N9H30 emWin Quick Start Guide

Document Information

Abstract	Introduce the steps to build and launch emWin for the N9H30 series microprocessor (MPU).
Apply to	N9H30 series

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

www.nuvoton.com

Table of Contents

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 N9H30 series microprocessor (MPU) supporting up to 800x480 (24 bpp) resolution. The emWin platform can be implemented on HMI for industrial, machines, appliances, etc.

2 emWin Installation and BSP Directory Structure

This chapter introduces emWin related files and directories in the N9H30 BSP.

2.1 emWin Installation

First, create a working folder and unzip "*N9H30_emWin_NonOS.zip*" into this folder. Then, move "*emWin_GUIDemo*" and "*emWin_SimpleDemo*" from working folder into "*BSP\SampleCode*" respectively. Finally, move "*emWin*" from working folder into "*BSP\ThirdParty*".

2.2 BSP\SampleCode

emWin_GUIDemo	Utilize emWin library to demonstrate widgets feature.
emWin_SimpleDemo	Utilize emWin library to demonstrate interactive feature.

2.3 BSP\ThirdParty\emWin\Config

GUI_X.c	Configuration and system dependent code for GUI.
GUIConf.c	Display controller initialization source code.
GUIConf.h	A header file configures emWins features, fonts, etc.
LCDConf.c	Display controller configuration source code.
LCDConf.h	Display driver configuration header file.

2.4 BSP\ThirdParty\emWin\Doc

AN03002_Custom_ Widget_Type.pdf	emWin custom widget type creation guide.
UM03001_emWin.pdf	emWin user guide and reference manual.
UM_Font_Architect_ EN_Rev1.02.pdf	Nuvoton font tool "*FontArchitect.exe*" user guide and reference manual in English.
UM_Font_Architect_ TC_Rev1.02.pdf	Nuvoton font tool "*FontArchitect.exe*" user guide and reference manual in Chinese.
Changelog.pdf	Introduce N9H20 emWin HMI change log.

Release.html	Release notes for emWin.

2.5 BSP\ThirdParty\emWin\Include

This directory contains header files for emWin project.

2.6 BSP\ThirdParty\emWin\Lib

NUemWin_ARM9_Keil.lib	emWin library for N9H30 series MPU.
libNUemWin_ARM9_GCC.a	emWin library for N9H30 series MPU. Note: for non-OS GCC toolchain ONLY.

2.7 BSP\ThirdParty\emWin\Tool

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

3 emWin Sample Code

There are two emWin sample code in the N9H30 BSP SampleCode directory:

- **emWin_GUIDemo**: utilizes the emWin library to demonstrate widgets feature;
- **emWin_SimpleDemo**: utilizes the emWin library to demonstrate interactive feature.

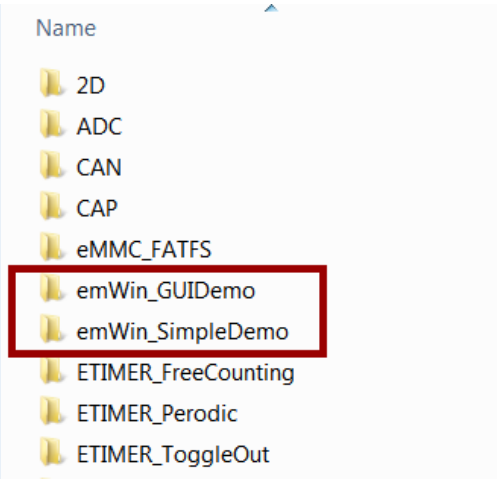

Figure 3-1 BSP emWin Sample Name

3.1 Development Environment

Keil IDE and Eclipse are used as Non-OS BSP development environment, which uses ULINK2 or J-Link ICE for debugging. This document uses Keil IDE to describe the project structure. To support ARM9, MDK Plus or Professional edition shall be used.

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 3-2 Keil MDK License Chart

3.2 Project Structure

The following uses emWin_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.

