emWin AppWizard 設計 Dashboard

NuMicro® 64位系列微控制器範例代碼介紹

文件資訊

應用簡述	本範例程式使用 SEGGER emWin AppWizard 工具設計 GUI，以儀表板介面示範 AppWizard API 功能，連動 MA35H0 系列開發板的 LED 及按鈕連動之功能設計。
BSP 版本	emWin Software Package for MA35 Family
開發平台	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

本範例程式將示範如何使用 SEGGER emWin AppWizard 工具，搭配 NuMaker-HMI-MA35H0-A1 V1.0 開發平台，以儀表板系統介面為例，逐步完成介面設計與人機互動功能，運用 AppWizard API 功能連動開發板上 LED 與按鈕事件處理並即時更新狀態於 GUI 上呈現，進而降低開發工時。

1.1 UI 設計

此範例程式使用新唐 GUI 開發平台 NuMaker-HMI-MA35H0-A1 開發板，其帶有尺寸為 7 吋的 TFT-LCD 電阻式觸控面板，解析度為 1024x600。於此專案中，UI 的介面設計工具採用 SEGGER emWin AppWizard (版本為 V1.40f_6.34g) 進行開發。AppWizard IDE 在開發上有非常直覺式 Editor 區塊，使用者可以透過將元件從物件拖曳到編輯器視窗來設計 UI 佈局。參考如圖 1-1 所示，編輯器視窗又帶有 Objects、Hierarchic Tree、Resource 和 Properties 提供開發者更簡易使用元件的屬性設置，達到應用功能的需求。

圖 1-1 SEGGER emWin AppWizard 開發工具

關於 Dashboard 專案，使用者可以參考其配置設置，如圖 1-2 所示。NuMaker-HMI-MA35H0-A1 V1.0 開發板是採用 7 吋的屏幕，調整 Display size 為 1024 x 600 像素值；Color Format 選擇"32bit, GUICC_M8888I"並啟動 Multibuffering 功能。

GUI 的架構如圖 1-3 所示。共分為三個頁面： Initialization Screen、Login Screen、Dashboard Screen。

Initialization Screen	顯示起始頁面，並設定 Timer，3 秒後跳轉至 Login Screen。
Login Screen	展示使用者登入介面，透過密碼輸入，正確即登入跳轉至 Dashboard Screen。密碼預設：1234。
Dashboard Screen	功能展示頁面，透過開發板上及 LED 開關按鈕連動控制，展示開發板當前狀態並即時更新顯示於畫面視窗。

