M55M1 通过 GDMA 实现 LCD 显示

NuMicro® 32 位系列微控制器范例代码介绍

文件信息

应用简述	本范例程序在 M55M1 开发平台上展示以 emWin appWizard 实现 GDMA 对 LCD 显示画面局部更新的功能设计
BSP 版本	M55M1_Series_BSP_CMSIS_V3.01.003
开发平台	NuMaker-X-M55M1

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

本范例代码将展示如何在 NuMaker-X-M55M1 开发平台上，结合 emWin AppWizard，通过 GDMA (Graphic Direct Memory Access，直接内存访问) 方式实现 LCD 画面更新。相较于传统软件内存拷贝 (Memory Copy)的方式，GDMA 的并行处理特性有效降低 CPU 运算占用率，减少 CPU 在图像渲染过程中的负担，释放处理器资源以执行其他任务，确保系统在执行 GUI (Graphical User Interface)显示的同时，仍能维持整体运行的流畅度与实时性，适合作为高效嵌入式系统的实务参考范例。

1.1 NuMaker-X-M55M1 开发平台

M55M1 系列微控制器采用 Arm® Cortex®-M55 核心，最高运作频率达 220 MHz，内建 Arm® Helium™ DSP 技术以增强数字讯号处理能力，并整合 Arm® Ethos™-U55 微型神经网络处理器，在数字讯号处理及机器学习推论方面表现优异，同时可于边缘装置加速机器学习推论。

本示例使用 Numaker-TFT-LCD5 V1.2 显示模块，其分辨率为 800x480 bit，色彩格式为 RGB565(16-bit)。



图 1-1 NuMaker-X-M55M1 开发板



图 1-2 NuMaker-TFT-LCD5 V1.2 显示模块

1.2 SEGGER emWin appWizard

本范例以 SEGGER 提供的 emWin appWizard 作为 GUI 展示，其具丰富的 Widget 库与多种 Interaction function 供使用者开发使用，能大幅降低在设计图形化接口的所需时间，并能整合程序设计，将基本框架完成后即可通过 IDE 导出档案并放入 Keil 项目，进行细节功能设计与测试。新唐有提供 emWin package，其包含使用 emWin 所需的所有 library、driver 及相关辅助开发工具。

使用者可于新唐官网下载最新版本的 emWin package (https://www.nuvoton.com/resource-download.jsp?tp_GUID=SW15202503130722211 9)，并通过教学手册了解如何将 emWin appWizard 项目导入 Keil 使用。

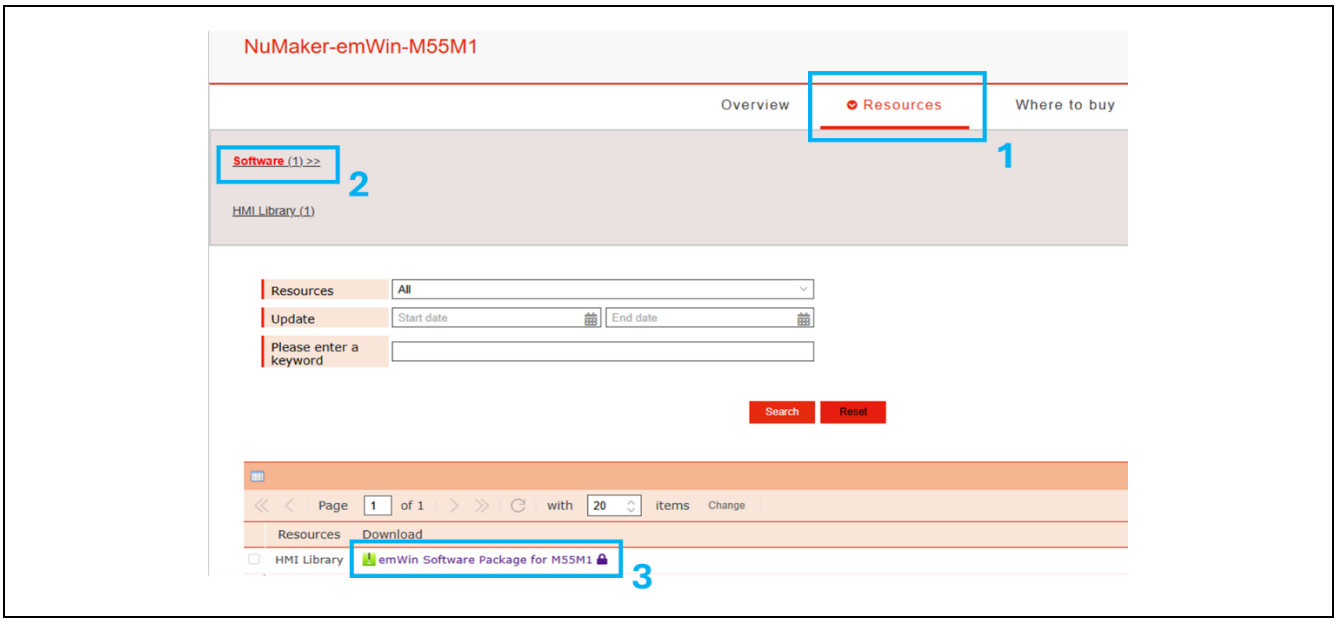


图 1-3 官网 EmWin Package 下载(需登入)

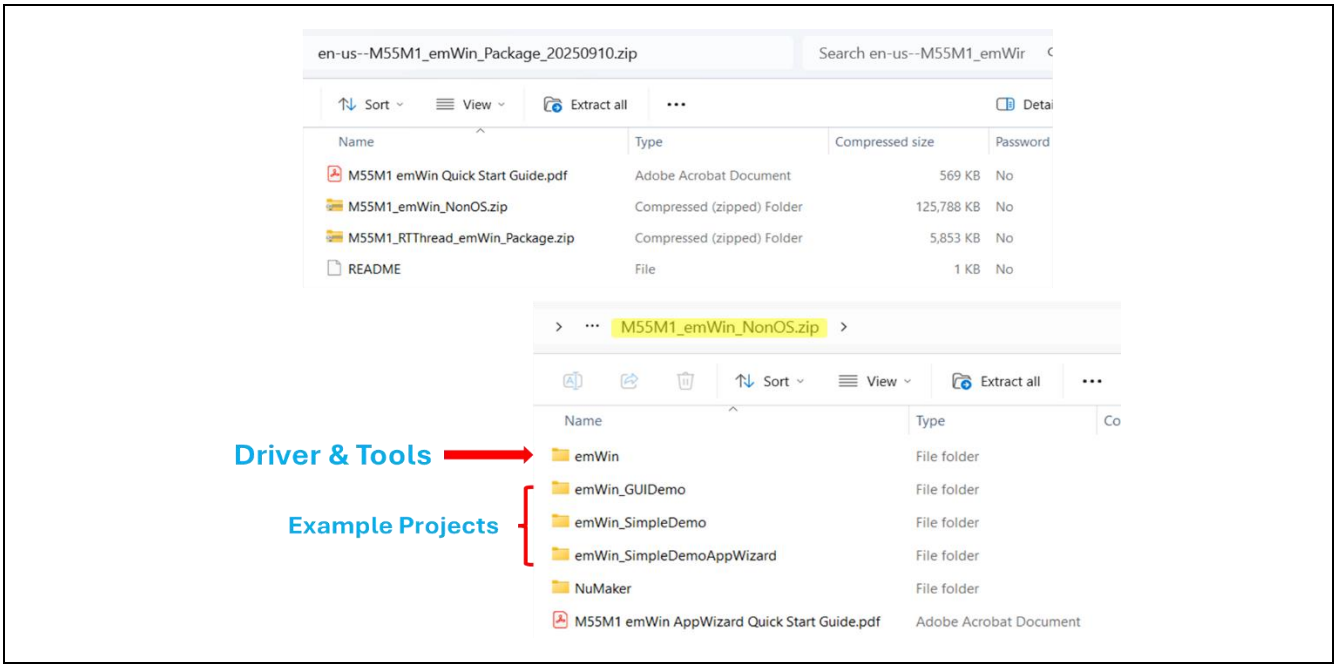


图 1-4 EmWin Package 简介

1.3 GDMA(Graphic Direct Memory Access) 工作原理

传统 CPU 传输必须逐一搬运每个像素(byte-by-byte)，会占用大量运算资源，导致 UI 更新卡顿。而 GDMA 的工作核心在于将「数据寻址」与「搬运执行」从 CPU 转移到专用的硬件控制器上。

CPU 仅需在 GDMA 的缓存器中设定一次性的任务(通过包含起始地址、目标地址、数据长度)启动传输，一旦通道被启用，GDMA 内部就会根据设定，自主地在总线上发起读取与写入请求。这意味着数据的流动不再需要 CPU 介入每一笔 byte 的存取指令，GDMA 会自动处理地址递增、突发传输以及与外设的同步，CPU 便可专注于其他更重要的数据处理任务。

如图 1-5 所示，GDMA 在进行 局部画面更新 (Dirty Region Update) 时，只复制画面中实际变动的区域(Dirty Region)到 LCD VRAM，而非整个 Frame Buffer。

- pSRC：来源指标(Pointer to source)。
- pDst：目标指针(Pointer to destination)。
- Dirty Region：表示需要更新的画面区域。
- Stride：每列像素的宽度(例：800 pixels)。
- Offset：计算起始传输地址，例如 pBG + (800 * y0 + x0)。
- DMA 2D Transfer：GDMA 依照 Dirty Region 的大小 (xSize × ySize) 与 Stride 自动搬移数据，跳过不需更新的像素区域(Skip Stride Gap)。
- 16-bit RGB Data：本范例使用 RGB565 色彩格式传输数据。

通过这种方式，GDMA 只更新实际变动的区域，大幅降低数据传输量与 CPU 负担，避免 UI 卡顿，并可保持高效能画面更新。

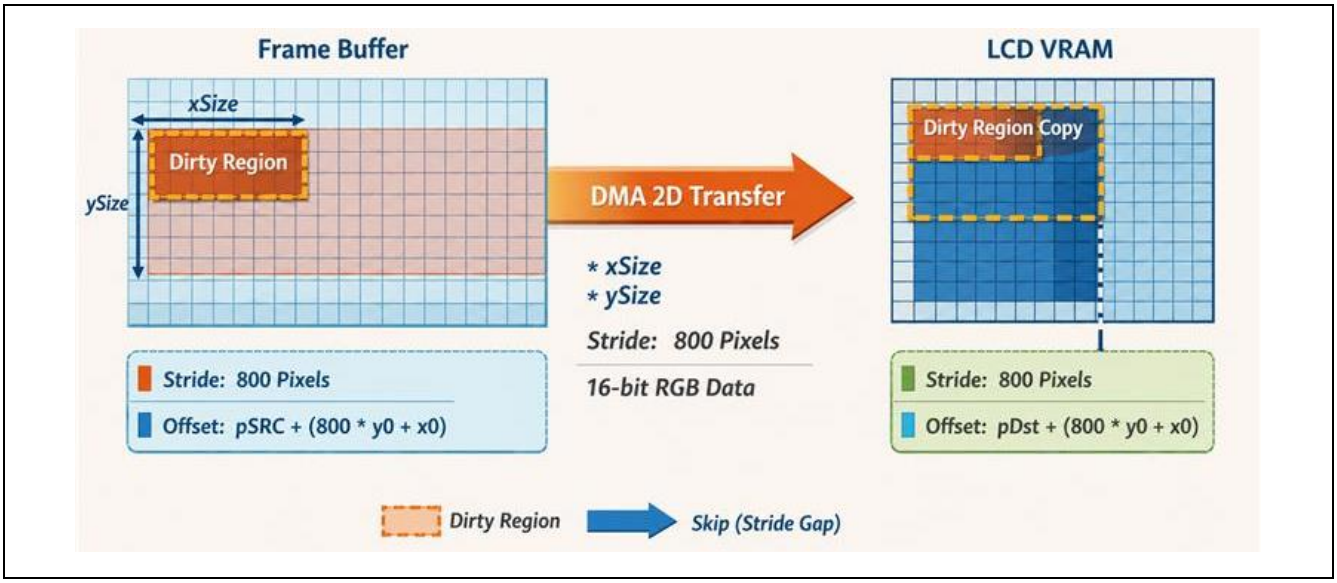


图 1-5 GDMA 工作示意图

1.4 系统流程与设计说明

本范例于 M55M1 开发平台上，通过 emWin AppWizard 建立图形接口，并结合 GDMA 实现 LCD 显示画面之局部更新(Partial Update)，降低 CPU 负载并提升画面更新效率。

