

## N9H26 emWin Quick Start Guide

### Document Information

|                 |  |
|-----------------|--|
| <b>Abstract</b> | Introduce the steps to build and launch emWin for the N9H26 series microprocessor (MPU). |
| <b>Apply to</b> | N9H26 series   |

*The information described in this document is the exclusive intellectual property of Nuvoton Technology Corporation and shall not be reproduced without permission from Nuvoton.*

*Nuvoton is providing this document only for reference purposes of N9H26 microprocessor based system design.  
Nuvoton assumes no responsibility for errors or omissions.*

*All data and specifications are subject to change without notice.*

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

[www.nuvoton.com](http://www.nuvoton.com)

## Table of Contents

|          |  |           |
|----------|--|-----------|
| <b>1</b> | <b>INTRODUCTION .....</b>                                  | <b>3</b>  |
| <b>2</b> | <b>EMWIN INSTALLATION AND BSP DIRECTORY STRUCTURE.....</b> | <b>4</b>  |
| 2.1      | emWin Installation .....                                   | 4         |
| 2.2      | BSP\SampleCode\emWin .....                                 | 4         |
| 2.3      | BSP\ThirdParty\emWin\Config .....                          | 4         |
| 2.4      | BSP\ThirdParty\emWin\Doc .....                             | 4         |
| 2.5      | BSP\ThirdParty\emWin\Include .....                         | 5         |
| 2.6      | BSP\ThirdParty\emWin\Lib.....                              | 5         |
| 2.7      | BSP\ThirdParty\emWin\Tool .....                            | 5         |
| <b>3</b> | <b>EMWIN SAMPLE CODE .....</b>                             | <b>6</b>  |
| 3.1      | Development Environment.....                               | 6         |
| 3.2      | Project Structure .....                                    | 6         |
| 3.3      | System Initialization .....                                | 8         |
| 3.4      | emWin Initialization .....                                 | 10        |
| 3.5      | Build emWin Project.....                                   | 10        |
| 3.6      | Download and Run .....                                     | 10        |
| 3.7      | Touch Screen .....   | 12        |
| <b>4</b> | <b>EMWIN GUIBUILDER .....</b>                              | <b>15</b> |
| 4.1      | Create Widget .....  | 15        |
| 4.2      | Handle Widget Event .....                                  | 15        |
| <b>5</b> | <b>CHANGE DISPLAY PANEL.....</b>                           | <b>19</b> |
| 5.1      | emWin Display Configuration.....                           | 19        |
| 5.2      | Display Driver .....                                       | 19        |
| <b>6</b> | <b>SUPPORTING RESOURCES .....</b>                          | <b>20</b> |

## **1 Introduction**

emWin is a graphic library with graphical user interface (GUI) designed to provide an efficient, processor and display controller-independent GUI for any application that operates with a graphical display.

Nuvoton provides emWin GUI library for free with the N9H26 series microprocessor (MPU) that supports up to 800x480 (24 bpp) resolution depending on frame rate. The emWin platform can be implemented on HMI for industrial, machines, appliance applications, etc.

## 2 emWin Installation and BSP Directory Structure

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

### 2.1 emWin Installation

First, create a working folder and unzip “N9H26\_emWin\_NonOS.zip” into this folder. Then, create a new directory called “emWin” under BSP path “BSP\SampleCode\”. Then, move “GUIDemo” and “SimpleDemo” from working folder into “BSP\SampleCode\emWin\” respectively. Finally, move “emWin” from working folder into “BSP\ThirdParty\”.

### 2.2 BSP\SampleCode\emWin

|                   |   |
|-------------------|---|
| <b>GUIDemo</b>    | Utilize emWin library to demonstrate widgets feature.     |
| <b>SimpleDemo</b> | Utilize emWin library to demonstrate interactive feature. |

### 2.3 BSP\ThirdParty\emWin\Config

|                  |  |
|------------------|--|
| <b>GUI_X.c</b>   | Configuration and system dependent code for GUI. |
| <b>GUIConf.c</b> | emWin heap memory initialization.                |
| <b>GUIConf.h</b> | A header file configures emWin features.         |
| <b>LCDConf.c</b> | Display controller configuration source code.    |
| <b>LCDConf.h</b> | Display driver configuration header file.        |

### 2.4 BSP\ThirdParty\emWin\Doc

|   |   |
|---|---|
| <b>AN03002_Custom_Widget_Type.pdf</b>   | emWin custom widget type creation guide.  |
| <b>UM03001_emWin.pdf</b>                | emWin user guide and reference manual.  |
| <b>UM_Font_Architect_EN_Rev1.02.pdf</b> | Nuvoton font tool “FontArchitect.exe” user guide and reference manual in English. |
| <b>UM_Font_Architect_TC_Rev1.02.pdf</b> | Nuvoton font tool “FontArchitect.exe” user guide and reference manual in Chinese. |
| <b>Changelog.pdf</b>                    | Introduce N9H26 emWin HMI change log.   |

|                     |                          |
|---------------------|--------------------------|
| <b>Release.html</b> | Release notes for emWin. |
|---------------------|--------------------------|

## 2.5 BSP\ThirdParty\emWin\Include

This directory contains header files for emWin project.

