Arm® 926-EJS

32-bit Microcontroller

N9H20
Elevator HMI
User Manual

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

www.nuvoton.com

Table of Contents

1 OVERVIEW

Elevator HMI for N9H20 is a GUI reference implementation.

This document utilizes Nuvoton N9H20 series geneal-purpose microprcessor N9H20K5 (32MB DDR) to implement elevator HMI with emWin GUI library. Elevator HMI image format supports PNG, GIF, JPG and BMP. Audio format supports WAV.

Figure 1-1 Elevator HMI Control

2 FEATURES

2.1 Elevator HMI Features

- Supports SEGGER licensed emWin GUI library
- Supports hardware timer to handle each events
- Supports resistive touch at 480x272 area with built-in touch ADC
- Supports high quality and contrast LCD panel with resolution up to 480 x 272
- Supports many popular image formats, e. g., PNG, GIF, JPG and BMP
- Supports audio and up to 48000Hz stereo 16-bit PCM in WAV format

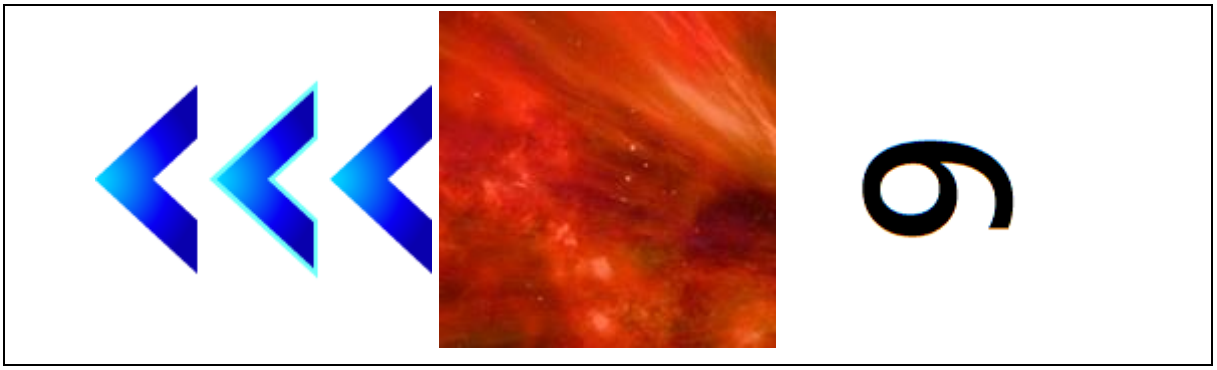

Figure 2-1 Typical Information for Elevator HMI

3 INSTALLATION AND ENVIRONMENT

3.1 Installing N9H20

First, download the latest N9H20 BSP from https://github.com/OpenNuvoton/N9H20_emWin_NonOS, and unzip "*N9H20_emWin_NonOS-master.zip*" to a working folder, e.g., unzip it to the path "*C:\N9H20*", where "N9H20" is the working folder.

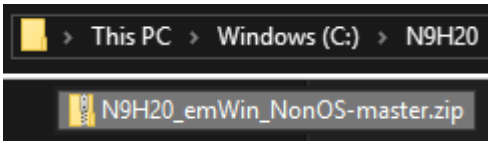

Figure 3-1 N9H20 BSP File Name and Working Folder

The detailed information of N9H20 BSP and emWin library can be found at "*\N9H20_emWin_NonOS-master\N9H20 Readme.pdf*" and "*emWinStartGuide-N9H20-Series.pdf*" respectively.

3.2 Installing Elevator HMI

First, download the latest "*N9H_emWin_Template-master.zip*" from https://github.com/OpenNuvoton/N9H_emWin_Template and unzip and copy "*Elevator_N9H20_NonOS*" to the BSP sample path "*\N9H20_emWin_NonOS-master\BSP\SampleCode\emWin*".

Then, open elevator project "*\N9H20_emWin_NonOS-master\BSP\SampleCode\emWin\Elevator_N9H20_NonOS\KEIL\SimpleDemo.uvproj*" and start compiling. The executable binary is in "*\N9H20_emWin_NonOS-master\BSP\SampleCode\emWin\Elevator_N9H20_NonOS\Bin*", called "*conprog.bin*". Next, connect the USB cable between PC/NB and N9H20 and power on. Then copy "*conprog.bin*" to "NAND1-1" USB disk. Finally, remove the USB disk safely and reboot N9H20.