本例同步比较 GDMA 与传统 Memory copy 之效能差异，详细比较与流程请参考图 1-6 ~ 图 1-7。

項目	CPU memory copy	GDMA copy
数据路径	Memory → CPU → Memory	Memory → GDMA → Memory
CPU参与程度	高（逐笔处理）	低（只须设定）
效率	较慢	高速（burst）
CPU负载	高	几乎没有
并行能力	无	可与CPU并行

图 1-6 Memory 与 GDMA 处理比较

上电后，会执行系统初始化及 LCD/GUI 设定相关的初始化(例如 *LCD_X_DisplayDriver()*)，接着通过 *MainTask()*执行 GUI 程序。本范例通过点击画面中的按钮(Button Trigger)触发复制数据的工作函数，完成后再以设定 *IMAGE_SetBitmap()*指定要更新显示的对象(Image Widget)并执行该范围的影像更新。流程图如图 1-7 所示。

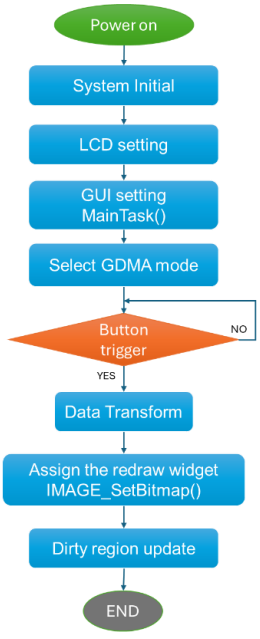


图 1-7 系统流程示意图

1.4.1 各阶段流程说明：

- 显示驱动呼叫 (LCD_X_DisplayDriver)
- GUI 事件触发(GUI Event Trigger)
  - 通过点击画面按钮(Button Widget)开始处理数据(data copy)。
- GDMA 设定
  - Source Address(来源影像数据)
  - Destination Address(LCD Frame Buffer)
  - Transfer Size(宽 × 高 × Bytes per pixel)
  - Line Offset(Stride，若为 2D copy)
  - Transfer Mode(Memory-to-Memory)：通过 GUI 选单切换，不同模式下可支持图片的旋转或镜射功能
- GDMA 执行(GDMA Execution)
- Frame Buffer 更新
  - GDMA 将指定区域的影像数据写入 LCD Frame Buffer。
- 窗口重绘请求(emWin Rendering/ Invalidation)
  - 当事件触发时(如数据更新)，emWin 会更新对应 Widget(如 Image Widget)，并在内部将该区域标记为无效区域(Invalid Area)。
- 重绘区域计算(Dirty Region / Invalid Area Calculation)
  - emWin 会精确计算需更新的矩形区域(Dirty Region)，系统仅针对该区域进行重绘，而非刷新整个屏幕，从而优化显示效能并降低 CPU 负载
- LCD 显示更新(Panel Refresh)
  - LCD 控制器持续从 Frame Buffer 读取数据并输出至面板，完成画面更新。

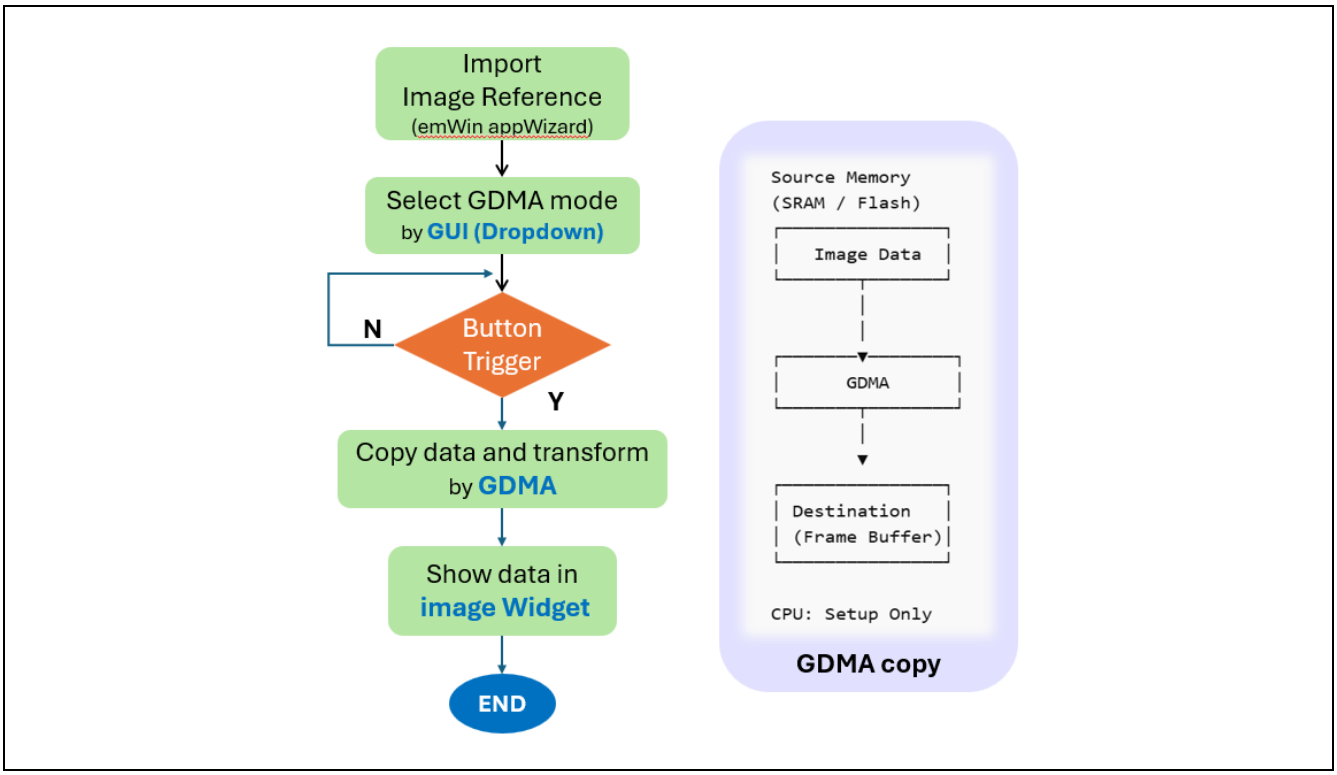


图 1-8 程序流程

1.4.2 emWin appWizard 物件设置

本范例使用 emWin appWizard 做 GUI 开发，通过下拉式选单(Dropdown Widget)选择需要的 GDMA 数据转换模式，并以按钮(Button Widget)作为数据处理的触发媒介。默认两个影像对象(Image Widget)分别存放目标影像及处理后影像。用户选定处理方式后，再通过点击按钮开始撷取影像数据，进行画面局部更新，将处理后影像绘制对应 Image Widget 显示。

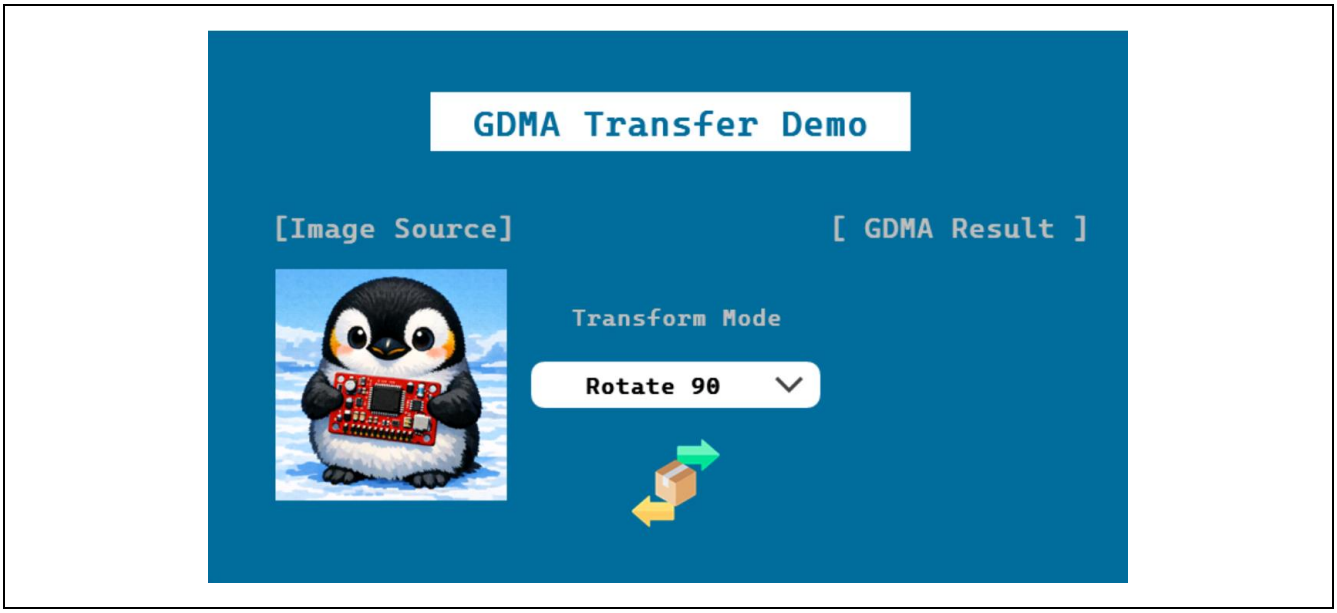


图 1-9 GUI 预设画面示意

GUI 对象设置说明如下表，详细操作步骤可参考新唐 emWin package 内的教学手册(档案位置如图 1-10)。

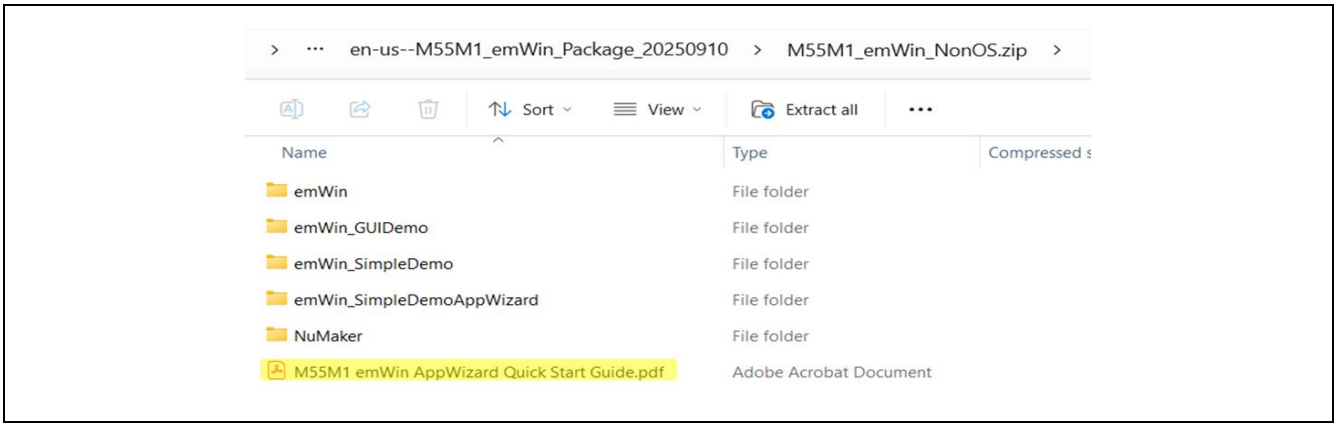


图 1-10 EmWin AppWizard Quick Start 手册位置

Description	Appearance	Widget Type	Widget ID
背景 800x480		Box	ID_BOX_00
说明标示		Text	ID_TEXT_00

