Arm® 926-EJS

32-bit Microcontroller

N9H26 Cooking HMI User Manual

Table of Contents

1 OVERVIEW

Cooking HMI for N9H26 is a GUI reference implementation.

This document utilizes Nuvoton N9H26 series geneal-purpose microprcessor N9H26K6 (64MB DDR) to implement cooking HMI with emWin GUI library. Cooking HMI supports hardware H.264 and AAC decoder.

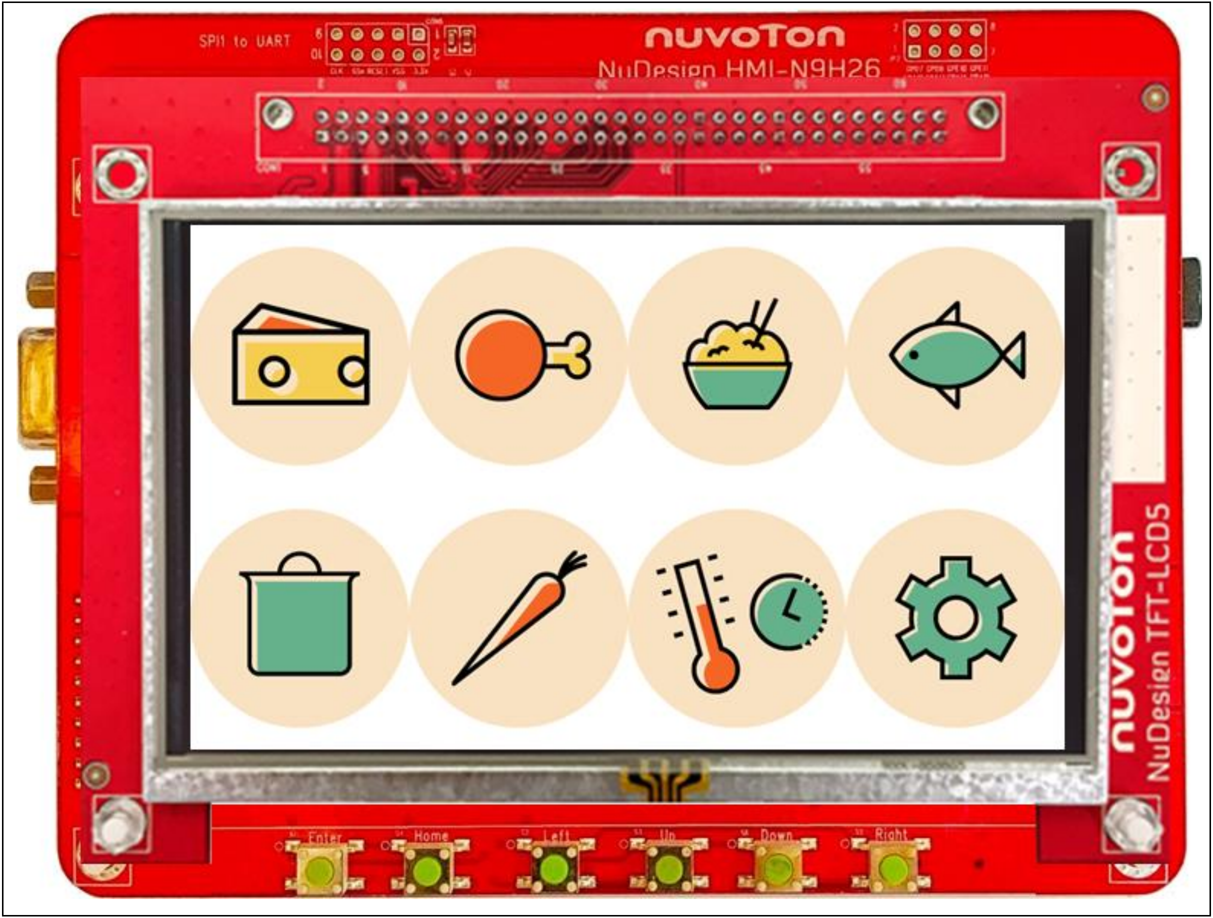


Figure 1-1 Cooking HMI Main Menu

2 FEATURES

2.1 Cooking HMI Features

- Support Nuvoton MPU N9H26K6
- Supports hardware H.264 decoder for baseline decoding
- Supports hardware AAC decoder with built-in audio codec
- Supports high quality and contrast LCD panel with resolution up to 800 x 480
- Supports resistive touch up to 800 x 480 area with built-in touch ADC
- Supports FreeRTOS
- Supports SEGGER licensed emWin GUI library
- Supports many popular image formats, e. g., PNG, GIF, JPG and BMP
- Supports user defined image as icon source

- Supports many popular image formats, e. g., PNG, GIF, JPG and BMP
- Supports user defined image as icon source

