使用 SquareLine Studio 设计 GUI 与执行

NuMicro® 32 位系列微控制器范例程序代码介绍

文件信息

应用简述	使用 SquareLine Studio 工具设计 LVGL GUI，并将产出的 UI 文件整合至 Nuvoton MCU 项目中编译与执行。
BSP 版本	M55M1_Series_BSP_CMSIS_V3.01.005
硬件平台	NuMaker-X-M55M1D

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

本范例程序与文件说明如何以 SquareLine Studio 设计 LVGL GUI，以及导览基本的 SquareLine Studio 画面设计功能，并将产出的 UI 文件整合至 Nuvoton MCU 项目中执行。

本范例以 NuMicro M55M1 平台为例，亦可应用于表 1-1 所述的其他平台使用。

1.1 原理

1.1.1 SquareLine Studio 介绍

SquareLine Studio（以下简称 SLS）是一套专为 LVGL 打造的所见即所得（WYSIWYG）GUI 设计工具。设计者只要在画布上以拖拽方式摆放按钮、标签、图片、滑杆等组件，并于属性面板调整版面、样式与事件，SLS 便会自动产生对应的 LVGL C 原始码，省去手写界面程序的繁琐工作。

SLS 其中的 LVGL Library 9.2 版内建部分 Nuvoton 开发板选项，如图 1-1，可直接选择对应平台建立项目；搭配 “lv_port_nuvoton” BSP，将导出的 UI 文件整合进项目即可在 Nuvoton 开发板上编译与执行，让画面设计与固件开发流程无缝接轨。

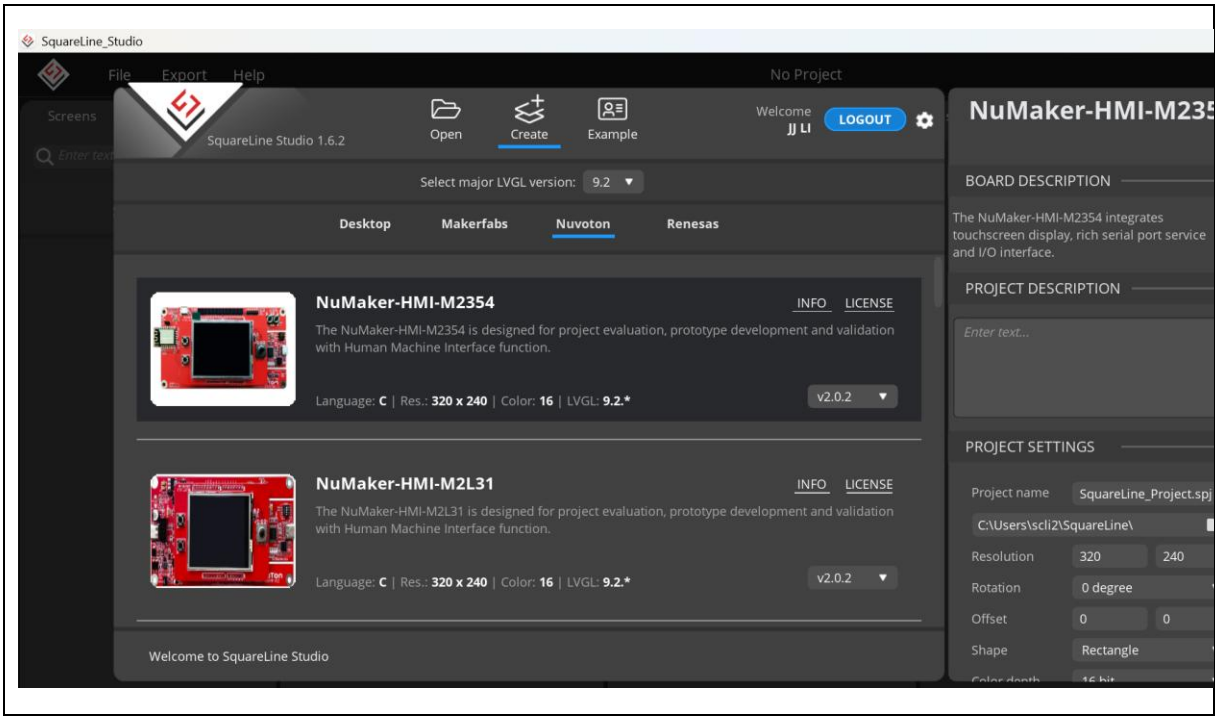


图 1-1 SquareLine Studio LVGL 9.2 Version

而在“lv_port_nuvoton” BSP 中目前已移植并支持的平台如表 1-1 所示，并且针对对应的开发板移植 LVGL 范例程序，支持平台会根据 LVGL 版本更新而异动，请依 GitHub 官方说明 为参考。

NuMicro Series	NuMaker Platforms
M2354 Series	NuMaker-HMI-M2354
M2L31 Series	NuMaker-HMI-M2L31
M3334 Series	NuMaker-HMI-M3334
M3351 Series	NuMaker-HMI-M3351
M487 Series	NuMaker-HMI-M487
M467 Series	NuMaker-HMI-M467
M55M1 Series	NuMaker-X-M55M1D
M5531 Series	NuMaker-HMI-M5531
N9H Series	NuMaker-HMI-N9H20 NuMaker-HMI-N9H26 NuMaker-HMI-N9H30 NuMaker-HMI-N9H31
NUC980 Series	NuMaker-IIoT-NUC980G1D NuMaker-IIoT-NUC980G2D
MA35D1 Series	NuMaker-HMI-MA35D1
MA35H0 Series	NuMaker-HMI-MA35H0