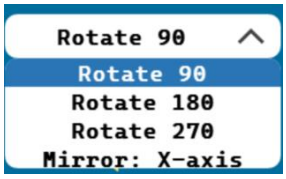
	[Image Source]		ID_TEXT_01
	[GDMA Result]		ID_TEXT_02
	Transform Mode		ID_TEXT_03
目标图片 200x200		Image	ID_IMG_SRC
影像显示 (GDMA result) 200x200	(無填色空物件)		ID_IMG_GDMA
下拉式选单 (250x40)		Dropdown	ID_DROPDOWN
操作按钮 80x80		Button	ID_BTN_MOVE

表 1-1 GUI 对象说明

新增对象后，可于属性字段(Properties)调整文字(Text 对象)显示格式，如下图所示。

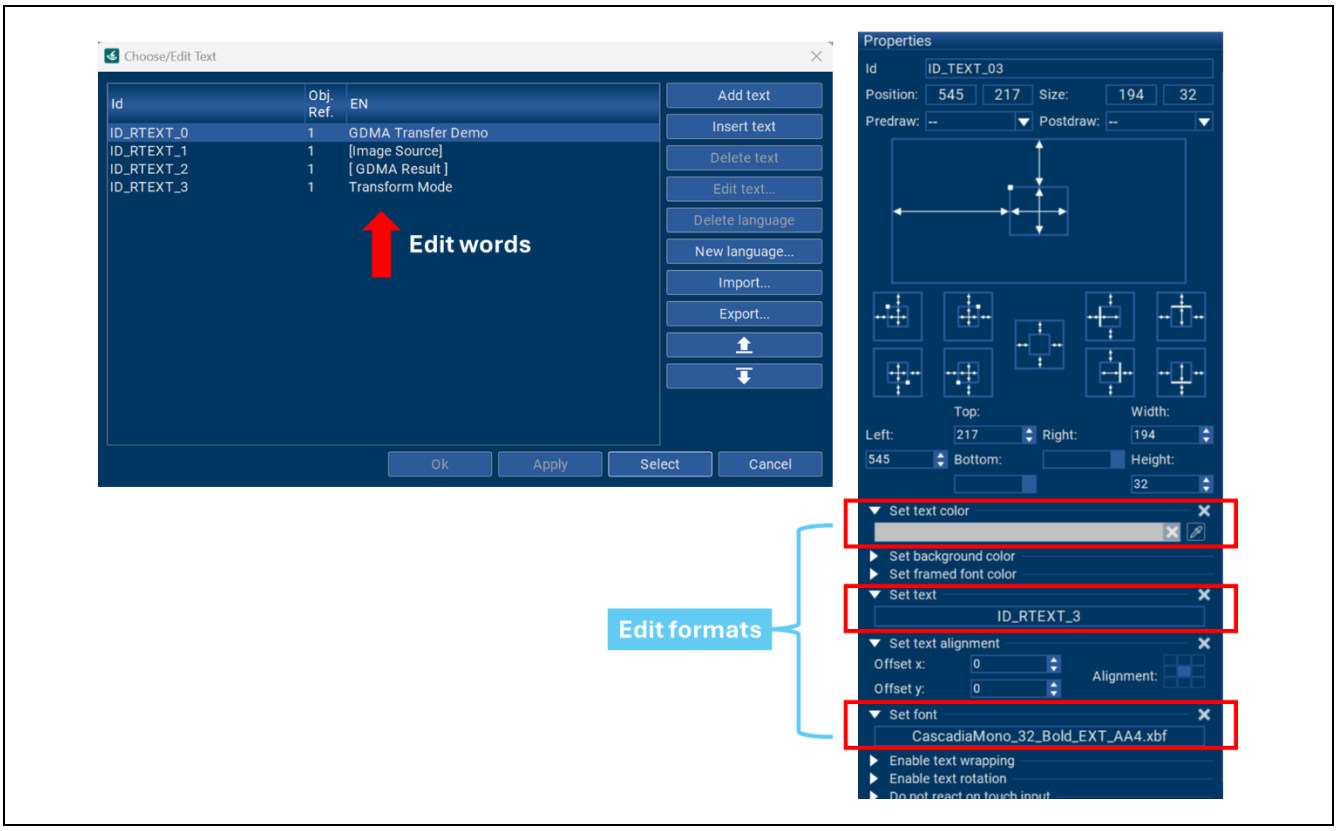


图 1-11 Text 对象属性字段

Button 及 Image 对象可通过「**Set Bitmap**」字段，设置影像作为对象外观。其中对象 ID_IMG_SRC 将做为目标影像，故需将影像格式设为「**High color(565), RB swap**」，避免处理后的影像色彩失真。

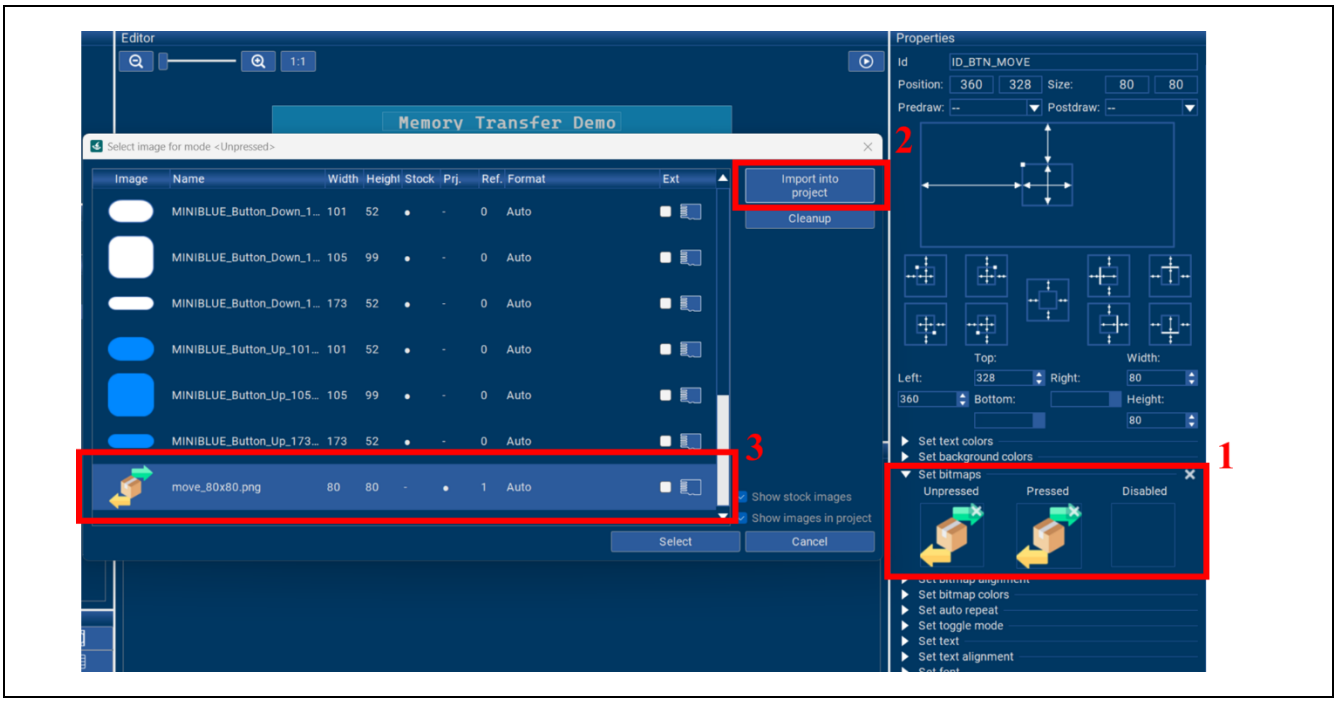


图 1-12 Button 对象属性设置

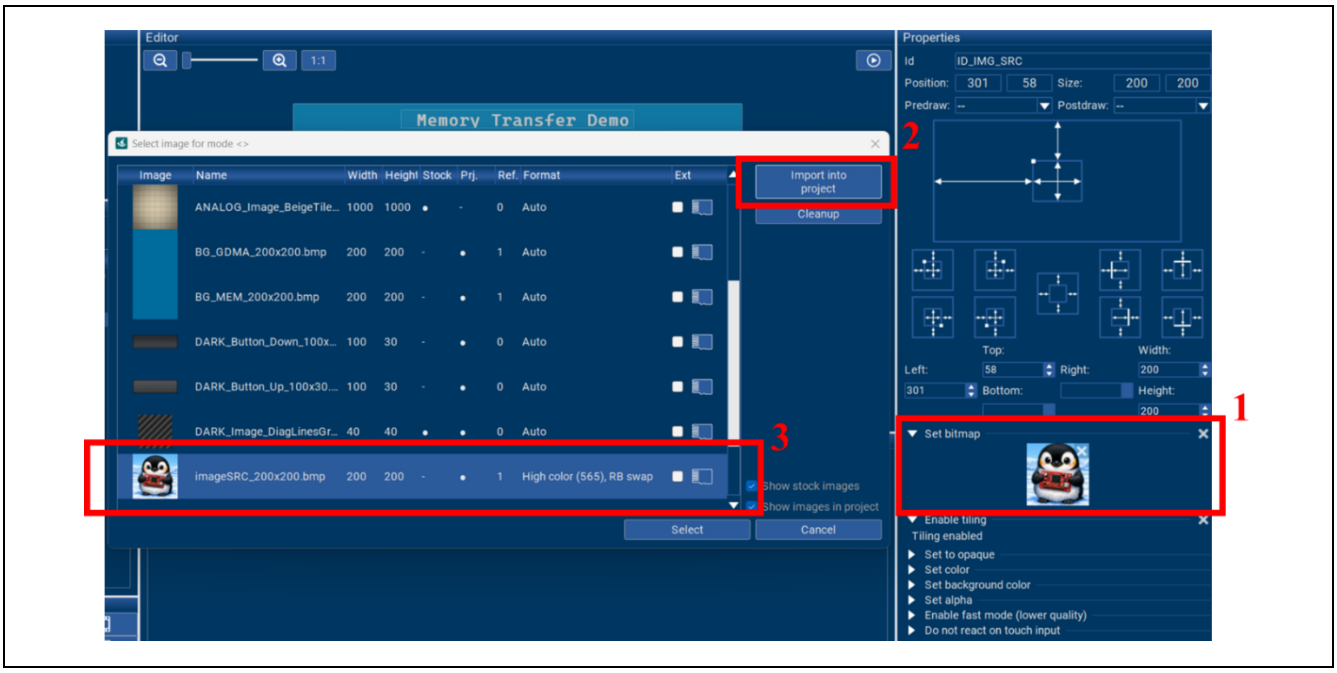


图 1-13 Image 对象属性设置

新增Dropdown物件後，可於屬性欄位對其外觀做調整，並透過「Set Content」加入欲顯示在選單中的內容，如下图 1-14~图 1-15图 1-14 Dropdown设置 新增label所示。

