Arm® 926-EJS

32-bit Microcontroller

N9H20K5
Elevator V2 HMI
User Manual

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

www.nuvoton.com

Table of Contents

1 OVERVIEW

Elevator V2 HMI for N9H20K5 is a GUI reference implementation.

This document utilizes Nuvoton N9H20K5 series geneal-purpose microprcessor N9H20K5 (32MB DDR) to implement second generation elevator HMI with emWin GUI library. Elevator V2 HMI image format supports PNG, GIF, JPG and BMP. Audio format supports WAV.

Arrow animation uses a series of pre-decoded PNG images to save computing power.

Floor animation uses emWin scaling-up effect.

In lang_en version, it utilizes RTC to show the month, day, time and week.

Figure 1-1 Elevator V2 HMI LANG_KO Version

Figure 1-2 Elevator V2 HMI LANG_ZH Version

Figure 1-3 Elevator V2 HMI LANG_EN Version

2 FEATURES

2.1 Elevator V2 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
- Supports two layers, one is background and the other is OSD layer respectively
- Supports true color format RGB888 and 8-bit alpha channel (per-pixel) in OSD layer
- Supports emWin memory device for animation
- Supports emWin scaling effect
- Supports RTC feature

Figure 2-1 Typical Information for Elevator V2 HMI LANG_KO Version

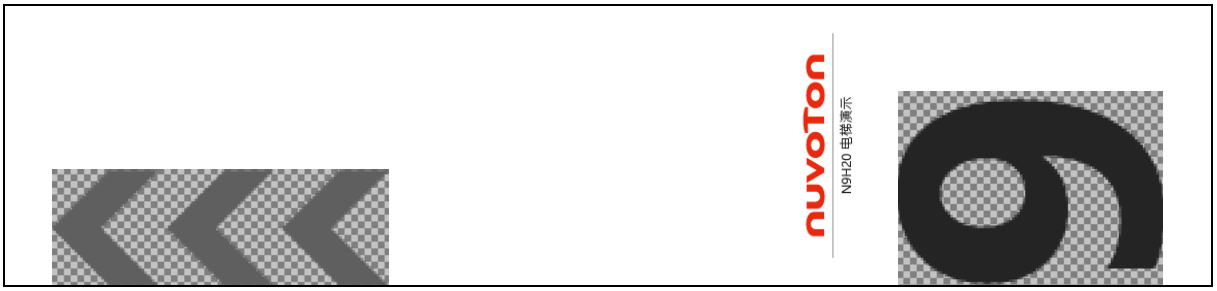

Figure 2-2 Typical Information for Elevator V2 HMI LANG_ZH Version

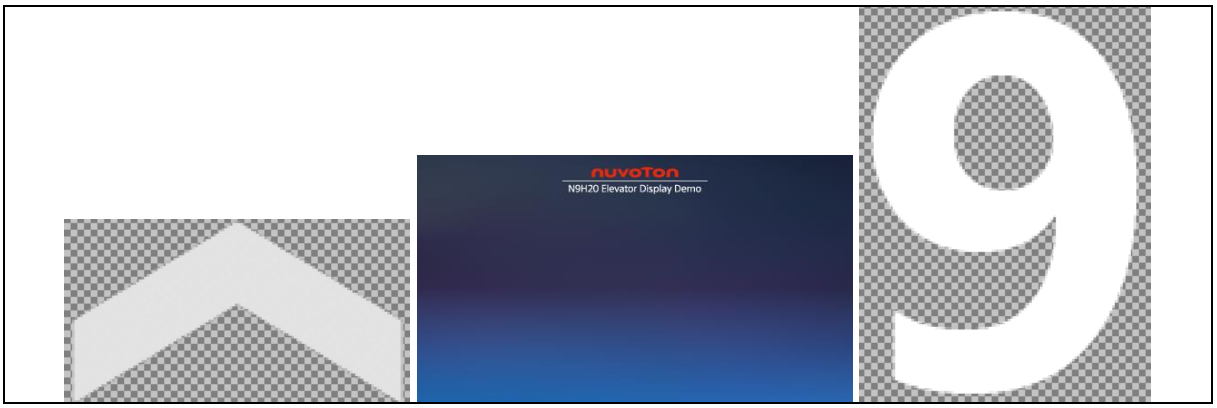

Figure 2-3 Typical Information for Elevator V2 HMI LANG_EN Version

3 INSTALLATION AND ENVIRONMENT

3.1 Installing N9H20K5

First, download the latest N9H20K5 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.

