.bin* to *Nuvoton.bin* for "**[SDLoader File Name]**" section. Please note that the file

name length MUST less than or equal to 511 bytes.

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

There are 3 options for section **[SD1-1 FAT FILE]** and **[SD1-2 FAT FILE]**:

- Option "**-1**": Skip to check the *SD1 or SD2* folder.
- Option "**0**": The SD writer copies disk image file *content.bin* on *SD1* or *SD2* folder in the SD card to *SD1-1* or *SD1-2* partition in the SD card on target port. It gets the best performance but it need to prepare the disk image file without MBR (Master Boot Record) by *NRomMaker* tool or Linux.
- Option "**1**": The SD writer copy those files on *SD1* or *SD2* folder in the SD card through FAT to *SD1-1* or *SD1-2* partition.

The SDWriter.ini also allows choosing SD port 1, SD port 2, SDIO port 0, or SDIO port 1 as target SD port.

```
[Target SD Port]
// [1, 2, 3, 4] 1 to write files to SD port 1,   2 to write files to SD port 2
//              3 to write files to SDIO port 0, 4 to write files to SDIO port 1
1
```

4.1.4 Turbo Writer Initialization File

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

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

4.2 Run SD Writer

After preparing the SD card, insert it into the SD card socket 0 on the device, insert a empty SD card into the target SD card socket on the device, and power on the device to execute the SD writer. The SD writer will display the burning progress on the UART port and the panel.

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

The output messages on UART port as below.

```
=====> N9H26 SDWriter (v1.3) Begin [0] <=====
Initial SIC/SD Non-OS Driver (20210315) for SD port 0
Detect SD0: 494,080 sectors * 512 Bytes = 247,040 KBytes
Process SDWriter.ini file...
  SDLoader  = SDLoader.bin
  Logo      = Logo.bin
  NvtLoader = NVTLoader_SD.bin
  SystemReservedMegaByte = 4
  SD1_1_SIZE = 200
  SD1_1_FAT  = 1
  SD1_2_FAT  = 1
  Target_SD_Port = 4
Process C:\TurboWriter.ini file ...
  OptionalMarker = 0xAA55AA55
  Counter        = 40
  Pair 0: Address = 0xB0003010, Value = 0x00000006
  Pair 1: Address = 0xB0000204, Value = 0xFFFFFFFF
  Pair 2: Address = 0xB0000208, Value = 0xFFFFFFFF
  Pair 3: Address = 0xB0003008, Value = 0x0000805A
  Pair 4: Address = 0xB0003028, Value = 0x2AFF3B4A
  Pair 5: Address = 0xB0003004, Value = 0x00000021
  Pair 6: Address = 0x55AA55AA, Value = 0x00000100
  Pair 7: Address = 0xB0003004, Value = 0x00000023
  Pair 8: Address = 0xB0003004, Value = 0x00000027
  Pair 9: Address = 0x55AA55AA, Value = 0x00000100
  Pair 10: Address = 0xB000301C, Value = 0x00002402
  Pair 11: Address = 0x55AA55AA, Value = 0x00000100
  Pair 12: Address = 0xB0003018, Value = 0x00000532
  Pair 13: Address = 0x55AA55AA, Value = 0x00000100
  Pair 14: Address = 0xB0003004, Value = 0x00000027
  Pair 15: Address = 0x55AA55AA, Value = 0x00000100
  Pair 16: Address = 0xB0003004, Value = 0x0000002B
  Pair 17: Address = 0x55AA55AA, Value = 0x00000100
  Pair 18: Address = 0xB0003004, Value = 0x0000002B
  Pair 19: Address = 0x55AA55AA, Value = 0x00000100
  Pair 20: Address = 0xB0003018, Value = 0x00000432
  Pair 21: Address = 0x55AA55AA, Value = 0x00000100
  Pair 22: Address = 0xB000301C, Value = 0x00002782
  Pair 23: Address = 0x55AA55AA, Value = 0x00000100
  Pair 24: Address = 0xB000301C, Value = 0x00002402
  Pair 25: Address = 0x55AA55AA, Value = 0x00000100
```