## 2.6 BSP\ThirdParty\emWin\Lib

|                              |  |
|------------------------------|--|
| <b>NUemWin_ARM9_Keil.lib</b> | emWin library for N9H26 series MPU.                                      |
| <b>libNUemWin_ARM9_GNU.a</b> | emWin library for N9H26 series MPU. Note: for non-OS GCC toolchain ONLY. |

## 2.7 BSP\ThirdParty\emWin\Tool

|                          |  |
|--------------------------|--|
| <b>BmpCvtNuvoton.exe</b> | The Bitmap Converter is designed for converting common image file formats like BMP, PNG or GIF into the desired emWin bitmap format. |
| <b>emWinPlayer.exe</b>   | This tool can show the previously created emWin Movie File (EMF) on a Computer with a Windows operating system.                      |
| <b>FontArchitect.exe</b> | A Nuvoton tool for creating emWin bitmap font format.  |
| <b>GUIBuilder.exe</b>    | A tool for creating dialogs by drag and drop operation.  |
| <b>JPEG2Movie.exe</b>    | A tool to convert JPEG files to an EMF file.   |

### 3 emWin Sample Code

There are two emWin sample codes in the *N9H26 BSP\SampleCode\emWin* directory:

- **GUIDemo**: utilizes the emWin library to demonstrate widgets feature;
- **SimpleDemo**: utilizes the emWin library to demonstrate interactive feature.

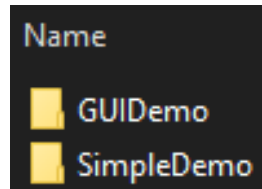


Figure 3-1 BSP emWin Sample Name

#### 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 |
| <b>Device Support</b>                           |  |   |                            |                                      |
| Arm Cortex-M0/M0+/M3/M4/M7                      | ✓  | ✓   | ✓                          | ✓                                    |
| Arm Cortex-M23/M33<br>Non-secure only           | ✓  | ✓   | ✓                          | ✗                                    |
| Arm Cortex-M23/M33<br>Secure and non-secure     | ✓  | ✓   | ✗                          | ✗                                    |
| Armv8-M Architecture Models including FastModel | ✓  | ✗   | ✗                          | ✗                                    |
| Arm SecurCore®                                  | ✓  | ✓   | ✗                          | ✗                                    |
| Arm7™, Arm9™, Arm Cortex-R4                     | ✓  | ✓   | ✗                          | ✗                                    |

Figure 3-2 Keil MDK License Chart

#### 3.2 Project Structure

The following uses SimpleDemo as a sample to explain the emWin project structure in BSP. This sample contains a frame window, four buttons, a text and a text editor. User can update the number shown in the text field by clicking four buttons shown on the display panel.