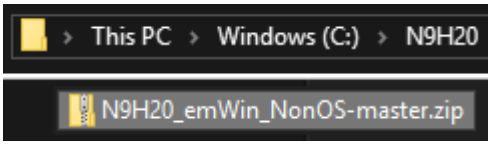

Figure 3-1 N9H20K5 BSP File Name and Working Folder

The detailed information of N9H20K5 BSP can be found at "*\N9H20_emWin_NonOS-master\N9H20 Readme.pdf*".

3.2 Installing emWin GUI Library

First, download the latest emWin GUI Library from https://www.nuvoton.com/products/microprocessors/arm9-mpus/-n9h-series/?group=Software&tab=2, and unzip "*N9H20_emWin_Package.zip*", then unzip "*N9H20_emWin_NonOS.zip*", finally, please follow the chapter 2 of "*N9H20 emWin Quick Start Guide.pdf*" to finish the installation.

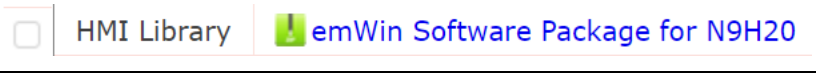

Figure 3-2 N9H20K5 emWin GUI Library

The detailed information of emWin GUI library can be found at "*\N9H20_emWin_Package\N9H20_emWin_NonOS\N9H20 emWin Quick Start Guide.pdf*".

3.3 Installing Elevator V2 HMI LANG_KO

First, download the latest "*N9H_emWin_Template-master.zip*" from https://github.com/OpenNuvoton/N9H_emWin_Template and unzip and copy "*ElevatorV2_N9H20_NonOS*" to the BSP sample path "*\N9H20_emWin_NonOS-master\BSP\SampleCode\emWin*".

For lang_ko version, copy "*lang_ko\ElevatorV2_N9H20_NonOS" to "\N9H20_NonOS_BSP-master\BSP\SampleCode\emWin\"*, and open *"\KEIL\SimpleDemo.uvproj*" and start compiling. The executable binary is in "*\Bin*", called "*conprog.bin*". Next, connect the USB cable between PC/NB and N9H20K5 and power on. Then copy "*conprog.bin*" to "NAND1-1" USB disk. Finally, remove the USB disk safely and reboot N9H20K5.

3.4 Installing Elevator V2 HMI LANG_ZH

For lang_zh version, copy "*lang_zh\ElevatorV2_N9H20_NonOS" to "\N9H20_NonOS_BSP-master\BSP\SampleCode\emWin\"*, and open *"\KEIL\SimpleDemo.uvproj*" and start compiling. The executable binary is in "*\Bin*", called "*2onprog.bin*". Next, connect the USB cable between PC/NB and N9H20K5 and power on. Then copy "*conprog.bin*" to "NAND1-1" USB disk. Finally, remove the USB disk safely and reboot N9H20K5.

3.5 Installing Elevator V2 HMI LANG_EN

For lang_en version, copy "*lang_en\ElevatorV2_N9H20_NonOS" to "\N9H20_NonOS_BSP-master\BSP\SampleCode\emWin\"*, and open *"\KEIL\SimpleDemo.uvproj*" and start compiling. The executable binary is in "*Bin*\", called "3*onprog.bin*". Next, connect the USB cable between PC/NB and N9H20K5 and power on. Then copy "*conprog.bin*" to "NAND1-1" USB disk. Finally, remove the USB disk safely and reboot N9H20K5.

3.6 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 "*ElevatorV2_N9H20_NonOS*" is described as follows.

Folder	Description
lang_ko/ElevatorV2_N9H20_NonOS lang_zh/ElevatorV2_N9H20_NonOS lang_en/ElevatorV2_N9H20_NonOS	Base folder ● main.c is elevator V2 code and platform related initilizations ● Changelog.pdf is version history ● Elevator_V2_Reference_Implementation.pdf is user manual
Application	emWin resource folder ● GUIConf.c is for emWin memory pool ● LCDConf.c is for emWin multiple buffers ● NVT_Config.c is for decoding JPEG and playing WAV ● bg.h is for background JPEG ● f0.c ~ f9.c are floor number ● d00 ~ are for down arrow ● u00 ~ are for up arrow ● c.c is for door closing ● o.c is for door opening
Bin	Pre-built binaries folder ● conprog.bin is elevator V2 lang_ko execution file ● 2onprog.bin is elevator V2 lang_zh execution file ● 3onprog.bin is elevator V2 lang_en execution file ● NAND1-1 is for device's NAND1-1 ● NAND1-2 is for device's NAND1-2
Bin / NAND1-1	Resource folder in lang_zh: ● WAV / FL is for floor voice ● WAV / DO is for door operation voice
Bin / NAND1-2	Resource folder in lang_en: ● WAV / FL is for floor voice ● WAV / DO is for door operation voice Resource folder in lang_ko:

	• WAV / 1 / FL is for floor voice • WAV / 1 / DO is for door operation voice
KEIL	Arm Keil MDK project folder
tslib	Touch folder • Resistive touch panel • Touch area is 480x272

Table 4-1 Elevator V2 HMI Folder Structure

5 DESIGN GUIDE

Elevator V2 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 RTC Control

The lang_en version utilizes RTC to show the month, day, time and week.

```
/*=========================================================================

*

*       RTC

*/

RTC_TIME_DATA_T g_sCurTime;


char g_ai8Date[64];

char g_ai8DayOfWeek[16];

char g_ai8AMPM[16];


VOID RTC_TickISR(VOID)

{

    //sysprintf("   Current
Time:%d/%02d/%02d %02d:%02d:%02d\n",g_sCurTime.u32Year,g_sCurTime.u32cMon
th,g_sCurTime.u32cDay,g_sCurTime.u32cHour,g_sCurTime.u32cMinute,g_sCurTim
e.u32cSecond);

}


static void RTC_TimeDisplay(void)

{

    RTC_TICK_T sTick;


    sysprintf("RTC Time Display \n");
```

```
    /* Set Tick property */
    sTick.ucMode = RTC_TICK_1_SEC;
    sTick.pfnTickCallBack = RTC_TickISR;

    g_sCurTime.u32Year         = 2022;
    g_sCurTime.u32cMonth       = 10;
    g_sCurTime.u32cDay         = 13;
    g_sCurTime.u32cHour        = 6;
    g_sCurTime.u32cMinute      = 10;
    g_sCurTime.u32cSecond      = 50;
    g_sCurTime.u32cDayOfWeek   = (4 - 1);   /* -1 */
    g_sCurTime.u8cClockDisplay = RTC_CLOCK_12;
    g_sCurTime.u8cAmPm         = RTC_PM;
    RTC_Write(RTC_CURRENT_TIME, &g_sCurTime);

    /* Set Tick setting */
    RTC_Ioctl(0,RTC_IOC_SET_TICK_MODE, (UINT32)&sTick,0);

    /* Enable RTC Tick Interrupt and install tick call back function */
    RTC_Ioctl(0,RTC_IOC_ENABLE_INT, (UINT32)RTC_TICK_INT,0);

}
```

Get the current time flow:

```
/* Get the current time */
RTC_Read(RTC_CURRENT_TIME, &g_sCurTime);
memset(g_ai8Date, 0x00, 64);
memset(g_ai8DayOfWeek, 0x00, 16);
memset(g_ai8AMPM, 0x00, 16);
switch (g_sCurTime.u32cDayOfWeek)
```

```
{
case 0:
    sprintf(g_ai8DayOfWeek, "Mon");
    break;
case 1:
    sprintf(g_ai8DayOfWeek, "Tue");
    break;
case 2:
    sprintf(g_ai8DayOfWeek, "Wed");
    break;
case 3:
    sprintf(g_ai8DayOfWeek, "Thu");
    break;
case 4:
    sprintf(g_ai8DayOfWeek, "Fri");
    break;
case 5:
    sprintf(g_ai8DayOfWeek, "Sat");
    break;
default:
    sprintf(g_ai8DayOfWeek, "Sun");
    break;
}

switch (g_sCurTime.u8cAmPm)
{
case 1:
    sprintf(g_ai8AMPM, "AM");
    break;
default:
```

```
    sprintf(g_ai8AMPM, "PM");

    break;

}

sprintf(g_ai8Date, "%02d/%02d %s. %02d:%02d %s", g_sCurTime.u32cMonth,
g_sCurTime.u32cDay, g_ai8DayOfWeek, g_sCurTime.u32cHour,
g_sCurTime.u32cMinute, g_ai8AMPM);



GUI_DispStringAt(g_ai8Date, 60, 160);
```

5.2 Arrow Control

The N9H20K5 utilizes emWin memory device to rendor series pre-decoded PNG data as arrow animation and to save more computing power. (larger arrow png may drop performance)

Note: LCD rotation may drop performance dramatically, hence, pre-rotated images are used in lang_zh version for portrait veiw.