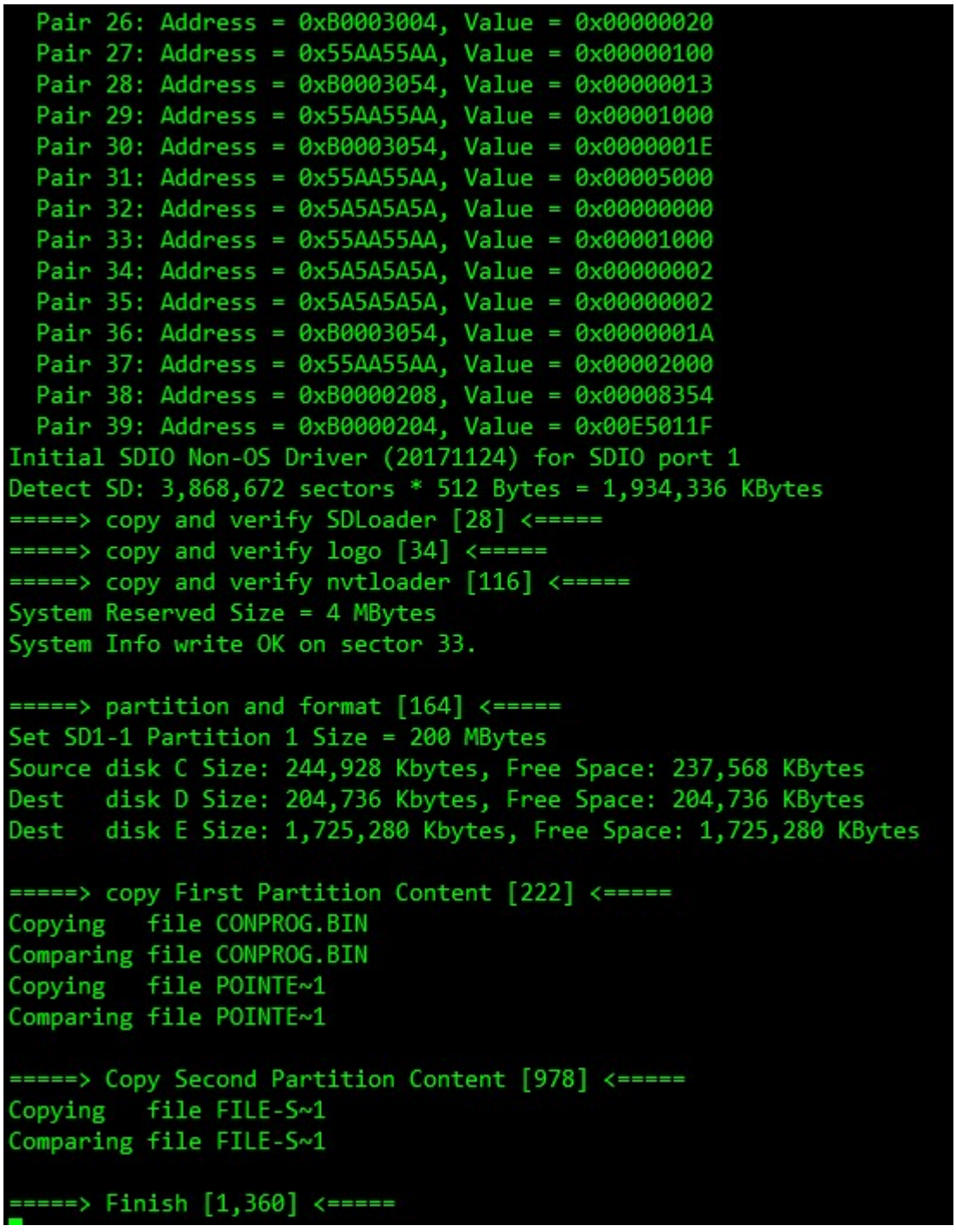

Figure 4-5 The SD Writer Output on UART Port

The output on panel can divide into several parts:

- Version Number: Display the SD 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 SD card Information: Display current SD card information in the format "SD1: *Total_Sector_Number* (Sec) * *Sector_Size* B = *Total_Size* MB".