表 1-1 GUI 架構表

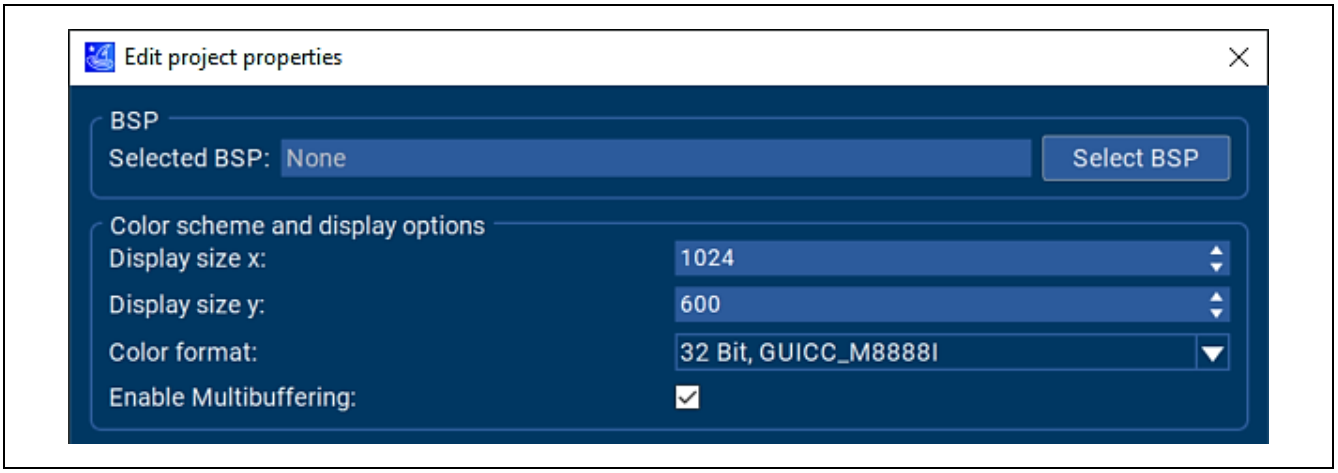

圖 1-2 Dashboard 專案的參數設定

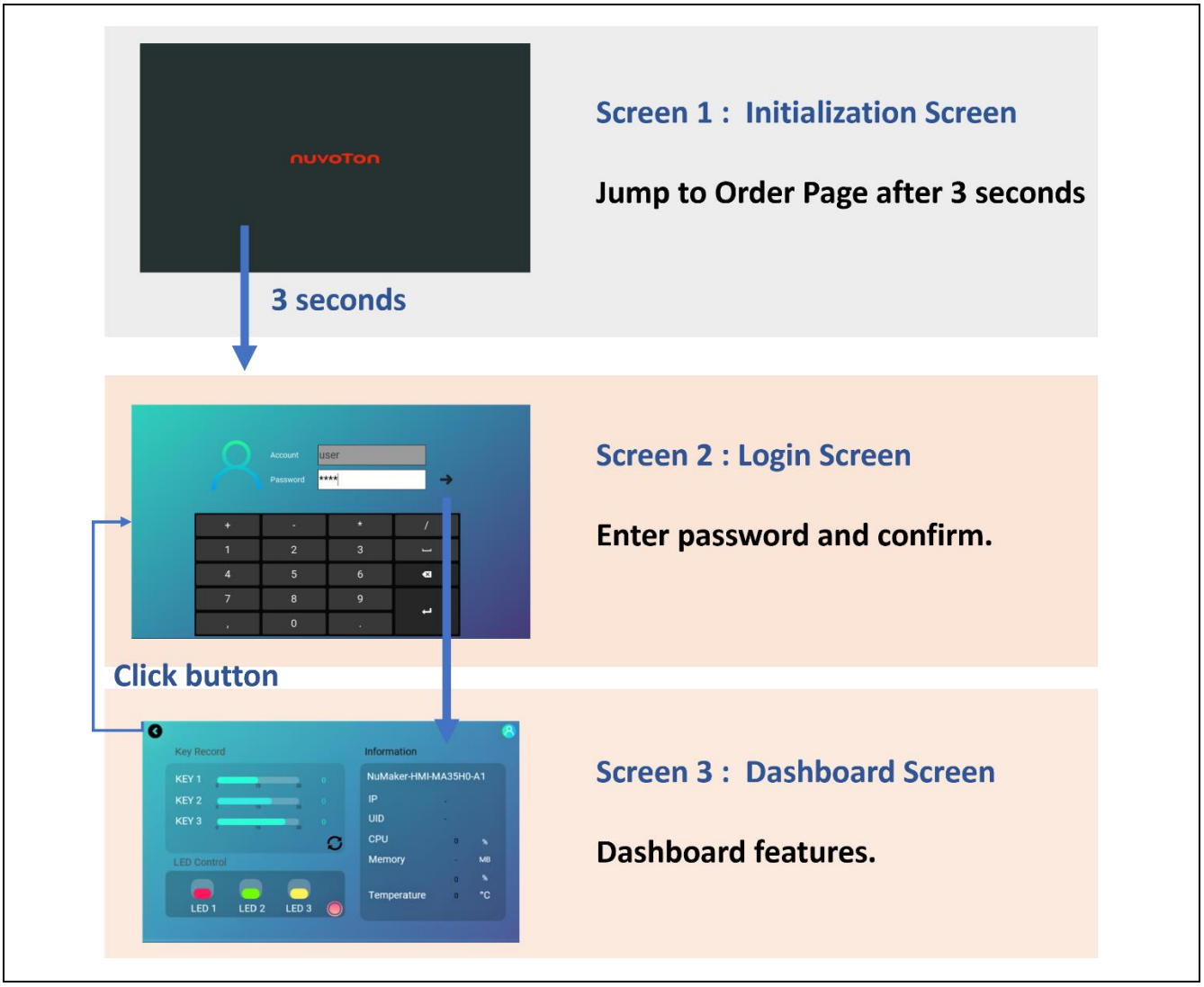

圖 1-3 Dashboard 專案的流程圖

在 Initialization Screen 設計上，配置如圖 1-4 所示。使用到 Screen、Box 及 Image 物件。其中，Screen size 為 1024 x 600 像素值，背景顏色則是使用 Box 物件選擇色彩 Red = 38，Green = 50，Blue = 50，Alpha = 255。