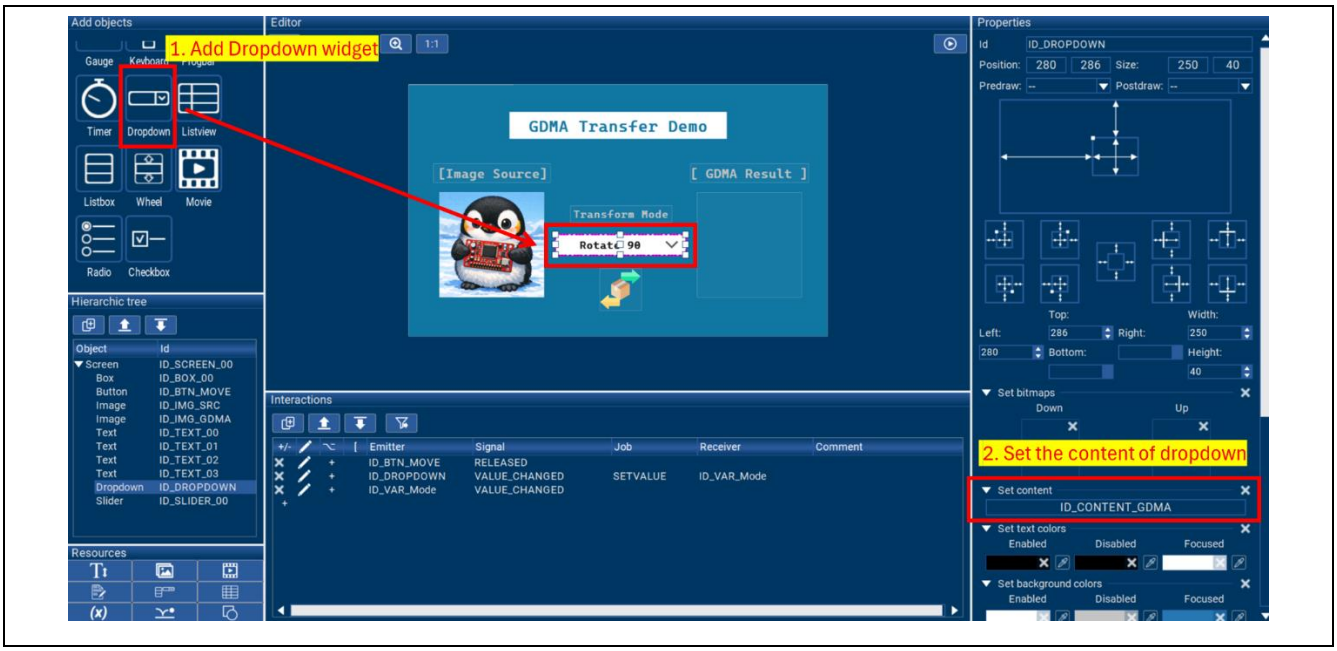


图 1-14 Dropdown 设置 新增 label

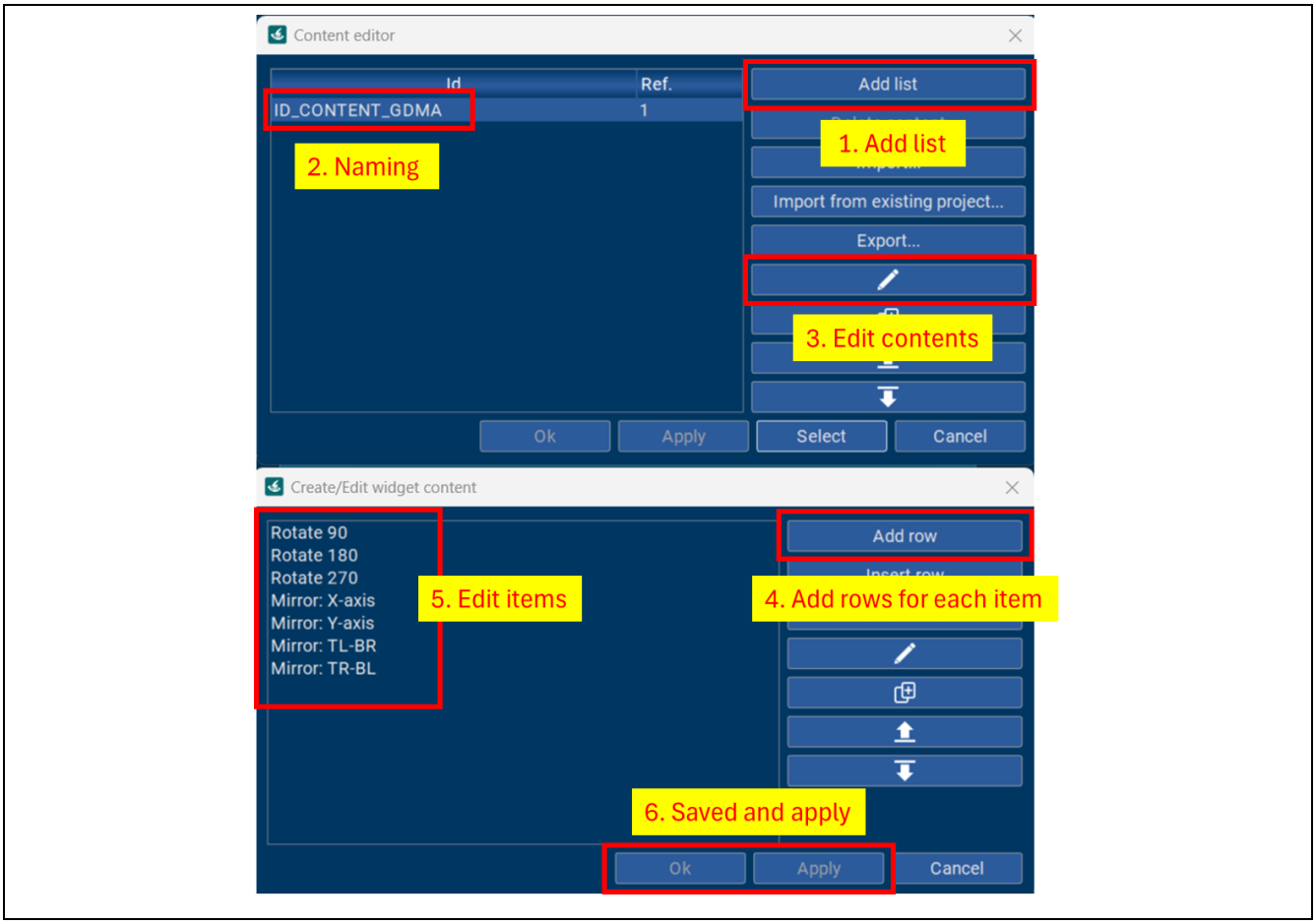


图 1-15 Dropdown 选单内容设置 新增 label

所有对象设置完成，在 Interaction 字段新增 Button 对象的交互功能，设定项为「Emitter (ID_BTN_MOVE)」、「Signal (RELEASED)」、「Job (NULL)」及「Comment (NULL)」，以此产生命令字 ***(ID_SCREEN_00__ID_BTN_MOVE__WM_NOTIFICATION_RELEASED)***，使用者也可于「Edit code」内新增程序(Optional)。

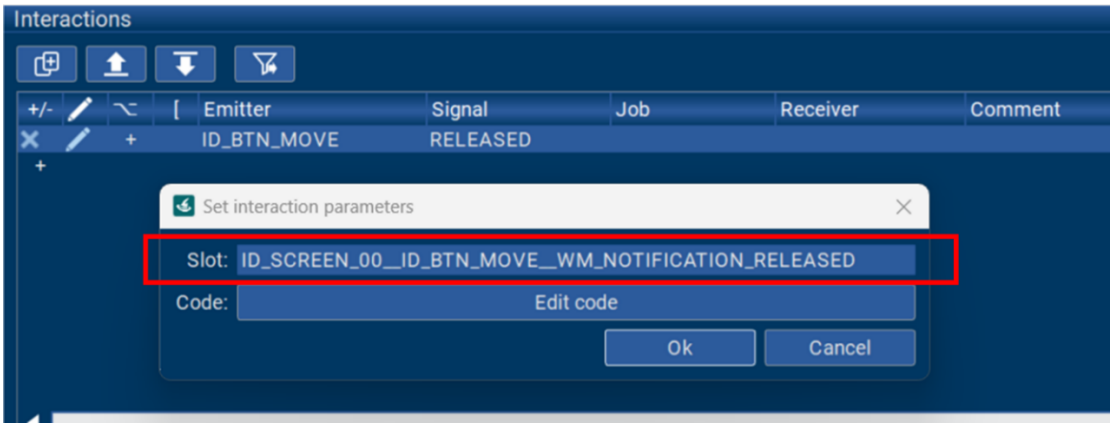


图 1-16 Interactions 设置与生成 Slot

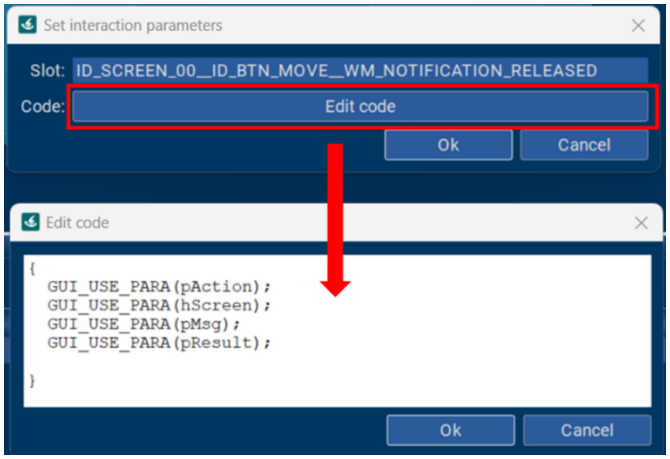


图 1-17 Slot 程序编辑

本范例会通过选单项目改变做 GDMA 处理时的呼叫参数(*dma350_lib_transform_t*)，故新增变量(ID_VAR_Mode)记录下拉式选单的数据。

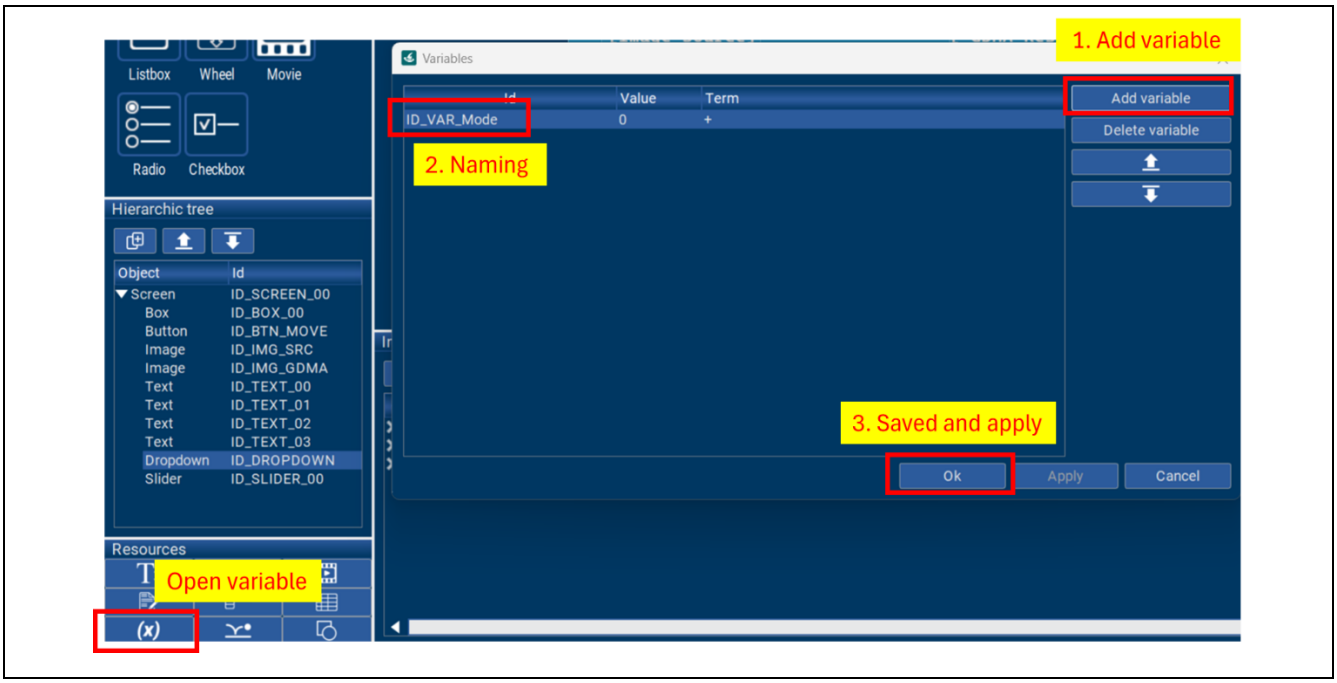


图 1-18 新增变数

新增变量后，通过设置 dropdown 对象与变量 ID_VAR_Mode 间的交互关系(如下表所示)记录 dropdown 更新状态，并于"**Edit code**"字段新增程序代码，使得程序中可通过 *APPW_GetVarDara()*函数读取 Dropdown 对象的 index 值(起始值为 0)识别用户所选择的项目。

Emitter	Signal	Job	Receiver	Emitter
ID_DROPDOWN	VALUE_CHANGED	SETVALUE	ID_VAR_Mode	ID_DROPDOWN
ID_VAR_Mode	VALUE_CHANGED	X	X	ID_VAR_Mode