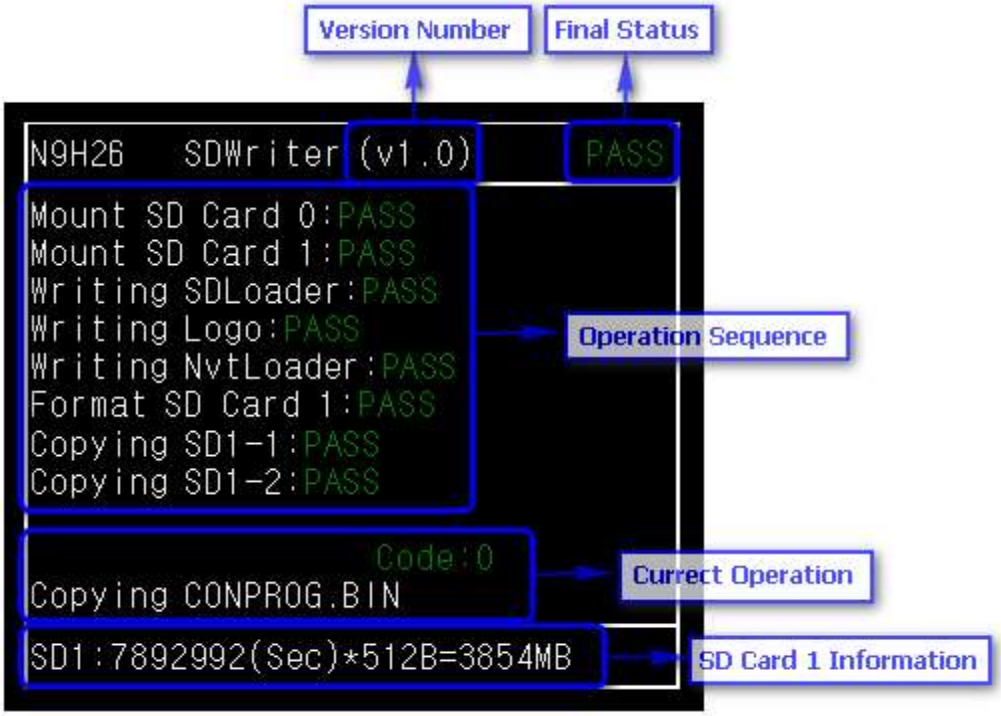

Figure 4-6 The SD Writer Output on Panel

5 Change Display Panel and Font

5.1 Display Panel and Font Configuration

The SD 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.

