emWin AppWizard 设计 Dashboard

NuMicro® 64位系列微控制器范例代码介绍

文件信息

应用简述	本范例程序使用 SEGGER emWin AppWizard 工具设计 GUI，以仪表盘接口示范 AppWizard API 功能，联动 MA35H0 系列开发板的 LED 及按钮联动之功能设计。
BSP 版本	emWin Software Package for MA35 Family
开发平台	NuMaker-HMI-MA35H04F70 V1.0

The information described in this document is the exclusive intellectual property of Nuvoton Technology Corporation and shall not be reproduced without permission from Nuvoton.

Nuvoton is providing this document only for reference purposes of NuMicro microcontroller and microprocessor based system design. Nuvoton assumes no responsibility for errors or omissions.

All data and specifications are subject to change without notice.

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

www.nuvoton.com

1. 概述

本范例程序将示范如何使用 SEGGER emWin AppWizard 工具，搭配 NuMaker-HMI-MA35H0-A1 V1.0 开发平台，以仪表盘系统接口为例，逐步完成接口设计与人机互动功能，运用 AppWizard API 功能联动开发板上 LED 与按钮事件处理并实时更新状态于 GUI 上呈现，进而降低开发工时。

1.1 UI 设计

此范例程序使用新唐 GUI 开发平台 NuMaker-HMI-MA35H0-A1 V1.0 开发板，其带有尺寸为 7 寸的 TFT-LCD 电阻式触控面板，分辨率为 1024x600。于此项目中，UI 的接口设计工具采用 SEGGER emWin AppWizard（ 版本为 V1.40f_6.34g ）进行开发。AppWizard IDE 在开发上有非常直觉式 Editor 区块，用户可以通过将组件从对象拖拽到编辑器窗口来设计 UI 布局。参考如图 1-1 所示，编辑器窗口又带有 Objects、Hierarchic Tree、Resource 和 Properties 提供开发者更简易使用组件的属性设置，达到应用功能的需求。

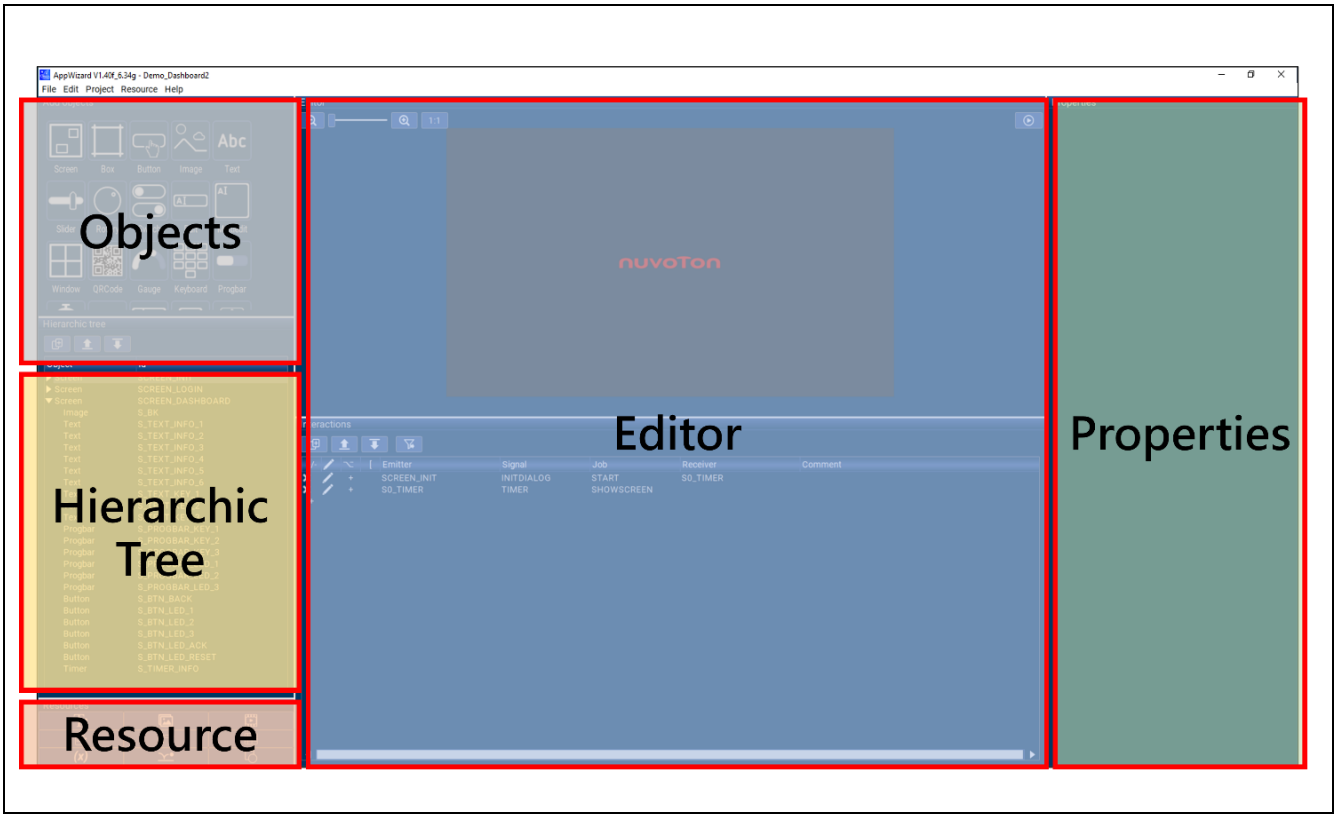

图 1-1 SEGGER emWin AppWizard 开发工具

关于 Dashboard 项目，用户可以参考其配置设置，如图 1-2 所示。NuMaker-HMI-MA35H0-A1 V1.0 开发板是采用 7 寸的屏幕，调整 Display size 为 1024 x 600 像素值；Color Format 选择 "32bit, GUICC_M8888I"并启动 Multibuffering 功能。