表 1-1 Supported Platform List

1.1.2 执行步骤

1. 下载与注册 SquareLine Studio，请参考 SLS 官方下载页面。

2. 建立 SLS 项目：

1) 在 SLS 开启后，点击 Create，并在 Select major LVGL version 选择 9.2，如图 1-2。

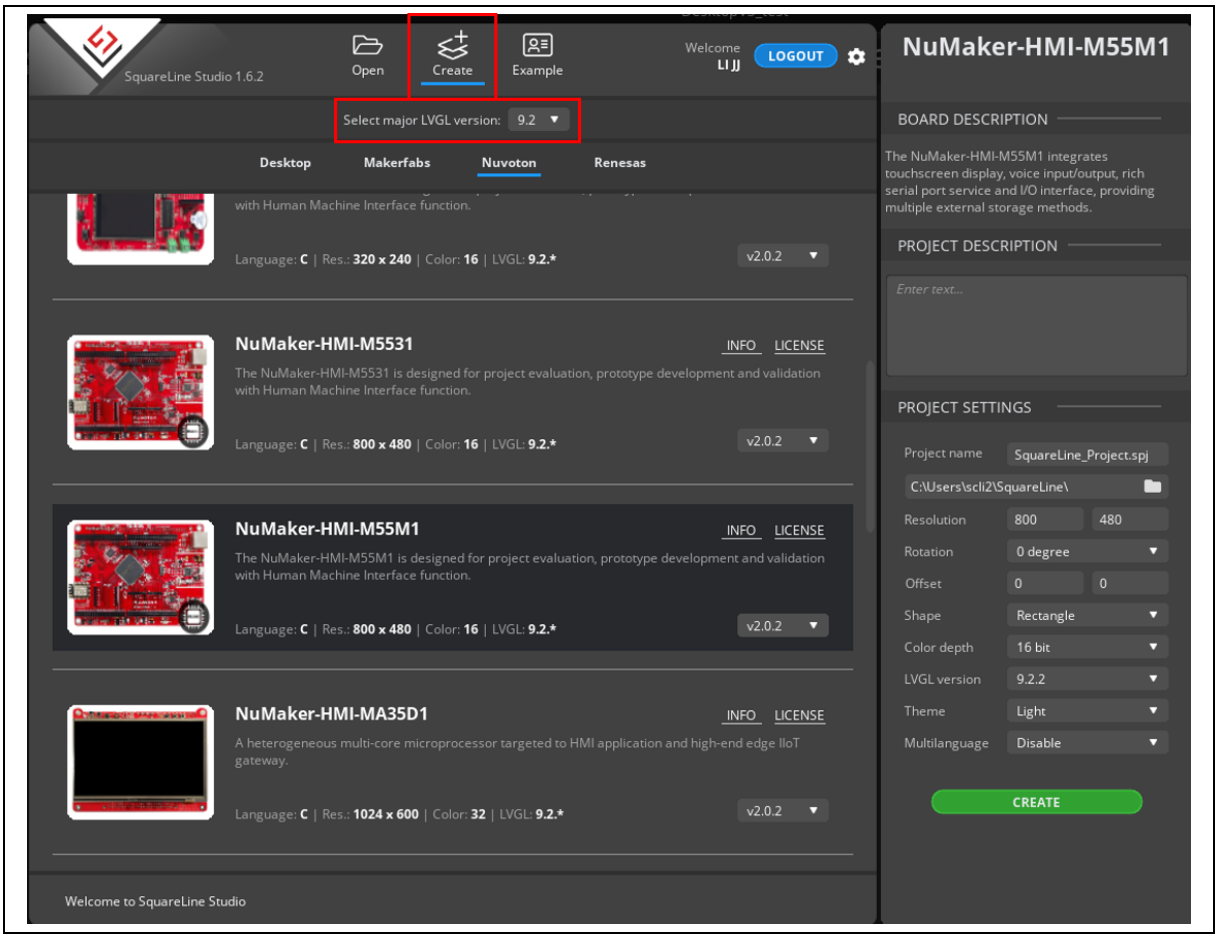


图 1-2 建立 SLS 项目

2) 若能在菜单中找到对应的平台与开发环境，可以直接挑选对应选项，参考图 1-2，如 M55M1 的 NuMaker-HMI-M55M1，同时可以看到右边的 Project Settings 区域已按照对应规格设定完成。

若无法在菜单中找到对应的平台与开发环境，可以先任选其他开发板或 Desktop 版本，再于 Project Settings 区域设定对应的项目属性即可。而项目属性规格设定内容与注意事项可参考表 1-2 说明。

Properties	Notes
Project Name/Path	名称与路径须避免任何空白或非英语的文字
Resolution	分辨率务必与实际 LCD 相同，否则画面比例会错误

Properties	Notes
Rotation	旋转角度一般皆为 0，若面板实体方向不同可再调整
Offset	偏移设定保持 0
Shape	面板形状可以设定为矩形或圆形
Color depth	颜色深度依据需求可设定为 16/32 bit
LVGL version	根据 Nuvoton GitHub 说明的同步版本，可向下兼容
Theme	主题可选择深/浅色，但仅影响外观默认值

表 1-2 Project Settings

3. SLS GUI 界面导览:

SLS 界面如图 1-3 所示，各项功能说明可参考表 1-3，高级界面设定与功能可参考 SLS 官方文件。

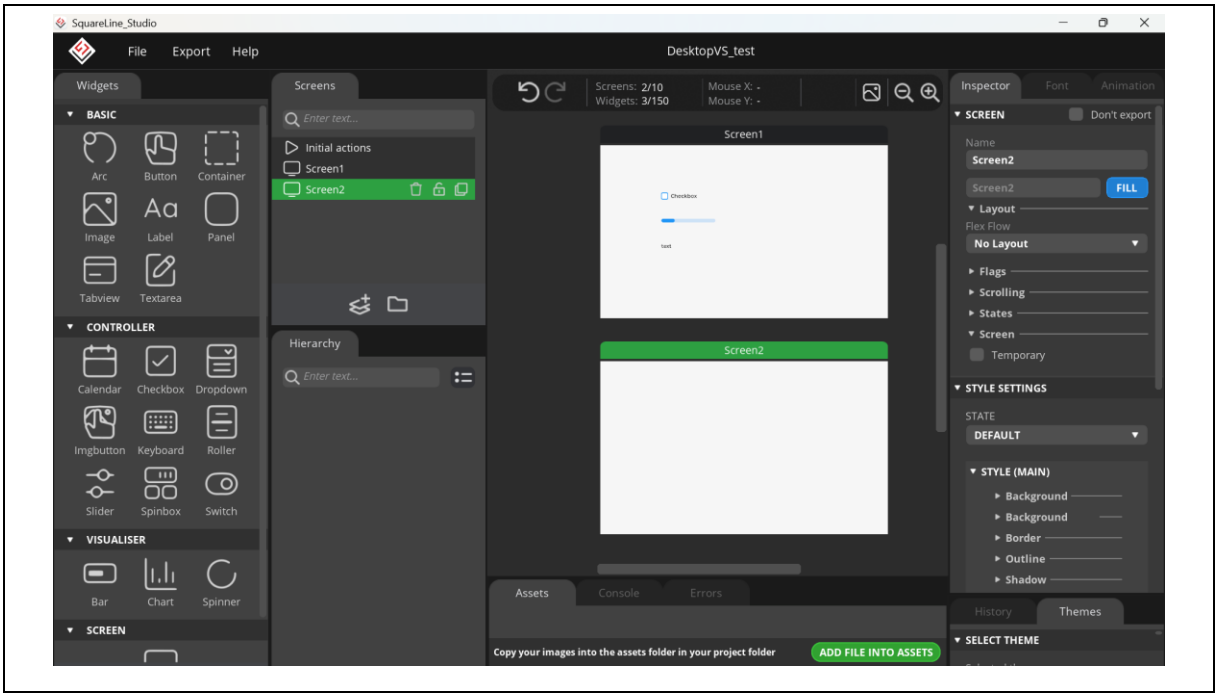


图 1-3 SLS 界面

名称	用途
Widgets	组件库，可以直接点击或拖拽至画布上
Screens	管理多个页面(Screens)设计

名称	用途
Canvas	实际摆放位置与排版的所见即所得区域
Hierarchy	以树状结构检视页面中所有组件的阶层关系
Inspector	设定选取组件的位置、大小、样式、事件等属性
Animations	为组件加入事件或动画

表 1-3 SLS 界面功能说明

4. SLS UI 画面设计：实现简易的仪表板功能

依照以下步骤，可完成如图 1-4 所示的简易仪表板画面设计。此范例包含两个画面：Screen1 用于显示动态数值与 Arc 连动效果；Screen2 则通过 Slider 控制数值显示。