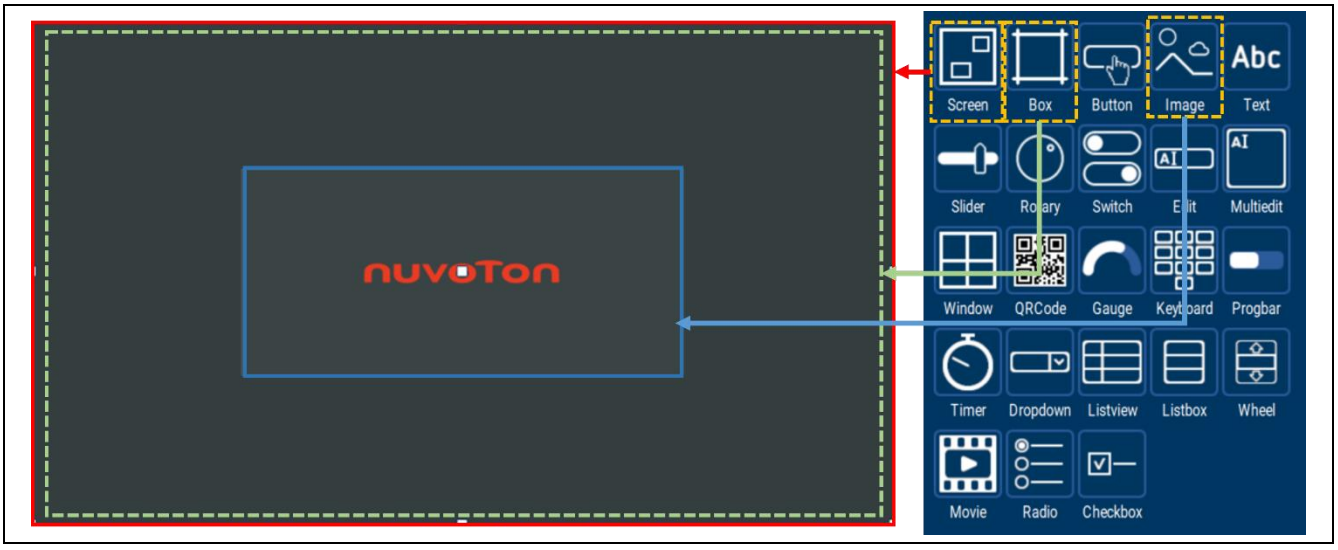

圖 1-4 Initialization Screen 頁面設計

在 Login Screen 頁面設計為使用者登入介面，配置如圖 1-5 所示。輸入帳戶及密碼欄位使用 Edit 物件並設定最大資料長度為 10，鍵盤輸入採用 keyboard 物件，設置 layout 為 SKEYBOARD_NUMPAD(Numeric Keypad)僅顯示數字鍵盤。使用 Image 物件作為背景圖，Image size 設置為 1024 x 600 像素大小。