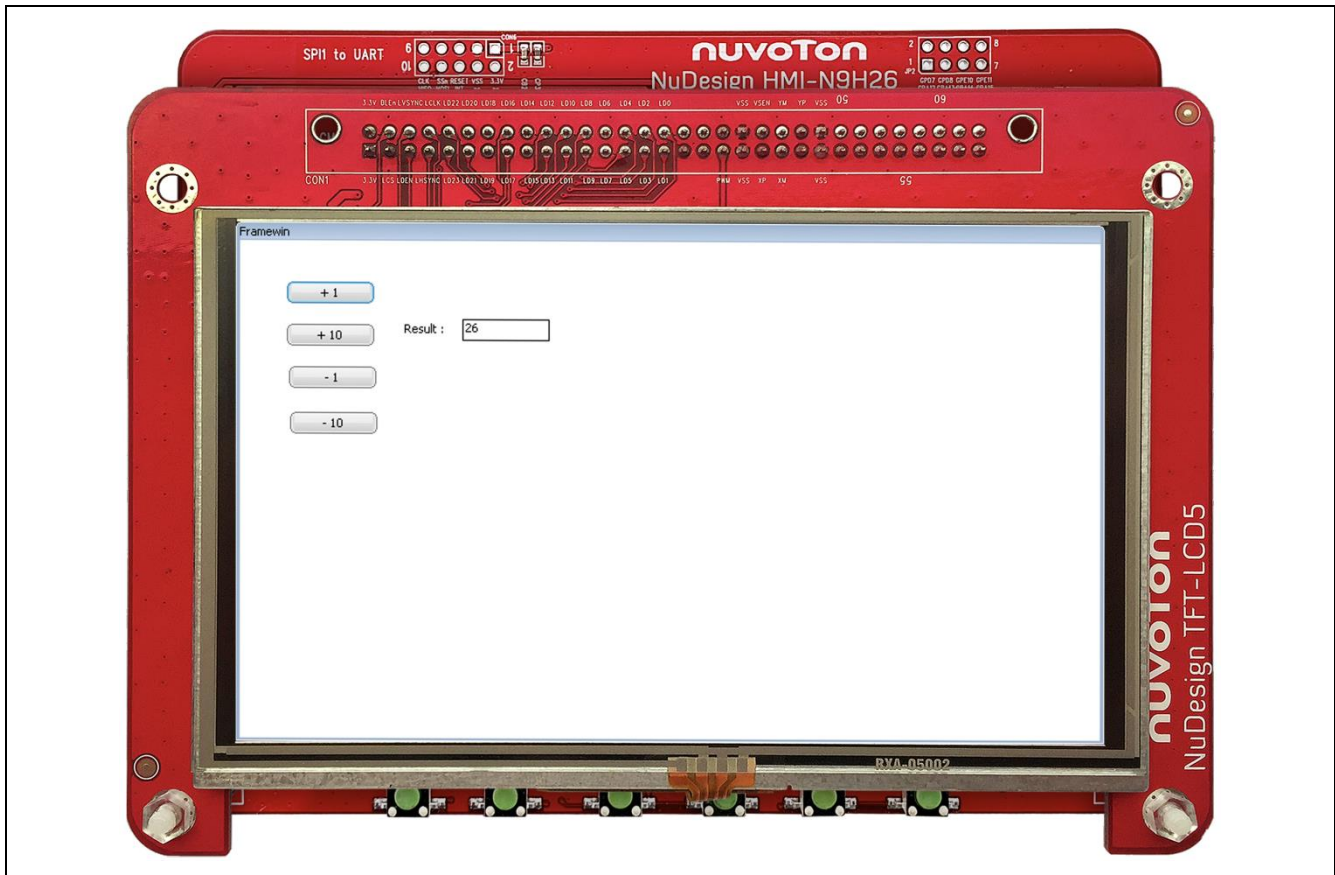


Figure 3-3 emWin SimpleDemo on NuMaker Board

The project structure is shown in the following figure. The project contains one target:

- **SimpleDemo\_N9H26K6\_SD\_800x480.bin:** Uses 800x480 16bpp LCD panel and stores touch screen calibration parameters in a SD card.

The Libraries group contains low level driver and system startup code. The emWin group contains emWin library and panel configuration for the N9H26. The emWin library will use BitBlt and JPEG codec to improve the graphic performance. Thus, the project file must include *N9H26\_BLT.lib* and *N9H26\_JPEG.lib*. The Application group contains the C code generated by emWin GUIBuilder. The tslib group is the touch screen library. The *N9H26\_NVTFAT.lib* contains the file system library to access a SD card. The Src group contains the *main.c* file.