GUI 的架构如图 1-3 所示。共分为三个页面： Initialization Screen、Login Screen、Dashboard Screen。

Initialization Screen	显示起始页面，并设定 Timer，3 秒后跳转至 Login Screen。
Login Screen	展示用户登入接口，通过密码输入，正确即登入跳转至 Dashboard Screen。密码默认：1234。
Dashboard Screen	功能展示页面，通过开发板上及 LED 开关按钮联动控制，展示开发板当前状态并实时更新显示于画面窗口。

表 1-1 GUI 架构表

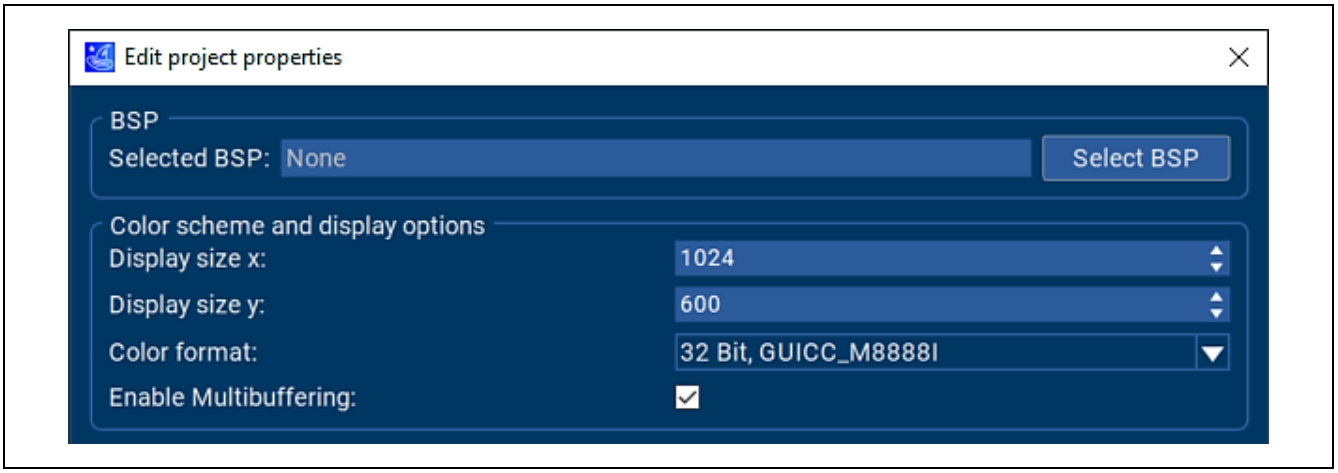

图 1-2 Dashboard 项目的参数设定

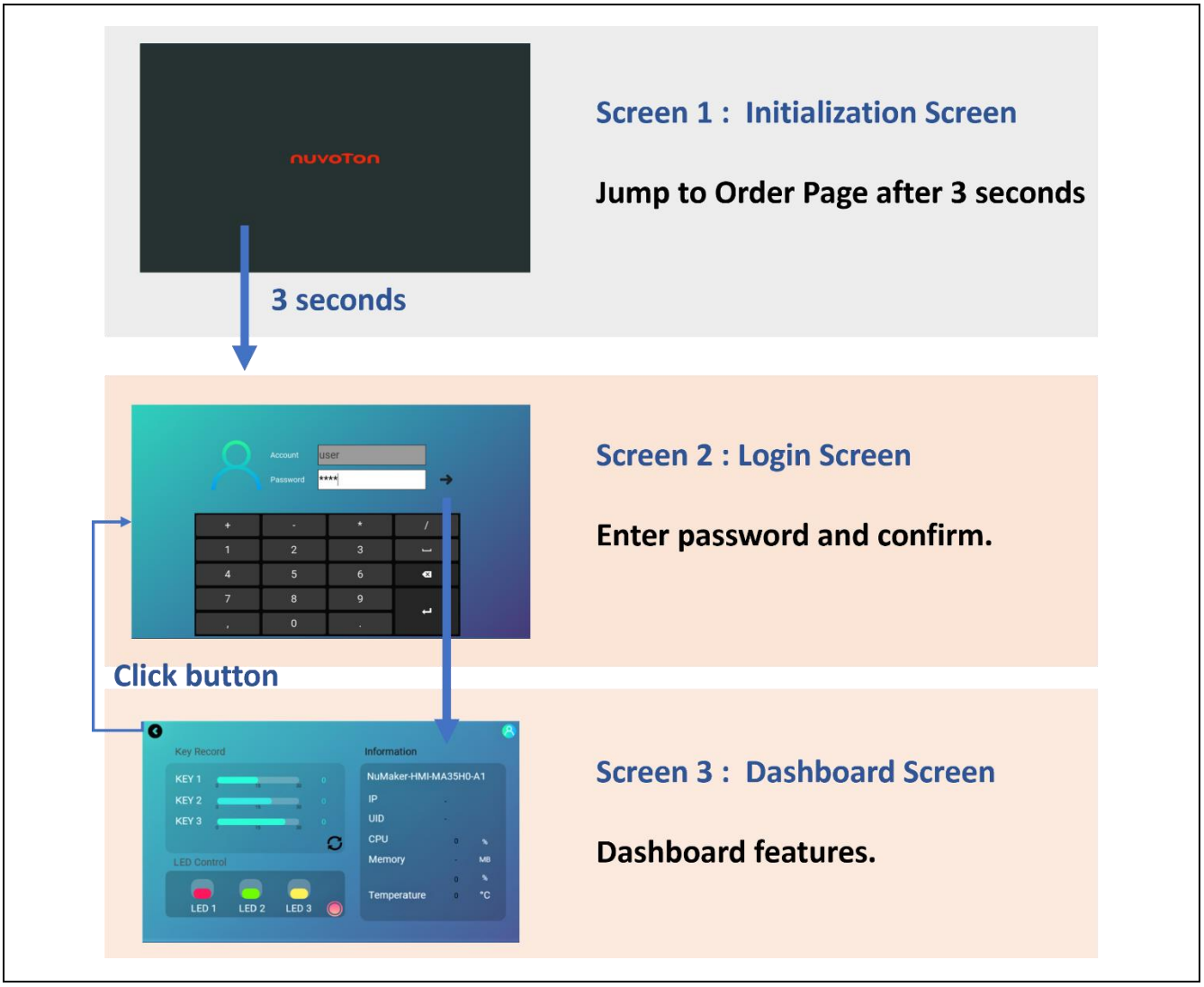

图 1-3 Dashboard 项目的流程图