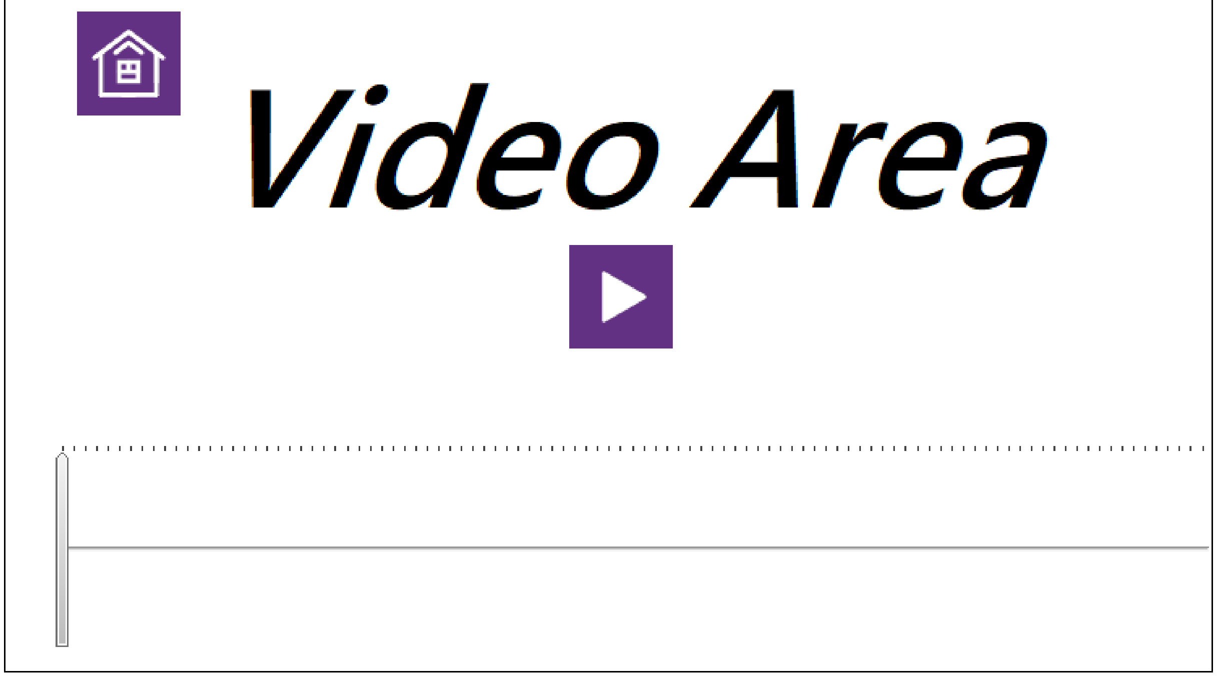
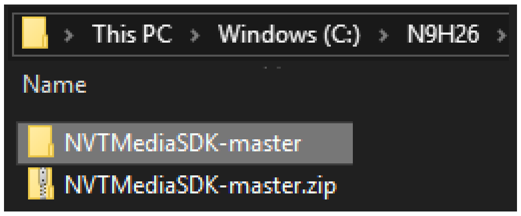


Figure 2-1 Cooking HMI MP4 Video Playback Menu

3 INSTALLATION AND ENVIRONMENT

3.1 Installing NVTMediaSDK

First, download the latest NVTMediaSDK from https://github.com/OpenNuvoton/NVTMediaSDK and unzip "*NVTMediaSDK-master.zip*" to a working folder, e. g., unzip it to the path "*C:\N9H26*", where "N9H26" is the working folder.



This PC › Windows (C:) › N9H26 ›

Name

NVTMediaSDK-master

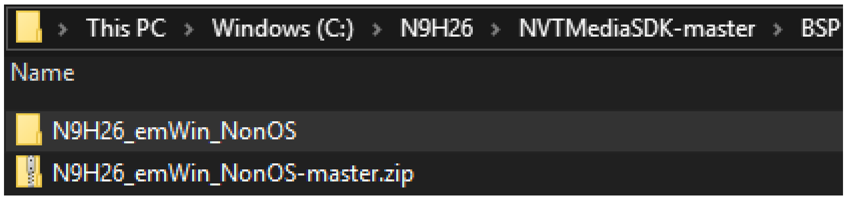
NVTMediaSDK-master.zip

Figure 3-1 NVTMediaSDK Folder

Then goto "*C:\N9H26\NVTMediaSDK-master\BSP*" and start to install N9H26 Non-OS BSP. It will describe in the next chapter. The detailed information of NVTMediaSDK can be found at "*C:\N9H26\NVTMediaSDK-master\Doc\NVTMediaSDK User Guide.pdf*".