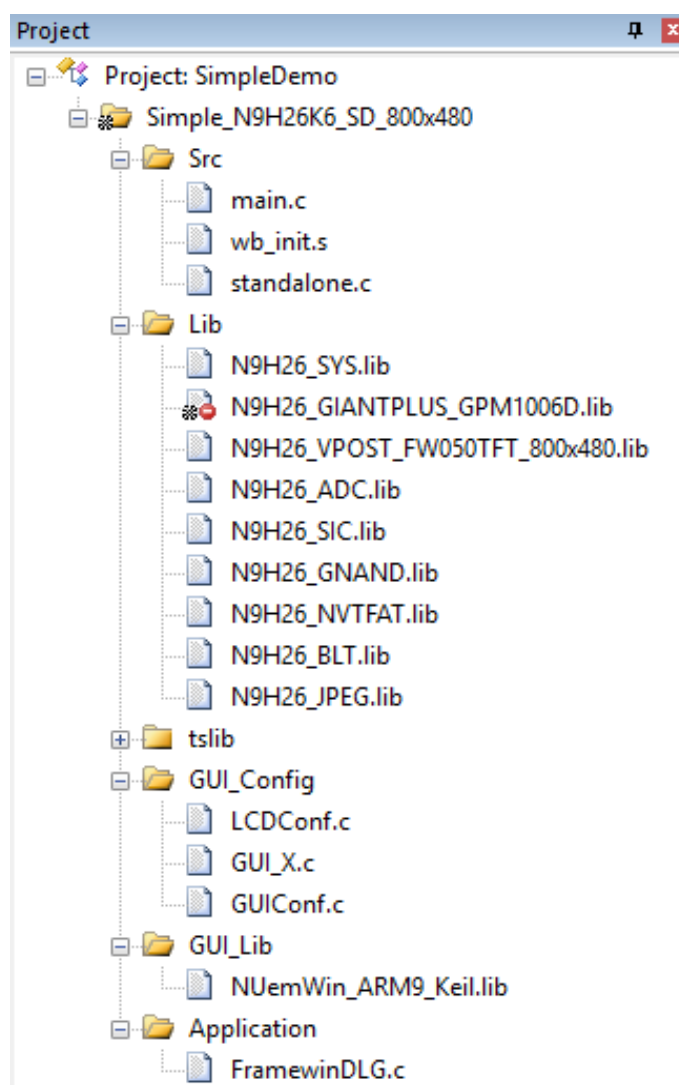


Figure 3-4 emWin SimpleDemo Project Tree on Keil MDK

### 3.3 System Initialization

The system initialization code is located in main function, including peripheral clock preparation, cache, LCD interface, touch screen interface and UART debug port setting. Also, a 1000Hz timer is configured to keep track of time elapsed.

```
int main(void)
{
    char szFileName[20];
    char szCalibrationFile[40];
    int hFile;

    #if GUI_SUPPORT_TOUCH
        g_enable_Touch = 0;
    #endif
}
```



```
#endif

// LCD interface & timer
_SYS_Init();

#if GUI_SUPPORT_TOUCH
Init_TouchPanel();

sprintf(szFileName, "C:\\ts_calib");
fsAsciiToUnicode(szFileName, szCalibrationFile, TRUE);
hFile = fsOpenFile(szCalibrationFile, szFileName, O_RDONLY | O_FSEEK);
sysprintf("file = %d\\n", hFile);
if (hFile < 0)
{
    // file does not exist, so do calibration
    hFile = fsOpenFile(szCalibrationFile, szFileName, O_CREATE|O_RDWR | O_FSEEK);
    if ( hFile < 0 )
    {
        sysprintf("CANNOT create the calibration file\\n");
        return -1;
    }
    GUI_Init();
    ts_calibrate(LCD_XSIZE, LCD_YSIZE);
    ts_writefile(hFile);
}
else
{
    ts_readfile(hFile);
}
fsCloseFile(hFile);

#endifdef STORAGE_SD
GNAND_UnMountNandDisk(&ptNDisk);
sicClose();
#endif

g_enable_Touch = 1;
#endif

MainTask();
```

```
return 0;
}
```

### 3.4 emWin Initialization

To initialize emWin GUI, the application needs to call GUI\_Init() and CreatFramewin() function. GUI\_Init() is called in main() and CreatFramewin() is called in MainTask() in *main.c*.

```
void MainTask(void)
{
    GUI_Init();

    CreateFramewin();

    while (1)
    {
        GUI_Delay(500);
    }
}
```

### 3.5 Build emWin Project

To build the emWin project in Keil MDK, click the **Rebuild** icon as shown below or press **F7** function key.

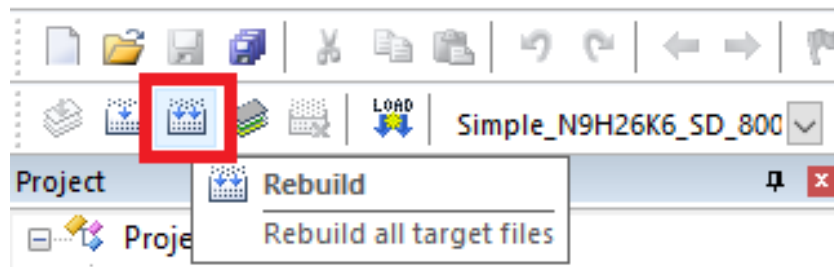


Figure 3-5 Shortcut Icon to Rebuild emWin Sample on Keil MDK

### 3.6 Download and Run

Please download the newly built image or pre-built image under *BSP/SampleCode/emWin/SimpleDemo/Bin* directory to DDR by TurboWriter, or download the newly built image by ICE. Nuvoton provides TurboWriter tool for downloading firmware to DDR, SPI Flash, NAND Flash, or SD card. To download images by TurboWriter, connect the N9H26 NuDesign board with PC via an USB cable and the execute TurboWriter. Further information can be found at *N9H26\_emWin\_NonOS-master/Tools/PC\_Tools/TurboWriter Tool User Guide.pdf*.

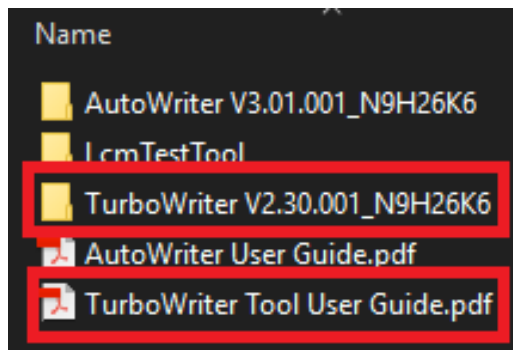


Figure 3-6 Nuvoton Windows Tool TurboWriter

Choose the type as DDR/SDRAM. Select the emWin sample binary image, set download and run address to 0x0, and then click the **Download** button. For more information, please refer to N9H26 TurboWriter User Manual under BSP's Documents/ directory.

