GUI emWin AppWizard Development Considerations

GUI Introduction for 64/32-bit NuMicro® Family

Document Information

Application	This readme shows emWin AppWizard development considerations for ensure smooth rendering without tearing or flickering.
BSP Version	Non-OS
Hardware	NuMaker-HMI-N9H30

1. Overview

This note will explain the application of emWin AppWizard on the Nuvoton N9H Human Machine Interface (HMI) Series platform, focusing on selecting appropriate hardware/software acceleration designs for different applications and designs. It aims to optimize the performance of the HMI interface and prevent issues such as screen distortion or tearing.

2. Code Description

The latest N9H30 Board Support Package at link: N9H30 BSP and the N9H30 emWin HMI library at link: emWin Software Package.

There are three emWin sample codes in the N9H30 BSP SampleCode directory:

- **emWin_GUIDemo**: utilizes emWin library to demonstrate widgets feature.
- **emWin_SimpleDemo**: utilizes emWin library to demonstrate interactive feature.
- **emWin_SimpleDemoAppWizard**: Utilizes AppWizard to demonstrate interactive feature.

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 src group contains the main file.

Take the project emWin_SimpleDemoAppWizard of Nos-OS version as an example, its software architecture is shown as follows:

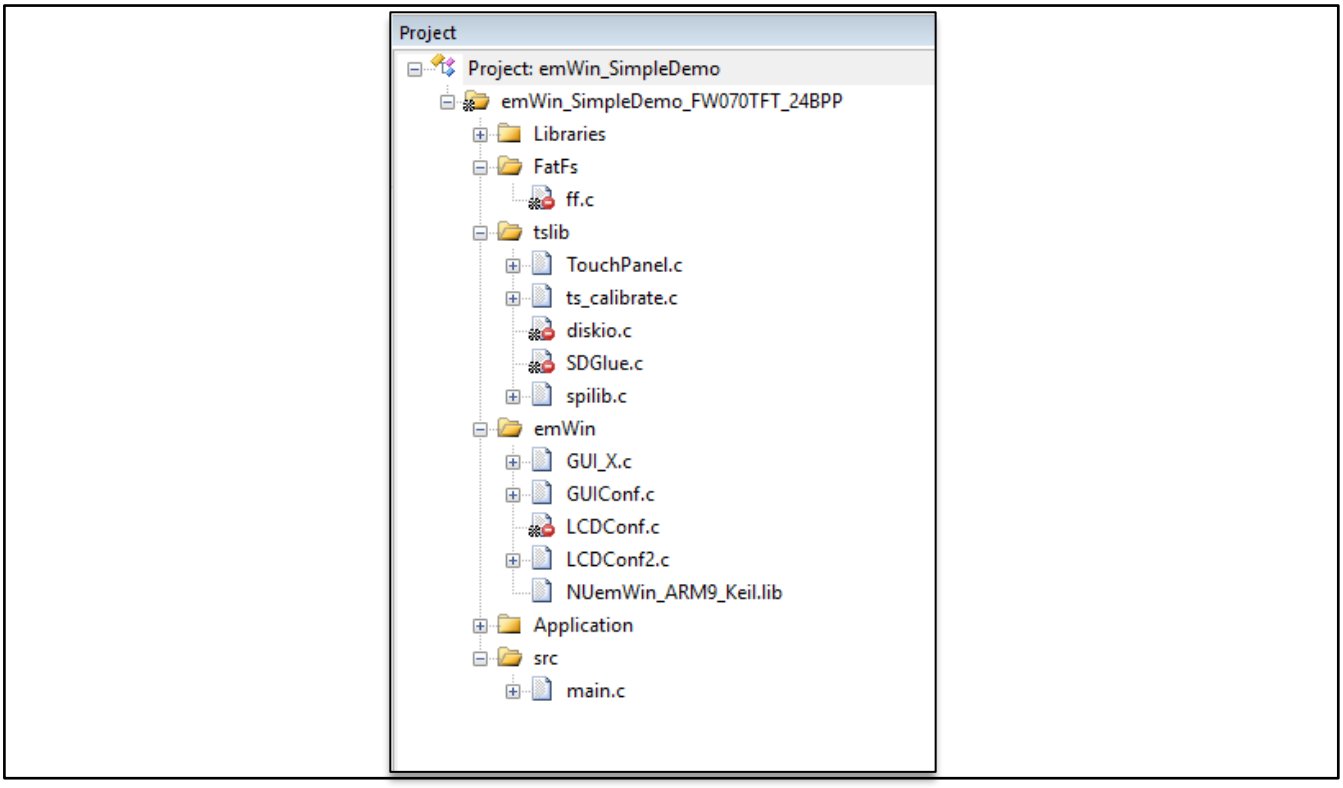

Figure 2 Software Framework

The samples already include the 2D Graphic Library, and its design incorporates a “pre-decode images to RAM” approach instead of real-time decoding to reduce CPU loading. There are two ways to avoid screen distortion, as follows:

- Enable the dual-buffer or Multi-buffer
- Reduce the pixel clock to solve the tearing of the screen

Multiple buffering is the use of more than one buffer to hold a block of data. Enable the Multi-buffer function, which is shown below.

Modify the LCD_initial function in ***main.c*** :

```
void LCD_initial(void)
{
    …
    // Get pointer of video frame buffer
    // Note: before get pointer of frame buffer, must set display color depth first

    g_VAFrameBuf = vpostGetFrameBuffer();         // set one frame buffer
    g_VAFrameBuf = vpostGetMultiFrameBuffer(2);   // set two frame buffers
    …
}
```

Note: Choose either vpostGetFrameBuffer or vpostGetMultiFrameBuffer with the development requirements

Set the NUM_BUFFERS in ***LCDConf2.c*** :

```
//
// Buffers / VScreens
//
#define NUM_BUFFERS  2 // Number of multiple buffers to be used
#define NUM_VSCREENS 1 // Number of virtual screens to be used
```

After initializing emWin, enable dual-buffering in ***APPW_MainTask.c***.

```
void MainTask(void) {

  WM_SetCreateFlags(WM_CF_MEMDEV);

  GUI_Init();
  WM_MULTIBUF_Enable(1); // Enable multiple buffers

  GUIDEMO_Main();
}
```

Suppose the emWin application is developed using GUIBuilder, AppWizard, or manually written code that utilizes Widget objects (with names starting with WIDGET_xxx), the emWin library will automatically handle buffer switching to avoid tearing and flickering.

if users call the 2D APIs by manually writing code (the ID names starting with GUI_xxx), they should handle buffer switching by settings functions *GUI_MULTIBUF_Begin()* and *GUI_MULTIBUF_End()* manually. The example of settings is shown as follows:

```
GUI_MULTIBUF_Begin();     // Switch to the next buffer

/* Perform custom 2D drawing operations using GUI_xxx APIs here */

GUI_MULTIBUF_End();       // End drawing operations and display the updated buffer
```

Using the *GUI_MULTIBUF_Begin()* and *GUI_MULTIBUF_End()* functions, users can manually switch buffers and ensure smooth rendering without tearing or flickering in custom applications.

3. Development Considerations

About N9H display capability, we measured still image display and animation display cases based on our EVB with 3 types of panels. And also measured the boundary of the pixel clock for reference, which is shown below. (All with parallel RGB "+ JPEG" means "hardware JPEG decode in the background".) For the details, please refer to Table 3-1~Table 3-7.

*** Underrun** means the pixel clock needs to be reduced below the screen specifications to prevent display tearing. Must refer to the panel's specifications. However, lowering it may cause other problems, such as frame rate issues.

- N9H30

	1024 x 600 + JPEG	1024 x 600	800 x 480 + JPEG	800 x 480	480 x 272 + JPEG	480 x 272
RGB565	OK	OK	OK	OK	OK	OK
RGB888	Underrun	OK	OK	OK	OK	OK
RGB565 + YUV422	Underrun	OK	OK	OK	OK	OK
RGB888 + YUV422	Underrun	Underrun	Underrun	OK	OK	OK
Pixel CLK (MHz)	42.8: OK 50: underrun 37: flicker	42.8: OK 50: underrun 37: flicker	27: OK 30: underrun 23: flicker	27: OK 30: underrun 23: flicker	4.6~15: OK 16: out of spec 4.5: flicker	4.6~15: OK 16: out of spec 4.5: flicker

Table 3-1 Still Image Display of N9H30

	1024 x 600 + JPEG	800 x 480 + JPEG	480 x 272 + JPEG
RGB565	OK	OK	OK
RGB888	Underrun	Underrun	OK
RGB565 + YUV422	Underrun	OK	OK
RGB888 + YUV422	Underrun	Underrun	OK
Pixel CLK (MHz)	42.8: OK 50: underrun 37: flicker	27: OK 30: underrun 23: flicker	4.6~15: OK 16: out of spec 4.5: flicker

Table 3-2 Animation Display of N9H30

Note:

1. Move data from offline fb to online fb (fps=30).
2. In RGB888 test item, 800x480 is underrun at both of pixel_clk=27MHz and 25MHz.
3. **Underrun** means the pixel clock needs to be reduced below the screen specifications to prevent display tearing. Must refer to the panel's specifications.