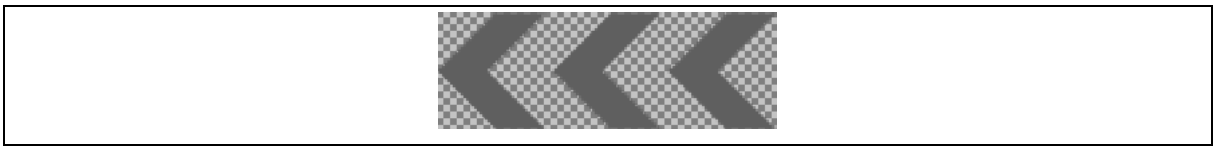

Figure 5-1 Arrow in Portrait View for LANG_ZH

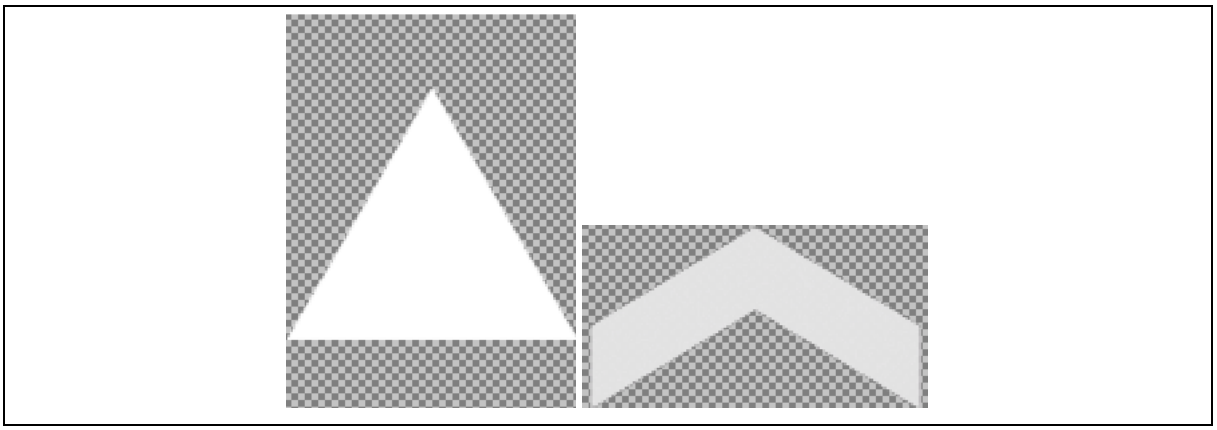

Figure 5-2 Normal Arrow View for LANG_KO and LANG_EN

```
void NVT_ShowUp(const GUI_BITMAP * pBM, int x0, int y0)

{

    if (g_flagM0S0 == 0)

    {

        GUI_MEMDEV_Select(hMemM0S0);

    }

    else

    {

        GUI_MEMDEV_Select(hMemM0S02);

    }


    GUI_SetBkColor(DEF_OSD_COLORKEY);
```

```
    GUI_ClearRect(x0, y0, x0 + pBM->XSize - 1, y0 + pBM->YSize - 1);


    GUI_DrawBitmap(pBM, x0, y0);
}


void NVT_ShowDown(const GUI_BITMAP * pBM, int x0, int y0)
{
    if (g_flagM0S0 == 0)
    {
        GUI_MEMDEV_Select(hMemM0S0);
    }
    else
    {
        GUI_MEMDEV_Select(hMemM0S02);
    }


    GUI_SetBkColor(DEF_OSD_COLORKEY);
    GUI_ClearRect(x0, y0, x0 + pBM->XSize - 1, y0 + pBM->YSize - 1);


    GUI_DrawBitmap(pBM, x0, y0);
}
```

Note: Count start from 0.

5.3 Background Control

The N9H20K5 utilizes H/W acceleration to decode JPEG as background.

Note: LCD rotation may drop performance dramatically, hence, pre-rotated images are used in lang_zh version for portrait veiw.