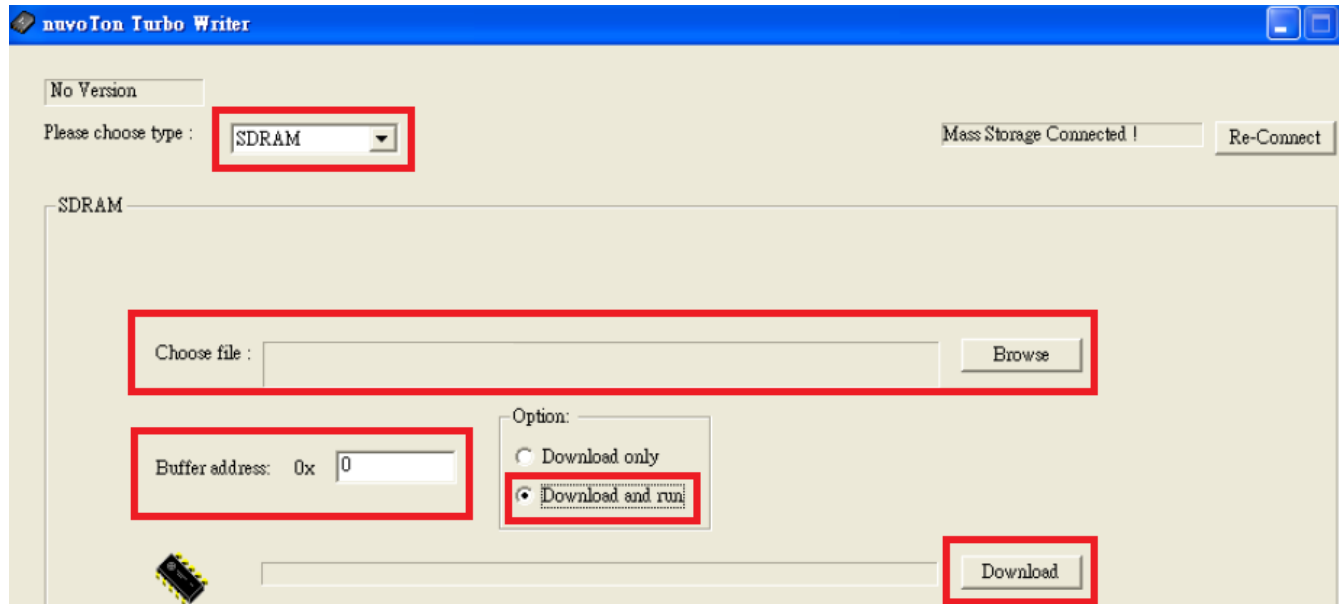


Figure 3-7 Nuvoton Windows Tool TruboWriter Main Page

The N9H26 JTAG interface configuration is shown below:

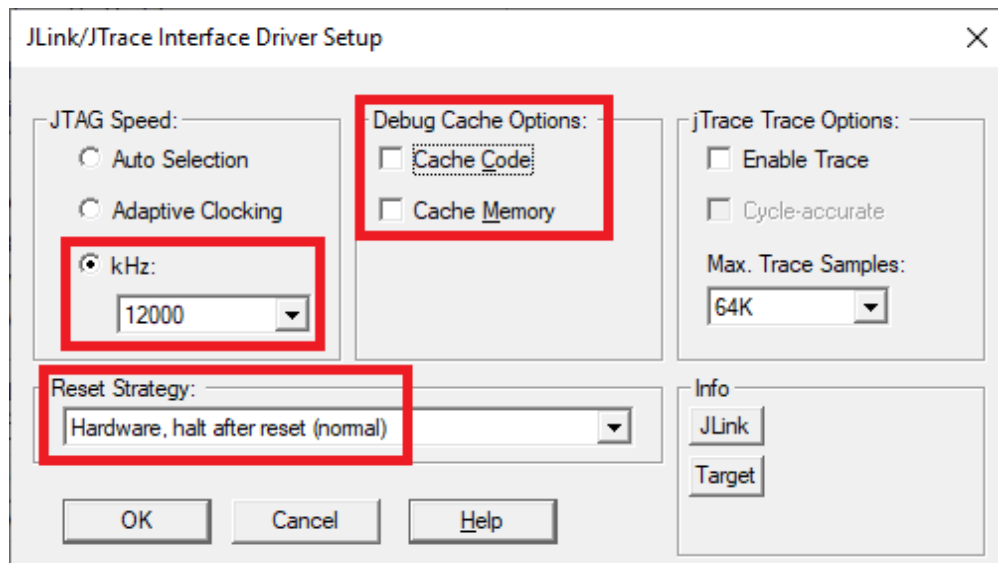


Figure 3-8 JLink Setting on Keil MDK

Press **Ctrl + F5** to download the application and start a debug session or click **Start/stop debug session** icon as shown below.



Figure 3-9 Shortcut Icon to Download Binary to Device and Start/Stop Debug Session

After entering debug session, press **F5** to start code execution.

### 3.7 Touch Screen

To support resistive touch screen, use ADC to convert the voltage of X axis and Y axis, and then map the ADC conversion result into the coordination. The conversion result can be affected by power noise, mechanical misalignment, etc. To overcome this issue, the tslib supports calibration function, and the calibration parameter is stored either in an on board NAND Flash or a SD card.

