M55M1 透過 GDMA 實現 LCD 顯示

NuMicro® 32 位元系列微控制器範例代碼介紹

文件資訊

應用簡述	本範例程式在 M55M1 開發平台上展示以 emWin appWizard 實現 GDMA 對 LCD 顯示畫面局部更新的功能設計
BSP版本	M55M1_Series_BSP_CMSIS_V3.01.003
開發平台	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

本範例代碼將展示如何在 NuMaker-X-M55M1 開發平台上，結合 emWin AppWizard，透過 GDMA (Graphic Direct Memory Access，直接記憶體存取) 方式實現 LCD 畫面更新。相較於傳統軟體記憶體拷貝 (Memory Copy)的方式，GDMA 的並行處理特性有效降低 CPU 運算占用率，減少 CPU 在圖像渲染過程中的負擔，釋放處理器資源以執行其他任務，確保系統在執行 GUI (Graphical User Interface)顯示的同時，仍能維持整體運行的流暢度與即時性，適合作為高效嵌入式系統的實務參考範例。

1.1 NuMaker-X-M55M1 開發平台

M55M1 系列微控制器採用 Arm® Cortex®-M55 核心，最高運作頻率達 220 MHz，內建 Arm® Helium™ DSP 技術以增強數位訊號處理能力，並整合 Arm® Ethos™-U55 微型神經網路處理器，在數位訊號處理及機器學習推論方面表現優異，同時可於邊緣裝置加速機器學習推論。

本示例使用 Numaker-TFT-LCD5 V1.2 顯示模組，其解析度為 800x480，色彩格式為 RGB565(16-bit)。



圖 1-1 NuMaker-X-M55M1 開發板



圖 1-2 NuMaker-TFT-LCD5 V1.2 顯示模組

1.2 SEGGER emWin appWizard

本範例以 SEGGER 提供的 emWin appWizard 作為 GUI 展示，其具豐富的 Widget 庫與多種 Interaction function 供使用者開發使用，能大幅降低在設計圖形化介面的所需時間，並能整合程式設計，將基本框架完成後即可透過 IDE 導出檔案並放入 Keil 專案，進行細節功能設計與測試。新唐有提供 emWin package，其包含使用 emWin 所需的所有 library、driver 及相關輔助開發工具。

使用者可於新唐官網下載最新版本的 emWin package (https://www.nuvoton.com/resource-download.jsp?tp_GUID=SW15202503130722211 9)，並透過教學手冊了解如何將 emWin appWizard 專案導入 Keil 使用。