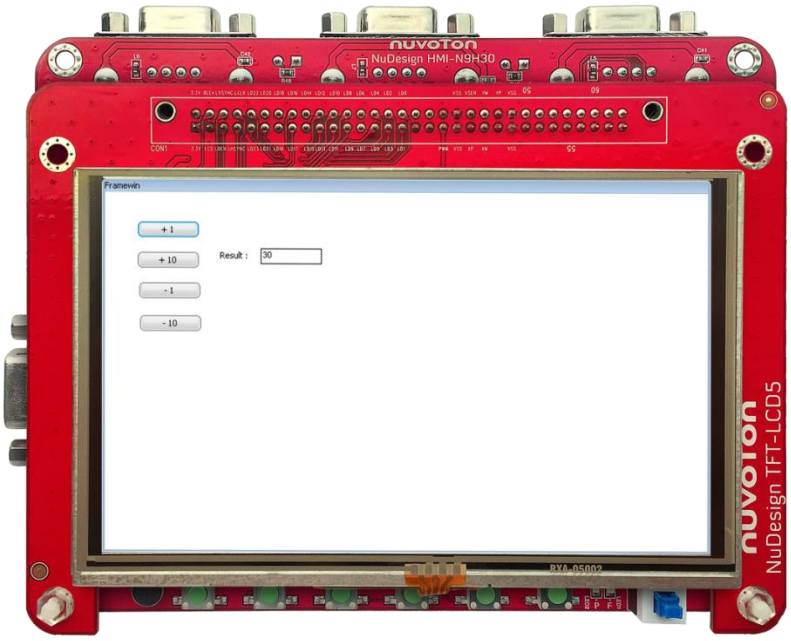

Figure 3-3 emWin SimpleDemo on NuMaker Board

The project structure is shown in the following figure. The project contains four targets:

- **emWin_SimpleDemo_E50A2V1_16BPP**: Uses 800x480 16bpp LCD panel and stores touch screen calibration parameters in SPI Flash.
- **emWin_SimpleDemo_E50A2V1_16BPP_SD**: Uses 800x480 16bpp LCD panel and stores touch screen calibration parameters in a SD card.
- **emWin_SimpleDemo_FW070TFT_24BPP**: Uses 800x480 24bpp LCD panel and stores touch screen calibration parameters in SPI Flash.
- **emWin_SimpleDemo_FW070TFT_24BPP_SD**: Uses 800x480 24bpp 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 N9H30. The emWin library will use 2D graphic engine and JPEG codec to improve the graphic performance. Thus, the project file must include *2d.c* and *jpegcodec.c*. The Application group contains the C code generated by emWin GUIBuilder. The tslib group is the touch screen library. The FatFs group contains the file system library to access the SD card. The src group contains the main file.