Screen1：动态数值与 Arc 显示

在 Screen1 中，数值会在 0 至 100 之间变化，并同步带动 Arc 的显示位置或进度变化。画面上同时配置 Button，可切换至 Screen2。

Screen2：Slider 数值控制

在 Screen2 中，可拖拽 Slider 调整 0 至 1000 的数值范围，画面上的数字会依照 Slider 目前值实时更新。

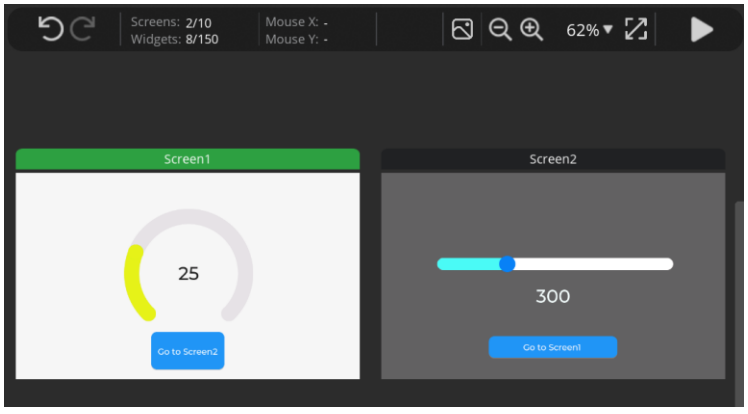


图 1-4 仪表板实现

1) 新增 Screen2：如图 1-5，在 Screen 窗口中，增加 Screen2，并在 Inspector 中 STYLE SETTINGS 对背景设定成不同的颜色。

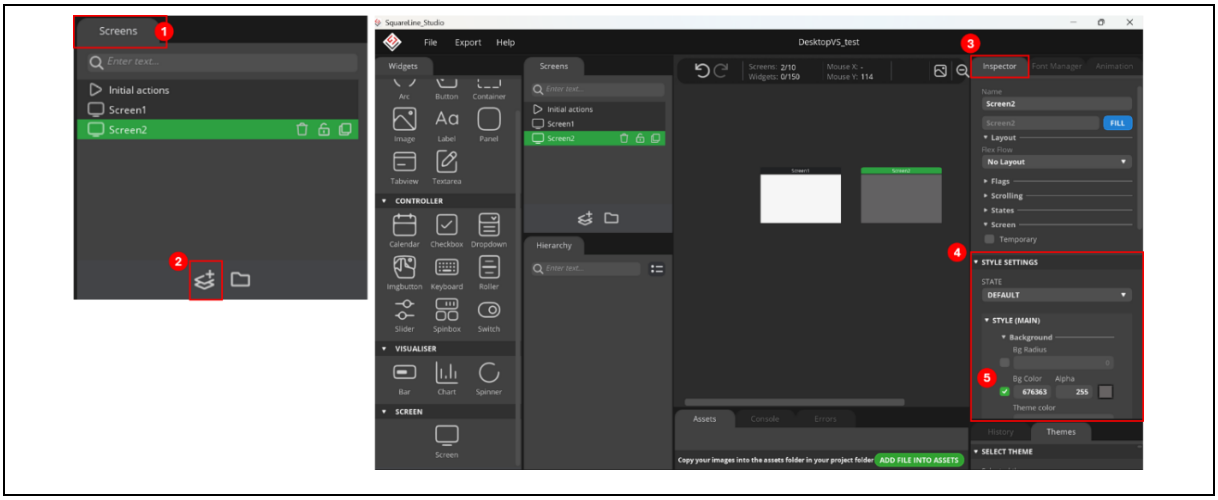


图 1-5 增加 Screen2，并设定背景色

2) 新增 Arc 组件：点击 Screen1，以增加在 Screen1 上的对象。参考图 1-6，从 Widgets 中点击 Arc，点击后即会出现在 Screen1 的画布上。并参考表 1-4 在 Inspector 中设定 Arc 属性，包括大小、数值的范围、显示的角度、颜色设定等。

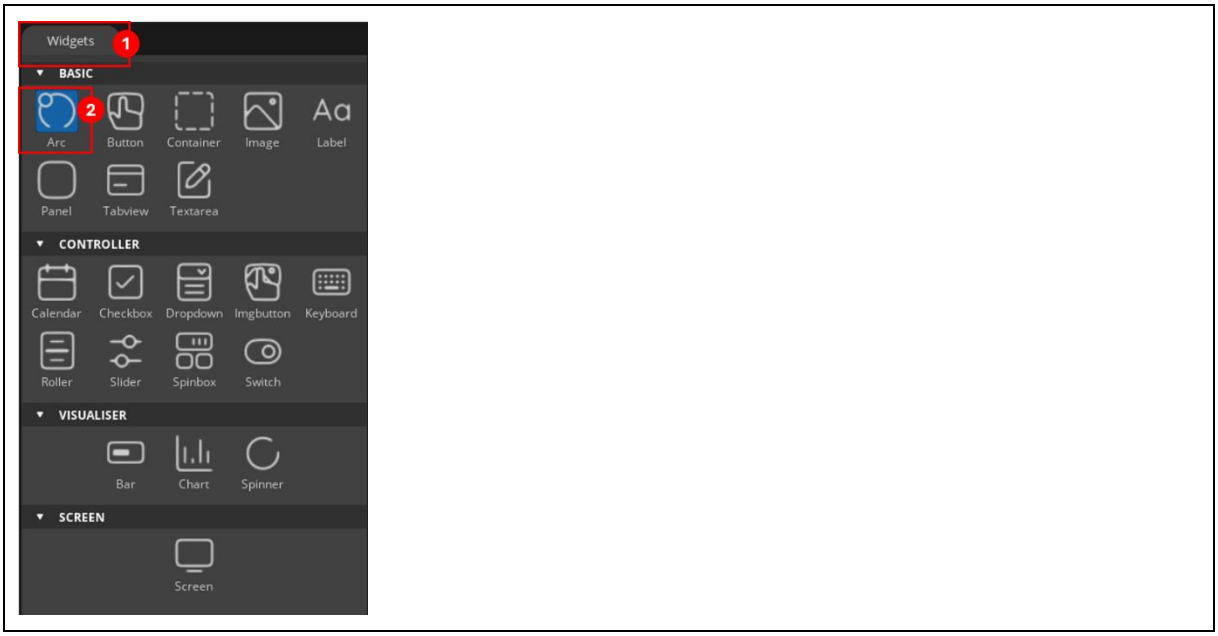
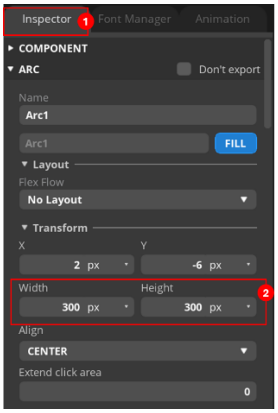
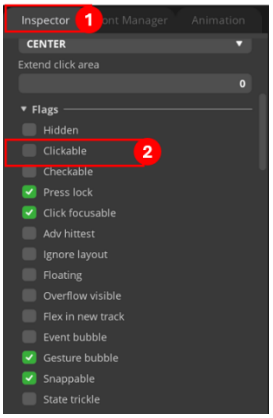
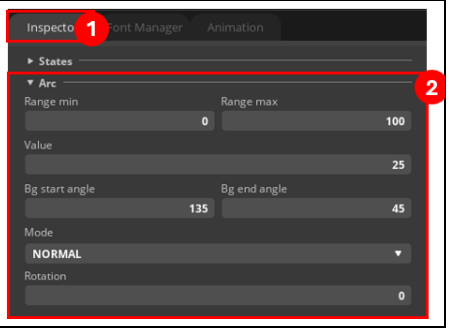