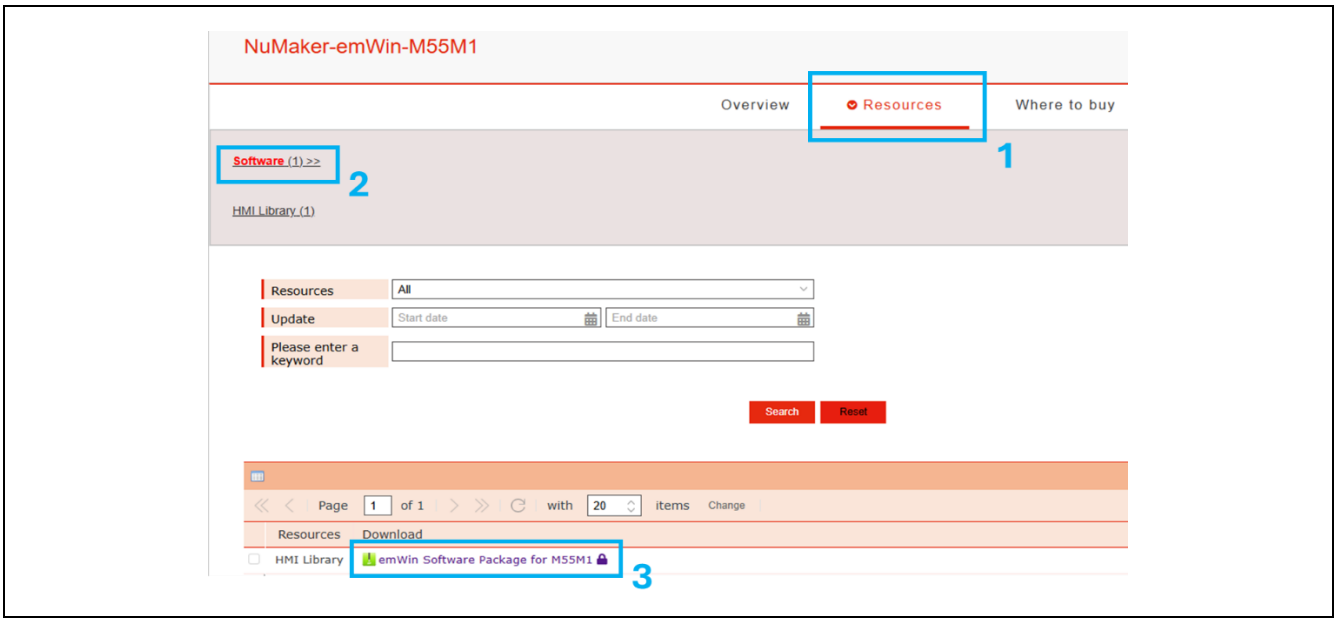


圖 1-3 官網 EmWin Package 下載(需登入)

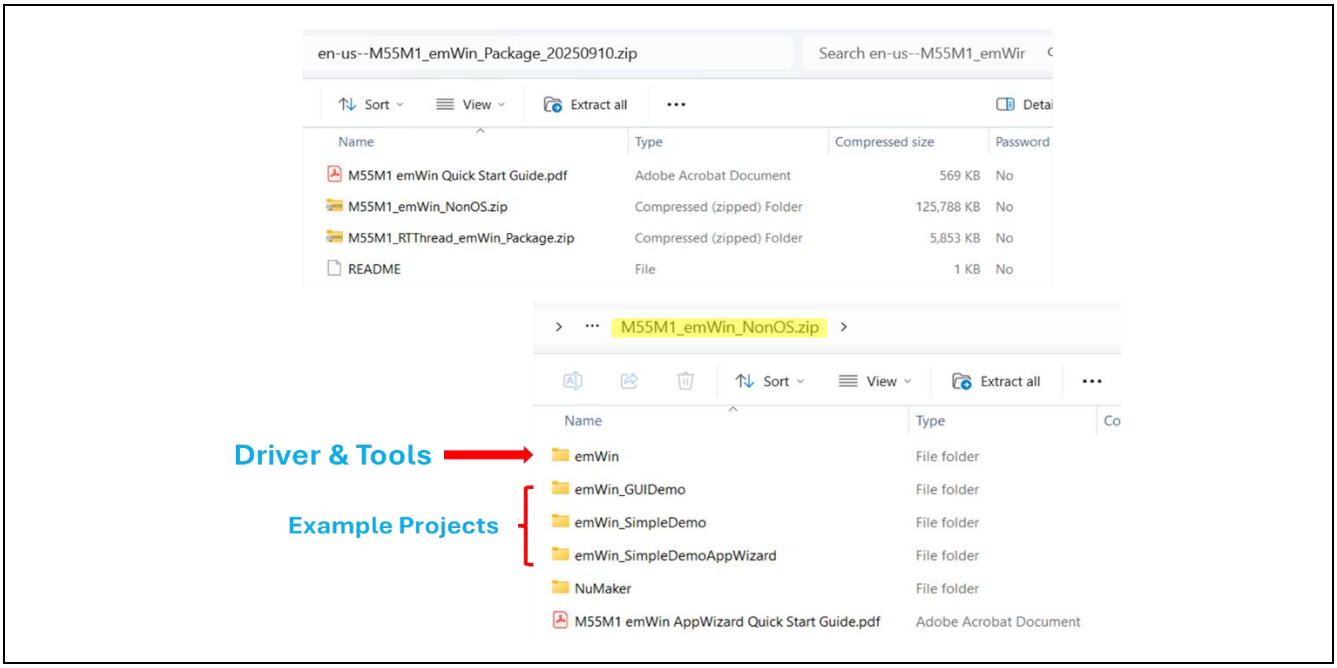


圖 1-4 EmWin Package 簡介

1.3 GDMA(Graphic Direct Memory Access) 工作原理

傳統 CPU 傳輸必須逐一搬運每個像素(byte-by-byte)，會占用大量運算資源，導致 UI 更新卡頓。而 GDMA 的工作核心在於將「資料定址」與「搬運執行」從 CPU 轉移到專用的硬體控制器上。