3.3 System Requirements

- KEIL IDE V5.xx and above with professional license
- Nuvoton N9H20K5 480 x 272 demo board (NuDesign HMI-N9H20 + NuDesign TFT-LCD4.3)

4 FOLDER STRUCTURE

4.1 Code Folder Structure

The content of "*Elevator_N9H20_NonOS*" is described as follows.

Folder	Description
Elevator_N9H20_NonOS	Base folder ● main.c is elevator code and platform related initilizations ● Changelog.pdf is version history ● Elevator_Reference_Implementation.pdf is user manual
Application	emWin resource folder ● GUIConf2.c is for emWin memory pool ● LCDConf2.c is for emWin multiple buffers
Bin	Pre-built binaries folder ● conprog.bin is elevator execution file ● NAND1-1 is for device's NAND1-1 ● NAND1-2 is for device's NAND1-2
Bin / NAND1-1	Resource folder ● IMG / AR is for arrow png ● IMG / BG is for background jpg ● WAV / FL is for floor voice ● WAV / ST is for public broadcast voice ● WAV / DO is for door operation voice
Bin / NAND1-2	Resource folder in English voice ● WAV / FL is for floor voice ● WAV / ST is for public broadcast voice ● WAV / DO is for door operation voice
GCC	NuEclipse project folder
KEIL	Arm Keil MDK project folder
V614	PNG lib folder, An adapted version of this library ready to use with emWin is available on the website. That library can be added to emWin in order to to use the PNG API. Licensing The use of 'libpng' library is subject to a BSD style license and copyright notice in the file GUI\PNG\png.h of the downloadable library. The original version of the library is available for free under http://www.libpng.org/.

tslib	Touch folder ● Resistive touch panel ● Touch area is 480x272

Table 4-1 Elevator HMI Folder Structure

4.2 Image and Wav Resource Folder Structure

The “NAND1-1” folder contains image and wav files.

Folder	Description
IMG / AR	Arrow png ● arrow_01.png ~ arrow_08.png ● 168 x 168 dimension ● Alpha-channel for alpha-blending ● Pre-rotated
IMG / BG	Background jpg ● background_01.jpg ~ background_03.jpg ● 480 x 272 dimension ● Utilize hardware JPEG deocder ● Pre-rotated
WAV / DO	Door operation wav (22050 Hz mono channel 16-bit) ● close001.wav ● open001.wav
WAV / FL	Floor information wav (22050 Hz mono channel 16-bit) ● 001.wav ~ 015.wav
WAV / ST	Situation for broadcast wav (16000 Hz mono channel 16-bit) ● em001.wav

Table 4-2 Elevator HMI Images and Wavs Folder Structure

5 DESIGN GUIDE

Elevator reference implementation guide assumes that you already have a mature knowledge of the following:

- IDE operation for editing and compiling
- The C programming language, how to use linker and C compiler
- The N9H20 Non-OS BSP programming knowledge
- The basic emWin programming knowledge

5.1 Timer Event Control

The N9H20 utilizes hardware TIMER0 to handle each event. Each event can be set to show which arrow, background and floor image.

```
// TIMER0 Init


//External Crystal at 12MHz

sysSetTimerReferenceClock(TIMER0, u32ExtFreq);


/* 1000 ticks/per sec ==> 1tick/1ms */

sysStartTimer(TIMER0, 1000, PERIODIC_MODE);


/* 1  tick  per call back for emWin */

sysSetTimerEvent(TIMER0,  1, (PVOID)TMR0_IRQHandler);


/* 20 ticks per call back for touch */

sysSetTimerEvent(TIMER0, 20, (PVOID)TMR0_IRQHandler_TouchTask);


/* 2000 ticks per call back for updating background */

g_BGHandle = sysSetTimerEvent(TIMER0, NVT_BG_UPDATE_TIME,
(PVOID)TMR0_IRQHandler_BGTask);


/* 1000 ticks per call back for updating floor imformation */

g_FLHandle = sysSetTimerEvent(TIMER0, NVT_FL_UPDATE_TIME,
(PVOID)TMR0_IRQHandler_FLTask);


/*  300 ticks per call back for updating arrow */
```

```
g_ARHandle = sysSetTimerEvent(TIMER0, NVT_AR_UPDATE_TIME,
(PVOID)TMR0_IRQHandler_ARTask);
```