在 Initialization Screen 设计上，配置如图 1-4 所示。使用到 Screen、Box 及 Image 对象。其中，Screen size 为 1024 x 600 像素值，背景颜色则是使用 Box 对象选择色彩 Red = 38，Green = 50，Blue = 50，Alpha = 255。

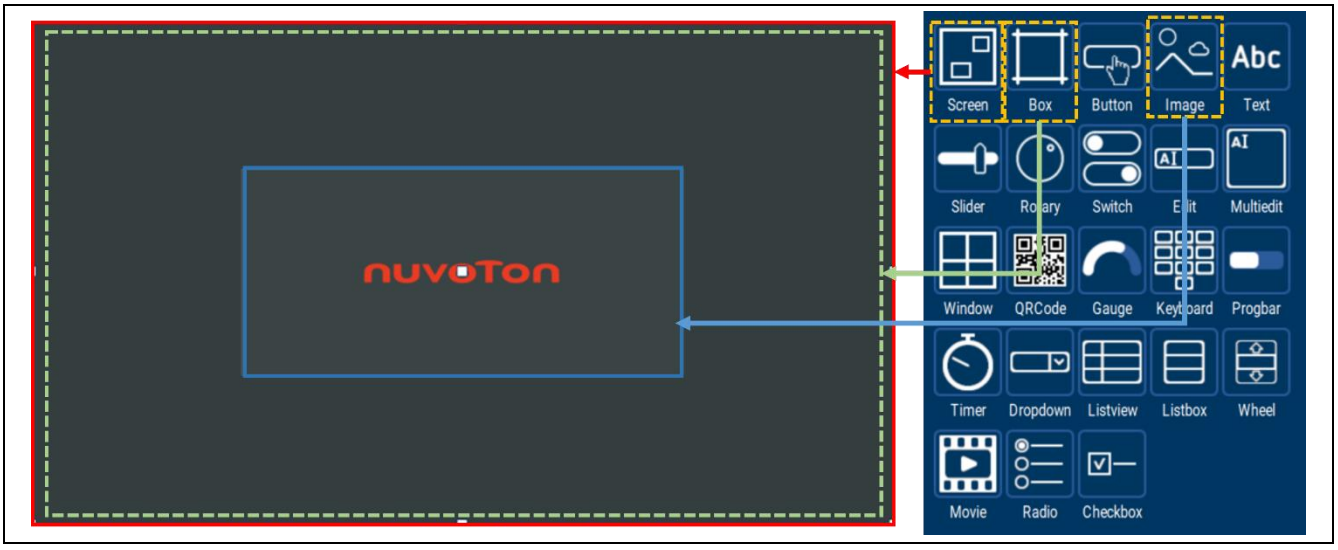

图 1-4 Initialization Screen 页面设计

在 Login Screen 页面设计为用户登入接口，配置如图 1-5 所示。输入帐户及密码字段使用 Edit 对象并设定最大数据长度为 10，键盘输入采用 keyboard 对象，设置 layout 为 SKEYBOARD_NUMPAD(Numeric Keypad)仅显示数字键盘。使用 Image 对象作为背景图，Image size 设置为 1024 x 600 像素大小。

