GUI emWin
Start Guide

V1.00.001

Publication Release Date: Feb. 2018

Support Chips:

N9H3xx Series

Support Platforms:

Non-OS

The information in this document is subject to change without notice.

The Nuvoton Technology Corp. shall not be liable for technical or editorial errors or omissions contained herein; nor for incidental or consequential damages resulting from the furnishing, performance, or use of this material.

Table of Contents

1. Introduction

1.1. Introduction

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

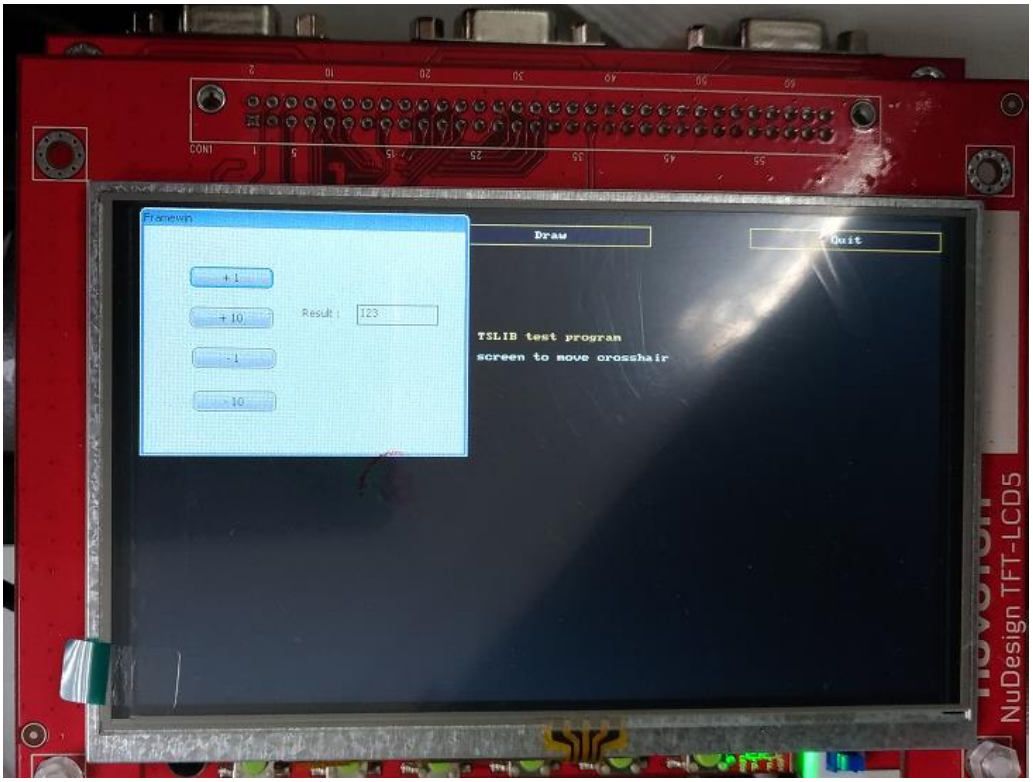

Figure 1.1-1 emWin runs on N9H3xx series.

2. Start emWin

2.1. Step 1: Open project

"emWin_SimpleDemo" is a sample code to demonstrate the emWin GUI system. It contains a frame window, four buttons, a text and a text editor.
We can click button by touch and check the result that shown on the text editor.
Here is the project path and structure:

Sample folder path: \N9H30_BSP\SampleCode\emWin_SimpleDemo

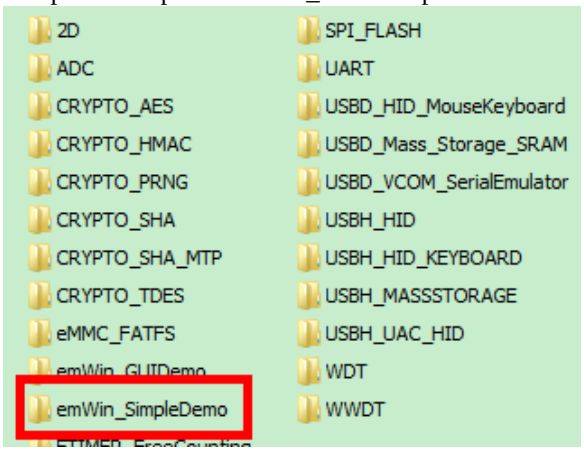

Figure 2.1-1 "emWin_SimpleDemo" sample path.