图 1-6 增加 Widgets – Arc

Contents	Functions	Settings
Width and Height	Arc 整体的宽与高	
Flags-Clickable	设定是否能被用户触控	
Range, Value, Angle	Range 是 Arc 值的范围，Value 是初始显示的值，Angle 是 Arc 显示的角度	

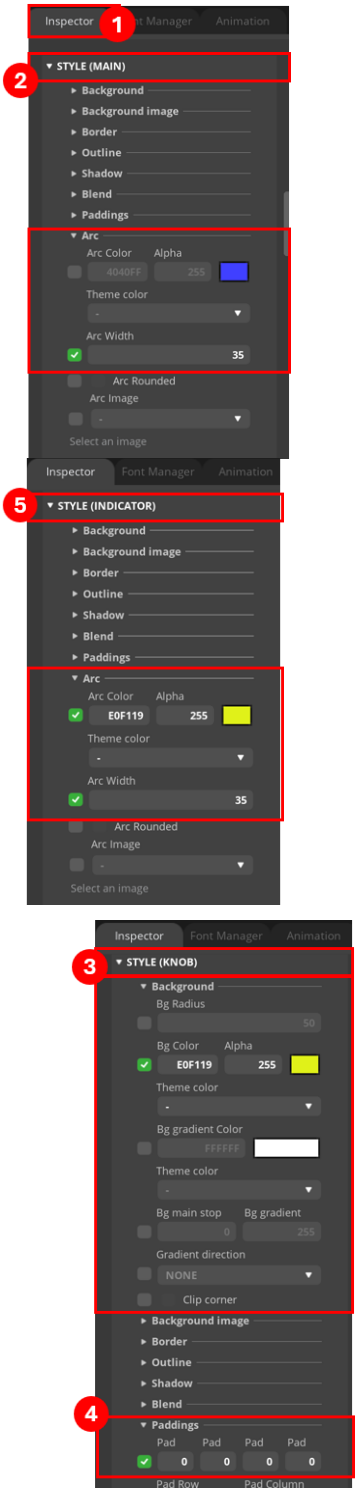
Contents	Functions	Settings
STYLE	Style 中有分成三项，MAIN 代表对 Arc 主体的控制设定、KNOB 代表控制旋钮的设计设定、INDICATOR 代表 Arc 值变动时指示的设计设定。	

表 1-4 Arc 属性设定

3) 新增 Screen1 的数值显示：从 Widgets 中点击 Label，点击后即会新增在 Screen1 的画布上。在画布上拖拽移动至 Arc 中间，参考表 1-5 说明，设定 Label 相关属性。

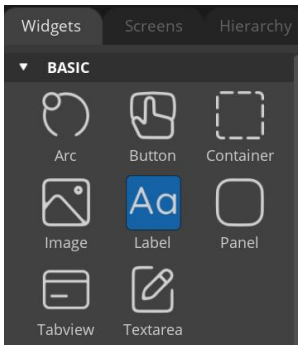


图 1-7 增加 Widgets – Label

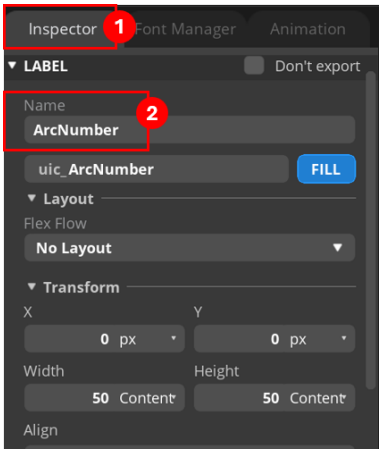
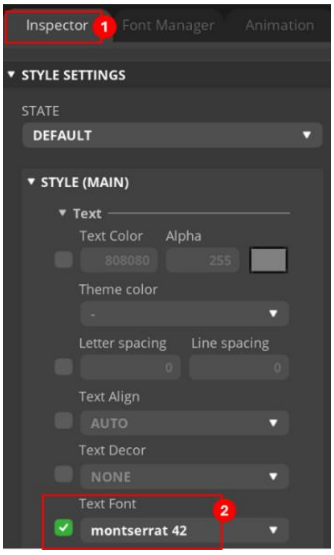
Contents	Functions	Settings
Name	可以自定义对象名称，在对象链接时或在项目中更方便辨认	
STYLE-Text Font	Style 中可以设定 Label 的颜色以及字体，字体包括大小。	

表 1-5 Label 属性设定

4) 新增 Screen1 跳至 Screen2 的 Button：参考图 1-8 从 Widgets 中点击 Button，并将 Button 改命名为 ”NextBTN”。若 Button 上需要显示文字，需另外新增 Label 重叠放置在 Button 上方。

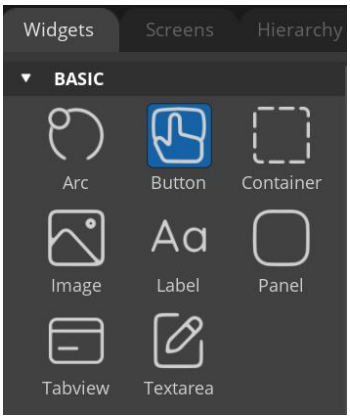


图 1-8 增加 Widgets – Button

5) 设定当按下 Screen1 的 ”NextBTN” 后，跳至 Screen2：参考图 1-9 步骤，点击 ”NextBTN” ，在 Inspector 属性设定按下 ADD EVENT，Trigger 方式是 Clicked，表示按下按钮时，Action 是选择 Change Screen，表示表现的动作是切换画面，加入动作后，可再进一步选择欲切换的目标-Screen2 以及切换的方式- Move left(往左移)。

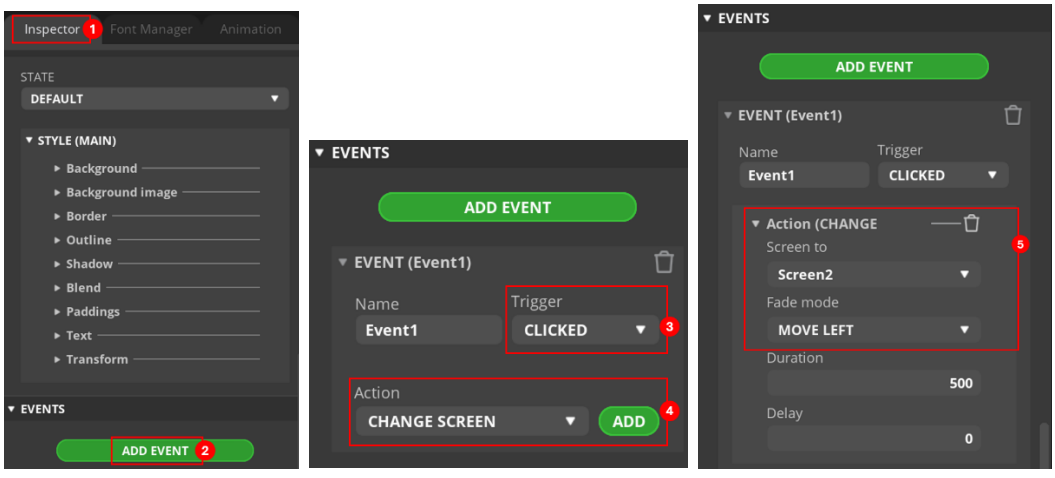


图 1-9 按键动作设定

6) 增加 Screen2 的组件：点击 Screen2，参考图 1-10 从 Widgets 中点击 Slider，并参考表 1-6 设定 Slider 相关属性。