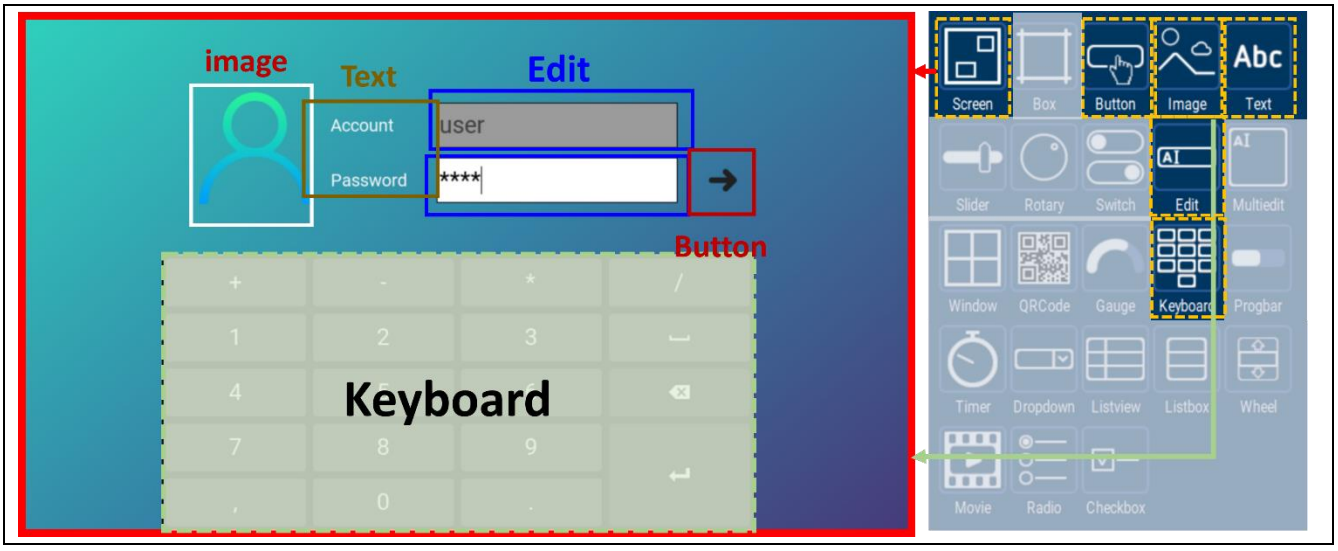

图 1-5 Login Screen 页面设计

在 Dashboard Screen 页面皆由 button 及 program bar 组合而成，在信息字段则由 text 对象依序排列。使用 Image 对象作为背景图，Image size 设置为 1024 x 600 像素大小，GUI 的对象配置如图 1-6 所示。

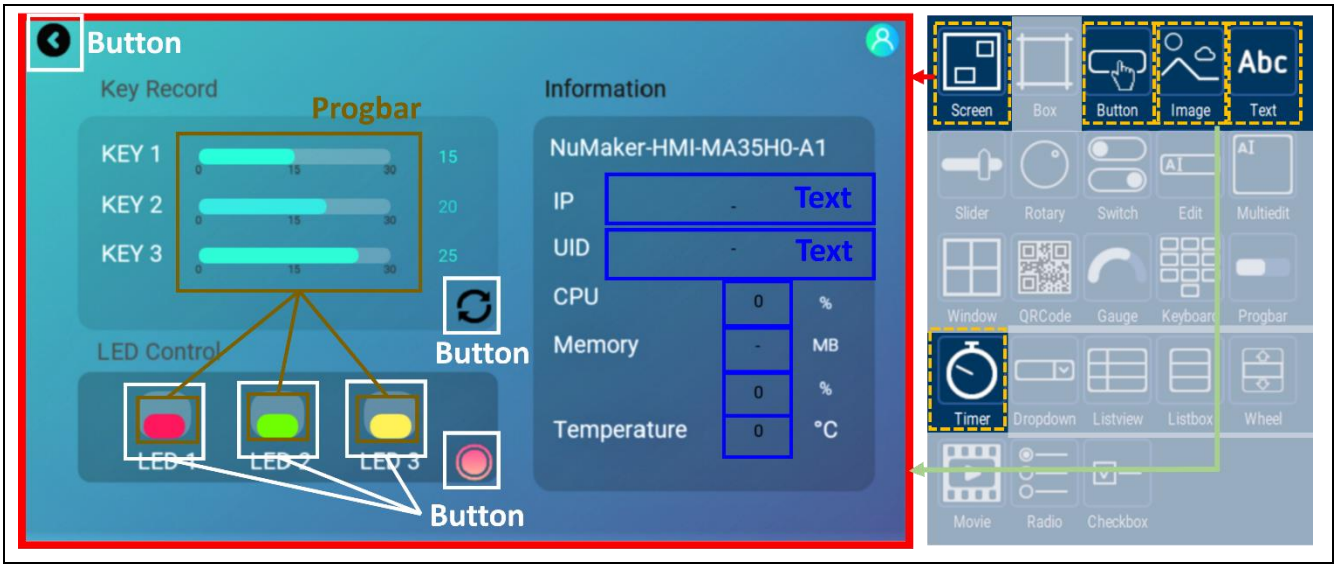

图 1-6 Dashboard Screen 页面设计

1.2 UI 功能设计

在 emWin AppWizard 有提供 Interactions 的功能，Interactions 可定义用户的行为操作及建立各对象间链接，如自动动画设计或滑动转场在屏幕之间切换。本项目的设计互动如下：

Initialization Screen 页面设定 timer = 3000，三秒后跳转至 Login Screen，设定 Interactions 如图 1-7 所示，Screen 初始化完成会启动 timer 并开始倒计时，三秒倒计时完成，会触发 SHOWSCREEN Job 跳转页面。

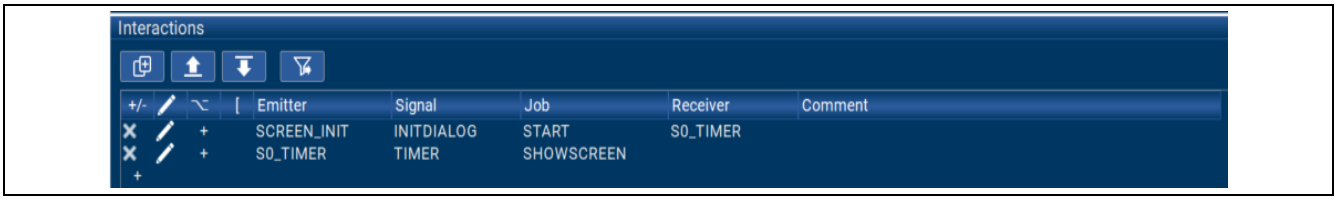

图 1-7 Initialization Screen 的 Interactions 设定

在 Screen 被建立时，自动填入帐户及密码的默认值，帐户设定「user」后设定 enable state = off，则无法被用户更改；密码则设定默认值「1234」。按下 S1_BUTTON_LOGIN 按钮后执行 NULL，设置呼叫函数，此函示识别用于密码的正确性，正确则跳转页面，错误则会清除 S1_EDIT_PASSWORD 当前输入的密码。