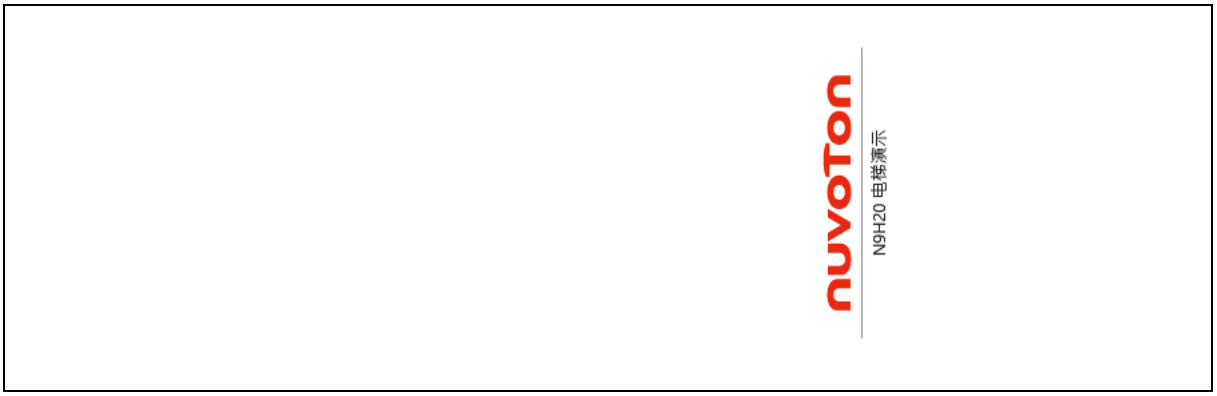

Figure 5-3 Background in Portrait View for LANG_ZH

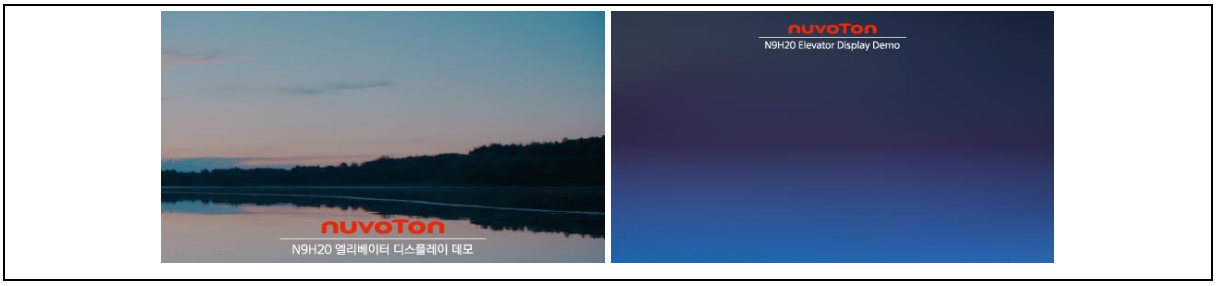

Figure 5-4 Normal Background View for LANG_KO and LANG_EN

```
int NVT_DecodeBGJPEG(void)

{

    jpegOpen();

    jpegInit();

    jpegIoctl(JPEG_IOCTL_SET_BITSTREAM_ADDR, (((unsigned int)_acbg) |
BIT31), 0);

    jpegIoctl(JPEG_IOCTL_SET_DECODE_MODE, JPEG_DEC_PRIMARY_PACKET_YUV422,
0);   // JPEG_DEC_PRIMARY_PACKET_YUV422 JPEG_DEC_PRIMARY_PACKET_RGB565
JPEG_DEC_PRIMARY_PACKET_RGB888

//    jpegIoctl(JPEG_IOCTL_SET_DECODE_DOWNSCALE, LCD_YSIZE, LCD_XSIZE);

    jpegIoctl(JPEG_IOCTL_SET_DECODE_STRIDE, LCD_XSIZE, 0);

    jpegIoctl(JPEG_IOCTL_SET_YADDR, (((unsigned int)u8FrameBufPtr) |
BIT31), 0);

    jpegIoctl(JPEG_IOCTL_DECODE_TRIGGER, 0, 0);

    jpegWait();

    jpegClose();
```

```
    return 0;
}
```

Note: The address of source and destination must non-cacheable respectively

5.4 Voice Control

The N9H20K5 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.