表 1-2 变数操作之 Interaction 设置

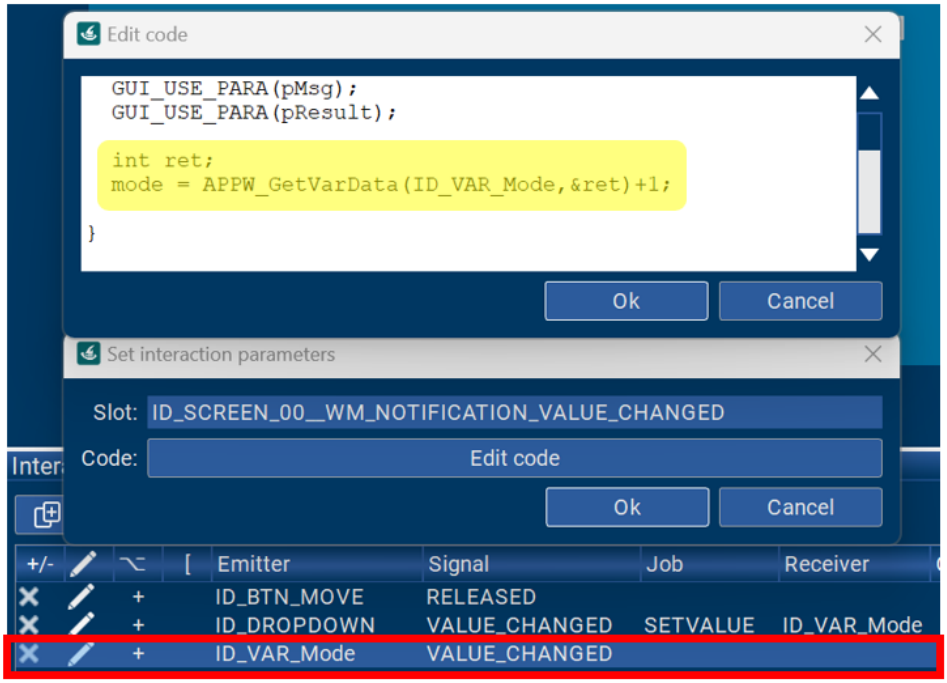


图 1-19 设定交互关系新增 label

1.4.3 导出 emWin appWizard 工程

完成所有 GUI 设置后，将项目导出 Tool bar ***File→ Export & Save***，并将导出的 Resource 与 Source 文件夹复制并取代至 Keil 工程的 Application 文件夹内。（建议先将预设档案删除后，再贴上要使用的新档案）。

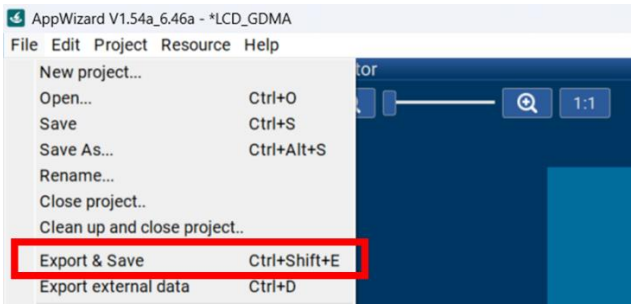


图 1-20 导出 GUI 设置档案

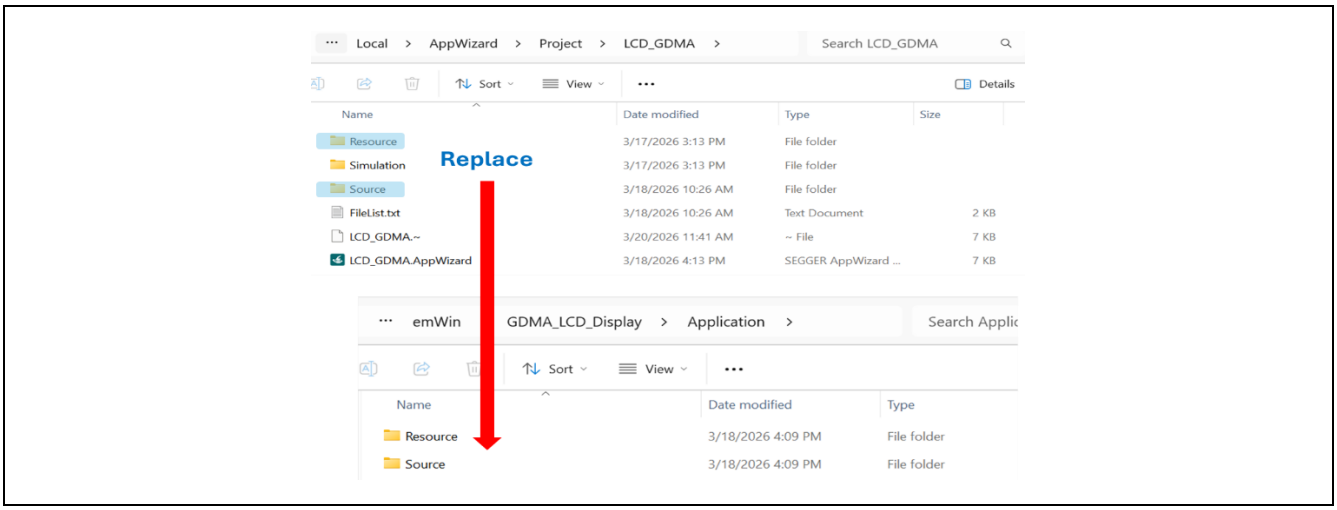


图 1-21 将 AppWizard 档案替换至 Keil 项目

1.4.4 将应用程序布署至开发板

开启 Keil 项目后，通过右键点选「*Add Existing Files to Group "Application"*」将 AppWizard 生成的档案全部放入项目，并确认 **Includes Path** 字段已新增所有档案的路径，最后执行编译。

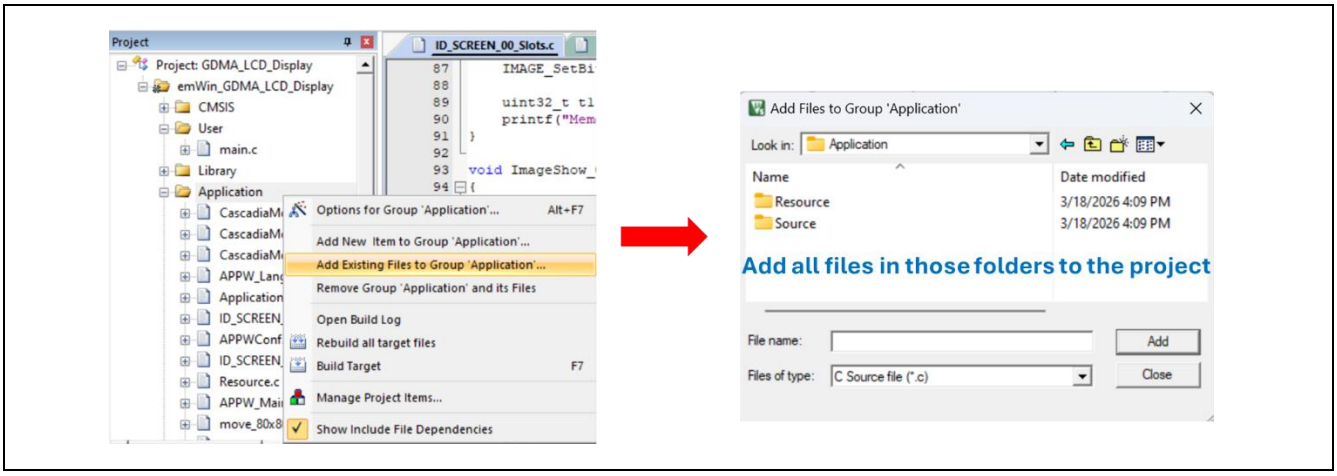


图 1-22 将 AppWizard 档案加入项目

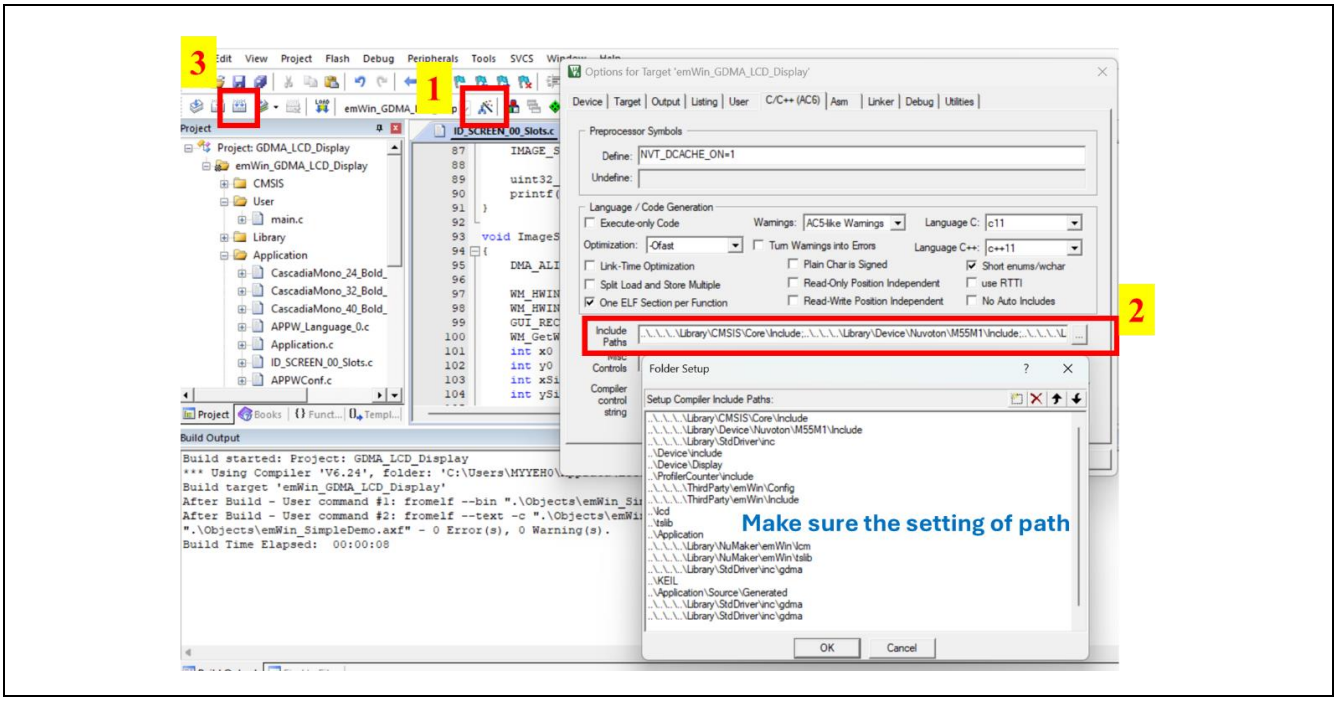


图 1-23 档案路径设定与编译

最后通过 type-C 线将计算机连接至 M55M1 开发板，上电后执行程序下载(Download)即可。

1.4.5 将 Keil 项目转换成 VS code 项目

若使用者有 VS Code 开发需求，可将已汇入 GUI 档案的 Keil 项目进行转换。首先开启 VS Code IDE，于左侧功能列选取”**Explorer”**分页，开启 Keil 项目所在的文件夹。