图 1-8 Login Screen 的 Interactions 设定

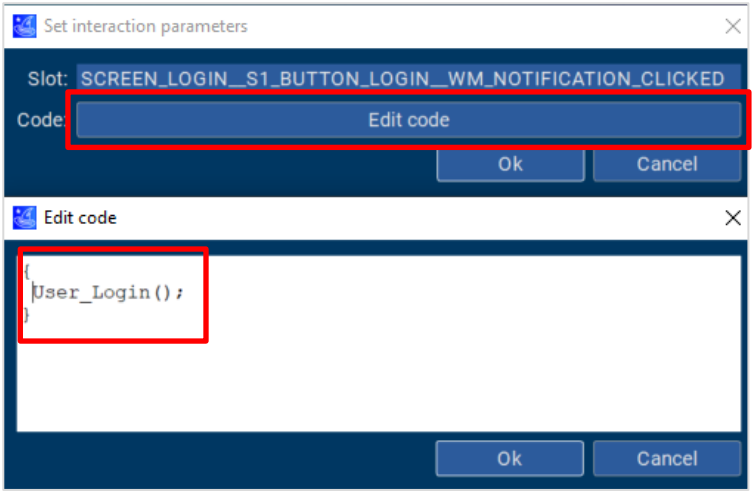

图1-9 按钮功能的呼叫函数设定

Dashboard Screen 页面被建立时，读取开发板系统信息设定更新 IP 及 UID。Key Control 及 LED Control 的控制上的 Interactions 设置如图 1-10 所示，设置按下 GUI 接口上的按钮，打开/关闭开发板上的 LED，反之联动开发板上的 Key，触发 GUI 接口上的计数功能。此 Dashboard Screen 的页面设定 timer = 2000，2 秒钟更新系统数值，更新 CPU loading、Memory 和 Temperature 数值。

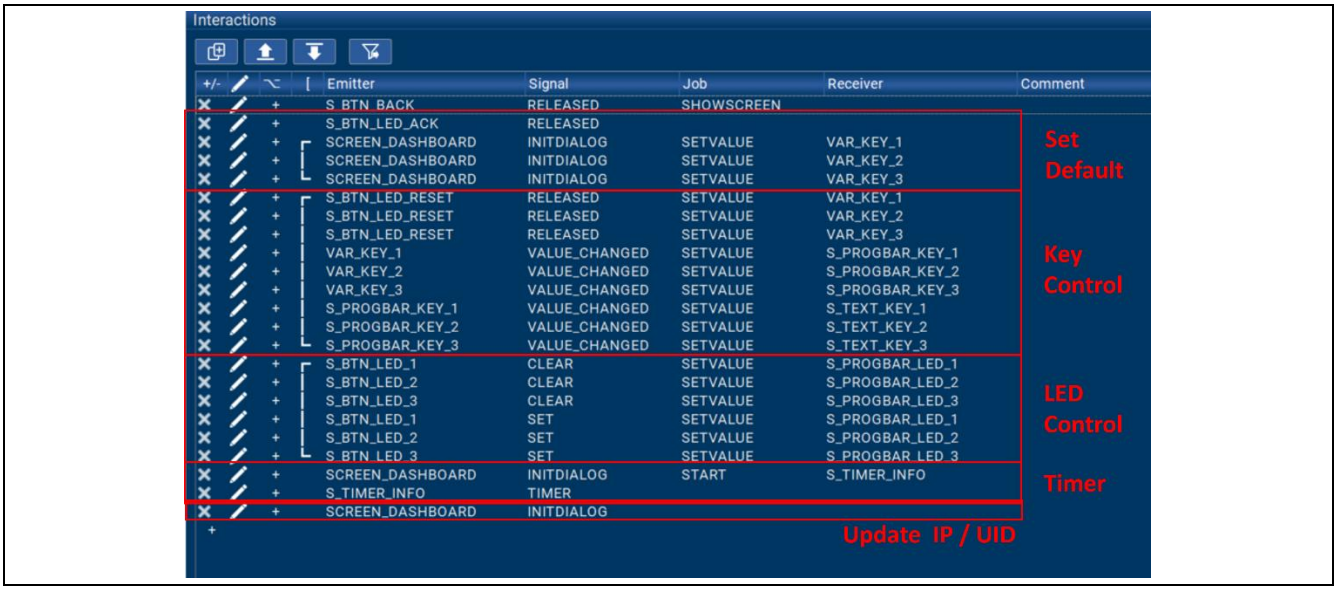

图 1-10 Screen 的 Properties 设置

完成 GUI 设计，AppWizard Tool 会自动将 GUI 的设计转成程序代码，用户按下 File > Export & Save 按钮，转换成对应的程序代码，如图 1-11。转换后的程序代码会放在 Source 与 Resource 的文件夹中。