The scope of BSP is in the blue part.
The scope of emWin is in the red part. (tslib is a third-party open source that modified for BSP)

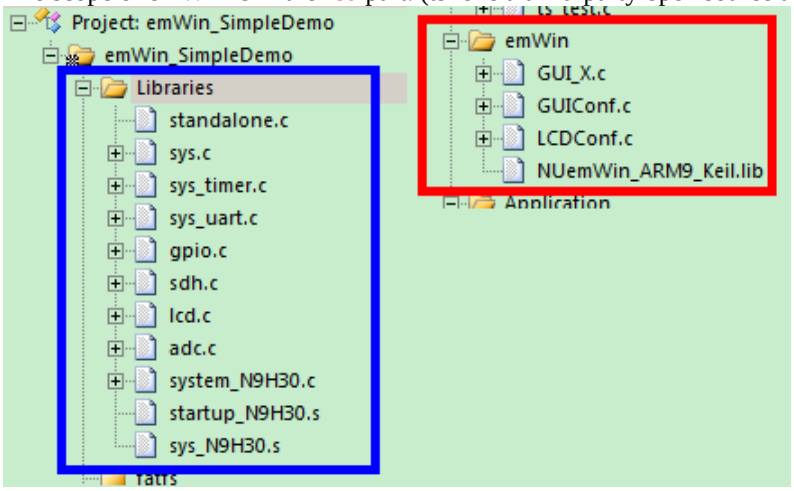

Figure 2.1-2 "emWin_SimpleDemo" project structure.

2.2. Step 2: BSP Initilization

Initialize N9H3xx series non-OS BSP to utilize the device system, e.g., Uart debug port, display output panel, vendor filesystem and resistor-type touch screen.

BSP initization descripted in \emWin_SimpleDemo\main.c.

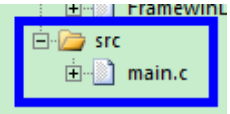

Figure 2.2-1 BSP initialization on main.c.

```
int main(void)
{
OS_TimeMS = 0;
sysSetTimerReferenceClock(TIMER0, 12000000);
/* 1000 ticks/per sec ==> 1tick/1ms */
sysStartTimer(TIMER0, 1000, PERIODIC_MODE);
sysSetTimerEvent(TIMER0, 1, (PVOID)TMR0_IRQHandler);
sysSetLocalInterrupt(ENABLE_IRQ);
LCD_initial();

#ifdef GUI_SUPPORT_TOUCH
Init_TouchPanel();
GUI_Init();
ts_calibrate(LCD_XSIZE, LCD_YSIZE);
ts_TestMain(LCD_XSIZE, LCD_YSIZE);
g_enable_Touch = 1;
#endif

MainTask();
while(1);
}
```

2.3. Step 3: emWin Initilization

To utilize emWin, we need to initialize emWin. GUI_Init() will start emWin GUI system.

\emWin_SimpleDemo\main.c:

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

2.4. Step 4: Build

To start working with the application, we need to utilize Keil MDK to build the project.

Press [F7] to compile the application or click “Rebuild”.

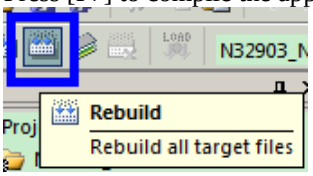

Figure 2.4-1 Build project.

2.5. Step 5: Download and run

Press CTRL + [F5] to download the application and start a debug session. After downloaded, it will halt at main() and we should see the similar screenshow below.

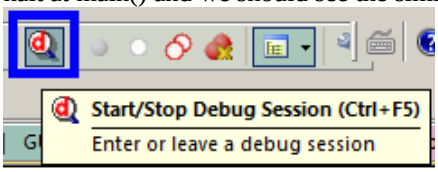

Figure 2.5-1 Download and run application.