3.2 Installing N9H26 Non-OS BSP

First, download the latest N9H26 Non-OS BSP from https://github.com/OpenNuvoton/N9H26_emWin_NonOS, and unzip "*N9H26_emWin_NonOS-master.zip*" to "*C:\N9H26\NVTMediaSDK-master\BSP*". Then rename from "*N9H26_emWin_NonOS-master*" to "*N9H26_emWin_NonOS*".



This PC › Windows (C:) › N9H26 › NVTMediaSDK-master › BSP

Name

N9H26_emWin_NonOS

N9H26_emWin_NonOS-master.zip

Figure 3-2 N9H26 Non-OS BSP Folder

The detailed information of N9H26 Non-OS BSP and emWin library can be found at "*\N9H26_emWin_NonOS-master\N9H26 Readme.pdf*" and "*N9H26 emWin Quick Start Guide.pdf*" respectively.

3.3 Installing Cooking HMI

First, download and unzip the latest "*N9H_emWin_Template-master.zip*" from https://github.com/OpenNuvoton/N9H_emWin_Template and copy "*Cooking_N9H26_FreeRTOS*" to the NVTMediaSDK sample path "*C:\N9H26\NVTMediaSDK-master\SampleCode*".

Then, open KEIL project file at "*C:\N9H26\NVTMediaSDK-master\SampleCode\Cooking_N9H26_FreeRTOS\KEIL\NMPlayer_GUI.uvproj*" and start compiling. The executable binary is in "*C:\N9H26\NVTMediaSDK-master\SampleCode\Cooking_N9H26_FreeRTOS\Bin*", called "*conprog.bin*". Next, connect the USB cable between PC/NB and N9H26 (press up & down key), then power on N9H26. Then copy "*conprog.bin*" to "*SD1-1*" USB disk. Next, copy "*C:\N9H26\NVTMediaSDK-master\SampleCode\Cooking_N9H26_FreeRTOS\Bin\SD1-2\*.**" to "*SD1-2*" USB disk. Finally, remove the USB disk safely and reboot N9H26.

3.4 System Requirements

- KEIL IDE V5.xx and above with professional license
- Nuvoton N9H26K6 800 x 480 demo board (NuDesign HMI-N9H26 + NuDesign TFT-LCD5)

4 FOLDER STRUCTURE

4.1 Code Folder Structure

The content of “*Cooking_N9H26_FreeRTOS*” is described as follows.

Folder	Description
Cooking_N9H26_FreeRTOS	Base folder ● Changelog.pdf is version history ● Cooking_Reference_Implementation.pdf is user manual
src / Application	HMI folder ● TaskGUIDemo.c is for play status control ● TaskTouching.c is for touch operation ● NVT_Config.c is for Nuvoton platform ● PlaybackDLG is for playback menu ● NVT_Playback1.c is for playback menu extension ● Window_CookingDLG.c is for cooking menu ● NVT_Cooking1.c is for cooking menu extension ● Window_TempDLG.c is for temperature menu ● NVT_Temp1.c is for temperature menu extension ● Window_SettingDLG.c is for setting menu ● NVT_Setting1.c is for setting menu extension
src / GUI_Config	emWin folder ● GUI_X.c is for emWin timer under FreeRTOS timer control ● GUIConf.c is for emWin memory pool ● LCDConf.c is for emWin display driver control
Bin	Pre-built binaries folder ● conprog.bin is for cooking execution file
Bin / SD1-1	Resource folder ● conprog.bin is for cooking execution file and identical with Bin’s conprog.bin
Bin / SD1-2	Resource image and dta ● DCIM is for 6 MP4 video files ● 01_dessert.dta ~ 08setting.dta are cooking menu icons ● Home_01.dta is back function to cooking menu for playback, temperature and setting menu ● Pause_01.dta is pause function in playback menu ● Play_01.dta is play function in playback menu

Bin / 9To565	Any image convert to dta file format
Bin / 28ToA565	Any image convert to dta file format with alpha-channel
Bin / AnyToMP4	Any video convert to base-line H.264 + AAC in MP4
Bin / LogGen	Any image convert to logo file format
Res	Image files
KEIL	Arm Keil MDK project folder
src / tslib	Touch folder ● Resistive touch panel ● Touch area is 480x272