As mentioned in section 3.2, there are two main targets in this project: N9H26K6\_SD\_800x480 uses the calibration parameter stored in a SD card called ts\_calib. User can switch between different targets using the pull down menu marked in the red rectangle shown below.

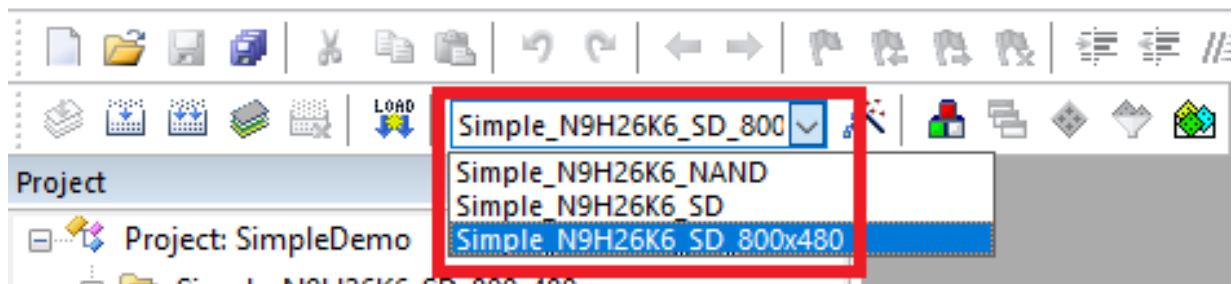


Figure 3-10 emWin SimpleDemo Targets on Keil MDK

A preprocessor symbol `STORAGE_SD` is defined to build the sample using calibration parameter stored in a SD card, and use preprocessor symbol `__800x480__` and target `_800x480` to select the support panel.

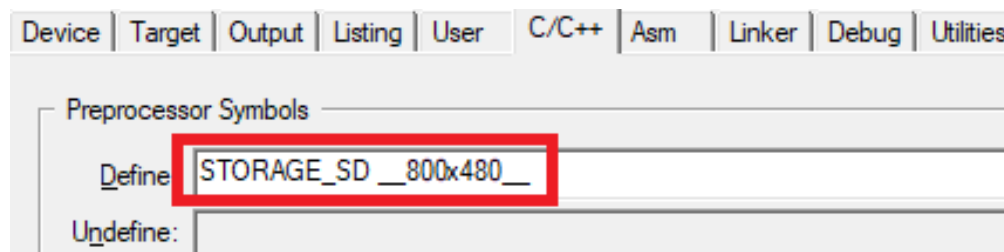


Figure 3-11 Define to Utilize SD Card and 800x480 LCD Panel

The touch screen resolution is defined in *N9H26TouchPanel.h*.

```
#ifndef __N9H26TOUCHPANEL_H__
#define __N9H26TOUCHPANEL_H__

#ifdef __800x480__
#define XSIZE_PHYS 800
#define YSIZE_PHYS 480
#else
#define XSIZE_PHYS 320
#define YSIZE_PHYS 240
#endif

#define LCD_XSIZE XSIZE_PHYS
#define LCD_YSIZE YSIZE_PHYS

#endif
```

If a SD card is used to store calibration parameters, main function will load the parameter file `ts_calib` from the SD card root directory. If the parameter doesn't exist, main function will call `ts_calibrate()` to generate a copy. This sample uses *N9H26\_NVTFAT.lib* to access FAT file

system.

```
#if GUI_SUPPORT_TOUCH
    g_enable_Touch = 0;
#endif

    // SD controller
    _SYS_Init();
#if GUI_SUPPORT_TOUCH
    Init_TouchPanel();

    sprintf(szFileName, "C:\\ts_calib");
    fsAsciiToUnicode(szFileName, szCalibrationFile, TRUE);
    hFile = fsOpenFile(szCalibrationFile, szFileName, O_RDONLY | O_FSEEK);
    sysprintf("file = %d\\n", hFile);
    if (hFile < 0)
    {
        // file does not exist, so do calibration
        hFile = fsOpenFile(szCalibrationFile, szFileName, O_CREATE|O_RDWR | O_FSEEK);
        if ( hFile < 0 )
        {
            sysprintf("CANNOT create the calibration file\\n");
            return -1;
        }
        GUI_Init();
        ts_calibrate(LCD_XSIZE, LCD_YSIZE);
        ts_writefile(hFile);
    }
    else
    {
        ts_readfile(hFile);
    }
    fsCloseFile(hFile);
#endif
    GNAND_UnMountNandDisk(&ptNDisk);
    sicClose();
#endif
    g_enable_Touch = 1;
#endif
```

## 4 emWin GUIBuilder

### 4.1 Create Widget

Segger provides a Windows tool GUIBuilder to create application with drag and drop interface. The tool is located under the *BSP\ThirdParty\emWin\Tool\* directory. This tool can generate a file named *FramewinDLG.c* for the widget of target application. Please refer to *GUI Builder chapter of UM03001\_emWin.pdf* for the usage of GUIBuilder.