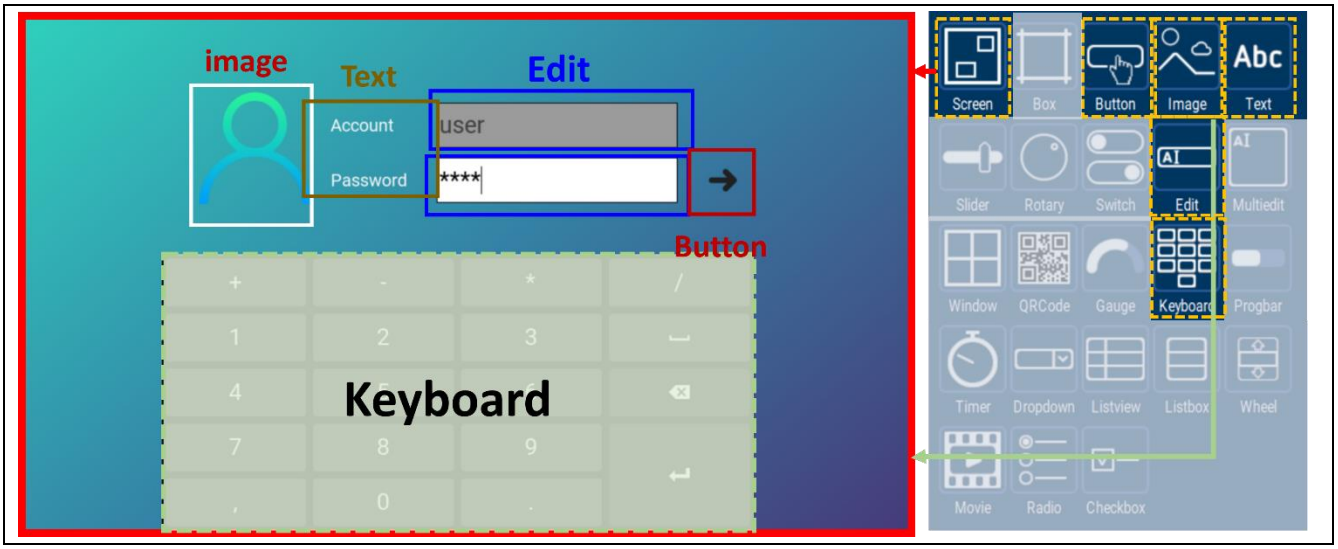

圖 1-5 Login Screen 頁面設計

在 Dashboard Screen 頁面皆由 button 及 program bar 組合而成，在資訊欄位則由 text 物件依序排列。使用 Image 物件作為背景圖，Image size 設置為 1024 x 600 像素大小，GUI 的物件配置如圖 1-6 所示。

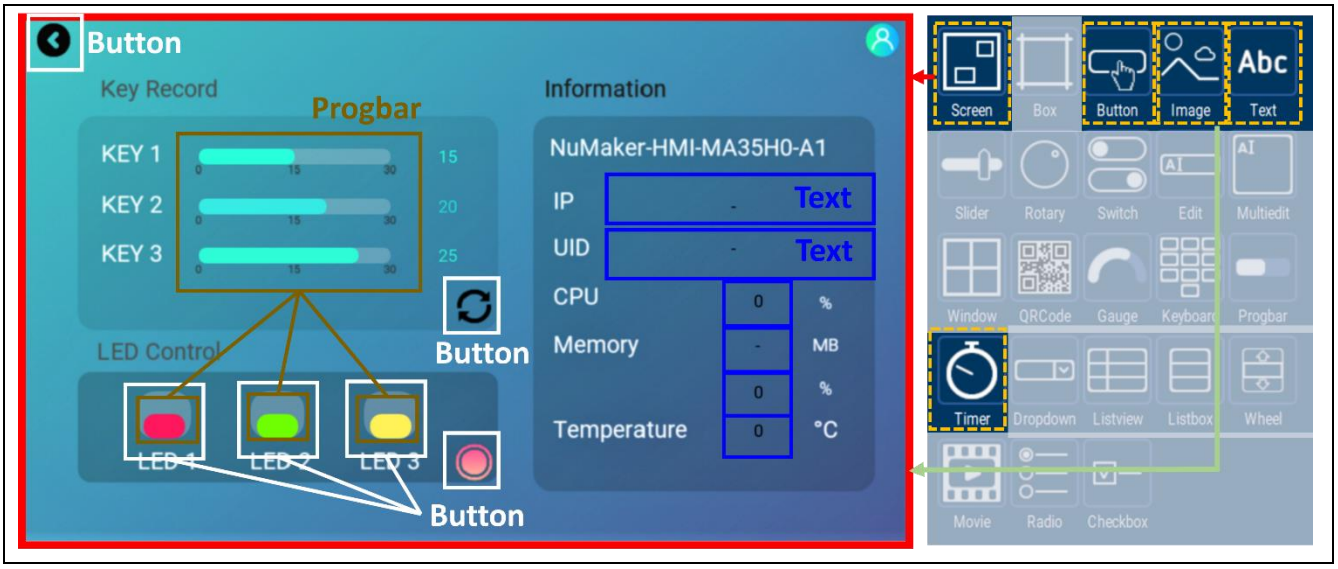

圖 1-6 Dashboard Screen 頁面設計

1.2 UI 功能設計

在 emWin AppWizard 有提供 Interactions 的功能，Interactions 可定義使用者的行為操作及建立各物件間連結，如自動動畫設計或滑動轉場在螢幕之間切換。本專案的設計互動如下：