Table 4-1 Cooking HMI Folder Structure

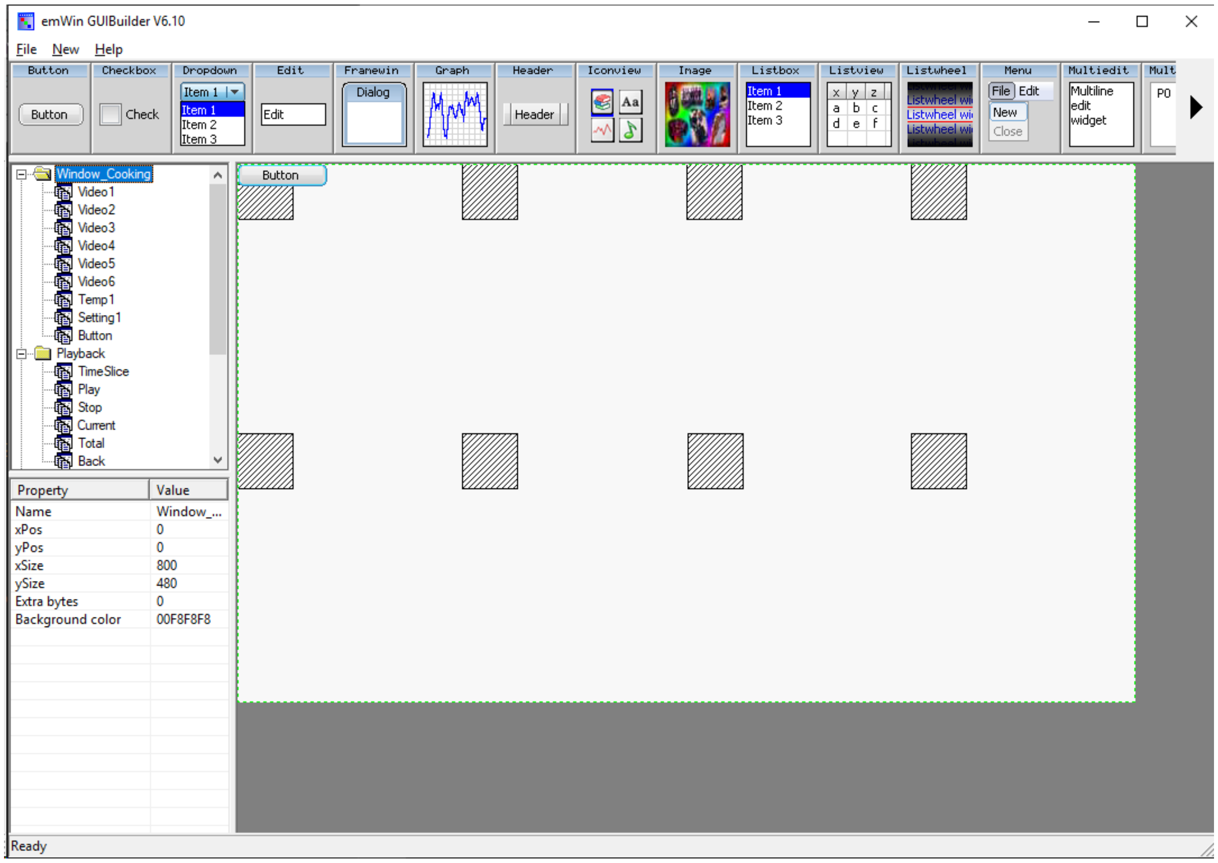
5 DESIGN GUIDE

Cooking 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 N9H26 Non-OS BSP & FreeRTOS programming knowledge
- The basic emWin programming knowledge

Note: cooking template is based on NVTMediaSDK - a Nuvoton Media SDK, more details can be found at "*NVTMediaSDK\Doc\NVTMediaSDK User Guide.pdf*".

5.1 Cooking Menu Control



Property	Value
Name	Window_...
xPos	0
yPos	0
xSize	800
ySize	480
Extra bytes	0
Background color	00F8F8F8

Figure 5-1 Cooking Menu

Cooking menu is generated from GUIBuilder, called "*Window_CookingDLG.c*". You can open this file by GUIBuilder and re-arrange widget postion, size and property. It contains 1 dummy BUTTON and 8 IMAGEs.

By "*NVT_Cooking1.c*", you can assign user define dta file to replace the original.

```
// USER START (Optionally insert additional code for further widget
initialization)

hItem = WM_GetDialogItem(pMsg->hWin, ID_IMAGE_0);

NVT_Cooking1Video1(hItem);
```

```
hItem = WM_GetDialogItem(pMsg->hWin, ID_IMAGE_1);

NVT_Cooking1Video2(hItem);


hItem = WM_GetDialogItem(pMsg->hWin, ID_IMAGE_2);

NVT_Cooking1Video3(hItem);


hItem = WM_GetDialogItem(pMsg->hWin, ID_IMAGE_3);

NVT_Cooking1Video4(hItem);


hItem = WM_GetDialogItem(pMsg->hWin, ID_IMAGE_4);

NVT_Cooking1Video5(hItem);


hItem = WM_GetDialogItem(pMsg->hWin, ID_IMAGE_5);

NVT_Cooking1Video6(hItem);


hItem = WM_GetDialogItem(pMsg->hWin, ID_IMAGE_6);

NVT_Cooking1Temp1(hItem);


hItem = WM_GetDialogItem(pMsg->hWin, ID_IMAGE_7);

NVT_Cooking1Setting1(hItem);


hItem = WM_GetDialogItem(pMsg->hWin, ID_BUTTON_0);

NVT_Cooking1SetButton(hItem, 1);


// USER END
```