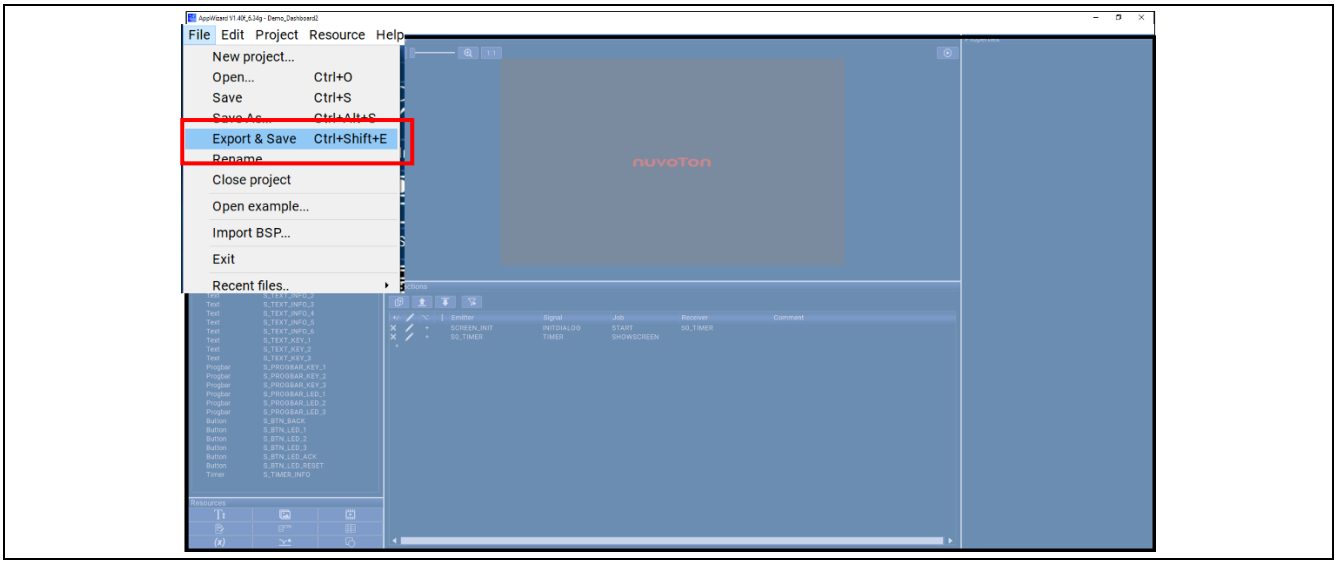

图 1-11 AppWizard Tool 的 Export & Save 位置

1.3 程序編譯

UI 设计完成后，需要将程序代码进行编译。待编译完成后，将生成执行文件 SimpleDemoAppWizard，该文件可在 NuMaker-HMI-MA35H0-A1 V1.0 开发平台上运行。依 NuMaker-HMI-MA35H0-A1 V1.0 为例，以下说明 Linux 环境编译流程：

1. 将项目 MA35_Family_Linux_emWin_Package 文件夹移动到 MA35H0 Linux 编译环境中。

2. 将 AppWizard 专案导出的 Source 与 Resource 文件夹覆盖到 MA35_Family_Linux_emWin_Package > Sample > Dashboard > Application 的同名文件夹。在交叉编译过程，会将 GUI 程序代码连同项目一同编译成执行文件。

3. 在运行交叉编译时，用户需要注意工具链(toolchain)的设置。在编译构建上，需手动修改 Makefile 文件中的 CC 和 STRIP 字段，将其修改为 MA35H0 工具链设置。

```
CC=~/MA35D1_Buildroot/output/host/bin/aarch64-nuvoton-linux-gnu-gcc
STRIP=~/MA35D1_Buildroot/output/host/bin/aarch64-nuvoton-linux-gnu-strip
```

如果缺少 toolchain 的文件，需执行环境的编译，请参考 MA35H0 Buildroot Project User Manual 文件。

4. 通过 make 指令进行编译：

```
~/MA35_Family_Linux_emWin_64bit_Package/Sample/Dashboard$ make
```

若要清除编译文件或重新编译，皆需执行以下指令：

```
~/MA35_Family_Linux_emWin_64bit_Package/Sample/Dashboard$ make clean
```

关于 Linux 编译，请参考 MA35H0 Linux User Manual 说明。

5. 编译完成后，将生成执行文件 SimpleDemoAppWizard，该文件可在 NuMaker-HMI-MA35H0-A1 V1.0 开发平台上运行。

6. 用户可以通过 U 盘来移动文件。将执行檔(SimpleDemoAppWizard)及脚本(getDeciceInfo.sh 和 getSysMonitor.sh)放到 U 盘。等待 NuMaker-HMI-MA35H0-A1 V1.0 开机后，运行挂载指令：

```
$ mount /dev/sda1 /mnt
```

移动到 mnt 的文件夹：

```
$ cd /mnt
```

运行执行文件

```
$ ./SimpleDemoAppWizard
```

倘若第一次运行 NuMaker-HMI-MA35H0-A1 V1.0 需要设置屏幕校正点。将 ts.conf 通过 U 盘复制到开发平台：

```
$ cp /mnt/ts.conf /etc/
```

运行 ts_calibrate 指令进行屏幕五点校正：

```
$ /usr/bin/ts_calibrate
```

2. 代码介绍

转换成对应的代码，用户可以在 ~/Application/Source/CustomCode/Application.c 加入自定义的函数功能。程序开头需 include 会使用到的页面。本程序代码使用 AppWizard API 请参考使用手册，下表列出本范例程序会使用到的 AppWizard API 函数。

Routine	Description
APPW_GetText()	This function stores the text of an object in the given buffer.
APPW_SetText()	This function sets a text to an object.
APPW_GetValue()	This function sets the value of an object.
APPW_SetValue()	This function returns the value of an object.