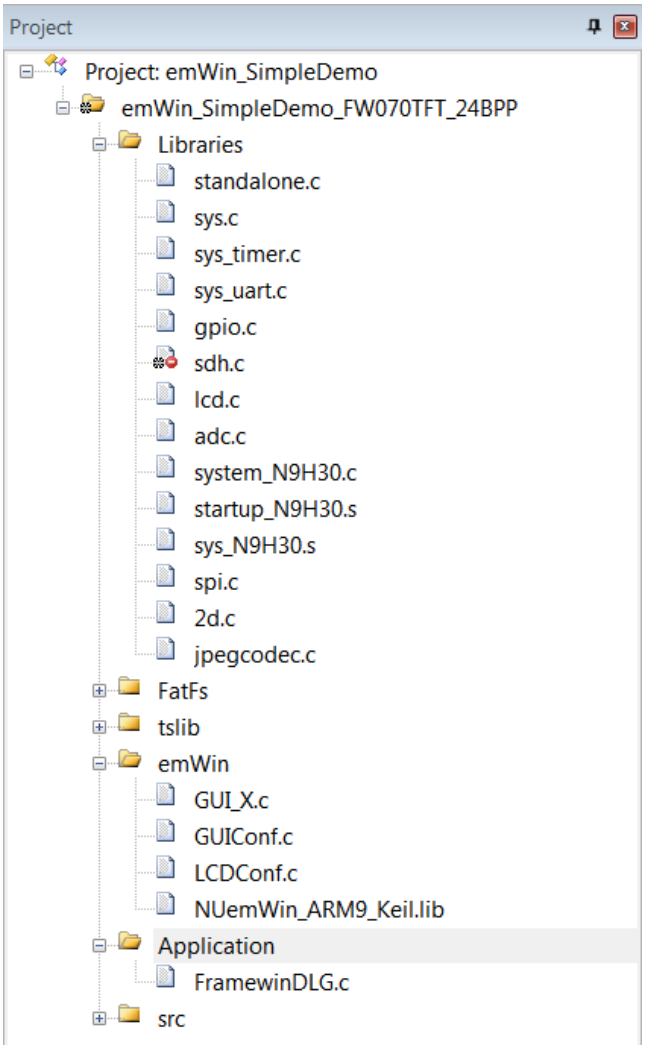

Figure 3-4 emWIn SImpleDemo Project Tree on Keil MDK

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)
{
#ifdef __USE_SD__
    FRESULT     res;
#else
    uint16_t u16ID;
#endif

    *(volatile unsigned int *)(CLK_BA+0x18) |= (1<<16); /* Enable UART0 clock */
```

```
    *(volatile unsigned int *)(CLK_BA+0x18) |= (1<<3);  /* Enable GPIO clock */
    /* Enable cache */
    sysDisableCache();
    sysFlushCache(I_D_CACHE);
    sysEnableCache(CACHE_WRITE_BACK);
    sysInitializeUART();

#ifdef GUI_SUPPORT_TOUCH
    g_enable_Touch = 0;
#endif

    OS_TimeMS = 0;

    /* Set TIMER0 to 1000 ticks per second and register callback function */
    sysSetTimerReferenceClock(TIMER0, 12000000);
    sysStartTimer(TIMER0, 1000, PERIODIC_MODE);
    sysSetTimerEvent(TIMER0, 1, (PVOID)TMR0_IRQHandler);

#ifdef __USE_SD__
    sysInstallISR(HIGH_LEVEL_SENSITIVE|IRQ_LEVEL_1, SDH_IRQn, (PVOID)SDH_IRQHandler);
    sysEnableInterrupt(SDH_IRQn);
#endif

    sysSetLocalInterrupt(ENABLE_IRQ);
    /* Initial LCD panel */
    LCD_initial();

#ifdef GUI_SUPPORT_TOUCH
    Init_TouchPanel();

#ifdef __USE_SD__
    SD_SetReferenceClock(300000);
    SD_Open_Disk(SD_PORT0 | CardDetect_From_GPIO);
    if (gCardInit)
    {
        gCardInit = 0;
        SD_Open_Disk(SD_PORT0 | CardDetect_From_GPIO);
    }

    if(!(SD_CardDetection(SD_Drv)))
        while(1);
```

```
    sysprintf("rc=%d\n", (WORD)disk_initialize(0));
    disk_read(0, Buff, 2, 1);
    f_mount(&FatFs[0], "", 0);  // for FATFS v0.11
#endif

    GUI_Init();

    ...
    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)
{
    WM_HWIN hWin;
    Char acVersion[40] = "Framewin: Version of emWin: ";

    hWin = CreateFramewin();
    strcat(acVersion, GUI_GetVersionString());
    FRAMEWIN_SetText(hWin, acVersion);
    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.

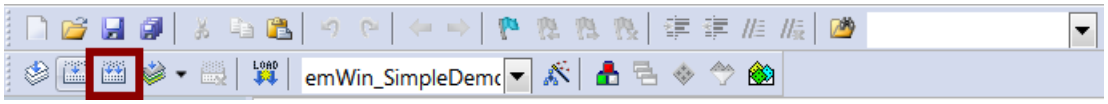

Figure 3-5 Shortcut Icon to Rebuild emWin Sample on Keil MDK

3.6 Download and Run

Users could download the newly built image or pre-built image under BSP *Image/emWin/* directory to DDR by NuWriter, or download the newly built image by ICE. Nuvoton provides NuWriter tool for downloading firmware to DDR, SPI Flash, NAND Flash, or eMMC. To download images by NuWriter, connect the N9H30 NuDesign board with PC via an USB cable and the execute NuWriter. Select the DDR parameter according to the MPU on board.