Click IMAGE will invoke related function, e. g., ID_IMAGE_0 is to play video and ID_IMAGE_7 is to go setting menu:

```
// USER START (Optionally insert additional code for further Ids)

case ID_IMAGE_0:
```

```
case ID_IMAGE_1:

case ID_IMAGE_2:

case ID_IMAGE_3:

case ID_IMAGE_4:

case ID_IMAGE_5:

case ID_IMAGE_6:

case ID_IMAGE_7:

    invoke_cooking_command(NCode, Id);

    break;


// USER END;
```

The implementation of invoke_cooking_command:

```
// USER START (Optionally insert additional static code)

static void invoke_cooking_command(int NCode, int Id)

{

    printf("[%s] id=%d, NCode=%d, \n", __func__, Id, NCode);

    if (NCode == WM_NOTIFICATION_CLICKED)

    {

        switch (Id)

        {

        case ID_IMAGE_0://Video1

            g_u8Flag1 = 1;

            break;

        case ID_IMAGE_1://Video2

            g_u8Flag1 = 2;

            break;

        case ID_IMAGE_2://Video3

            g_u8Flag1 = 3;

            break;
        case ID_IMAGE_3://Video4
            g_u8Flag1 = 4;
            break;
        case ID_IMAGE_4://Video5
            g_u8Flag1 = 5;
            break;
        case ID_IMAGE_5://Video6
            g_u8Flag1 = 6;
            break;
        case ID_IMAGE_6://Temp1
            g_u8Flag1 = 7;
            break;
        case ID_IMAGE_7://Setting1
            g_u8Flag1 = 8;
            break;
        }
    }
}

// USER END
```

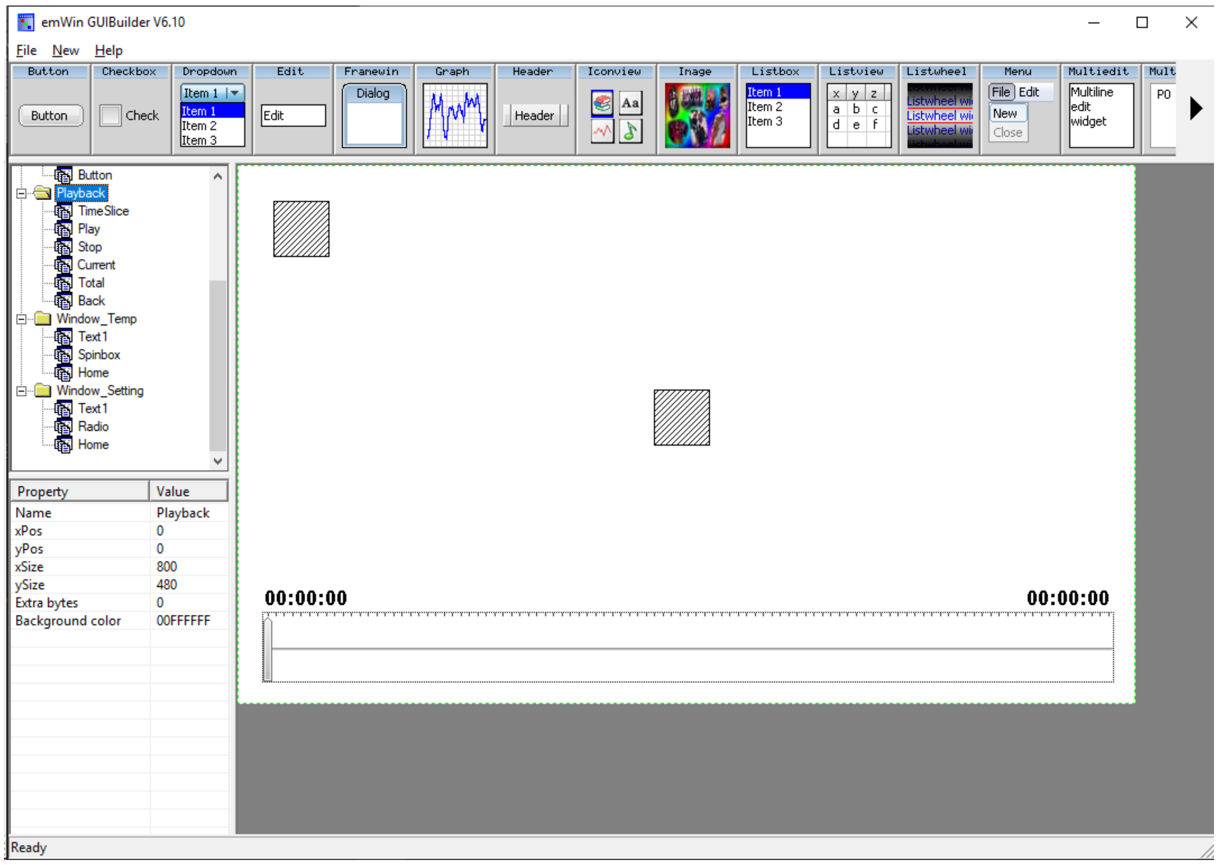
User defined dta in “*NVT_Cooking1.c*”:

Note: the size of dta with alpha-channel is bigger than dta.

```
#define NVT_COOKING_VIDEO1 "D:\\01_dessert.dta"
#define NVT_COOKING_VIDEO2 "D:\\02_meat.dta"
#define NVT_COOKING_VIDEO3 "D:\\03_rice.dta"
#define NVT_COOKING_VIDEO4 "D:\\04_seafood.dta"
#define NVT_COOKING_VIDEO5 "D:\\05_soup.dta"
#define NVT_COOKING_VIDEO6 "D:\\06_vegetable.dta"
#define NVT_COOKING_TEMP1 "D:\\07_temp_time.dta"
```

```
#define NVT_COOKING_SETTING1 "D:\\08_setting.dta"
```

5.2 Playback Menu Control


emWin GUIBuilder V6.10
File New Help
Button Checkbox Dropdown Edit Framewin Graph Header Iconview Image Listbox Listview Listwheel Menu Multiedit
00:00:00
00:00:00
Ready

- Button
- Playback
  - TimeSlice
  - Play
  - Stop
  - Current
  - Total
  - Back
- Window_Temp
  - Text1
  - Spinbox
  - Home
- Window_Setting
  - Text1
  - Radio
  - Home

Property	Value
Name	Playback
xPos	0
yPos	0
xSize	800
ySize	480
Extra bytes	0
Background color	00FFFFFF

Figure 5-2 Playback Menu

Playback menu is generated from GUIBuilder, called “*PlaybackDLG.c*”. You can open this file by GUIBuilder and re-arrange widget postion, size and property. It contains 1 SCROLLBAR, 2 TEXT and 3 IMAGE.

Note: PlaybackDLG.c contains NVTMediaSDK API control flow, details can be found at “*NVTMediaSDK\Doc\NVTMediaSDK User Guide.pdf*”.

By “*NVT_Playback1.c*”, you can assign user define dta file to replace the original.

```
// USER START (Optionally insert additional code for further widget
initialization)

hItem = pMsg->hWin;

WINDOW_SetBkColor(hItem, DEF_OSD_COLORKEY);



// Set SLIDER width

hItem = WM_GetDialogItem(pMsg->hWin, ID_SLIDER_0);
```

```
SLIDER_SetWidth(hItem, 32);


hItem = WM_GetDialogItem(pMsg->hWin, ID_IMAGE_0);

NVT_Playback1Play(hItem);


hItem = WM_GetDialogItem(pMsg->hWin, ID_IMAGE_1);

NVT_Playback1Stop(hItem);


hItem = WM_GetDialogItem(pMsg->hWin, ID_IMAGE_2);

NVT_Playback1Back(hItem);


// Create file list.

filelist_create(DEF_PATH_MEDIA_FOLDER);


// Dump media file name in file list.

filelist_dump();


play_first_media();


// USER END
```

Click IMAGE will invoke related function, e. g., ID_IMAGE_1 is to pause video and ID_IMAGE_2 is to go cooking menu:

```
// USER START (Optionally insert additional code for further Ids)

case ID_IMAGE_0://Play

case ID_IMAGE_1://Stop

case ID_IMAGE_2://Back

    invoke_player_command(NCode, Id);

    break;
```

```
// USER END
```

The implementation of invoke_player_command:

```
// USER START (Optionally insert additional static code)
static void invoke_player_command(int NCode, int Id)
{
    printf("[%s] id=%d, NCode=%d, \n", __func__, Id, NCode);
    if (NCode == WM_NOTIFICATION_CLICKED)
    {
        switch (Id)
        {
        case ID_IMAGE_0://Play
            player_play();
            break;
        case ID_IMAGE_1://Stop
            player_pause();
            break;
        case ID_IMAGE_2://Back
            player_stop();
            break;
        }
    }
}


// USER END
```