表 2-1 AppWizard API 函数表

```
// Main application header file
#include "Application.h"

// Header for dashboard screen generation
#include "Application/Source/Generated/SCREEN_DASHBOARD.h"
#include "Application/Source/Generated/SCREEN_LOGIN.h"
#include "Application/Source/Generated/Resource.h

// Path to the input device (e.g., keyboard or mouse)
#define DEVICE "/dev/input/event1"

// Pointer to an integer that stores an error code
int *error;
```

LED 控制程序码如下所示，取得当前 GUI LED 状态后决定 LED 的亮灭行为。如果当前 LED 状态是恒亮，则按下按钮则会让LED熄灭，反之LED状态是熄灭，则按下按钮则会让LED点亮：

```
/*=============================================================================================*
** Function : Control LED on and off
** Output   : None
** use      : LED_Toggle(1);
**=============================================================================================*/
void LED_Toggle(char led) {

    int status;
    char command[256];
    char meg[256];
```

```
    // Check the status of the specified LED
    if (led == 0) {
        status = APPW_GetValue(SCREEN_DASHBOARD, S_PROGBAR_LED_1, error);
    } else if (led == 1) {
        status = APPW_GetValue(SCREEN_DASHBOARD, S_PROGBAR_LED_2, error);
    } else if (led == 2) {
        status = APPW_GetValue(SCREEN_DASHBOARD, S_PROGBAR_LED_3, error);
    } else {
        // Invalid LED number, log an error message and return
        printf("[E] Invalid LED number %d \n", led + 1);
        return;
    }

    // Toggle the LED state based on its current status
    if (status == 100) {
        // If LED is on (status 100), turn it off
        snprintf(command, sizeof(command), "echo 0 > /sys/class/leds/LED%d/brightness",
led);
        // Execute the command to toggle the LED
        system(command);
        printf("[L] Control LED%d OFF ! \n", led + 1);

    } else {
        // If LED is off, turn it on
        snprintf(command, sizeof(command), "echo 1 > /sys/class/leds/LED%d/brightness",
led);
        // Execute the command to toggle the LED
        system(command);
        printf("[L] Control LED%d ON ! \n", led + 1);
    }
}
```

开发板上的按钮读取程序如下所示，启动 thread 反复检测开发板上按钮是否被按下，当按下按钮后，则更新计数到 GUI 上的次数显示。

```
/**
 * Thread function to monitor button events.
 * This function continuously reads input events from a device and updates
 * the application variables based on specific key events.
 */

void* button_event(void* arg) {
    int fd, count = 0;
    struct input_event ev;

    // Open the input device for reading
    fd = open(DEVICE, O_RDONLY);
    if (fd == -1) {
        printf("[E] Failed to open device \n");
        pthread_exit(NULL);  // Use pthread_exit instead of return
    }

    printf("[!] Monitoring key events...\n");
    while (1) {
        // Read input event from the device
```

```
        ssize_t bytesRead = read(fd, &ev, sizeof(struct input_event));

        if (bytesRead < sizeof(struct input_event)) {
            printf("[E] Failed to read input event \n");
            close(fd);
            pthread_exit(NULL);
        }

        // Process the key event
        if (ev.type == EV_KEY) {
            // Update the application variable based on the key code
            if (ev.code == 105 && ev.value == 1) {
                count = APPW_GetVarData(VAR_KEY_1, error) + 1;
                APPW_SetVarData(VAR_KEY_1, count);
            }

            … …
        }
    }

    close(fd);
    pthread_exit(NULL);  // End thread
}
```

代码刚开始运行时，会通过读取脚本方式取得开发板上的系统信息，代码如下：

```
/*=============================================================================================*
** Function : Device information update
** Output   : 0 | 1: error
** use      : UpdateDeviceInfo();
**=============================================================================================*/

int UpdateDeviceInfo(void) {
    FILE *fp;
    char output[100];
    char IP[256], UUID[256];
    char *delimiter;
    char meg[256];

    // Run the shell script to get device information
    fp = popen("./getDeciceInfo.sh", "r");
    if (fp == NULL)  return 1;

    … …

    pclose(fp);

    // Remove the newline character from the output
    output[strcspn(output, "\n")] = '\0';

    // Split the output string using ';' as a delimiter
    delimiter = strchr(output, ';');
    if (delimiter != NULL) {
        *delimiter = '\0';

        // Get UUID text
```

```
        strncpy(UUID, delimiter + 1, sizeof(UUID) - 1);
        // Get IP text
        strncpy(IP, output, sizeof(IP) - 1);
        // Clean up the strings
        IP[sizeof(IP) - 1] = '\0';
        UUID[sizeof(UUID) - 1] = '\0';

        // Update the screen with the new IP and UUID
        APPW_SetText(SCREEN_DASHBOARD, S_TEXT_INFO_1, IP);
        APPW_SetText(SCREEN_DASHBOARD, S_TEXT_INFO_2, UUID);

        … …
    }
    … …

    return 0;
}
```

本范例程序通过 command line 方式取得开发板上 IP 及 UID 信息，脚本 getDeciceInfo.sh 的程序代码如下所示：

```
#!/bin/sh

# Get the IP address for eth0
eth0_IP="$(ifconfig | grep -Eo 'inet (addr:)?([0-9]*\.){3}[0-9]*' | grep -Eo '([0-
9]*\.){3}[0-9]*' | grep -v '127.0.0.1')"

# Read memory addresses and concatenate them to form UID
uid_part0="$(devmem 0x40460180)"
uid_part1="$(devmem 0x40460184)"
uid_part2="$(devmem 0x40460188)"
UID="${uid_part0#0x}${uid_part1#0x}${uid_part2#0x}"

# Print the eth0 IP and the UID variable
echo "${eth0_IP};${UID}"
```