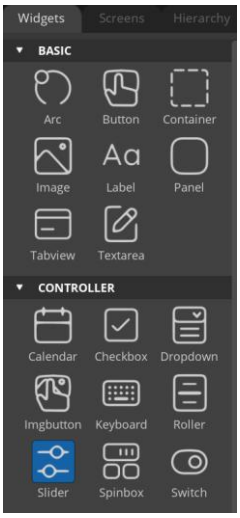


图 1-10 增加 Widgets – Slider

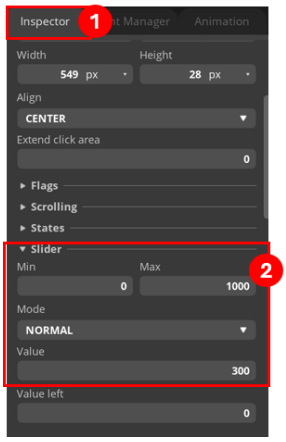
Contents	Functions	Settings
Min/Max	表示 Slider 数值的范围	
Value	Slider 显示的初始位置	

表 1-6 Slider 属性设定

7) 新增 Screen2 的数值显示：从 Widgets 中点击 Label，点击后即会新增在 Screen2 的画布上。参考图 1-7 的设定步骤，将此 Label 命名为 SliderNumber。

8) 将 Slider 的变化跟 SliderNumber 数值连动：点击 Screen2 的 Slider 对象，参考图 1-11 步骤，在 Inspector 中按下 ADD EVENT，设定 Trigger 方式为 VALUE_CHANGED，Action 动作选为 SET VALUE FROM SLIDER，Target 则是选定为前面设定好的 SliderNumber。

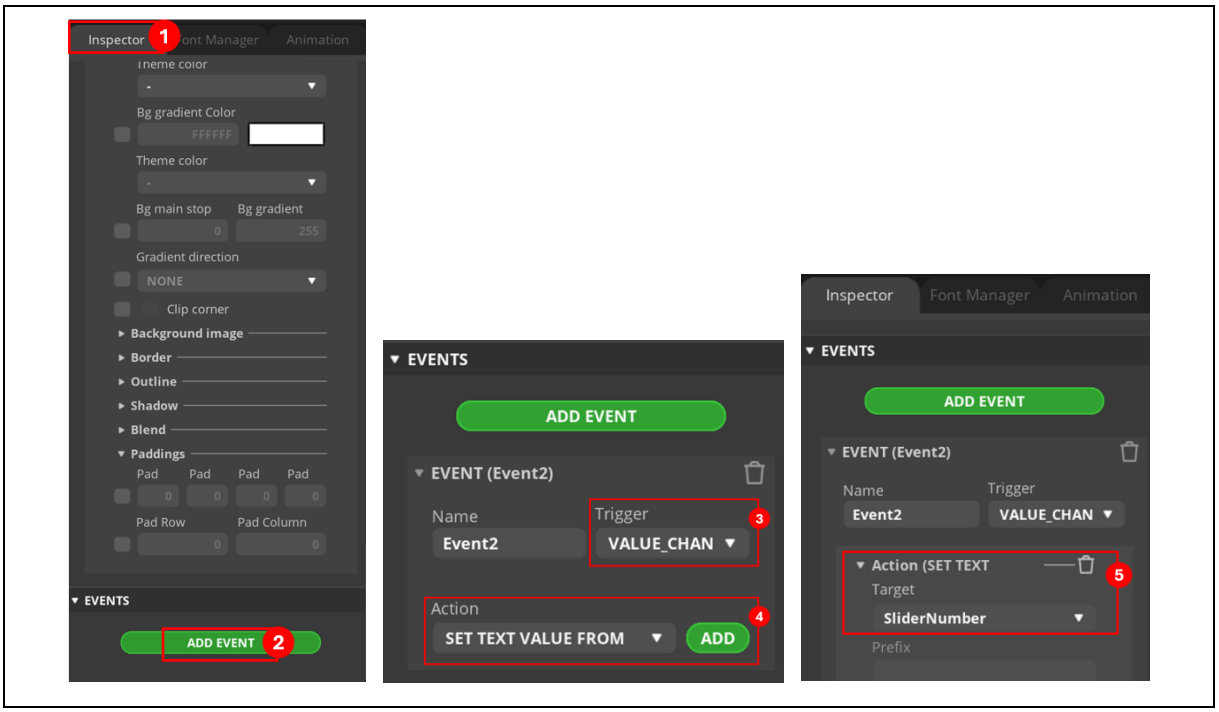


图 1-11 Slider 与数值连动设定

9) 增加返回 Screen1 的 Button：与步骤 5)相同，新增 Button，并参考图 1-9 设定连动。

10) Play Mode 模拟：完成设计时，可如图 1-12 所示，按下 Play，在计算机上仿真画面。

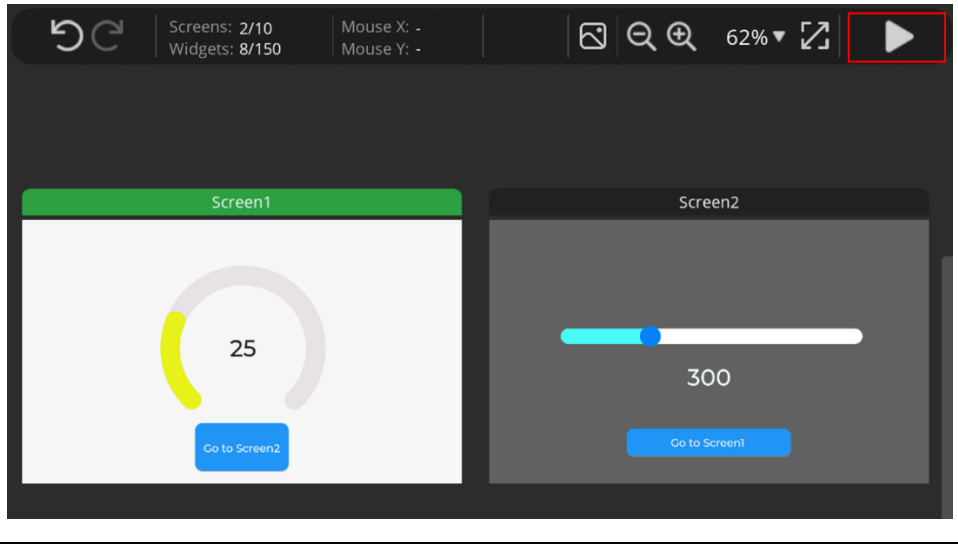
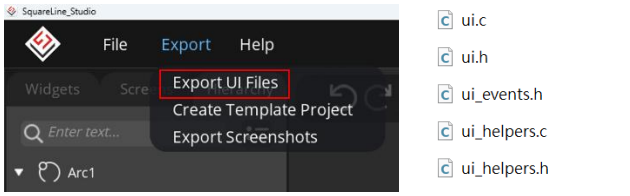