Initialization Screen 頁面設定 timer = 3000，三秒後跳轉至 Login Screen，設定 Interactions 如圖 1-7 所示，Screen 初始化完成會啟動 timer 並開始倒計時，三秒倒計時完成，會觸發 SHOWSCREEN Job 跳轉頁面。

圖 1-7 Initialization Screen 的 Interactions 設定

在 Screen 被建立時，自動填入帳戶及密碼的預設值，帳戶設定「user」後設定 enable state = off，則無法被用戶更改；密碼則設定預設值「1234」。按下 S1_BUTTON_LOGIN 按鈕後執行 NULL，設置呼叫函式，此函示識別用於密碼的正確性，正確則跳轉頁面，錯誤則會清除 S1_EDIT_PASSWORD 當前輸入的密碼。

圖 1-8 Login Screen 的 Interactions 設定

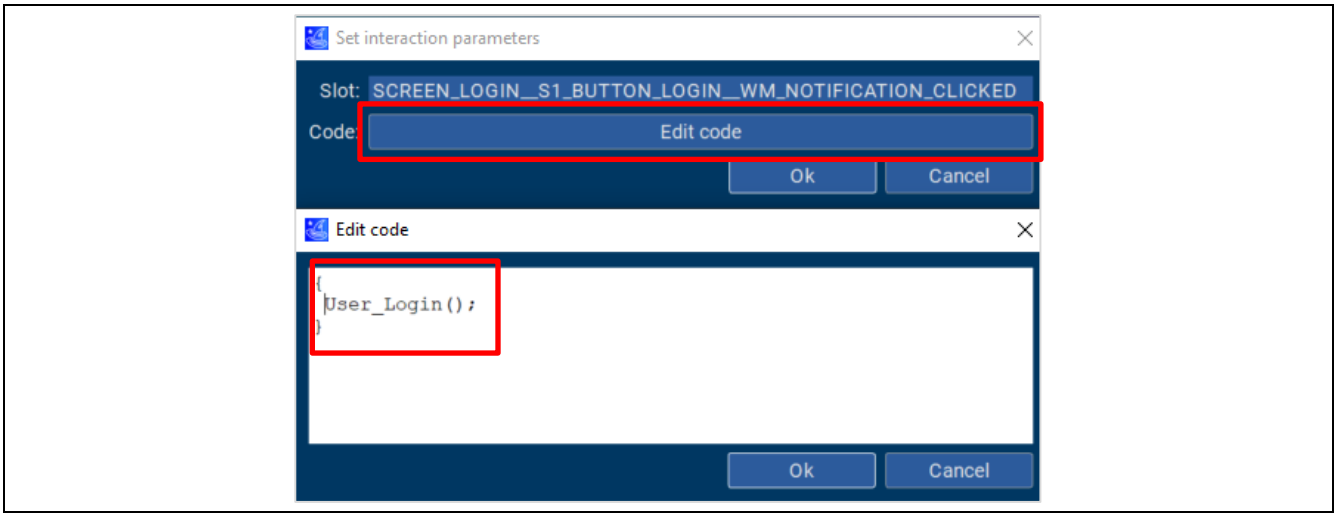

圖 1-9 按鈕功能的呼叫函式設定

Dashboard Screen 頁面被建立時，讀取開發板系統資訊設定更新 IP 及 UID。Key Control 及 LED Control 的控制上的 Interactions 設置如圖 1-10 所示，設置按下 GUI 介面上的按鈕，打開/關閉開發板上的 LED，反之連動開發板上的 Key，觸發 GUI 介面上的計數功能。此 Dashboard Screen 的頁面設定 timer = 2000，2 秒鐘更新系統數值，更新 CPU loading、Memory 和 Temperature 數值。