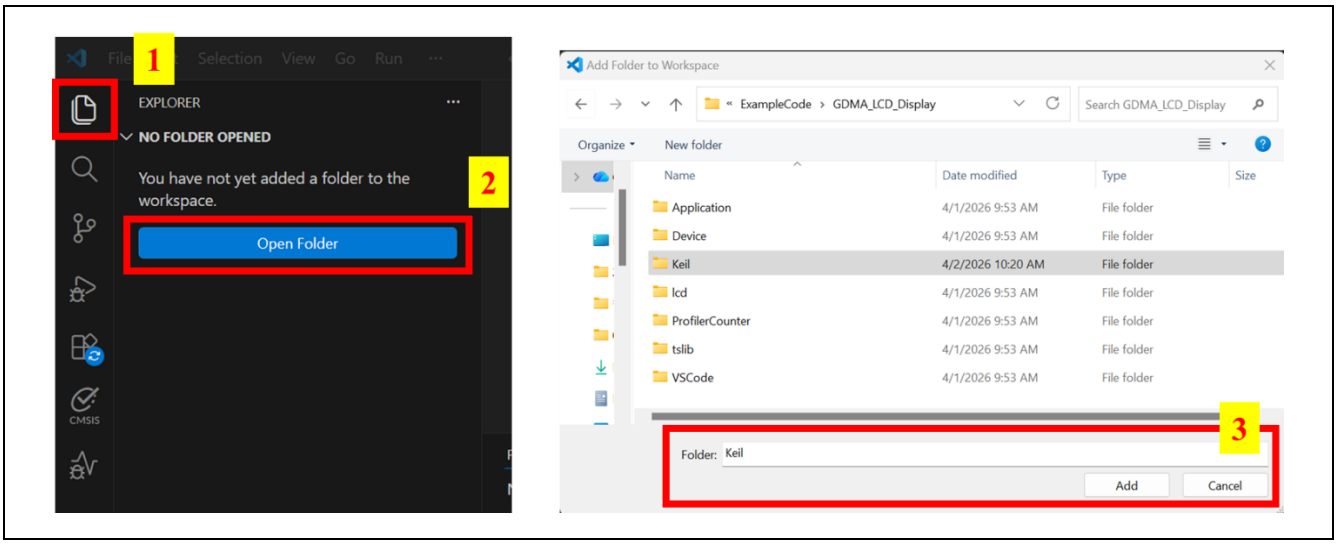


图 1-24 在 VS Code IDE 开启 Keil 文件夹

汇入文件夹后，选择 Keil 工程的***.uvprojx*** 档并点选右键 ”*Convert µVision project to CMSIS solution*” 开始转换。

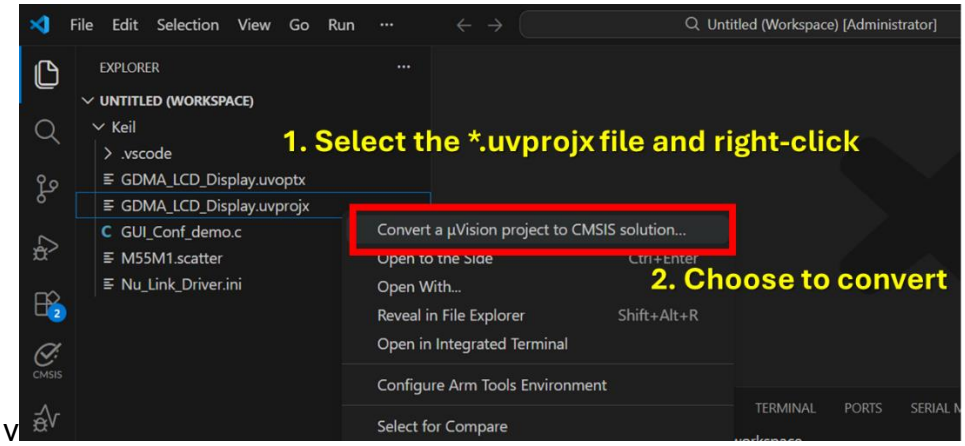


图 1-25 通过 VS Code 内建功能进行转换

转换后将自动生成 *launch.json* 、*tasks.json* 、*vcpkg-configuration.json* 、**.cproject.yml* 与 **.csolution.yml* 等，开发 VS Code 项目所需的档案。

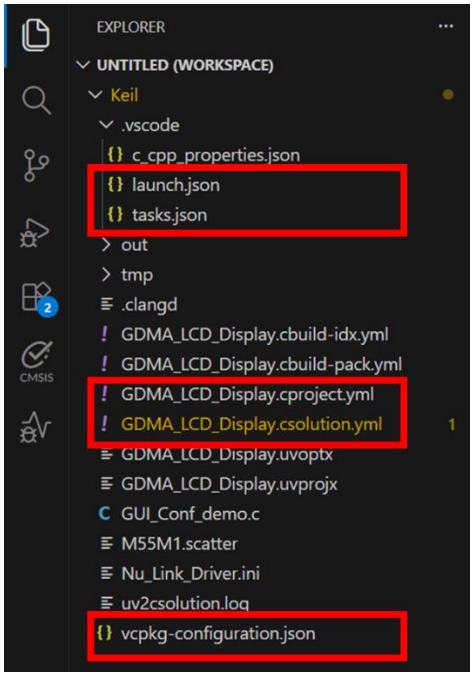


图 1-26 转换后的 VS Code Files

1.5 执行结果

本范例代码执行编译和刻录的结果如图 1-27~图 1-28 所示，点选画面中间的按钮通过 GDMA 复制目标影像(Image Source)，并依使用者选择的方式将处理结果绘至影像对象中。

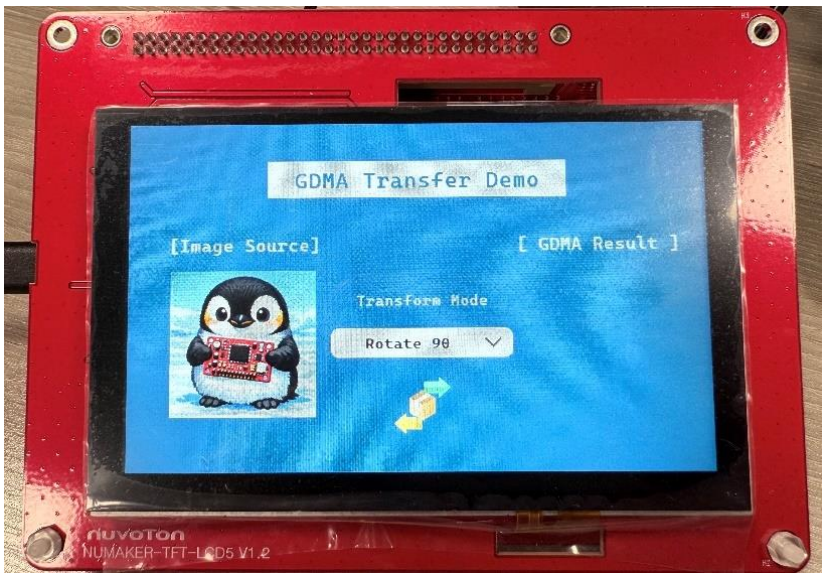


图 1-27 程序执行结果 (GUI 画面)




GDMA Transfer Demo
[Image Source]
[GDMA Result]
Transform Mode
Mirror: Y-axis
GDMA Transfer Demo
[Image Source]
[GDMA Result]
Transform Mode
Mirror: TR-BL
GDMA Transfer Demo
[Image Source]
[GDMA Result]
Transform Mode
Mirror: TL-BR

图 1-28 操作结果

2. 代碼介紹

本范例通过 emWin 设计 GUI 并将档案汇入 Keil 项目中。其中 *main.c* 会对程序进行初始化，设定触控屏幕、frame buffer 与 Multiple Buffering 等，接着导向应用主程序 *MainTask()*开始执行项目。下列程序分别位于 *main.c* 与 *APPW_MainTask.c*。

```
int main(void)
{
    /* Init System, peripheral clock and multi-function I/O */
    SYS_Init();

    /* Init Debug UART to 115200-8N1 for print message */
    InitDebugUart();

    g_pu8FrameBufPtr  = (U8 *)((U32)g_u8FrameBuf);
    g_enable_Touch = 0;
    //
    // Initialize timer
    //
    tmr_init();
    //
    // NVT_LCM_VERSION defined in lcm.h
    //
#if NVT_LCM_VERSION == 1

    FT5316_low_level_init();
    FT5316_startup_sequence();

#elif NVT_LCM_VERSION == 2
    GUI_Init();
    WM_MULTIBUF_Enable(1);
    Init_TouchPanel();
    #if 0   /* set 1 to do touch calibration */
    ts_calibrate(__DEMO_TS_WIDTH__, __DEMO_TS_HEIGHT__);
    #else   /* set 0 to utilize default touch parameters */
    ts_init();
    #endif
#else
    #error "NVT_LCM_VERSION in lcm.h defined as unknown version!"
#endif
    g_enable_Touch = 1;

    printf("\n\nCPU @ %dHz\n", SystemCoreClock);
    printf("+-----------------------------------------------------+ \n");
    printf("|                   Example Code Demo                 | \n");
    printf("|                  GDMA Transform Ways                | \n");
    printf("+-----------------------------------------------------+ \n");

    MainTask();
}
```

APPW_MainTask.c

```
#include "Generated/Resource.h"

void MainTask(void) {
  //
  // Setup configuration dependent pointers
  //
  APPW_X_Setup();
  //
  // Initialize AppWizard
  //
  APPW_Init(APPW_PROJECT_PATH);
  //
  // Create all persistent screens except initial screen
  //
  APPW_CreatePersistentScreens();
  //
  // Create initial screen...
  //
  APPW_CreateRoot(APPW_INITIAL_SCREEN, WM_HBKWIN);
  //
  // ...and keep it alive
  //
  for (APPW_Run = 1; APPW_Run; ) {
    APPW_Exec();
    GUI_X_Delay(5);
  }
}
```

通过 emWin appWizard 制作的对象，将自动定义所有对象在程序中的名称与信息，并为每个画面(Screen Widget)生成对应程序文件供编辑使用，其中对象相关定义将位于 *ID_SCREEN_00.c*，可于此查询各对象在程序中的数据定义名称，例如此范例中 ID_IMG_SRC 内容被定义为 ***acimageSRC_200x200***(名称会因 emWin appWizard 汇入图档档名变动)。