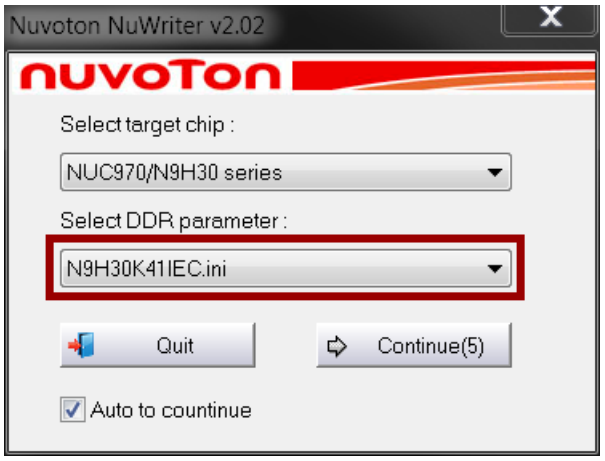

Figure 3-6 Nuvoton Windows Tool NuWriter INI Select Window

Choose the type as DDR/SRAM. 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 NUC970 N9H30 NuWriter User Manual under BSP's Documents directory.

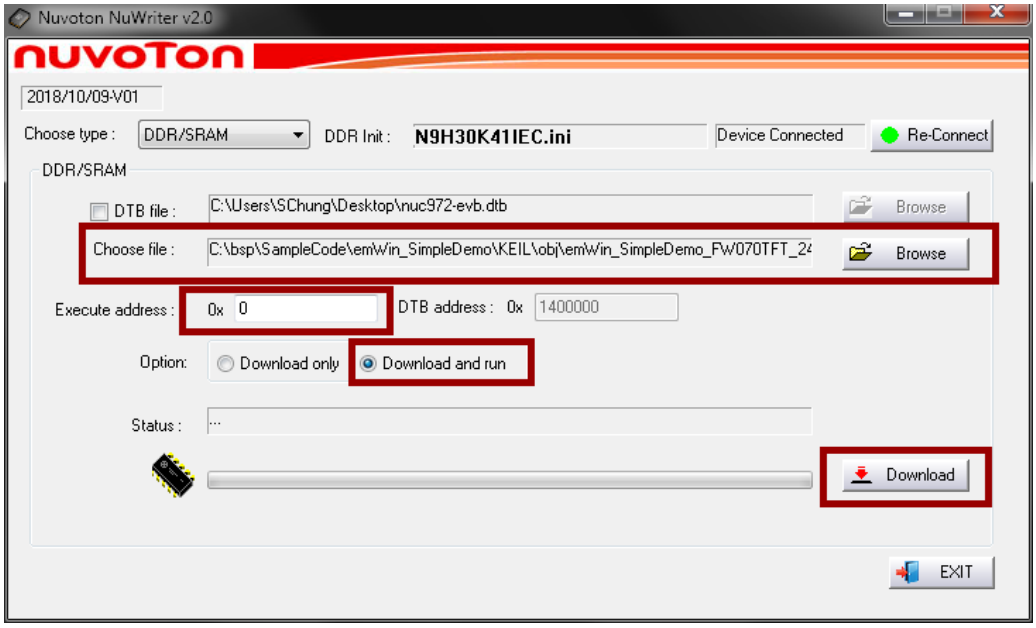

Figure 3-7 Nuvoton Windows Tool NuWriter Main Function

The N9H30 JTAG interface is disabled by default. To enable JTAG interface, please connect N9H30 with NuWriter and ICE reset must be disabled; otherwise the JTAG interface will be disabled immediately after reset. To disable ULINK2 reset, uncheck Use Reset at Startup under the Misc Options as shown below.