CPU 僅需在 GDMA 的暫存器中設定一次性的任務(透過包含起始位址、目標位址、資料長度)啟動傳輸，一旦通道被啟用，GDMA 內部就會根據設定，自主地在匯流排上發起讀取與寫入請求。這意味著資料的流動不再需要 CPU 介入每一筆 byte 的存取指令，GDMA 會自動處理地址遞增、突發傳輸以及與外設的同步，CPU 便可專注於其他更重要的資料處理任務。

如圖 1-5 所示，GDMA 在進行 局部畫面更新 (Dirty Region Update) 時，只複製畫面中實際變動的區域(Dirty Region)到 LCD VRAM，而非整個 Frame Buffer。

- pSRC：來源指標(Pointer to source)。
- pDst：目標指標(Pointer to destination)。
- Dirty Region：表示需要更新的畫面區域。
- Stride：每列像素的寬度(例：800 pixels)。
- Offset：計算起始傳輸位址，例如 pBG + (800 * y0 + x0)。
- DMA 2D Transfer：GDMA 依照 Dirty Region 的大小 (xSize × ySize) 與 Stride 自動搬移資料，跳過不需更新的像素區域(Skip Stride Gap)。
- 16-bit RGB Data：本範例使用 RGB565 色彩格式傳輸資料。

透過這種方式，GDMA 只更新實際變動的區域，大幅降低資料傳輸量與 CPU 負擔，避免 UI 卡頓，並可保持高效能畫面更新。

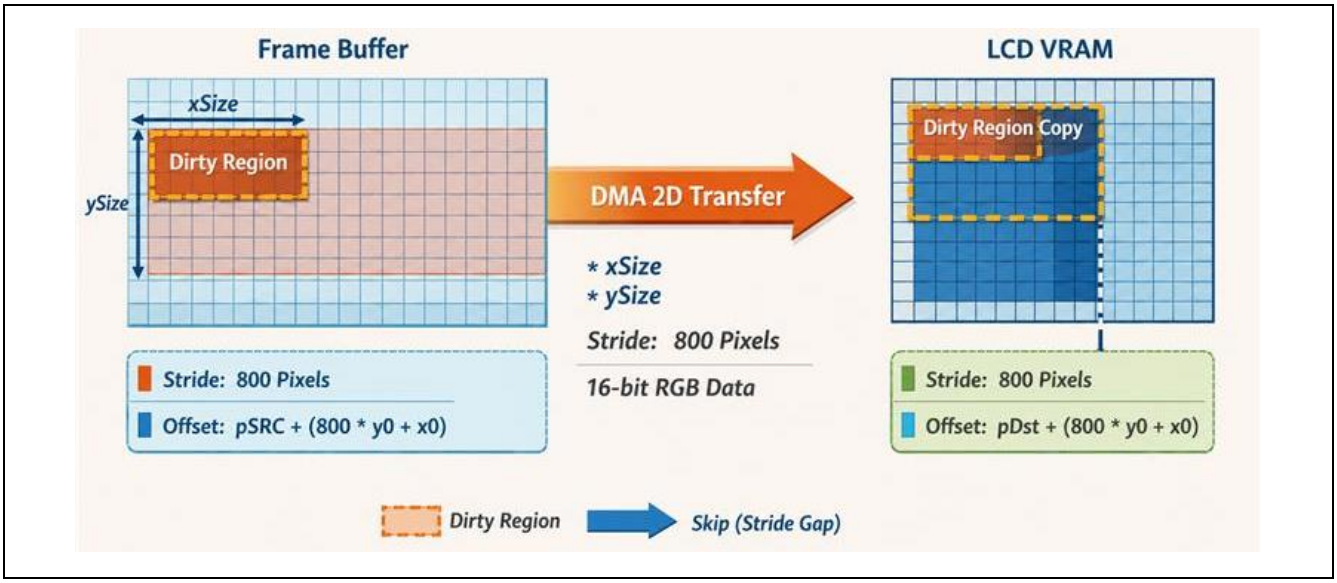


圖 1-5 GDMA 工作示意圖

1.4 系統流程與設計說明

本範例於 M55M1 開發平台上，透過 emWin AppWizard 建立圖形介面，並結合 GDMA 實現 LCD 顯示畫面之局部更新(Partial Update)，降低 CPU 負載並提升畫面更新效率。