图 1-12 Play Mode

5. 输出并开启程序项目：在 SLS 窗口左上角，选择 Export，共有两种输出方式。

- Export UI Files – SLS 将只输出 UI 设定相关文件。本范例亦以此方式输出，输出流程与开启程序项目的方式为：

1) 参考图 1-13，若选择 Export UI Files，会输出的文件包括表 1-7 所示。储存路径可以于 Project Setting 中设定与查询，如图 1-14。



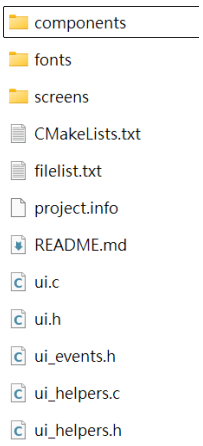


图 1-13 Export UI Files

Contents	Notes
Components	存放 Component（如 Status Bar、Navigation Bar），供多个 Screen 共享。
Fonts	字体资源
Screens	每个 Screen 对应一组 ui_ScreenX.c / .h，建立画面上所有 Widget
ui.c / ui.h	是 ui 的主入口，包括 ui_init()、Screen 初始化、加载默认画面、Widget 全局变量定义
ui_events.h	event 设定为 CALL FUNCTION 时产生的函数原型声明，须将程序执行内容由用户自行写于 ui_events.c
ui_helpers.c ui_helpers.h	由 SLS 产生的通用辅助函数

表 1-7 UI Files List

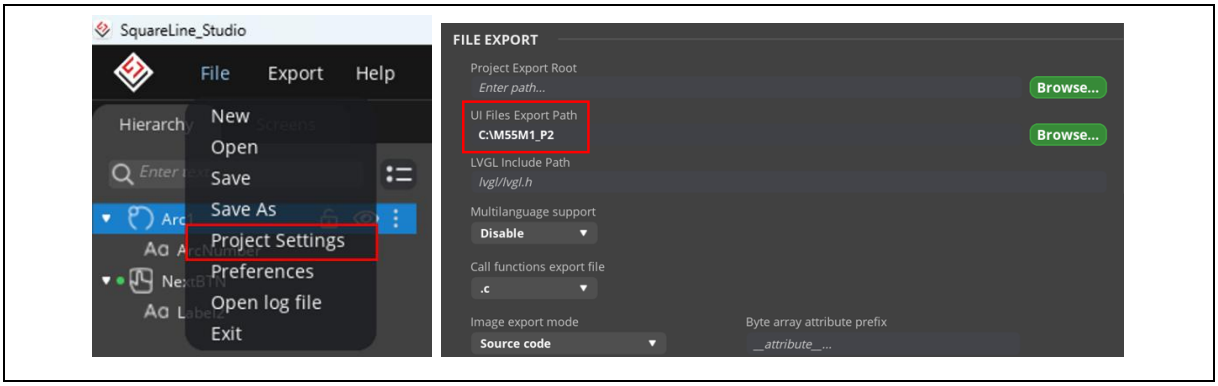


图 1-14 Project Settings

2) 复制 SLS UI 输出文件至项目目录：将 SLS 产生的所有 UI 相关文件，完整复制到项目中的以下路径：\board\numaker-x-m55m1d\sls_files。

3) 如章节 5 所述，开启项目。

若使用 Keil MDK 开启，需依照图 1-15 步骤，将 UI 相关文件加入项目中。

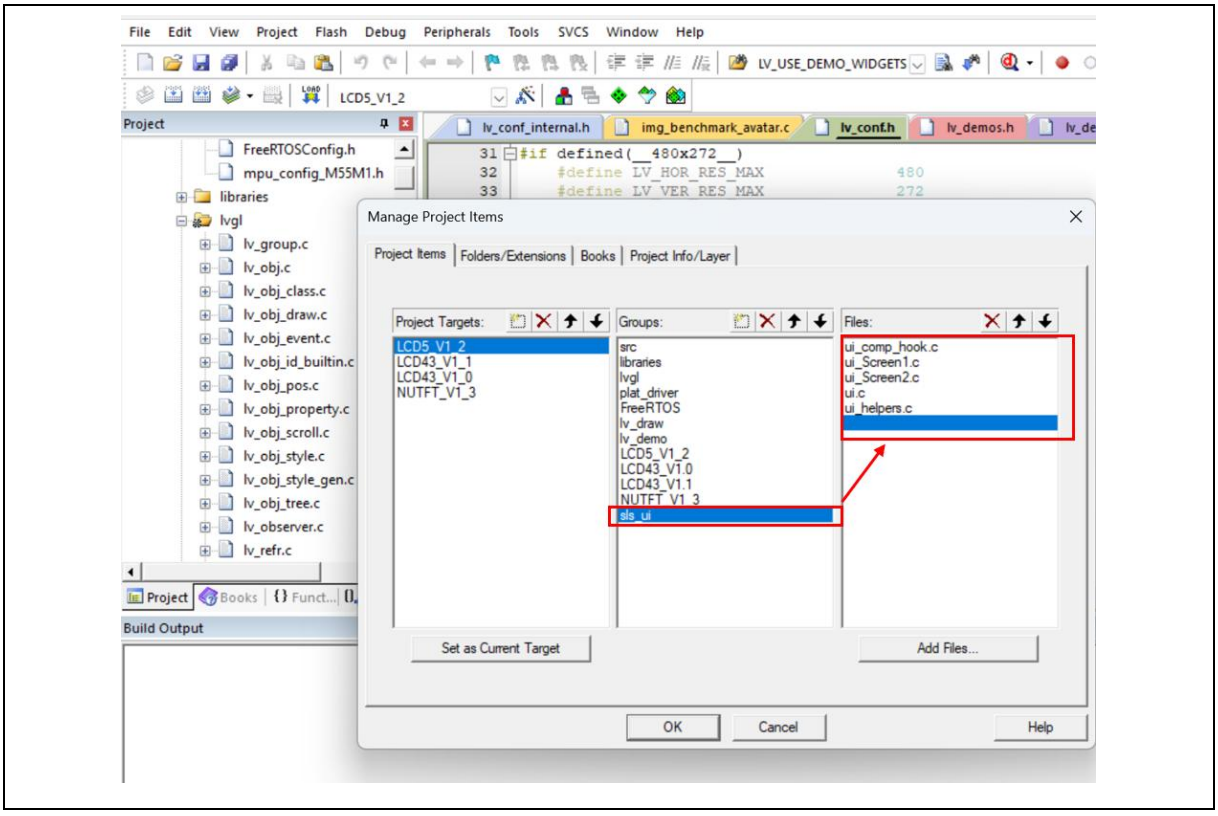


图 1-15 于 Keil 嵌入 UI 文件

若使用 VS Code 开启项目，如图 1-16 所示，若有新增 UI 文件需编译，需手动加入文件于 sls_files 文件夹内，并确保.cproject.yml 编译档亦包括该文件。