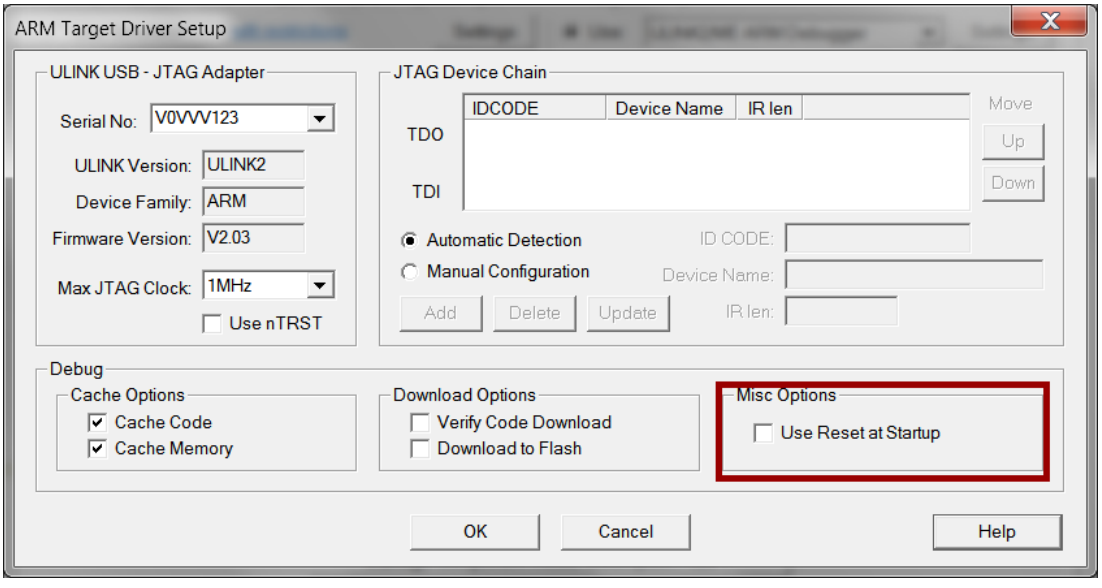

Figure 3-8 ULINK Setting on Keil MDK

To disable J-Link reset, set Reset Strategy to No Reset as shown below.

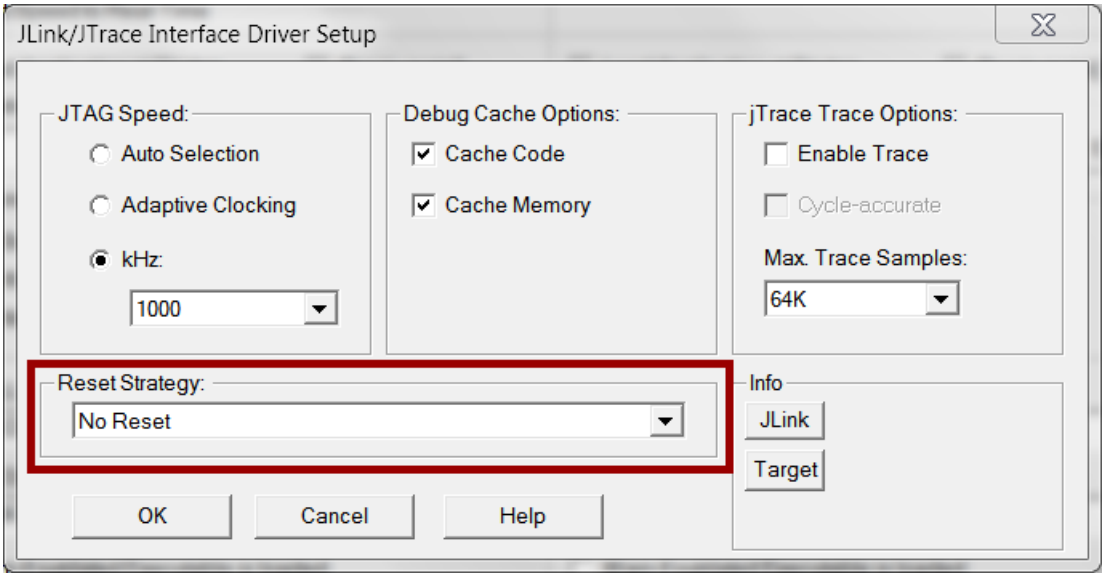

Figure 3-9 JLink Setting on Keil MDK

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

Figure 3-10 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 SPI Flash or a SD card.

As mentioned in section 3.1, there are four targets in this project: emWin_SimpleDemo_E50A2V1_16BPP and emWin_SimpleDemo_FW070TFT_24BPP use the calibration parameter in the on board SPI Flash offset 0x00180000, and the other targets use the calibration parameter results 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.