本例同步比較 GDMA 與傳統 Memory copy 之效能差異，詳細比較與流程請參考圖 1-6 ~ 圖 1-7。

項目	CPU memory copy	GDMA copy
資料路徑	Memory → CPU → Memory	Memory → GDMA → Memory
CPU參與程度	高（逐筆處理）	低（只須設定）
效率	較慢	高速（burst）
CPU負載	高	幾乎沒有
並行能力	無	可與CPU並行

圖 1-6 Memory 與 GDMA 處理比較

上電後，會執行系統初始化及 LCD/GUI 設定相關的初始化(例如 *LCD_X_DisplayDriver()*)，接著透過 *MainTask()*執行 GUI 程式。本範例透過點擊畫面中的按鈕(Button Trigger)觸發複製資料的工作函式，完成後再以設定 *IMAGE_SetBitmap()*指定要更新顯示的物件(Image Widget)並執行該範圍的影像更新。流程圖如圖 1-7 所示。

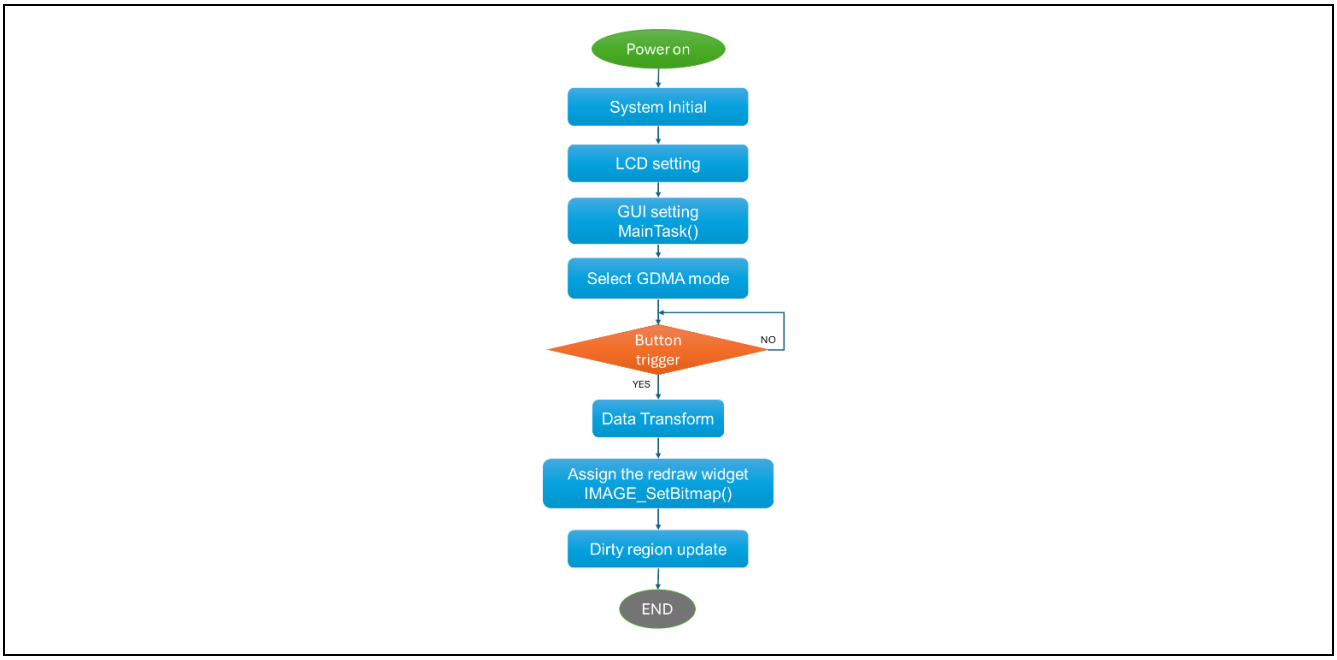


圖 1-7 系統流程示意圖

1.4.1 各階段流程說明：

- 顯示驅動呼叫(LCD_X_DisplayDriver)
- GUI 事件觸發(GUI Event Trigger)
  - 透過點擊畫面按鈕(Button Widget)開始處理資料(data copy)。
- GDMA 設定
  - Source Address(來源影像資料)
  - Destination Address(LCD Frame Buffer)
  - Transfer Size(寬 × 高 × Bytes per pixel)
  - Line Offset(Stride，若為 2D copy)