User can adjust updating interval by modifying those definitions show below.

```
//

// Define update interval

// Please note NVT_AR_UPDATE_TIME must less than elevator loop time

//

#define NVT_BG_UPDATE_TIME 2000

#define NVT_FL_UPDATE_TIME 1000

#define NVT_AR_UPDATE_TIME 300
```

For this demo, user needs to reset events after key pressed.

```
//

// Restart timer 0 all events

//

sysStartTimer(TIMER0, 1000, PERIODIC_MODE);

sysSetTimerEvent(TIMER0,  1, (PVOID)TMR0_IRQHandler);

sysSetTimerEvent(TIMER0, 20, (PVOID)TMR0_IRQHandler_TouchTask);

g_BGHandle = sysSetTimerEvent(TIMER0, NVT_BG_UPDATE_TIME,
(PVOID)TMR0_IRQHandler_BGTask);

g_FLHandle = sysSetTimerEvent(TIMER0, NVT_FL_UPDATE_TIME,
(PVOID)TMR0_IRQHandler_FLTask);

g_ARHandle = sysSetTimerEvent(TIMER0, NVT_AR_UPDATE_TIME,
(PVOID)TMR0_IRQHandler_ARTask);
```

5.2 Arrow Control

The N9H20 utilizes software decoder to decode PNG file for alpha-blending and arrow animation. (larger arrow png may drop performance)

Note: images are landscape orientation.

Figure 5-1 Landscape Arror Image

```
//
// Read arrow image (png file) and S/W decode
//
sprintf(szFileName, "C:\\IMG\\AR\\arrow_%02d.png", g_ARCnt);
_PNG_Decode(szFileName);
```

Set event handler to control index.

```
//
// Define total counts
// Need correct number of images, respectively
//
#define NVT_AR_CNT 8
//
// Update arrow index
//
volatile int g_ARCnt;
volatile int g_ARHandle;


void TMR0_IRQHandler_ARTask(void)
{
```

```
    g_ARCnt++;
    if (g_ARCnt > NVT_AR_CNT)
        g_ARCnt = 1;
}
```

Note: Count start from 1.

5.3 Background Control

The N9H20 utilizes hardware decoder to decode JPG file for background animation.

Note: images are landscape orientation.

Figure 5-2 Landscape Background Image

```
//
// Read background image (jpeg file) and H/W decode
//
sprintf(szFileName, "C:\\IMG\\BG\\background_%02d.jpg", g_BGCnt);
_JPG_Decode(szFileName);
```

Set event handler to control index.

```
//
// Define total counts
```

```
// Need correct number of images, respectively
//
#define NVT_BG_CNT 3
//
// Update background index
//
volatile int g_BGCnt;
volatile int g_BGHandle;


void TMR0_IRQHandler_BGTask(void)
{
    g_BGCnt++;
    if (g_BGCnt > NVT_BG_CNT)
        g_BGCnt = 1;
}
```

Note: Count start from 1.

5.4 Voice Control

The N9H20 utilizes internal audio codec to decode PCM. PCM data stored in WAV file container. The maximum format is 48000 Hz for sampling rate, stereo for channel and 16-bit for sample size.

Note: voice format is PCM.

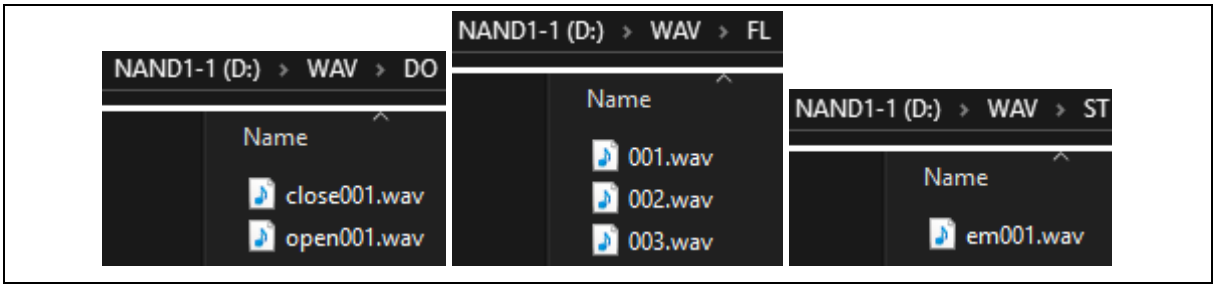

Figure 5-3 Voice Wav Files

```
//
// Play floor
```

```
//
sprintf(szFileName, "C:\\WAV\\FL\\%03d.wav", g_FLCnt);
_WAV_Decode(szFileName);
sprintf(szFileName, "C:\\WAV\\DO\\open001.wav");
_WAV_Decode(szFileName);
sprintf(szFileName, "C:\\WAV\\DO\\close001.wav");
_WAV_Decode(szFileName);
```