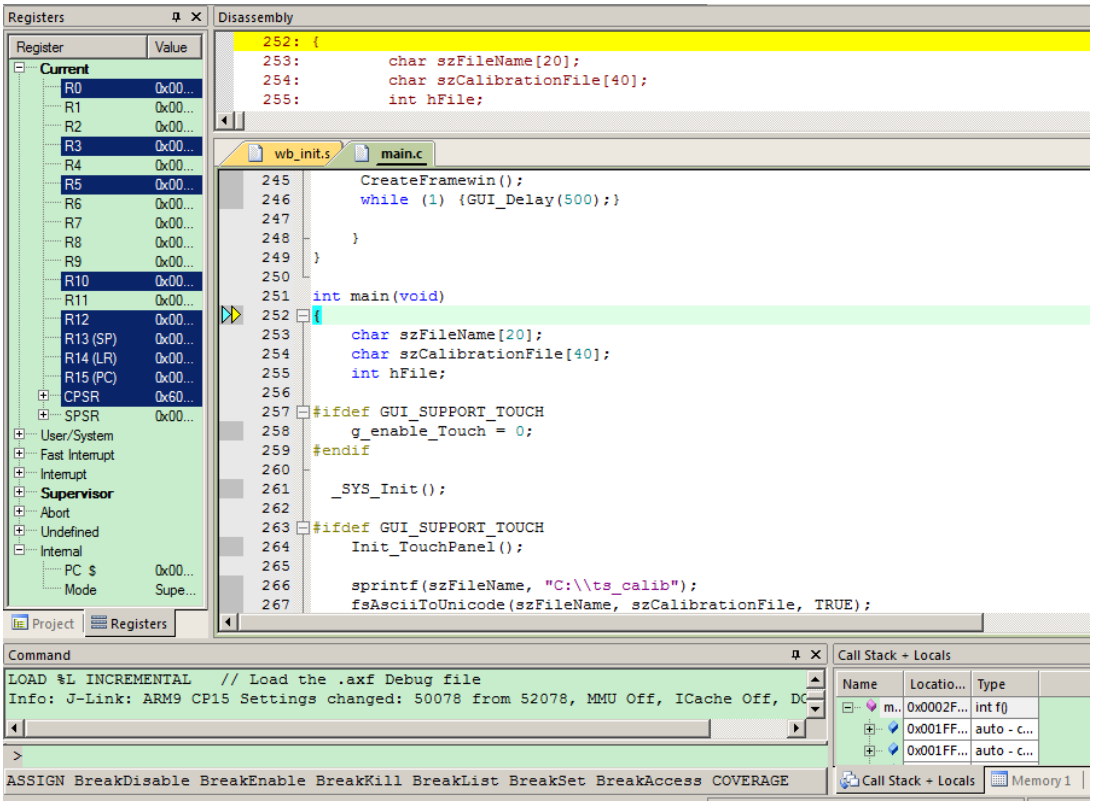

Figure 2.5-2 Debug session.

2.6. Touch screen

To use resistor-type touch screen, we can use tslib, a third-part open source library that modified for BSP.

\emWin_SimpleDemo\tslib\

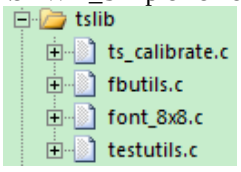

Figure 2.6-1 tslib, a third-party open source library.

3. Start emWin GUIBuilder

3.1. Step 1: Create widget

To create widget, we can use windows tool “GUIBuilder” to generate to a source file.

\ThirdParty\emWin\Tool\GUIBuilder.exe:

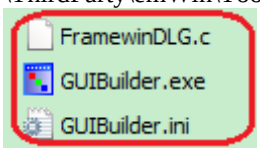

Figure 3.1-1 emWin GUIBuilder.

After execute “File” → “Save…”, we can get the source file called “FramewinDLG.c”.