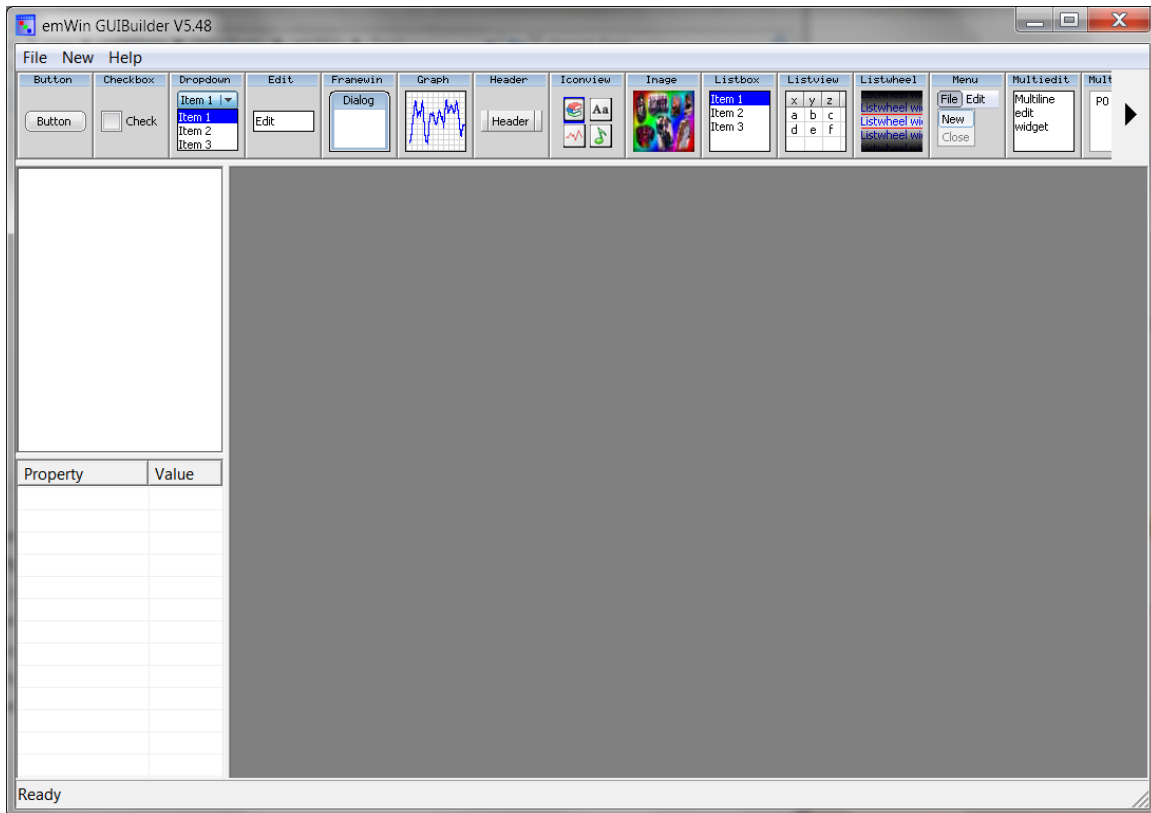


Figure 4-1 emWin Windows Tool GUIBuilder Main Page

### 4.2 Handle Widget Event

*FramewinDLG.c* is only the framework of widget and programmers still need to add their desired widget event handler in this file after copying the *FramewinDLG.c* file into the project directory. Below is the event handling code of SimpleDemo.

```
...
case WM_NOTIFY_PARENT:
    Id      = WM_GetId(pMsg->hWinSrc);
    NCode = pMsg->Data.v;
    switch(Id) {
        case ID_BUTTON_0: // Notifications sent by 'NVT_Button_+1'
```

```

switch(NCode) {
case WM_NOTIFICATION_CLICKED:
    // USER START (Optionally insert code for reacting on notification message)
    // USER END
    break;
case WM_NOTIFICATION_RELEASED:
    // USER START (Optionally insert code for reacting on notification message)
    value += 1;
    sprintf(sBuf,"%d  ", value);
    hItem = WM_GetDialogItem(pMsg->hWin, ID_EDIT_0);
    EDIT_SetText(hItem, sBuf);
    // USER END
    break;
    // USER START (Optionally insert additional code for further notification handling)
    // USER END
}
break;
case ID_BUTTON_1: // Notifications sent by 'NVT_Button_+10'
    switch(NCode) {
    case WM_NOTIFICATION_CLICKED:
        // USER START (Optionally insert code for reacting on notification message)
        // USER END
        break;
    case WM_NOTIFICATION_RELEASED:
        // USER START (Optionally insert code for reacting on notification message)
        value += 10;
        sprintf(sBuf,"%d  ", value);
        hItem = WM_GetDialogItem(pMsg->hWin, ID_EDIT_0);
        EDIT_SetText(hItem, sBuf);
        // USER END
        break;
        // USER START (Optionally insert additional code for further notification handling)
        // USER END
    }
    break;
case ID_BUTTON_2: // Notifications sent by 'NVT_Button_-1'
    switch(NCode) {
    case WM_NOTIFICATION_CLICKED:
        // USER START (Optionally insert code for reacting on notification message)
        // USER END