Play English voice version. Those voice files are stored in NAND1-2.

```
//
// Play floor in English
//
sprintf(szFileName, "D:\\WAV\\FL\\%03d.wav", g_FLCnt);
_WAV_Decode(szFileName);
sprintf(szFileName, "D:\\WAV\\DO\\open001.wav");
_WAV_Decode(szFileName);
sprintf(szFileName, "D:\\WAV\\DO\\close001.wav");
_WAV_Decode(szFileName);
```

5.5 Floor Information Control

The N9H20 utilizes emWin draw text feature to display rotated string in text box.

Note: utilize definition GUI_ROTATE_CCW to draw rotation text.

Figure 5-4 Floor Information Text

```
//
// Define text box x-y & width-height
//
```

```
#define NVT_TEXT_BOX_X 240

#define NVT_TEXT_BOX_Y 0

#define NVT_TEXT_BOX_WIDTH (NVT_TEXT_BOX_X + 128)

#define NVT_TEXT_BOX_HEIGHT (NVT_TEXT_BOX_Y + 128)

//

// Text box

//

Rect.x0 = NVT_TEXT_BOX_X;

Rect.y0 = NVT_TEXT_BOX_Y;

Rect.x1 = NVT_TEXT_BOX_WIDTH;

Rect.y1 = NVT_TEXT_BOX_HEIGHT;

//

// Text background is transparent

//

GUI_SetTextMode(GUI_TM_TRANS);

//

// Font type

//

GUI_SetFont(GUI_FONT_D80);

//

// Draw rotate text for floor number

//

sprintf(szFileName, "%d", g_FLCnt);

GUI_DispStringInRectEx(szFileName, &Rect, GUI_TA_HCENTER |
GUI_TA_VCENTER, strlen(szFileName), GUI_ROTATE_CCW);
```

5.6 Key Control

You can press key to play different voices, to hear floor number, door operation and broadcast voice.

Note: NuMaker key control utilizes "*N9H20_GPIO.lib*" and "*N9H20_KPI_2x3.lib*", respectively.

Figure 5-5 NuMaker Key Conrol

```
//
// From left to right:
// UP: Chinese voice for emergency broadcast
// DN: English voice for emergency broadcast
// LF: Chinese voice for floor information
// RT: Same as LF
// HOME: English voice for floor information
// ENTER: Same as HOME
//
// Key action
//
if (key != 0)
{
//
// Stop timer 0
// Clear background, floor & arrow event
//
sysStopTimer(TIMER0);
sysClearTimerEvent(TIMER0, g_BGHandle);
sysClearTimerEvent(TIMER0, g_FLHandle);
sysClearTimerEvent(TIMER0, g_ARHandle);
//
// Play wav
```

```
// Key 1 mapping to UP
//
if (key == 1)
{
//
// Play status
//
}
//
// Key 2 mapping to DN
//
else if (key == 2)
{
//
// Play status in English
//
}
//
// Key 4 and 8 mapping to LF and RT
//
else if ((key == 4) || (key == 8))
{
//
// Play floor
//
}
//
// Other mapping is HOME and ENTER
//
else
```

```
{
//
// Play floor in English
//
}
//
// Restart timer 0 all events
//
}
```

6 FAQ

6.1 How to replace image?

Use the same image filename, width and height, then copy to related folder, please see the chapter 4.2.

Note: to utilize PNG, you need the png library, please see the chapter 4.1.

6.2 How to replace voice?

Use the same wav filename, then copy to related folder, please see the chapter 4.2.

Note: Make sure the audio format is wav. And the highest sampling rate is 48000 Hz.

6.3 Why the animation effect looks so slow and laggy?

It is caused by softwwre decoder for PNG or GIF file on the device, please to use the suggested dimensiton for image.

7 REVISION HISTORY

Date	Revision	Description
2020.07.27	1.00	1. Initially release.
2020.12.18	1.01	1. Update document image

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.