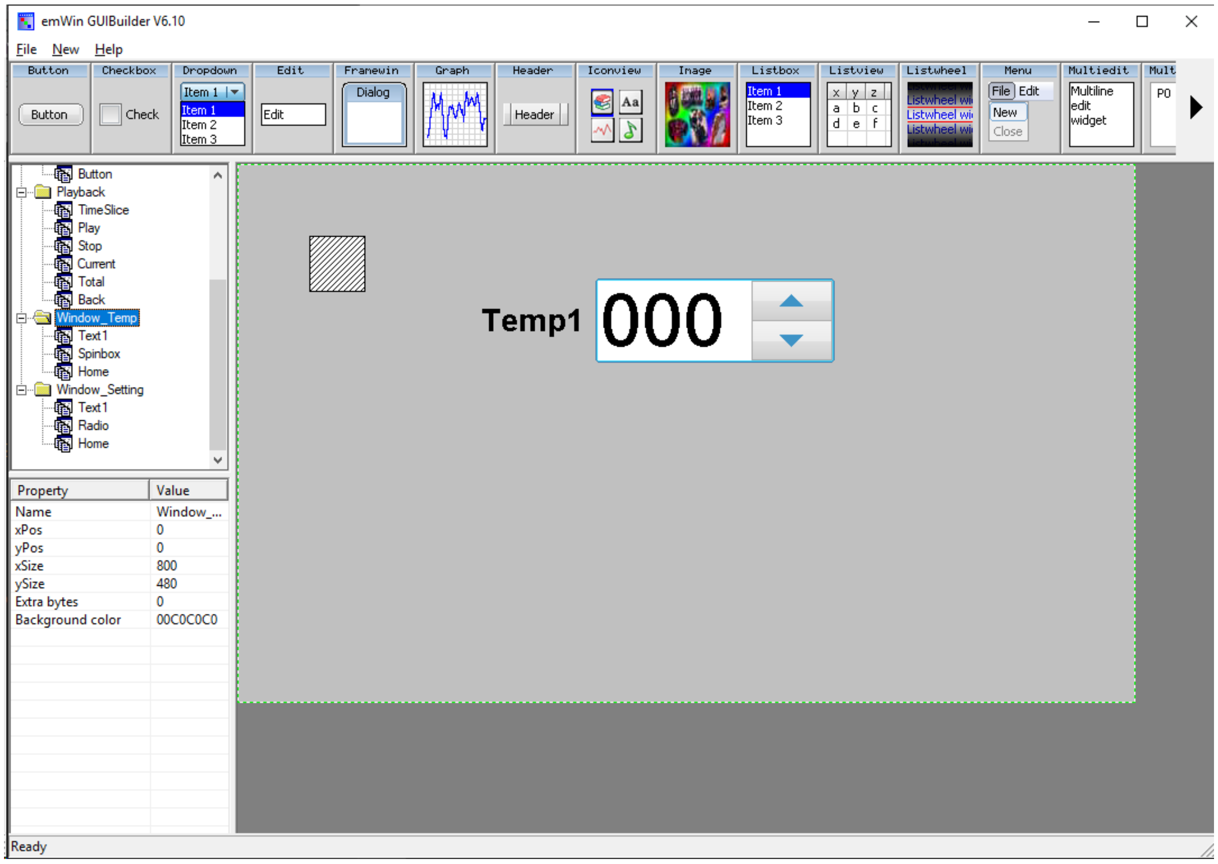
User defined dta file name in “*NVT_Playback1.c*”:

Note: the size of dta with alpha-channel is bigger than dta.

```
#define NVT_PLAYBACK_PLAY "D:\\Play_01.dta"
#define NVT_PLAYBACK_STOP "D:\\Pause_01.dta"
```

```
#define NVT_PLAYBACK_BACK "D:\\Home_01.dta"
```

5.3 Temperature and Setting Menu Control



Property	Value
Name	Window_...
xPos	0
yPos	0
xSize	800
ySize	480
Extra bytes	0
Background color	00C0C0C0

Figure 5-3 Temperature Menu

Temperature and setting menu are generated from GUIBuilder, called “*Window_TempDLG.c*” and “Window_*SettingDLG.c*”, respectively. You can open these files by GUIBuilder and re-arrange widget postion, size and property. “*Window_TempDLG.c*” contains 1 back BUTTON, 1 TEXT and 1 SPINBOX.

By “*NVT_Temp1.c*”, you can assign user define dta file to replace the original.