  - Transfer Mode(Memory-to-Memory)：透過 GUI 選單切換，不同模式下可支援圖片的旋轉或鏡射功能
- GDMA 執行(GDMA Execution)
- Frame Buffer 更新
  - GDMA 將指定區域的影像資料寫入 LCD Frame Buffer。
- 視窗重繪請求(emWin Rendering/ Invalidation)
  - 當事件觸發時(如資料更新)，emWin 會更新對應 Widget(如 Image Widget)，並在內部將該區域標記為無效區域(Invalid Area)。
- 重繪區域計算(Dirty Region / Invalid Area Calculation)
  - emWin 會精確計算需更新的矩形區域(Dirty Region)，系統僅針對該區域進行重繪，而非刷新整個螢幕，從而優化顯示效能並降低 CPU 負載
- LCD 顯示更新(Panel Refresh)
  - LCD 控制器持續從 Frame Buffer 讀取資料並輸出至面板，完成畫面更新。

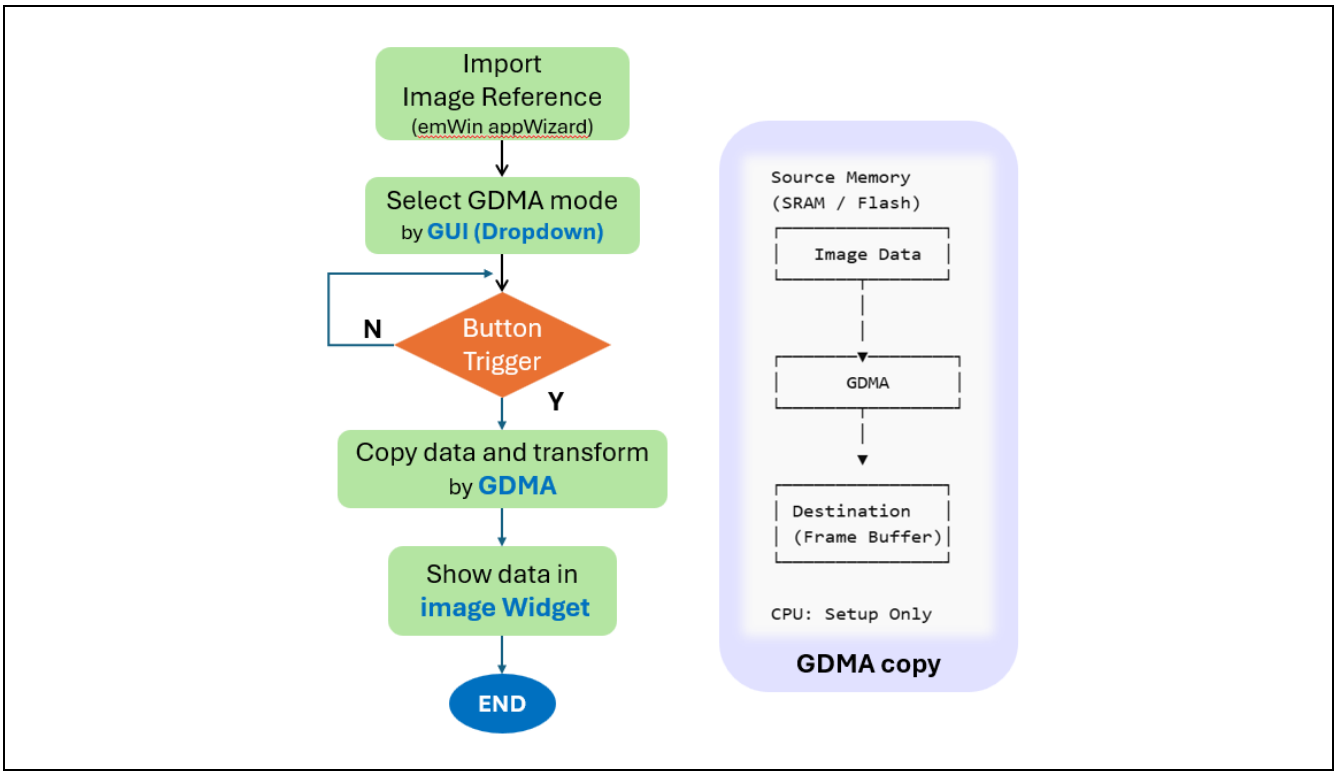


圖 1-8 程式流程

1.4.2 emWin appWizard 物件設置

本範例使用 emWin appWizard 做 GUI 開發，透過下拉式選單(Dropdown Widget)選擇需要的 GDMA 資料轉換模式，並以按鈕(Button Widget)作為資料處理的觸發媒介。預設兩個影像物件(Image Widget)分別存放目標影像及處理後影像。使用者選定處理方式後，再透過點擊按鈕開始擷取影像資料，進行畫面局部更新，將處理後影像繪製對應 Image Widget 顯示。