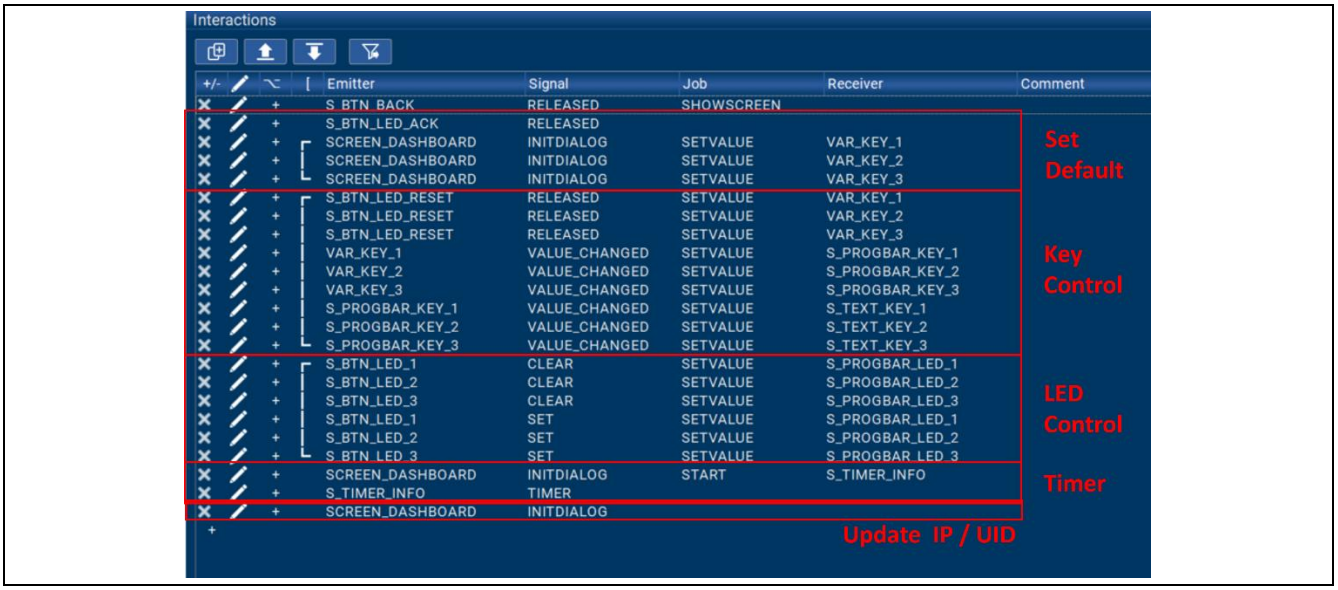

圖 1-10 Dashboard Screen 的 Interactions 設定

完成 GUI 設計，AppWizard Tool 會自動將 GUI 的設計轉成程式碼，使用者按下 File > Export & Save 按鈕，轉換成對應的程式碼，如圖 1-11。轉換後的程式碼會放在 Source 與 Resource 的資料夾中。