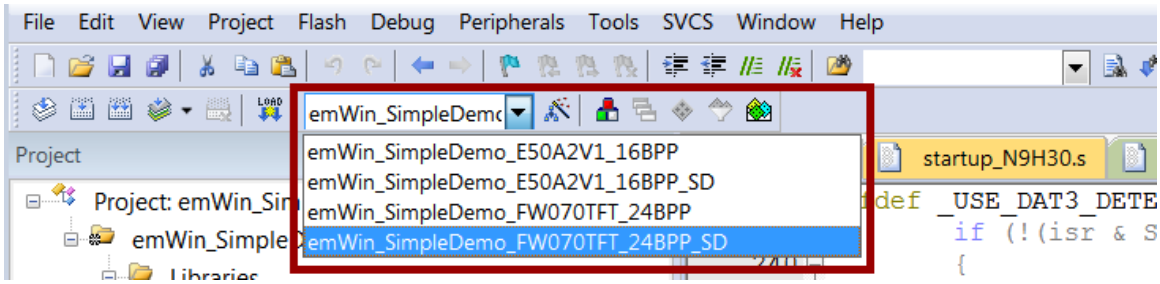

Figure 3-11 emWin SimpleDemo Targets on Keil MDK

A preprocessor symbol __USE_SD__ is defined to build the sample using calibration parameter stored in a SD card, and use two preprocessor symbols _PANEL_FW070TFT_24BPP_ and _PANEL_E50A2V1_16BPP_ to select the support panel.

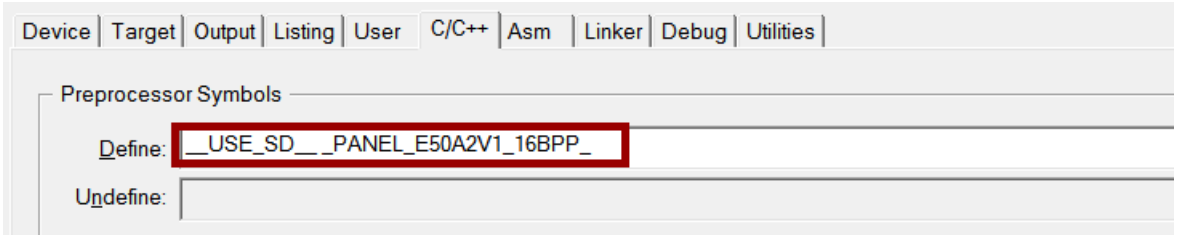

Figure 3-12 Define to Utilize SD Card and 800x480 LCD Panel

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

```
#ifndef __TOUCHPANEL_H__
#define __TOUCHPANEL_H__

#define __DEMO_TS_WIDTH__       800
#define __DEMO_TS_HEIGHT__      480
...
#endif
```

The SPI Flash offset store calibration parameter is defined in *spilib.h*.

```
#ifndef __SPILIB_H__
#define __SPILIB_H__

#define __DEMO_TSFILE_ADDR__    0x00180000 /* SPI flash 1.5MB offset */
...

#endif
```

If SPI Flash is used to store the calibration parameter, main function will load the parameter from SPI Flash. If the parameter doesn't exist, main function will call ts_calibrate() to generate a copy.

```
    _DemoSpiInit();

    // check flash id
    if((u16ID = SpiFlash_ReadMidDid()) == 0xEF17)
        sysprintf("Flash found: W25Q128BV ...\n");
    else
        sysprintf("Flash ID, 0x%x\n", u16ID);

    SpiFlash_NormalRead(__DEMO_TSFILE_ADDR__, DestArray);
    g_pu32Res = (uint32_t *)DestArray;
    sysprintf("%x\n", g_pu32Res[7]);
    if (g_pu32Res[7] != 0x55AAA55A)
    {
        ts_calibrate(XSIZE_PHYS, YSIZE_PHYS);
        sysprintf("Sector Erase ...");

        /* Sector erase SPI flash */
        SpiFlash_EraseSector(__DEMO_TSFILE_ADDR__);

        /* Wait ready */
        SpiFlash_WaitReady();

        ts_writefile();
        sysprintf("[OK]\n");
    }
    else
        ts_readfile();
```