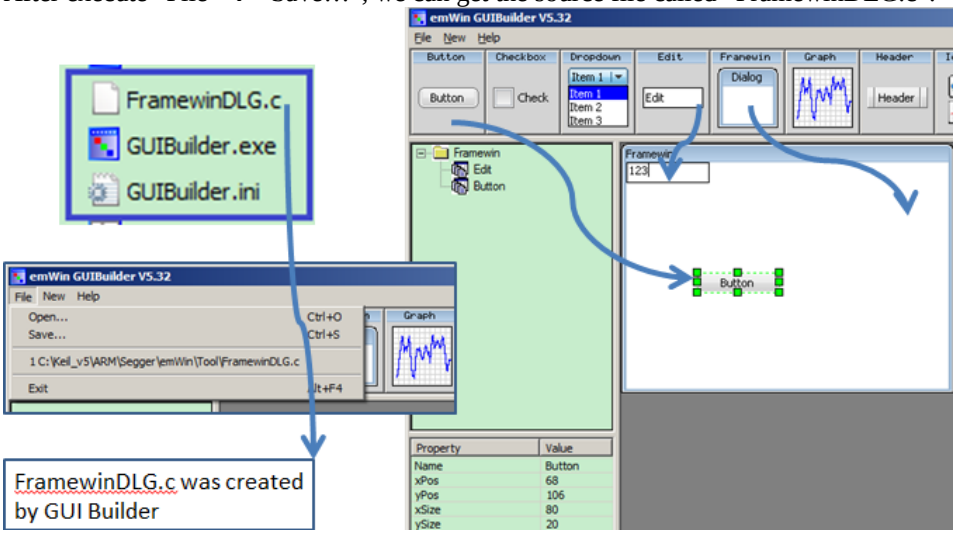

Figure 3.1-2 emWin GUIBuilder can generate a GUI layout and source file.

3.2. Step 2: Handle widget event

In “FramewinDLG.c”, we can add code to utilize widget event, e.g., initialization, button click, release and change the content data of text editor.

\emWin_SimpleDemo\Application\FramewinDLG.c:

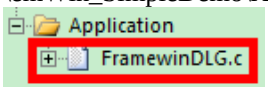

Figure 3.2-1 emWin GUI application source file.

```
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
sysprintf("clicked\n");
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);
sysprintf("released\n");
// USER END
break;
```

4. How to change display panel

4.1. Step 1: emWin display

emWin LCDConf.h defines resolution of the display panel.

\ThirdParty\emWin\Config\LCDConf.c and .h:

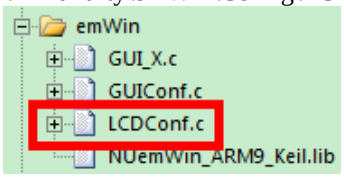

Figure 4.1-1 emWin display define.

Modify the "XSIZE_PHYS" and "YSIZE_PHYS" to fit the request LCD panel.

```
#ifndef LCDCONF_H
#define LCDCONF_H

#define XSIZE_PHYS 800
#define YSIZE_PHYS 480
```

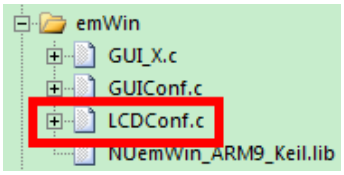

Figure 4.1-2 emWin display source code.

In LCDConf.c, we need to assign frame buffer address, e.g., Sync-type LCD 800x480, the frame buffer size in RGB565 is 800x480x2=750KB.

```
/* assign BSP display frame buffer address to emWin */
LCD_SetVRAMAddrEx(0, (void *)g_VAFrameBuf);
```

4.2. Step 2: BSP display

BSP lcd.c defines the library of display device.

\emWin_SimpleDemo\Libraries\lcd.c and lcd.h

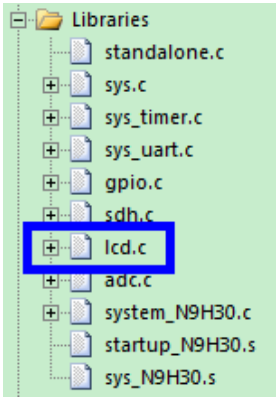

Figure 4.2-1 BSP display source code.

5. Revision History

Version	Date	Description
V1.00.001	Feb. 23, 2018	● Created

Important Notice

Nuvoton products are not designed, intended, authorized or warranted for use as components in equipment or systems intended for surgical implantation, atomic energy control instruments, aircraft or spacecraft instruments, transportation instruments, traffic signal instruments, combustion control instruments, or for any other applications intended to support or sustain life. Furthermore, Nuvoton products are not intended for applications whereby failure could result or lead to personal injury, death or severe property or environmental damage.

Nuvoton customers using or selling these products for such applications do so at their own risk and agree to fully indemnify Nuvoton for any damages resulting from their improper use or sales.