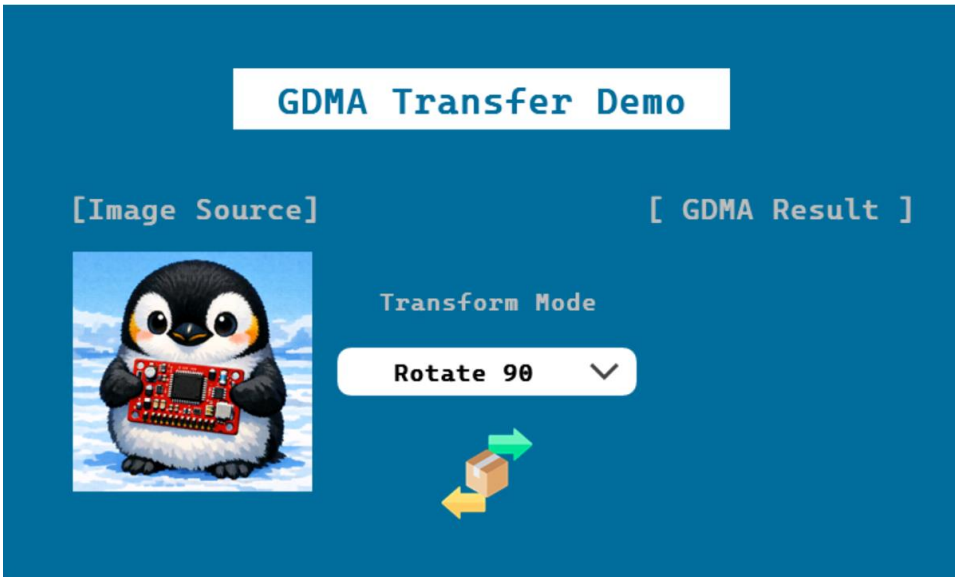


圖 1-9 GUI 預設畫面示意

GUI 物件設置說明如下表，詳細操作步驟可參考新唐 emWin package 內的教學手冊(檔案位置如圖 1-10)。

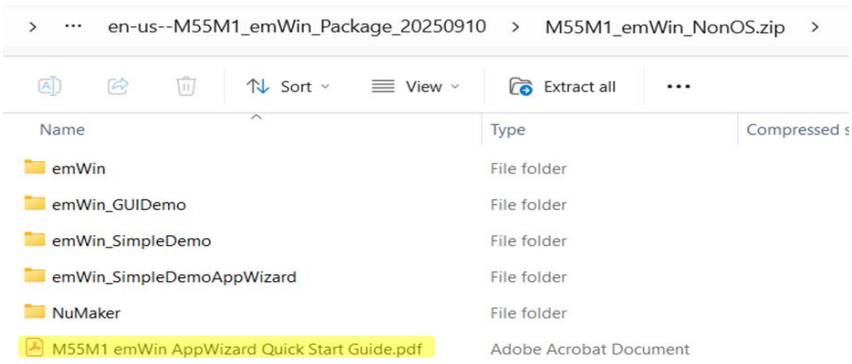


圖 1-10 EmWin AppWizard Quick Start 手冊位置

Description	Appearance	Widget Type	Widget ID
背景 800x480		Box	ID_BOX_00

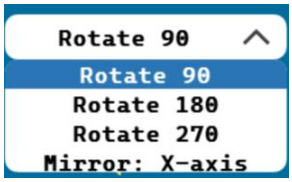
說明標示	GDMA Transfer Demo	Text	ID_TEXT_00
	[Image Source]		ID_TEXT_01
	[GDMA Result]		ID_TEXT_02
	Transform Mode		ID_TEXT_03
目標圖片 200x200		Image	ID_IMG_SRC
影像顯示 (GDMA result) 200x200	(無填色空物件)		ID_IMG_GDMA
下拉式選單 (250x40)		Dropdown	ID_DROPDOWN
操作按鈕 80x80		Button	ID_BTN_MOVE