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 FatFS to access FAT file system.

```
    SD_SetReferenceClock(300000);
    SD_Open_Disk(SD_PORT0 | CardDetect_From_GPIO);
    if (gCardInit)
    {
        gCardInit = 0;
        SD_Open_Disk(SD_PORT0 | CardDetect_From_GPIO);
    }

    if(!(SD_CardDetection(SD_Drv)))
        while(1);
    sysprintf("rc=%d\n", (WORD)disk_initialize(0));
    disk_read(0, Buff, 2, 1);
    f_mount(&FatFs[0], "", 0);  // for FATFS v0.11

    res = f_open(&hFile, "0:\\ts_calib", FA_OPEN_EXISTING | FA_READ);
    if (res)
    {
        // file does not exists, so do calibration
        res = f_open(&hFile, "0:\\ts_calib", FA_CREATE_ALWAYS | FA_WRITE);
        if ( res )
        {
            f_close(&hFile);
            GUI_DispStringAt("CANNOT create the calibration file.\nPlease insert a SD card
then reboot.", 0, 0);
            while(1);
        }

        ts_calibrate(__DEMO_TS_WIDTH__, __DEMO_TS_HEIGHT__);
        ts_writefile();
    }
    else
    {
        ts_readfile();
    }
    f_close(&hFile);
```

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 *ThirdParty\emWin\Tool* directory. This tool can generate a file named *FramewinDLG.c* for the widget of target application. Please refer to chapter 20 of *UM03001_emWin5.pdf* for the usage of GUIBuilder.