	1024 x 600	800 x 480 (7")	480 x 272
Panel Name	ST70IPS1024600	FORWORLD FW070TFT-V02TP	FW043TFT
Pixel CLK(MHz)	min: 44.9 typ: 51.2 max: 63	min: 26.4 typ: 33.3 max: 46.8	typ: 9 max: 15

Table 3-3 List of Panels

Note:

1. PLL= 300MHz, pixel clock= 300/(N+1).
2. VPOST: underrun bit could work.

- N9H20

	1024 x 600 + JPEG	1024 x 600	800 x 480 + JPEG	800 x 480	480 x 272 + JPEG	480 x 272
RGB565	OK	OK	OK	OK	OK	OK
RGB888	Underrun	Underrun	OK	OK	OK	OK
RGB565 + YUV422	Underrun	Underrun	OK	OK	OK	OK
RGB888 + YUV422	Underrun	Underrun	Underrun	Underrun	OK	OK
Pixel CLK (MHz)	48: OK 64: underrun 38: flicker	48: OK 64: underrun 38: flicker	27: OK 32: underrun 24: flicker	27: OK 32: underrun 24: flicker	4.6~15: OK 16: out of spec 4.5: flicker	4.6~15: OK 16: out of spec 4.5: flicker

Table 3-4 Still Image Display of N9H20

	1024 x 600 + JPEG	800 x 480 + JPEG	480 x 272 + JPEG
RGB565	OK	OK	OK
RGB888	Underrun	OK	OK
RGB565 + YUV422	Underrun	OK	OK
RGB888 + YUV422	Underrun	Underrun	OK
Pixel CLK (MHz)	48: OK 64: underrun 38: flicker	27: OK 32: underrun 24: flicker	4.6~15: OK 16: out of spec 4.5: flicker

Table 3-5 Animation Display of N9H20

Note:

1. Move data from offline fb to online fb (fps=30).
2. PLL= 192MHz, pixel clock= 192/(N+1)
3. VPOST: underrun bit not work
4. **Underrun** means the pixel clock needs to be reduced below the screen specifications to prevent display tearing. Must refer to the panel's specifications.

- N9H26

	1024 x 600 + JPEG	1024 x 600	800 x 480 + JPEG	800 x 480	480 x 272 + JPEG	480 x 272
RGB565	OK	OK	OK	OK	OK	OK
RGB888	OK	OK	OK	OK	OK	OK
RGB565 + YUV422	OK	OK	OK	OK	OK	OK
RGB888 + YUV422	Underrun	Underrun	OK	OK	OK	OK
Pixel CLK (MHz)	48 : OK 60: underrun 40-: flicker	48 ~ 60: OK 80: underrun 40-: flicker	26.6 ~ 40: OK 48: underrun 24: flicker	26.6 ~ 40: OK 48: underrun 24: flicker	4.6~15: OK 16: out of spec 4.5: flicker	4.6~15: OK 16: out of spec 4.5: flicker

Table 3-6 Still Image Display of N9H26

	1024 x 600 + JPEG	800 x 480 + JPEG	480 x 272 + JPEG
RGB565	OK	OK	OK
RGB888	OK	OK	OK
RGB565 + YUV422	OK	OK	OK
RGB888 + YUV422	Underrun	OK	OK
Pixel CLK (MHz)	48: OK 60: underrun 40-: flicker	26.6 ~ 40: OK 48: underrun 24: flicker	4.6~15: OK 16: out of spec 4.5: flicker

Table 3-7 Animation Display of N9H26

Note:

1. Move data from offline fb to online fb (fps=30).
2. PLL= 240 MHz, pixel clock= 240/(N+1)
3. DRAM: 360 MHz
4. VPOST: underrun bit could work
5. **Underrun** means the pixel clock needs to be reduced below the screen specifications to prevent display tearing. Must refer to the panel's specifications.

4. Revision History

Date	Revision	Description
2023.08.02	1.00	Initial version.

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.