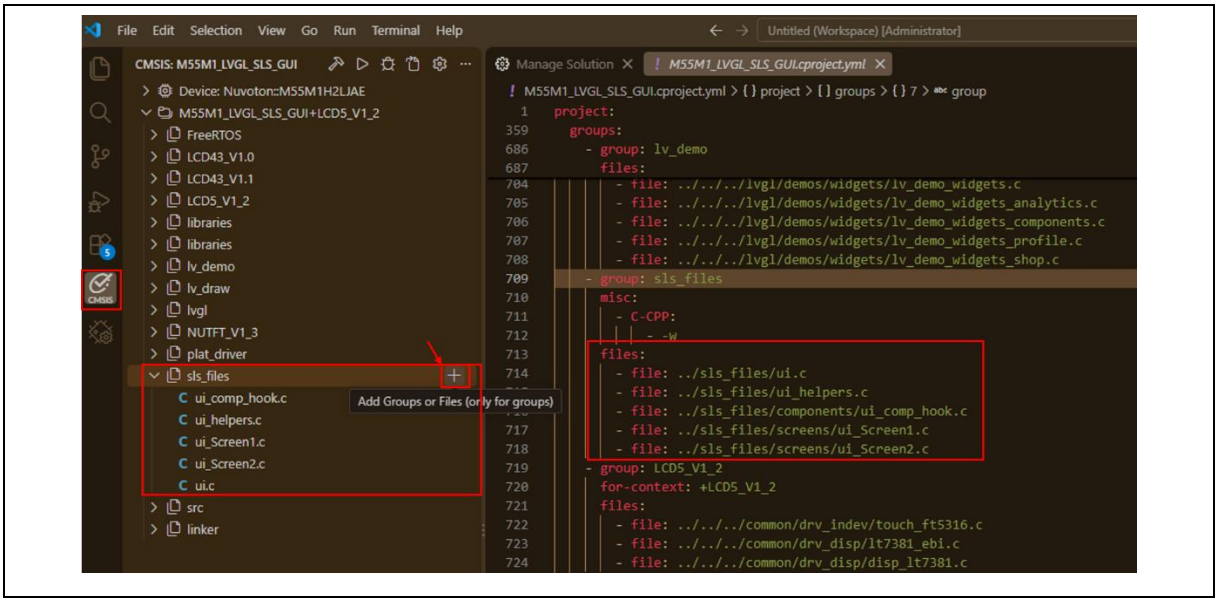


图 1-16 于 VS Code 嵌入 UI 文件

6. 加入数值读取，并显示于 Screen1 的 Label 对象与 Arc 对象：当 MCU 有接 sensor 或是计算数值，可于 task_lv.c 中写入相关数值至欲显示的组件，API 与相关程序流程请参考章节 2。

1.2 执行结果

本范例程序代码执行编译和烧录后，Screen1 会显示自动播放动画，Screen2 则是可以控制画面，如图 1-17 所示。

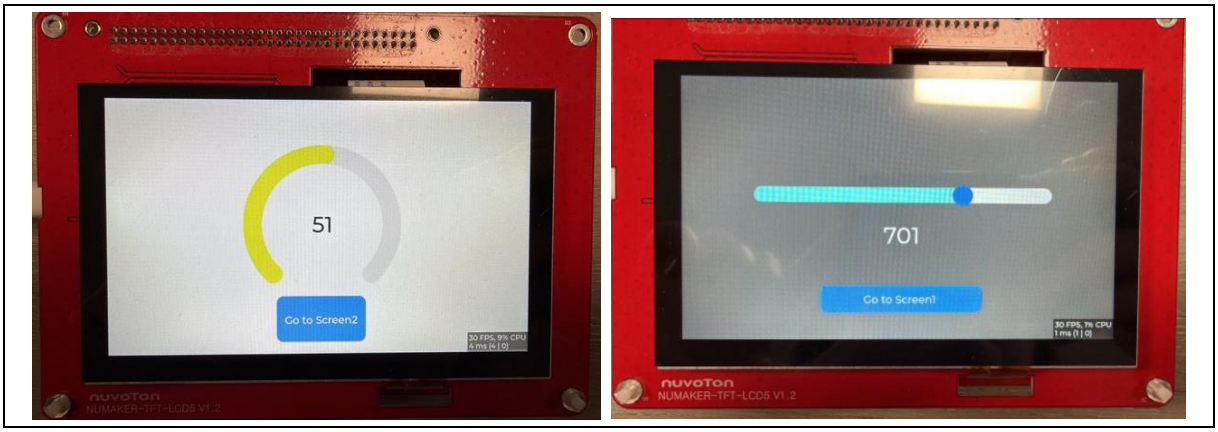


图 1-17 执行编译和烧录结果

2. 程序代码介绍

以下将介绍此示例的关键程序代码部分。主程序 main.c 流程，包括三部分，分别是系统初始化、LVGL 初始化以及 UI 程序流程。

```
int main(void)
{
    sys_init();

    int task_lv_init(void);
    task_lv_init();

    xTaskCreate(sensor_task, "sensor", configMINIMAL_STACK_SIZE, NULL, tskIDLE_PRIORITY +
1, NULL);

    /* Start scheduling. */
    vTaskStartScheduler();

    for (;;);

    return 0;
}
```

1. 系统初始化 sys_init()

   此范例所使用到的硬件脚位初始化，主要包括 EBI 用于驱动 LCD、I2C1 驱动 LCD 的触控，SPI2、PDMA0/1、TMR0、EADC0 则供其他周边功能使用，另外使用 GDMA350，作为 LVGL 绘图的加速功能。

```
static void sys_init(void)
{
…
    /* Enable EBI clock */
    CLK_EnableModuleClock(EBI0_MODULE);

    /* Enable I2C1 clock */
    CLK_EnableModuleClock(I2C1_MODULE);

    /* Enable SPI2 clock */
    CLK_EnableModuleClock(SPI2_MODULE);

    /* Enable PDMA clock */
    CLK_EnableModuleClock(PDMA0_MODULE);
    CLK_EnableModuleClock(PDMA1_MODULE);

    /* Select TIMER clock source */
    CLK_SetModuleClock(TMR0_MODULE, CLK_TMRSEL_TMR0SEL_HIRC, 0);

    /* Enable TIMER module clock */
    CLK_EnableModuleClock(TMR0_MODULE);

    /* EADC Analog Pin */
```

```
    CLK_EnableModuleClock(EADC0_MODULE);

    /* Select EADC peripheral clock source. */
    CLK_SetModuleClock(EADC0_MODULE, CLK_EADCSEL_EADC0SEL_PCLK0, CLK_EADCDIV_EADC0DIV(8));

    /*---------------------------------------------------------------------------------*/
    /* Init I/O Multi-function                                                         */
    /*---------------------------------------------------------------------------------*/
    SET_SPI2_SS_PA11();
 …
    SET_I2C1_SDA_PB10();
    SET_I2C1_SCL_PB11();

    SET_EBI_AD0_PA5();
  …

    SYS->GPB_MFP1 &= ~(SYS_GPB_MFP1_PB7MFP_Msk | SYS_GPB_MFP1_PB6MFP_Msk);
    SYS->GPB_MFP1 |= (SYS_GPB_MFP1_PB7MFP_EADC0_CH7 | SYS_GPB_MFP1_PB6MFP_EADC0_CH6);
    SYS->GPB_MFP2 &= ~(SYS_GPB_MFP2_PB9MFP_Msk | SYS_GPB_MFP2_PB8MFP_Msk);
    SYS->GPB_MFP2 |= (SYS_GPB_MFP2_PB9MFP_EADC0_CH9 | SYS_GPB_MFP2_PB8MFP_EADC0_CH8);

    /* Disable digital path on these EADC pins */
    GPIO_DISABLE_DIGITAL_PATH(PB, BIT6 | BIT7 | BIT8 | BIT9);

    /* Enable SysTick clock */
    CLK_EnableSysTick(CLK_STSEL_ST0SEL_HIRC_DIV2, 0);

    /*---------------------------------------------------------------------------*/
    /* Init I/O Multi-function                                                   */
    /*---------------------------------------------------------------------------*/
    SetDebugUartMFP();

    InitDebugUart();

    /* Enable SPIM module clock */
    CLK_EnableModuleClock(SPIM0_MODULE);

    /* Enable OTFC module clock */
    CLK_EnableModuleClock(OTFC0_MODULE);

    DMA350DevInit();

}
```