ID_SCREEN_00.c

```
static APPW_CREATE_ITEM _aCreate[] = {
  { WM_OBJECT_WINDOW_Create,
    ID_SCREEN_00, 0,
    { { { DISPOSE_MODE_REL_PARENT, 0, 0, 0 },
        { DISPOSE_MODE_REL_PARENT, 0, 0, 0 },
        { DISPOSE_MODE_NULL, 0, 0, 0 },
        { DISPOSE_MODE_NULL, 0, 0, 0 },
      },
      800, 480, 0, 0, 0, 0
    },
    { 0, 0 }
  },
  { WM_OBJECT_BOX_Create,
    ID_BOX_00, ID_SCREEN_00,
    { { { DISPOSE_MODE_REL_PARENT, 0, 0, 0 },
        { DISPOSE_MODE_REL_PARENT, 0, 0, 0 },
        { DISPOSE_MODE_REL_PARENT, 0, 0, 0 },
        { DISPOSE_MODE_REL_PARENT, 0, 0, 0 },
```

```
      },
      0, 0, 0, 0, 0, 0
    },
    { 0, 0 }
  },
  { WM_OBJECT_BUTTON_Create,
    ID_BTN_MOVE, ID_SCREEN_00,
    { { { DISPOSE_MODE_REL_PARENT, 365, 0, 0 },
        { DISPOSE_MODE_REL_PARENT, 351, 0, 0 },
        { DISPOSE_MODE_NULL, 0, 0, 0 },
        { DISPOSE_MODE_NULL, 0, 0, 0 },
      },
      80, 80, 0, 0, 0, 0
    },
    { 0, 0 }
  },
  { WM_OBJECT_IMAGE_Create,
    ID_IMG_SRC, ID_SCREEN_00,
    { { { DISPOSE_MODE_REL_PARENT, 59, 0, 0 },
        { DISPOSE_MODE_REL_PARENT, 206, 0, 0 },
        { DISPOSE_MODE_NULL, 0, 0, 0 },
        { DISPOSE_MODE_NULL, 0, 0, 0 },
      },
      200, 200, 0, 0, 0, 0
    },
    { 0, 0 }
  },
  { WM_OBJECT_IMAGE_Create,
    ID_IMG_GDMA, ID_SCREEN_00,
    { { { DISPOSE_MODE_REL_PARENT, 551, 0, 0 },
        { DISPOSE_MODE_REL_PARENT, 206, 0, 0 },
        { DISPOSE_MODE_NULL, 0, 0, 0 },
        { DISPOSE_MODE_NULL, 0, 0, 0 },
      },
      200, 200, 0, 0, 0, 0
    },
    { 0, 0 }
  },
  { WM_OBJECT_TEXT_Create,
    ID_TEXT_00, ID_SCREEN_00,
    { { { DISPOSE_MODE_REL_PARENT, 193, 0, 0 },
        { DISPOSE_MODE_REL_PARENT, 53, 0, 0 },
        { DISPOSE_MODE_NULL, 0, 0, 0 },
        { DISPOSE_MODE_NULL, 0, 0, 0 },
      },
      415, 51, 0, 0, 0, 0
    },
    { 0, 0 }
  },
  { WM_OBJECT_TEXT_Create,
    ID_TEXT_01, ID_SCREEN_00,
    { { { DISPOSE_MODE_REL_PARENT, 47, 0, 0 },
        { DISPOSE_MODE_REL_PARENT, 151, 0, 0 },
        { DISPOSE_MODE_NULL, 0, 0, 0 },
        { DISPOSE_MODE_NULL, 0, 0, 0 },
      },
```

```
      224, 36, 0, 0, 0, 0
    },
    { 0, 0 }
  },
  { WM_OBJECT_TEXT_Create,
    ID_TEXT_02, ID_SCREEN_00,
    { { { DISPOSE_MODE_REL_PARENT, 538, 0, 0 },
        { DISPOSE_MODE_REL_PARENT, 151, 0, 0 },
        { DISPOSE_MODE_NULL, 0, 0, 0 },
        { DISPOSE_MODE_NULL, 0, 0, 0 },
      },
      226, 36, 0, 0, 0, 0
    },
    { 0, 0 }
  },
  { WM_OBJECT_TEXT_Create,
    ID_TEXT_03, ID_SCREEN_00,
    { { { DISPOSE_MODE_REL_PARENT, 309, 0, 0 },
        { DISPOSE_MODE_REL_PARENT, 231, 0, 0 },
        { DISPOSE_MODE_NULL, 0, 0, 0 },
        { DISPOSE_MODE_NULL, 0, 0, 0 },
      },
      194, 32, 0, 0, 0, 0
    },
    { 0, 0 }
  },
  { WM_OBJECT_DROPDOWN_Create,
    ID_DROPDOWN, ID_SCREEN_00,
    { { { DISPOSE_MODE_REL_PARENT, 280, 0, 0 },
        { DISPOSE_MODE_REL_PARENT, 286, 0, 0 },
        { DISPOSE_MODE_NULL, 0, 0, 0 },
        { DISPOSE_MODE_NULL, 0, 0, 0 },
      },
      250, 40, 0, 0, 0, 0
    },
    { 0, 0 }
  },
};

/*********************************************************************
*
*       _aSetup
*/
static GUI_CONST_STORAGE APPW_SETUP_ITEM _aSetup[] = {
  { ID_BOX_00,    APPW_SET_PROP_COLOR,        { ARG_V(0xff006c9e) } },
  { ID_BTN_MOVE,  APPW_SET_PROP_SBITMAPS,     { ARG_VP(0, acmove_80x80),
                                                ARG_VP(0, acmove_80x80), } },
  { ID_IMG_SRC,   APPW_SET_PROP_TILE,         { ARG_V(0) } },
  { ID_IMG_SRC,   APPW_SET_PROP_SBITMAP,      { ARG_VP(0, acimageSRC_200x200),
                                                ARG_V(160054), } },
  { ID_IMG_GDMA,  APPW_SET_PROP_TILE,         { ARG_V(0) } },
  { ID_IMG_GDMA,  APPW_SET_PROP_SBITMAP,      { ARG_VP(0, NULL), } },
  { ID_TEXT_00,   APPW_SET_PROP_COLOR,        { ARG_V(0xff006c9e) } },
  { ID_TEXT_00,   APPW_SET_PROP_ALIGNTEXT,    { ARG_V(GUI_ALIGN_HCENTER |
GUI_ALIGN_VCENTER),
                                                ARG_V(0),
                                                ARG_V(0) } },
```

```
  { ID_TEXT_00,   APPW_SET_PROP_TEXTID,       { ARG_V(ID_RTEXT_0) } },
  { ID_TEXT_00,   APPW_SET_PROP_FONT,         { ARG_VPF(0, acCascadiaMono_40_Bold_EXT_AA4,
&APPW__aFont[0]) } },
  { ID_TEXT_00,   APPW_SET_PROP_BKCOLOR,      { ARG_V(GUI_WHITE) } },
  { ID_TEXT_01,   APPW_SET_PROP_COLOR,        { ARG_V(0xffc0c0c0) } },
  { ID_TEXT_01,   APPW_SET_PROP_ALIGNTEXT,    { ARG_V(GUI_ALIGN_HCENTER |
GUI_ALIGN_VCENTER),
                                                ARG_V(0),
                                                ARG_V(0) } },
  { ID_TEXT_01,   APPW_SET_PROP_TEXTID,       { ARG_V(ID_RTEXT_1) } },
  { ID_TEXT_01,   APPW_SET_PROP_FONT,         { ARG_VPF(0, acCascadiaMono_32_Bold_EXT_AA4,
&APPW__aFont[1]) } },
  { ID_TEXT_02,   APPW_SET_PROP_COLOR,        { ARG_V(0xffc0c0c0) } },
  { ID_TEXT_02,   APPW_SET_PROP_ALIGNTEXT,    { ARG_V(GUI_ALIGN_HCENTER |
GUI_ALIGN_VCENTER),
                                                ARG_V(0),
                                                ARG_V(0) } },
  { ID_TEXT_02,   APPW_SET_PROP_TEXTID,       { ARG_V(ID_RTEXT_2) } },
  { ID_TEXT_02,   APPW_SET_PROP_FONT,         { ARG_VPF(0, acCascadiaMono_32_Bold_EXT_AA4,
&APPW__aFont[1]) } },
  { ID_TEXT_03,   APPW_SET_PROP_COLOR,        { ARG_V(0xffc0c0c0) } },
  { ID_TEXT_03,   APPW_SET_PROP_ALIGNTEXT,    { ARG_V(GUI_ALIGN_HCENTER |
GUI_ALIGN_VCENTER),
                                                ARG_V(0),
                                                ARG_V(0) } },
  { ID_TEXT_03,   APPW_SET_PROP_TEXTID,       { ARG_V(ID_RTEXT_3) } },
  { ID_TEXT_03,   APPW_SET_PROP_FONT,         { ARG_VPF(0, acCascadiaMono_26_Bold_EXT_AA4,
&APPW__aFont[2]) } },
  { ID_DROPDOWN,  APPW_SET_PROP_SBITMAPS,     { ARG_VP(0,
acDARK_Angle_Arrow_Down_Gray_24x14),
                                                ARG_VP(0,
acDARK_Angle_Arrow_Up_Gray_24x14), } },
  { ID_DROPDOWN,  APPW_SET_PROP_BKCOLORS,     { ARG_V(GUI_WHITE),
                                                ARG_V(0xffc4c4c4),
                                                ARG_V(0xff2777b3) } },
  { ID_DROPDOWN,  APPW_SET_PROP_COLORS,       { ARG_V(GUI_BLACK),
                                                ARG_V(GUI_BLACK),
                                                ARG_V(GUI_WHITE) } },
  { ID_DROPDOWN,  APPW_SET_PROP_COLOR,        { ARG_V(0xff3c3c3c) } },
  { ID_DROPDOWN,  APPW_SET_PROP_FONT,         { ARG_VPF(0, acCascadiaMono_26_Bold_EXT_AA4,
&APPW__aFont[2]) } },
  { ID_DROPDOWN,  APPW_SET_PROP_CONTENT,      { ARG_P(&ID_CONTENT_GDMA_Data) } },
  { ID_DROPDOWN,  APPW_SET_PROP_ALIGN,        { ARG_V(GUI_ALIGN_HCENTER |
GUI_ALIGN_VCENTER) } },
  { ID_DROPDOWN,  APPW_SET_PROP_RADIUS,       { ARG_V(11) } },
  { ID_DROPDOWN,  APPW_SET_PROP_BORDER,       { ARG_V(15) } },
  { ID_DROPDOWN,  APPW_SET_PROP_SIZE,         { ARG_V(100) } },

};
```

emWin appWizard 中所使用的图片会在导出档案时，自动生成 C 档，例如本例的 *imageSRC_200x200.c*。而为避免数据不对齐导致 GDMA 处理数据时发生影像偏移现象，故需在此新增对齐数据的定义 ***#define EMWIN_SUFFIX_DTA DMA***:

```
#include "AppWizard.h"
```

```
#define EMWIN_SUFFIX_DTA   __attribute__((aligned(4)))   //word-alignment for DMA

#ifndef EMWIN_PREFIX_DTA
  #define EMWIN_PREFIX_DTA
#endif

#ifndef EMWIN_SUFFIX_DTA
  #define EMWIN_SUFFIX_DTA
#endif
EMWIN_PREFIX_DTA GUI_CONST_STORAGE unsigned char acimageSRC_200x200[80016UL + 1]
EMWIN_SUFFIX_DTA = {
…
…
};
```

新唐的 BSP 有提供使用 GDMA library，本范例即使用函数 ***dma350_draw_from_canvas()***执行数据搬移，用户可于程序顶端通过定义 include "***dma350_lib.h”及#include "gdma.h***"使用，并可于 *dma350_lib.h* 内参阅其他相关 library 的使用说明。

dma350_lib.h

```
/* DMA350 DMA Library transformation enumeration type */
enum dma350_lib_transform_t
{
    DMA350_LIB_TRANSFORM_NONE = 0,
    DMA350_LIB_TRANSFORM_ROTATE_90,   /*!< Rotate 90 degrees clockwise */
    DMA350_LIB_TRANSFORM_ROTATE_180,  /*!< Rotate 180 degrees clockwise */
    DMA350_LIB_TRANSFORM_ROTATE_270,  /*!< Rotate 270 degrees clockwise */
    DMA350_LIB_TRANSFORM_MIRROR_VER,  /*!< Mirror along X axis. Top to bottom */
    DMA350_LIB_TRANSFORM_MIRROR_HOR,  /*!< Mirror along Y axis. Left to right */
    DMA350_LIB_TRANSFORM_MIRROR_TLBR, /*!< Mirror along diagonal.TopLeft to BottomRight */
    DMA350_LIB_TRANSFORM_MIRROR_TRBL, /*!< Mirror along diagonal.TopRight to BottomLeft */
};

…

/**
 * \brief 2D Copy from canvas (area within a source bitmap) to within a
 *        destination bitmap, while applying various possible transformations.
 *        If the destination size is larger than the source, the source image
 *        will be wrapped (repeated).
 *
 * \param[in] dev             DMA350 channel device struct \ref dma350_ch_dev_t
 * \param[in] src             Source address, top left corner
 * \param[in] des             Destination address, top left corner
 * \param[in] src_width       Source width
 * \param[in] src_height      Source height
 * \param[in] src_line_width  Source line width
 * \param[in] des_width       Destination width
 * \param[in] des_height      Destination height
 * \param[in] des_line_width  Destination line width
 * \param[in] pixelsize       Size of a pixel as in \ref dma350_ch_transize_t
 * \param[in] transform       Transform type as in \ref dma350_lib_transform_t
 * \param[in] exec_type       Execution type as in \ref dma350_lib_exec_type_t
 *
 * \return Result of the operation \ref dma350_lib_error_t
 *
```

```
 * \note Destination width and height denote the area which will be filled at
 *       the destination address. The copy always starts from the top left
 *       corner of the source. If the requested destination size does not match
 *       the source, the resulting image will be repeated / cropped.
 */
enum dma350_lib_error_t dma350_draw_from_canvas(struct dma350_ch_dev_t *dev,
                                                const void *src, void *des,
                                                uint32_t src_width, uint16_t src_height,
                                                uint16_t src_line_width,
                                                uint32_t des_width, uint16_t des_height,
                                                uint16_t des_line_width,
                                                enum dma350_ch_transize_t pixelsize,
                                                enum dma350_lib_transform_t transform,
                                                enum dma350_lib_exec_type_t exec_type);
```