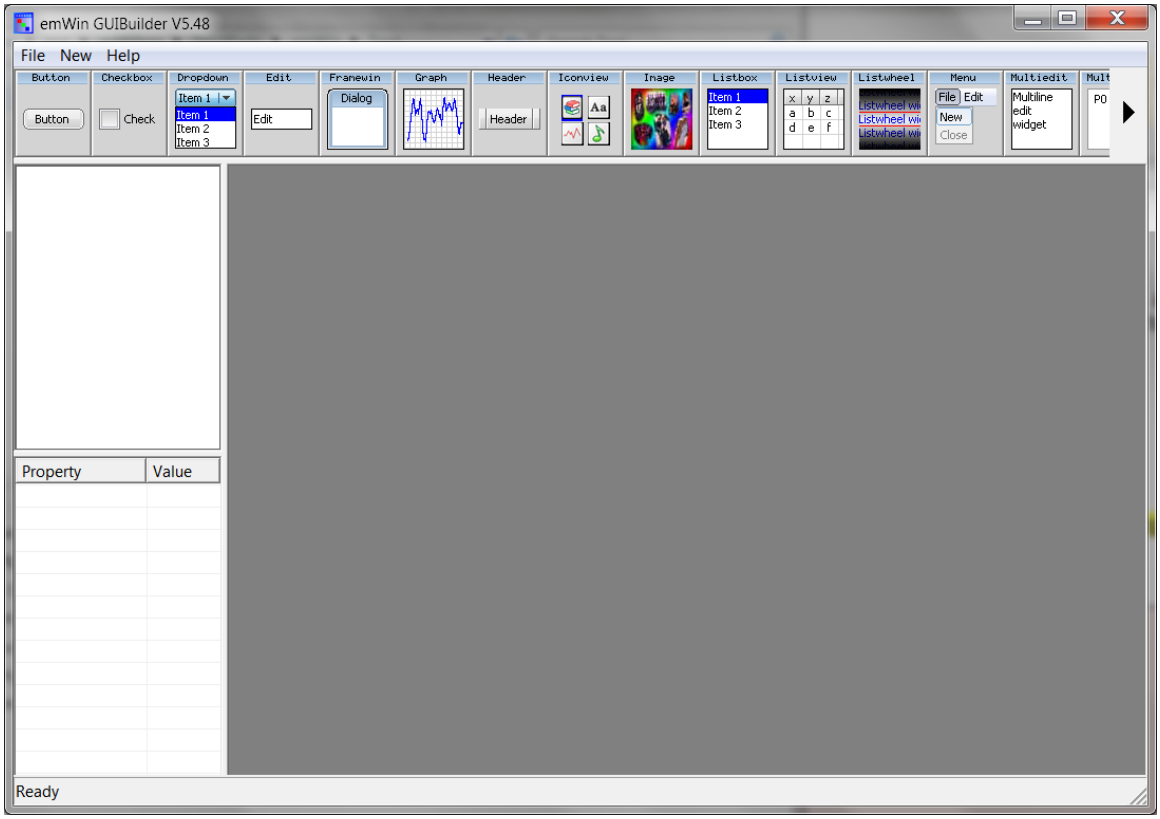

Figure 4-1 emWin Windows Tool GUIBuilder Main Page

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 emWin_SimpleDemo.

```
......
switch (pMsg->MsgId)
{
case WM_INIT_DIALOG:
    //
    // Initialization of 'Edit'
```

```
    //
    value = 123;
    sprintf(sBuf,"%d    ", value);
    hItem = WM_GetDialogItem(pMsg->hWin, ID_EDIT_0);
    EDIT_SetText(hItem, sBuf);

    // USER START (Optionally insert additional code for further widget initialization)
    // USER END
    break;
case WM_NOTIFY_PARENT:
    Id    = WM_GetId(pMsg->hWinSrc);
    NCode = pMsg->Data.v;
    switch(Id)
    {
    case ID_BUTTON_0: // Notifications sent by '+ 1'
        switch(NCode)
        {
        case WM_NOTIFICATION_CLICKED:
            // USER START (Optionally insert code for reacting on notification message)
            // USER END
             value += 1;
             sprintf(sBuf,"%d    ", value);
             hItem = WM_GetDialogItem(pMsg->hWin, ID_EDIT_0);
             EDIT_SetText(hItem, sBuf);
             break;
        case WM_NOTIFICATION_RELEASED:
             // 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;
......
```

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 *ThirdParty\emWin\Config* directory.

```
......
// Color depth
#ifdef _PANEL_E50A2V1_16BPP_
#define COLOR_CONVERSION GUICC_M565
#endif
#ifdef _PANEL_FW070TFT_24BPP_
#define COLOR_CONVERSION GUICC_M888
#endif

// Display resolution
#define XSIZE_PHYS 800
#define YSIZE_PHYS 480

#define LCD_XSIZE     XSIZE_PHYS
#define LCD_YSIZE     YSIZE_PHYS
```

5.2 Display Driver

The emWin project includes the *lcd.c* to support E50A2V1 and FW070TFT LCD panels. For system connection with other panel, *lcd.c* has to be updated to add the LCD controller register setting according to the panel datasheet.

```
......

/* LCD attributes */
static VPOST_T DEF_E50A2V1 = {
    800,                            /*!< Panel width */
    480,                            /*!< Panel height */
    0,                              /*!< MPU command line low indicator */
    0,                              /*!< MPU command width */
    0,                              /*!< MPU bus width */
    VPOSTB_DATA16or18,              /*!< Display bus width */
    0,                              /*!< MPU mode */
    VPOSTB_COLORTYPE_64K,           /*!< Display colors */
    VPOSTB_DEVICE_SYNC_HIGHCOLOR,   /*!< Type of display panel */
```

```
    0x020d03a0,                     /*!< CRTCSIZE register value */
    0x01e00320,                     /*!< CRTCDEND register value */
    0x03250321,                     /*!< CRTCHR register value */
    0x03780348,                     /*!< CRTCHSYNC register value */
    0x01f001ed                      /*!< CRTCVR register value */
};
......
static VPOST_T DEF_FW070TFT = {
    800,                            /*!< Panel width */
    480,                            /*!< Panel height */
    0,                              /*!< MPU command line low indicator */
    0,                              /*!< MPU command width */
    0,                              /*!< MPU bus width */
    VPOSTB_DATA16or18,              /*!< Display bus width */
    0,                              /*!< MPU mode */
    VPOSTB_COLORTYPE_16M,           /*!< Display colors */
    VPOSTB_DEVICE_SYNC_HIGHCOLOR,   /*!< Type of display panel */
    0x020d0420,                     /*!< CRTCSIZE register value */
    0x01e00320,                     /*!< CRTCDEND register value */
    0x033e0339,                     /*!< CRTCHR register value */
    0x040c03f8,                     /*!< CRTCHSYNC register value */
    0x020001f6                      /*!< CRTCVR register value */
};

/* LCD build-in support list */
static VPOST_T* DisplayDevList[4] = {&DEF_E50A2V1, &DEF_ILI9341_MPU80,
&DEF_LSA40AT9001,&DEF_FW070TFT};
......
```

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 N9H30 system related issues can be posted in Nuvoton's

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
2018.10.12	1.00	1. Initially issued.
2018.12.20	1.01	1. Add IDE and forum description.
2019.06.10	1.02	1. Add Font Tool description
2020.05.21	1.03	1. Update chapter 2.3, 2.5 & 6.
2022.02.24	1.04	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.