2. LVGL 初始化

LVGL 会在 FreeRTOS 调度启动后，于 lv_hdler 任务内执行函数 lv_nuvoton_task()。

```
int task_lv_init(void)
{
    printf("stack addr = %p\n", lvgl_stack);
    xTaskCreate(lv_tick_task, "lv_tick", configMINIMAL_STACK_SIZE, NULL,
CONFIG_LV_TASK_PRIORITY, NULL);
    xTaskCreateStatic(lv_nuvoton_task, "lv_hdler", CONFIG_LV_TASK_STACKSIZE, NULL,
CONFIG_LV_TASK_PRIORITY, lvgl_stack, &lvgl_tcb);
    return 0;
}
```

函数 lv_nuvoton_task() 依序完成 LVGL 核心初始化、log 注册、GDMA 绘图单元注册、时间与延迟绑定 FreeRTOS 的 tick、显示设备移植以对应欲显示的 LCD 设备、输入设备移植、用户界面建立，最后建立数据更新用的 LVGL timer 并进入主循环。

其中与用户界面建立有关的 ui_init()，是由 SLS 产生，会直接取得位于路径 sls_files/ui.c 文件进入并执行加载。

```
void lv_nuvoton_task(void *pdata)
{
    lv_init();

#if LV_USE_LOG
    lv_log_register_print_cb(lv_nuvoton_log);
#endif

#if LV_USE_DRAW_GDMA
    void lv_draw_gdma_init(void);
    lv_draw_gdma_init();
#endif

    lv_tick_set_cb(xTaskGetTickCount);
    lv_delay_set_cb(vTaskDelay);

    extern void lv_port_disp_init(void);
    lv_port_disp_init();

    extern void lv_port_indev_init(void);
    lv_port_indev_init();

    extern void ui_init(void);
    ui_init();

    /* Sample faster than the producer so no sensor value is skipped */
    lv_timer_create(sensor_ui_timer_cb, 10, NULL);

    while (1)
    {
        lv_task_handler();
        vTaskDelay(pdMS_TO_TICKS(1));
    }
}
```

3. UI 程序流程

要让 MCU 端取得的数值，例如 EADC 转换结果、I²C 传感器读值、定时器计数等，显示在 Screen1 的弧形量表与标签上，共需两步骤。

1) 在 MCU 端产生数值：在 main.c 中，以 sensor_task 函数模拟一颗 I²C 传感器，模拟数值为 g_i32I2CSensorValue，此值会在 0 与 100 之间来回扫描，中间每 50 ms 前进一步，抵达两端时停留 1 秒以便观察。

```
static void sensor_task(void *pdata)
{
    int32_t value = 0;
    int32_t step = 1;

    (void)pdata;

    for (;;)
    {
        g_i32I2CSensorValue = value;

        if (value >= 100)
        {
            step = -1;
            vTaskDelay(pdMS_TO_TICKS(1000));
        }
        else if (value <= 0)
        {
            step = 1;
            vTaskDelay(pdMS_TO_TICKS(1000));
        }
        else
        {
            vTaskDelay(pdMS_TO_TICKS(50));
        }

        value += step;
    }
}
```

2) 在 LVGL 侧结合数值：在可客制不会被 SLS 覆盖的 task_lv.c 中，调用 lv_arc_set_value() 更新 Arc 数值，以及 lv_label_set_text_fmt() 更新数字。

其余 Widgets 的 API 可通过 LVGL 官方文档查询。

```
static void sensor_ui_timer_cb(lv_timer_t *timer)
{
    static int32_t last_value = -1;
    int32_t value = g_i32I2CSensorValue;

    (void)timer;

    if (value == last_value) return;

    last_value = value;

    if (ui_Arc1)      lv_arc_set_value(ui_Arc1, value);
    if (ui_ArcNumber) lv_label_set_text_fmt(ui_ArcNumber, "%d", (int)value);
}
```

3. 软件与硬件需求

3.1 软件需求

- BSP 版本
  - M55M1_Series_BSP_CMSIS_V3.01.005
- IDE 版本
  - Keil uVision 5.43
  - VS Code 1.138.0
- SquareLine Studio 版本
  - SquareLine Studio 1.6.2

3.2 硬件需求

- 电路组件
  - NuMaker-X-M55M1D

4. 目录信息

目录结构如下图所示。

EC_M55M1_LVGL_SLS_GUI_V1.00	
board	Board-level files including the main application.
numaker-x-m55m1d	
Keil	Keil Project
VSCode	VSCode Project
components	task_lv
sls_files	ui files generated from SLS
bsp	Board-level for board support package
m55m1	
common	Shared source code, including display drivers, drawing, touch input.
lvgl	Open-source LVGL graphics library
SLS	SquareLine Studio project file
EC_M55M1_LVGL_SLS_GUI.spj	
thirdparty	Third-party components, primarily the FreeRTOS library

图 4-1 目录信息

5. 范例程序执行

5.1 Keil MDK

1. 在 Windows 资源管理器中进入 **Keil** 项目文件夹，再开启 *M55M1_LVGL_SLS_GUI.uvprojx* 项目档。
2. 编译程序代码
3. 下载程序代码
4. 进入调试模式
   - 执行程序代码

5.2 VS Code

1. 在 **VS Code** IDE 中开启 **File** 菜单
   - 选取 Open Folder
   - 选取项目目录下的 **VSCode** 文件夹
2. 选择 CMSIS
   - 点击 **Build solution** 编译项目
   - 点击 **Load & Run application** 下载映像文件至目标芯片
   - 点击 **Load & Debug application** 进入调试模式

6. 修订记录

Date	Revision	Description
2026.09.30	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.