本范例相关操作皆位于 *ID_SCREEN_00_slots.c*，emWin appWizard 设置的 interaction 触发函数也定义于此(*ID_SCREEN_00__ID_BTN_MOVE__WM_NOTIFICATION_RELEASED*)，使用者可于此新增欲执行的程序及相关定义宣告。若想将数据汇入形成 Widget 外观，需定义格式为 GUI_BITMAP 的数据缓冲区，格式定义位于 *GUI_Type.h* 中。本例中宣告型态为 unsigned short 的 buffer 并指定给特定 Image 对象，通过更新 buffer 数据更新影像内容。

注意：每次更新时，需以 *IMAGE_SetBitmap()*做画面刷新。

ID_SCREEN_00_slots.c

```
#include "dma350_lib.h"
#include "gdma.h"

#define IMG_W 200                                                    //image width
#define IMG_H 200                                                    //image height
#define DMA_ALIGNMENT   __attribute__((aligned(DCACHE_LINE_SIZE))) // alignment

int mode = 1;                 //default mode of GDMA

DMA_ALIGNMENT static unsigned short GDMA_Buffer01[ DCACHE_ALIGN_LINE_SIZE(200 * 200) ];
static GUI_BITMAP GDMA_IMG_200x200 = {
             200,
             200,
             200*2,
             16,
             (const unsigned char*)GDMA_Buffer01,
           NULL,
             GUI_DRAW_BMPM565
};

// customer's code
void ImageShow_GDMA(WM_HWIN hScreen, const unsigned char *src, int mode)
{
    DMA_ALIGNMENT const uint8_t *image = src + 16;   // skip header

    WM_HWIN ImgWidget1 = WM_GetDialogItem(hScreen, ID_IMG_GDMA);

    GUI_RECT ImgRect;
```

```
    WM_GetWindowRectEx(ImgWidget1, &ImgRect);
      int x0    = ImgRect.x0;
    int y0    = ImgRect.y0;
    int xSize = ImgRect.x1 - ImgRect.x0 + 1;
    int ySize = ImgRect.y1 - ImgRect.y0 + 1;

    // GDMA transformation
    dma350_draw_from_canvas(
        GDMA_CH_DEV_S[0],
        (const void *)image,
        GDMA_Buffer01,            // destination line start,
        xSize, ySize, xSize,      // width, height, source stride(line width of image)
        xSize, ySize, xSize,      // dest width, height, destination stride(line width of
buffer)
        DMA350_CH_TRANSIZE_16BITS,
        mode,          // transformation enumeration type, refer to dma350_lib_transform_t
        DMA350_LIB_EXEC_BLOCKING);// Assign a GUI_BITMAP to the Image widget for display
on redraw

#if (NVT_DCACHE_ON == 1)
    // Cache maintenance: Invalidate the DMA buffer to ensure CPU reads the latest data
before redrawing.
     SCB_InvalidateDCache_by_Addr (GDMA_Buffer01, xSize*ySize*2 );
#endif

    IMAGE_SetBitmap(ImgWidget1, &GDMA_IMG_200x200);
}

…

// Interaction function
void ID_SCREEN_00__WM_NOTIFICATION_VALUE_CHANGED(APPW_ACTION_ITEM * pAction, WM_HWIN
hScreen, WM_MESSAGE * pMsg, int * pResult) {
  GUI_USE_PARA(pAction);
  GUI_USE_PARA(hScreen);
  GUI_USE_PARA(pMsg);
  GUI_USE_PARA(pResult);

  int ret;
  mode = APPW_GetVarData(ID_VAR_Mode,&ret)+1;
}
```

GUI_Type.h

```
typedef struct {
  I16 xSize;
  I16 ySize;
} GUI_SIZE;

typedef struct {
  void      (* pfDraw)  (int x0,
                         int y0,
                         int xsize,
                         int ysize,
                         const U8 * pPixel,
                         const LCD_LOGPALETTE * pLogPal,
```

```
                          int xMag,
                          int yMag);
                          GUI_COLOR (* pfIndex2Color)(LCD_PIXELINDEX Index);
                          void      (* pfDrawHW)(int x0,
                          int y0,
                          int xsize,
                          int ysize,
                          const U8 * pPixel,
                          const LCD_LOGPALETTE * pLogPal,
                          int xMag,
                          int yMag);
                          const LCD_API_COLOR_CONV * pColorConvAPI;
} GUI_BITMAP_METHODS;

typedef struct {
  U16P XSize;
  U16P YSize;
  U16P BytesPerLine;
  U16P BitsPerPixel;
  const U8 * pData;
  const GUI_LOGPALETTE * pPal;
  const GUI_BITMAP_METHODS * pMethods;
} GUI_BITMAP;
```

3. 软件与硬件需求

3.1 软件需求

- BSP 版本
  - M55M1_Series_BSP_CMSIS_V3.01.003
- IDE 版本
  - Keil uVision 5.43
  - Visual Studio Code V1.112.0
    - CMSIS Solution extension
    - Arm Tools Environment Manager
    - Arm Compiler (ARMClang 6.x)
- AppWizard 版本
  - AppWizard_V154a_646a

3.2 硬件需求

- 电路组件
  - NuMaker-X-M55M1
- 线路示意图
  - 通过 Type-C USB 连接 NuMaker-X-M55M1 与 PC。

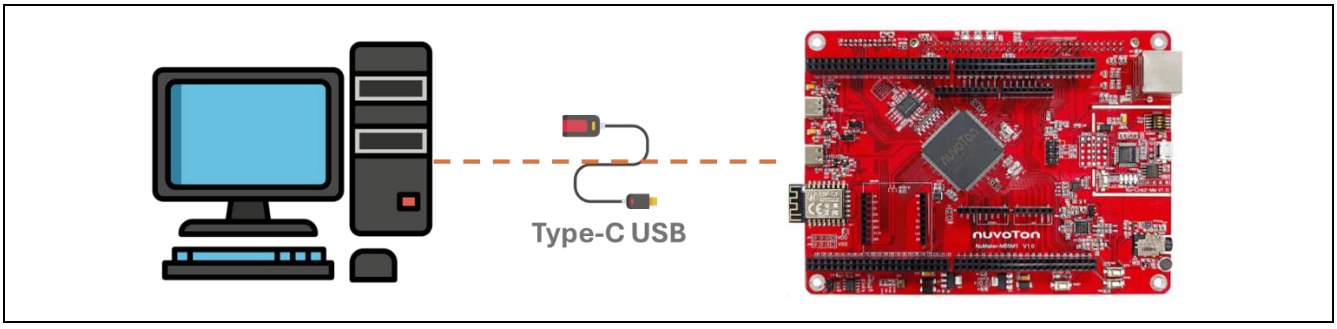


图 3-1 M55M1 接线示意图

4. 目录信息

目录结构如下图所示。

EC_M55M1_GDMA_LCD_Display_V1.00	
Library	Sample code header and source files
CMSIS	Cortex® Microcontroller Software Interface Standard (CMSIS) by Arm® Corp.
Device	CMSIS compliant device header file
NuMaker	
emWin	Libraries of touch screen
StdDriver	All peripheral driver header and source files
ThirdParty	
emWin	EmWin libraries and drivers
LCD_GDMA	GUI project of AppWizard. Please open it by *AppWizard_V154a_646a*.
ExampleCode	
GDMA_LCD_Display	
Keil	Source code of Keil project
VSCode	Source code of VS code project

图 4-1 目录信息

5. 范例程序执行

5.1 Keil 项目：

1. 根据目录信息章节进入 ExampleCode 路径中的 Keil 文件夹，双击 *GDMA_LCD_Display.uvprojx*。
2. 进入编译模式接口
   - 编译
   - 下载代码至内存
   - 进入 / 离开调试模式
3. 进入调试模式接口
   - 执行代码

5.2 VS Code 项目：

1. 开启 VS code，汇入本范例的 VSCode 文件夹。

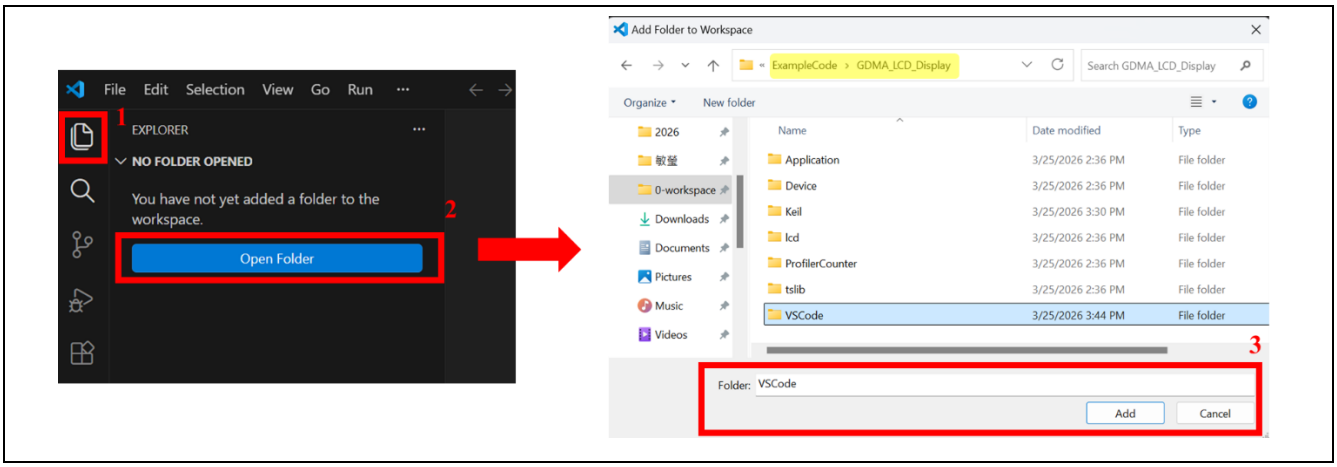


图 5-1 开启 VS code 工程文件夹

2. 点选”**CMSIS**”页面，依序执行编译(***Build solution***)与下载(***Load and run application***)按钮，完成后即可于 LCD 上看见执行结果。

1.

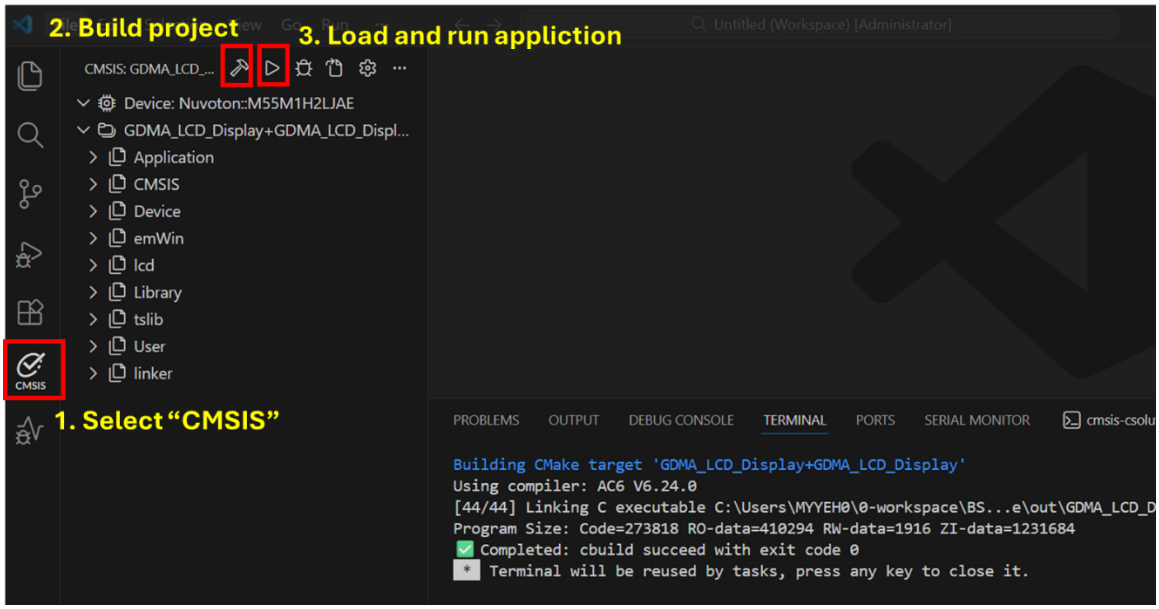


图 5-2 编译并下载程序至开发板

6. 修订纪录

Date	Revision	Description
2026.06.08	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.