```



```

        break;
    case WM_NOTIFICATION_RELEASED:
        // USER START (Optionally insert code for reacting on notification message)
        value -= 1;
        sprintf(sBuf, "%d", value);
        hItem = WM_GetDialogItem(pMsg->hWin, ID_EDIT_0);
        EDIT_SetText(hItem, sBuf);
        // USER END
        break;
    // USER START (Optionally insert additional code for further notification handling)
    // USER END
}
break;
case ID_BUTTON_3: // Notifications sent by 'NVT_Button_-10'
    switch(NCode) {
    case WM_NOTIFICATION_CLICKED:
        // USER START (Optionally insert code for reacting on notification message)
        // USER END
        break;
    case WM_NOTIFICATION_RELEASED:
        // USER START (Optionally insert code for reacting on notification message)
        value -= 10;
        sprintf(sBuf, "%d", value);
        hItem = WM_GetDialogItem(pMsg->hWin, ID_EDIT_0);
        EDIT_SetText(hItem, sBuf);
        // USER END
        break;
    // USER START (Optionally insert additional code for further notification handling)
    // USER END
    }
    break;
case ID_EDIT_0: // Notifications sent by 'NVT_Edit'
    switch(NCode) {
    case WM_NOTIFICATION_CLICKED:
        // USER START (Optionally insert code for reacting on notification message)
        // USER END
        break;
    case WM_NOTIFICATION_RELEASED:
        // USER START (Optionally insert code for reacting on notification message)
        // USER END
    }

```

```

        break;
    case WM_NOTIFICATION_VALUE_CHANGED:
        // USER START (Optionally insert code for reacting on notification message)
        // USER END
        break;
        // USER START (Optionally insert additional code for further notification handling)
        // USER END
    }
    break;
    // USER START (Optionally insert additional code for further Ids)
    // USER END
    }
    break;
    // USER START (Optionally insert additional message handling)
    // USER END
default:
    WM_DefaultProc(pMsg);
    break;
}
...

```

## 5 Change Display Panel

### 5.1 emWin Display Configuration

emWin declares its display panel resolution in *LCDConf.h* and color depth in *LCDConf.c*. Both files can be found at *BSP\ThirdParty\emWin\Config* directory.

```
...
// Example for RGB565 in LCDConf.c
//
// Color conversion
//
#define COLOR_CONVERSION GUICC_M565
//
// Display driver
//
#define DISPLAY_DRIVER GUIDRV_LIN_16
...
```

### 5.2 Display Driver

The emWin project includes the lcd library to support different LCD resolution. For system connection with other panel, lcd library has to be enabled to the project.

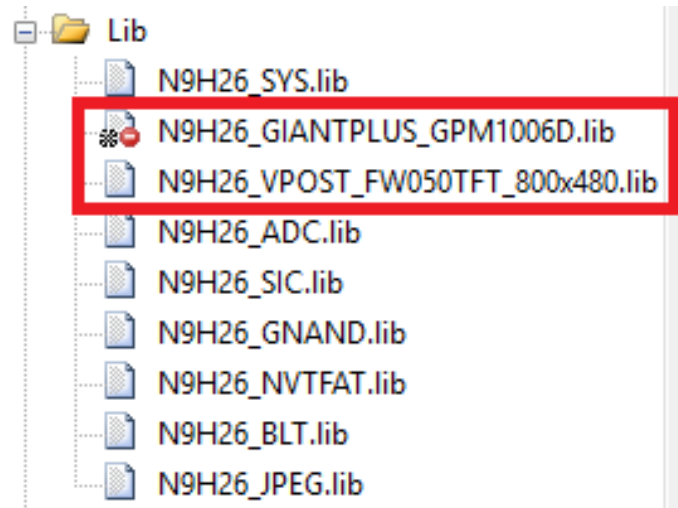


Figure 5-1 BSP VPOST Display Driver

## 6 Supporting Resources

Segger provides an emWin supporting forum. Questions regarding emWin usage are discussed at: <https://forum.segger.com/index.php/Board/12-emWin-related/>.

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

- ARM7/9 forum at: <http://forum.nuvoton.com/viewforum.php?f=12>.
- HMI/GUI forum at: <http://forum.nuvoton.com/viewforum.php?f=31>.

**Revision History**

| Date       | Revision | Description  |
|------------|----------|--|
| 2020.10.7  | 1.00     | 1. Initially issued.                                       |
| 2020.02.24 | 1.01     | 1. Add emWin installation and modify touch screen chapter. |

### **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.

---

*Please note that all data and specifications are subject to change without notice.  
All the trademarks of products and companies mentioned in this datasheet belong to their respective owners.*