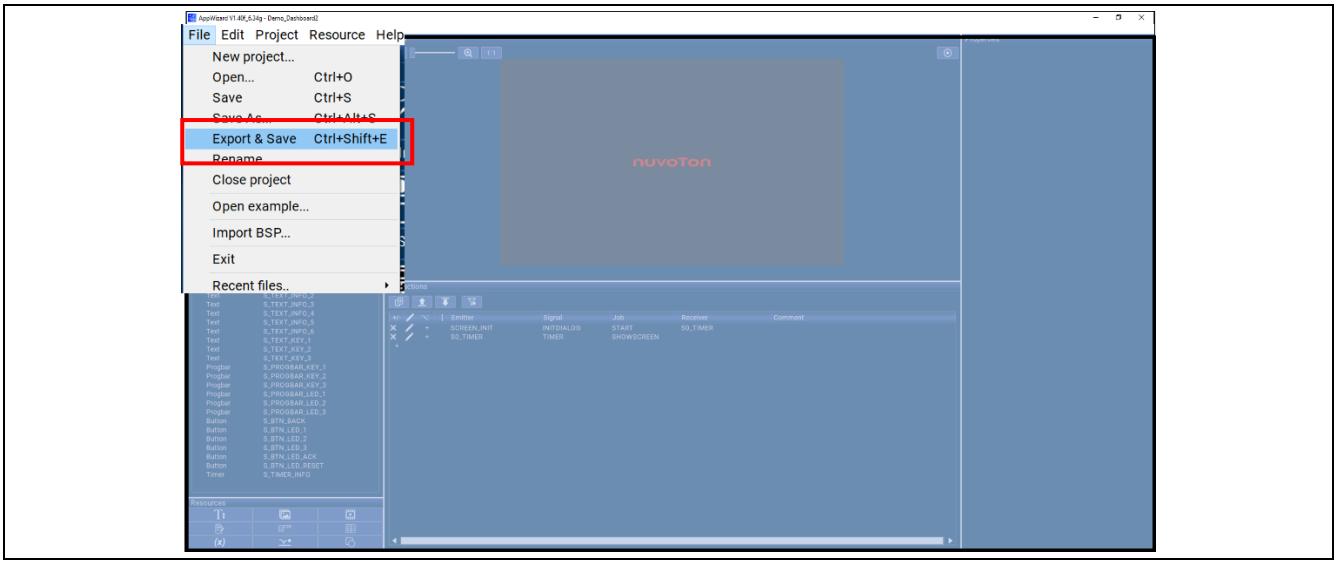

圖 1-11 AppWizard Tool 的 Export & Save 位置

1.3 程式編譯

UI 設計完成後，需要將程式碼進行編譯。待編譯完成後，將生成執行檔 SimpleDemoAppWizard，該檔案可在 NuMaker-HMI-MA35H0-A1 V1.0 開發平台上運行。依 NuMaker-HMI-MA35H0-A1 V1.0 為例，以下說明 Linux 環境編譯流程：

1. 將專案 MA35_Family_Linux_emWin_Package 資料夾移動到 MA35H0 Linux 編譯環境中。

2. 將 AppWizard 專案導出的 Source 與 Resource 資料夾覆蓋到 MA35_Family_Linux_emWin_Package > Sample > Dashboard > Application 的同名資料夾。在交叉編譯過程，會將 GUI 程式碼連同專案一同編譯成執行檔。

3. 在運行交叉編譯時，用戶需要注意工具鏈(toolchain)的設置。在編譯構建上，需手動修改 Makefile 文件中的 CC 和 STRIP 欄位，將其修改為 MA35H0 工具鏈設置。

```
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 的檔案，需執行環境的編譯，請參考 MA35H0 Buildroot Project User Manual 文件。

4. 透過 make 指令進行編譯：

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

若要清除編譯檔案或重新編譯，皆需執行以下指令：

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

關於 Linux 編譯，請參考 MA35H0 Linux User Manual 說明。

5. 編譯完成後，將生成執行檔 SimpleDemoAppWizard，該檔案可在 NuMaker-HMI-MA35H0-A1 開發平台上運行。

6. 用戶可以透過 USB 隨身碟來移動檔案。將執行檔(SimpleDemoAppWizard)及腳本(getDeciceInfo.sh 和 getSysMonitor.sh)放到 USB 隨身碟。等待 NuMaker-HMI-MA35H0-A1 V1.0 開機後，運行掛載指令：

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

移動到 mnt 的資料夾：

```
$ cd /mnt
```

運行執行檔案

```
$ ./SimpleDemoAppWizard
```

倘若第一次運行 NuMaker-HMI-MA35H0-A1 V1.0 需要設置螢幕校正點。將 ts.conf 透過 USB 隨身碟方式複製到開發平台：

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

運行 ts_calibrate 指令進行螢幕五點校正:

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

2. 程式介紹

轉換成對應的程式碼，用戶可以在 ~/Application/Source/CustomCode/Application.c 加入自定義的函式功能。程式開頭需 include 會使用到的頁面。本程式碼使用 AppWizard API 請參考使用手冊，下表列出本範例程式會使用到的 AppWizard API 函式。

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 函式表

```
// 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 控制程式碼如下所示，取得當前 GUI LED 狀態後決定 LED 的亮滅行為。如果當前 LED 狀態是恆亮，則按下按鈕則會讓LED熄滅，反之LED狀態是熄滅，則按下按鈕則會讓LED點亮：

```
/*=============================================================================================*
** 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);
    }
}
```

開發板上的按鈕讀取程式如下所示，啟動 thread 反覆監視開發板上按鈕是否被按下，當按下按鈕後，則更新計數到 GUI 上的次數顯示。

```
/**
 * 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
}
```