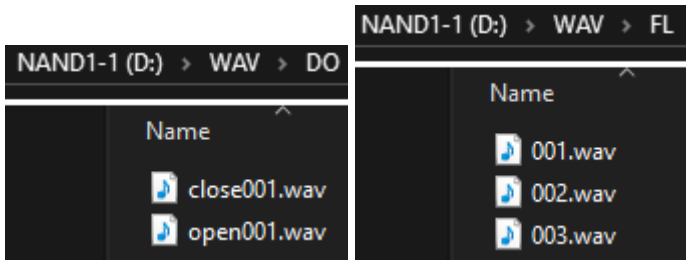

Figure 5-5 Voice Wav Files

```
//
// Play floor
//
_WAV_Decode("D:\\WAV\\FL\\001.wav");
_WAV_Decode("D:\\WAV\\DO\\open001.wav");
```

5.5 Floor Information Control

The N9H20K5 utilizes emWin to draw and scale up floor number as special effect.

Note: larger floor number may drop performance.

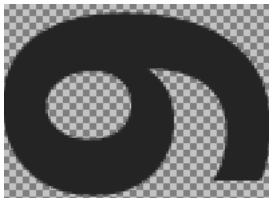

Figure 5-6 Floor Number in Portrait View for LANG_ZH

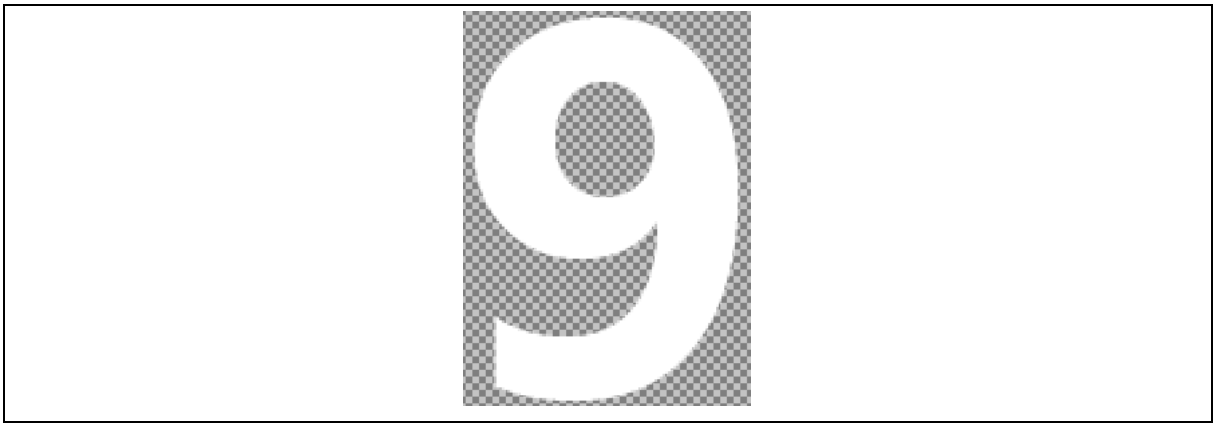

Figure 5-7 Floor Number in Normal View for LANG_KO and LANG_EN

```
void NVT_ShowFloorNumber(const GUI_BITMAP * pBM, int dx, int dy, int Mag)

{

    if (g_flagMOS0 == 0)

    {

        GUI_MEMDEV_Select(hMemMOS0);

    }

    else

    {

        GUI_MEMDEV_Select(hMemMOS02);

    }



    GUI_SetBkColor(DEF_OSD_COLORKEY);

    GUI_ClearRect(dx, dy, dx + pBM->XSize + 4 + pBM->XSize - 1, dy +
pBM->YSize - 1);



    if (g_flagMOS0 == 0)

    {

        GUI_MEMDEV_RotateHR(hMemFloorNumber, hMemMOS0, dx * 8, dy * 8, 0,
Mag);

    }

    else
```

```
    {

        GUI_MEMDEV_RotateHR(hMemFloorNumber, hMemMOS02, dx * 8, dy * 8,
0, Mag);

    }

}
```

Note: In high resolution mode, dx and dy need to multiply 8-pixel respectively.

5.6 Key Control

You can press any key to switch multiple languages FW, conprog.bin is for lang_ko, 2onprog.bin is for lang_zh and 3onprog.bin is for lang_en.

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

Note: Please make sure the NAND1-1 only contains "*conprog.bin*", "*2onprog.bin*" and "*3onprog.bin*".

Figure 5-8 NuMaker Key Conrol

6 REVISION HISTORY

Date	Revision	Description
2022.10.31	1.00	1. Initially release.

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.