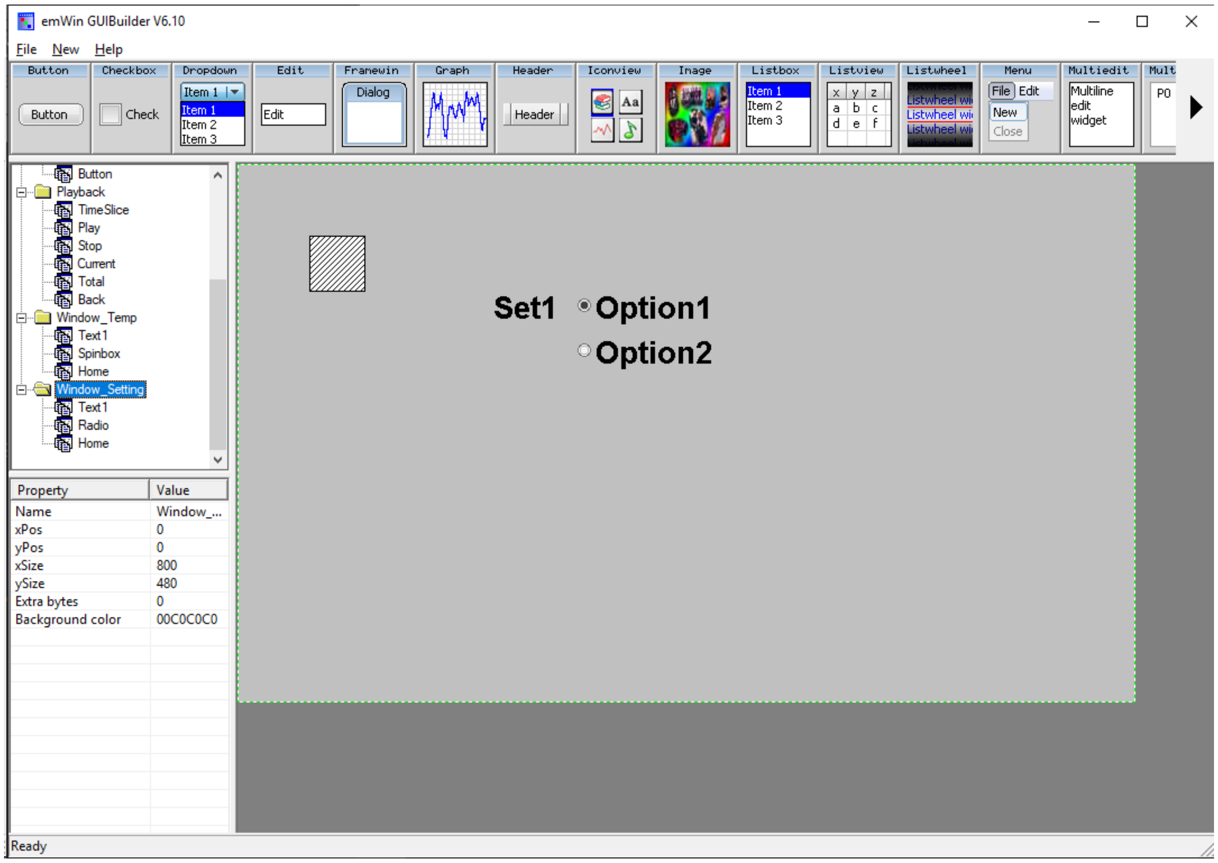
```
// USER START (Optionally insert additional code for further widget
initialization)

g_u8Flag1 = 0;



hItem = WM_GetDialogItem(pMsg->hWin, ID_IMAGE_0);

NVT_Temp1Back(hItem);



// USER END
```

Click IMAGE will back to cooking menu::

```
// USER START (Optionally insert additional code for further Ids)

case ID_IMAGE_0:

    g_u8Flag1 = 7;

    break;


// USER END
```

User defined dta file name in “*NVT_Temp1.c*”:

```
#define NVT_TEMP_BACK "D:\\Home_01.dta"
```



Property	Value
Name	Window_...
xPos	0
yPos	0
xSize	800
ySize	480
Extra bytes	0
Background color	00C0C0C0

Figure 5-4 Setting Menu

"*Window_SettingDLG.c*" contains 1 back BUTTON, 1 TEXT and 1 RADIO.

By "*NVT_Setting1.c*", you can assign user define dta file to replace the original.

```
// USER START (Optionally insert additional code for further widget
initialization)

g_u8Flag1 = 0;



hItem = WM_GetDialogItem(pMsg->hWin, ID_IMAGE_0);

NVT_Setting1Back(hItem);



// USER END
```

Click IMAGE will back to cooking menu::

```
// USER START (Optionally insert additional code for further Ids)

case ID_IMAGE_0:

    g_u8Flag1 = 8;
    break;


// USER END
```

User defined dta file name in “*NVT_Setting1.c*”:

```
#define NVT_SETTING_BACK "D:\\Home_01.dta"
```

6 FAQ

6.1 How to replace dta?

Use the same image filename, width and height and utilize 9To565 to convert to dta then copy to SD1-2.

6.2 How to replace MP4?

Use AnyToMp4 to convert to MP4 then copy to SD1-2 / DCIM.

6.3 How to utilize dta with alpha-channel?

Use 28ToA565 to convert to dta with alpha-channel then copy to SD1-2.

Note: need same image filename, width and height.

7 REVISION HISTORY

Date	Revision	Description
2020.12.10	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.