程式剛開始運行時，會透過讀取腳本方式取得開發板上的系統訊息，程式碼如下：

```
/*=============================================================================================*
** 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;
}
```

本範例程式透過 command line 方式取得開發板上 IP 及 UID 訊息，腳本 getDeciceInfo.sh 的程式碼如下所示：

```
#!/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}"
```

本範例程式透過 command line 方式取得開發板上 CPU loading、memory 使用率與溫度訊息，腳本 getSysMonitor.sh 的程式碼如下所示：

```
#!/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
```

溫度資訊，請參考 MA35H0_Linux User Manual 說明，於設置 MA35H0 開發平台時，用戶需自行設置開啟溫度感測器功能 ：

```
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
```

開啟後，透過 command line 方式，以獲取開發板溫度資訊:

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

3. 軟體與硬體需求

3.1 軟體需求

- AppWizard 版本
  - AppWizard_V1.40f_6.34g

3.2 硬體需求

- 電路元件
  - NuMaker-HMI-MA35H0-A1 V1.0

4. 目錄資訊

目錄結構如下圖所示。

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

圖 4-1 目錄資訊

5. 範例程式執行

1. 將 AppWizard_Project > Firmware 資料夾下的執行檔及腳本複製到 NuMaker-HMI-MA35H0-A1 V1.0 開發板上運行。

   - 執行檔： EC_MA35_Dashboard
   - 腳本： getDeciceInfo.sh、getSysMonitor.sh
   - 如何移動執行檔，請參閱章節 1.3 第 6 項說明。

2. 執行檔案 EC_MA35_Dashboard。

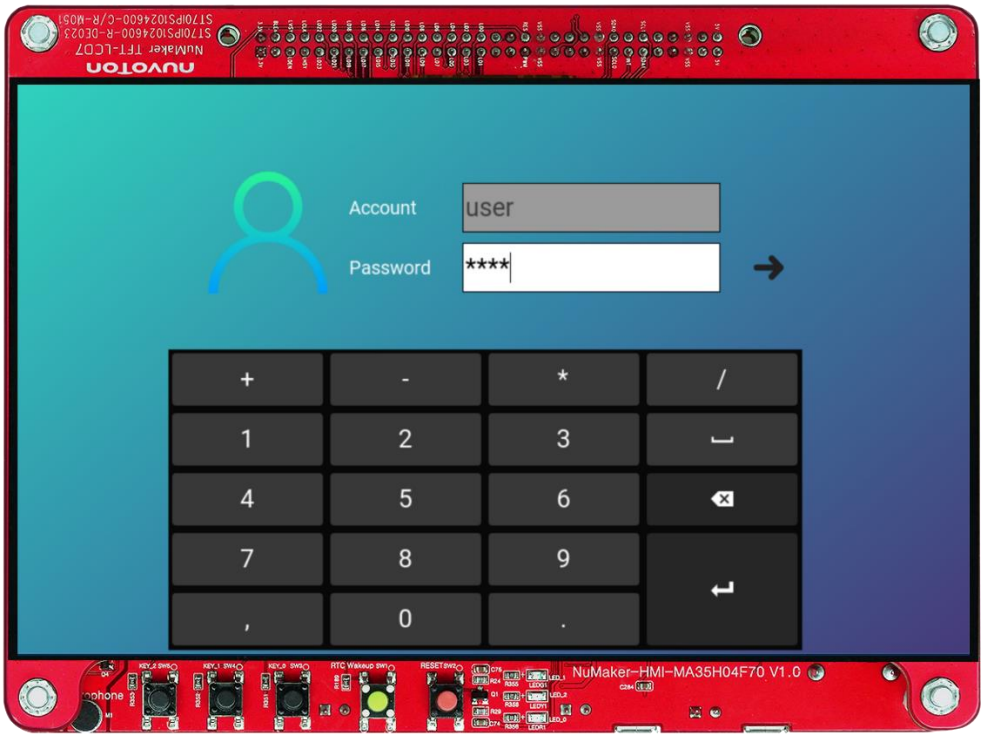

圖 5-1 執行結果

6. 修訂紀錄

Date	Revision	Description
2024.09.10	1.00	初始發布。

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.