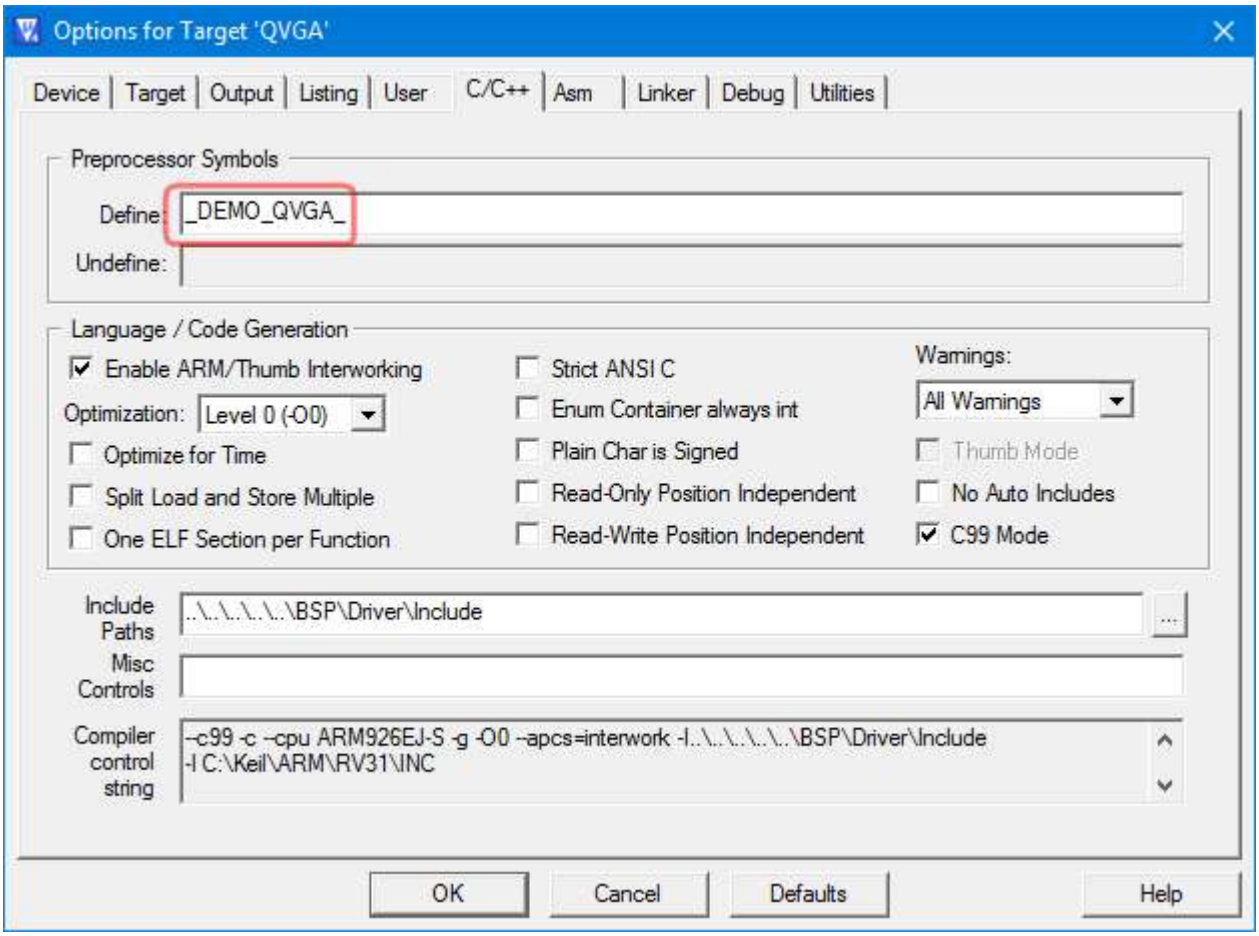

Figure 5-1 The Macro Definition about Font and Panel Resolution

5.2 Display Panel and Font Driver

The SD writer project includes the Font library and the VPOST library to support different LCD resolution. For system connected to panel that SD 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 SD 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.

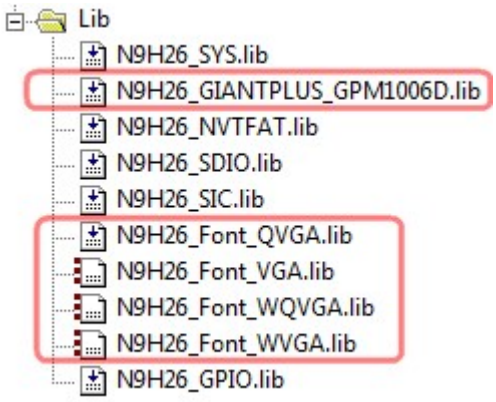

Figure 5-2 The Display Panel and Font Library in Project

6 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.

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.