本范例程序通过 command line 方式取得开发板上 CPU loading、memory 使用率与温度信息，脚本 getSysMonitor.sh 的程序代码如下所示：

```
#!/bin/sh

# Define a function to read the system status
sys_status() {
  # Output the CPU usage
  echo -n "$(cpu_usage);"

  # Output the RAM usage
  echo -n "$(ram_usage);"

}

# Retrieve CPU usage
cpu_usage() {
  # Get the CPU idle value
```

```
  cpu=$(top -bn1 | grep "CPU:" | grep -v "grep" | awk '{ print $2}' | sed 's/%//')

  # Get the user value
  usr=$(top -bn1 | grep "CPU:" | grep -v "grep" | awk '{ print $4}' | sed 's/%//')

  # Output the sum of CPU and user values
  echo $((cpu + usr))
}

# Retrieve RAM usage
ram_usage() {
  free -m | awk '/Mem:/ { printf("%d/%d MB (%.2f%%)", $3, $2, $3/$2 * 100) }'
}


# Output information
# Decide which function to execute based on the argument
case "$1" in
  "Sys Status")
    sys_status
    ;;
  *)
    sys_status
    ;;
esac
```

温度信息，请参考 MA35H0_Linux User Manual 说明，于设置 MA35H0 开发平台时，用户需自行设置开启温度传感器功能 ：

```
Device Drivers --->
  [*] Thermal drivers --->
    [*] Expose thermal sensors as hwmon device
    [*] APIs to parse thermal data out of device tree
    [*] Enable writable trip points
    [*] User_space thermal governor
    [*] Generic device cooling support
          Nuvoton thermal drivers --->
              <*> MA35H0 Temperature Sensor
```

开启后，通过 command line 方式，以获取开发板温度信息：

```
$ cat /sys/class/thermal/thermal_zone0/temp
58
```

3. 软件与硬件需求

3.1 软件需求

- AppWizard 版本
  - AppWizard_V1.40f_6.34g

3.2 硬件需求

- 电路组件
  - NuMaker-HMI-MA35H0-A1 V1.0

4. 目录信息

目录结构如下图所示。

EC_MA35_AppWizard_Dashboard_V1.00	
AppWizard_Project	
Firmware	Program scripts and executable file
Resource	Image files used for this example
Source	Source file of example code
Simulation	AppWizard simulation file
MA35_Family_Linux_emWin_Package	
Config	LCD configuration file
Include	GUI related function library
Lib	AppWizard library file
Sample	
Dashboard	
Application	
Resource	Image files used for this example
Source	Source file of example code

图 4-1 目录信息

5. 范例程序执行

1. 将 AppWizard_Project > Firmware 文件夹下的执行文件及脚本复制到 NuMaker-HMI-MA35H0-A1 V1.0 开发板上运行。

   - 执行文件： EC_MA35_Dashboard
   - 脚本： getDeciceInfo.sh、getSysMonitor.sh
   - 如何移动执行文件，请参阅章节 1.3 第 6 项说明。

2. 执行代码 EC_MA35_Dashboard。

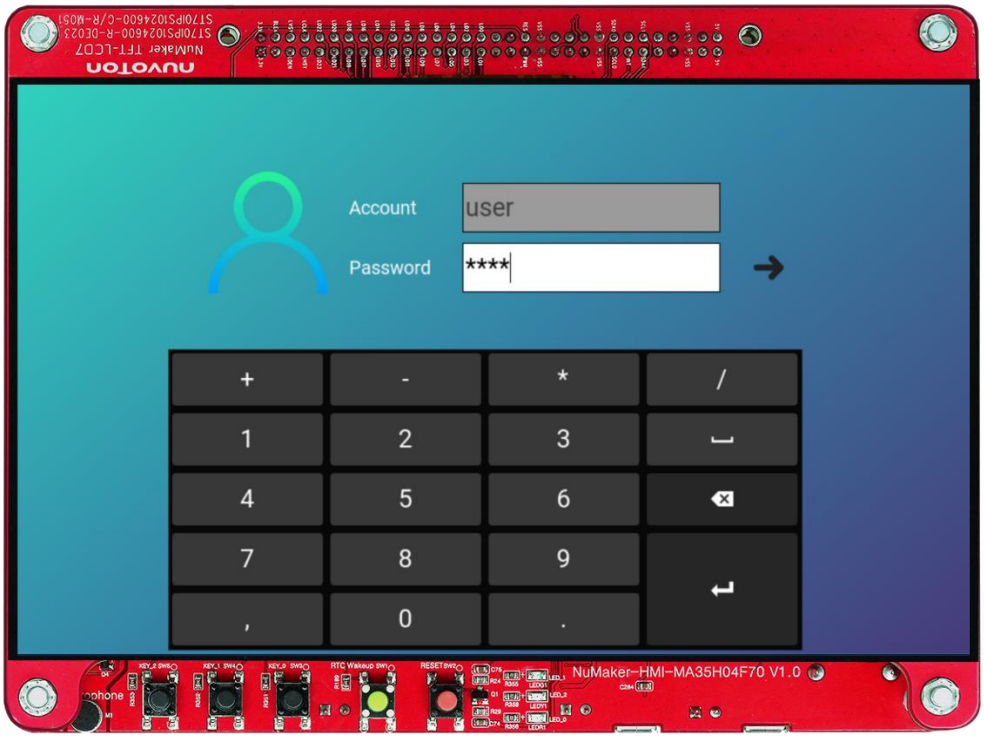

图 5-1 执行结果

6. 修订纪录

Date	Revision	Description
2024.09.10	1.00	初始发布。

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.

Please note that all data and specifications are subject to change without notice.
All the trademarks of products and companies mentioned in this datasheet belong to their respective owners.