表 1-1 GUI 物件說明

新增物件後，可於屬性欄位(Properties)調整文字(Text 物件)顯示格式，如下圖所示。

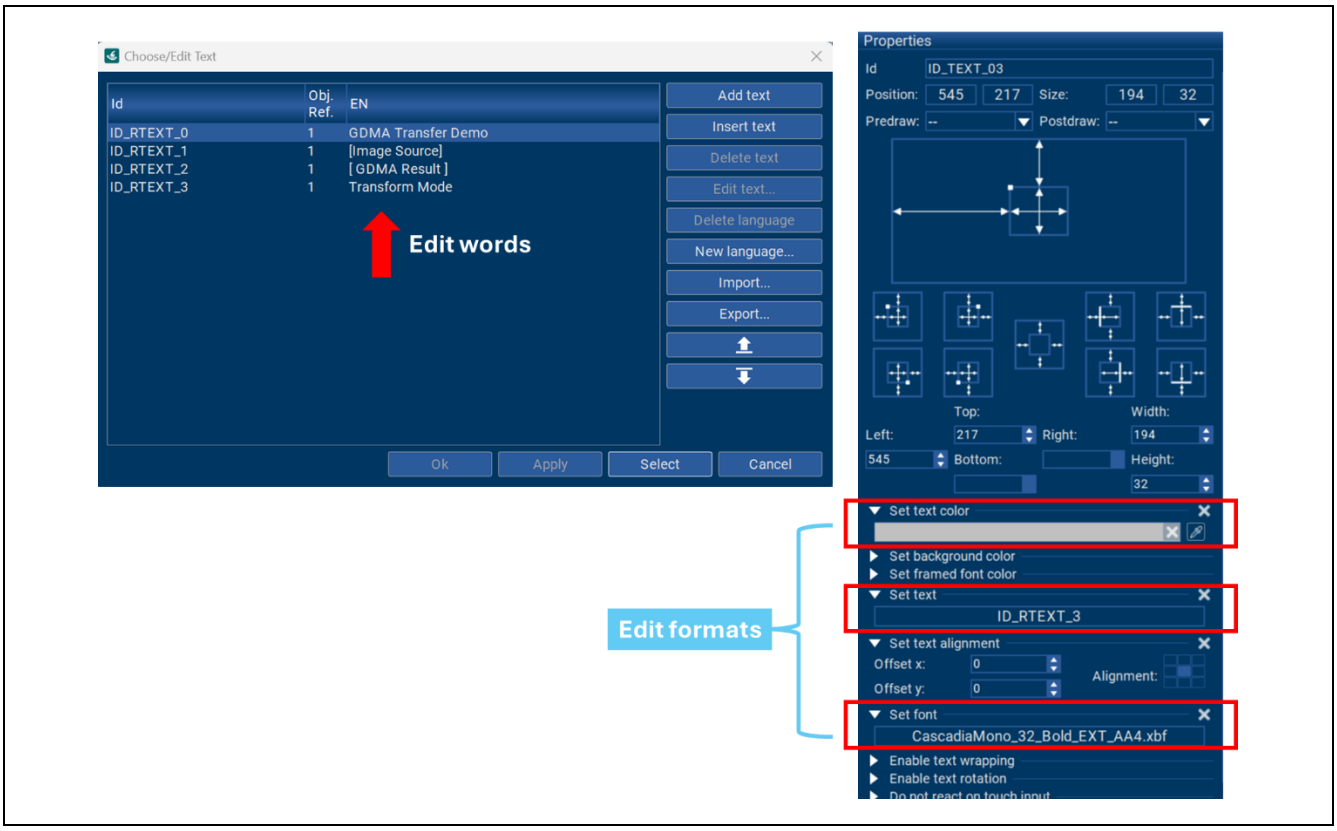


圖 1-11 Text 物件屬性欄位

Button 及 Image 物件可透過「**Set Bitmap**」欄位，設置影像作為物件外觀。其中物件 ID_IMG_SRC 將做為目標影像，故需將影像格式設為「**High color(565), RB swap**」，避免處理後的影像色彩失真。

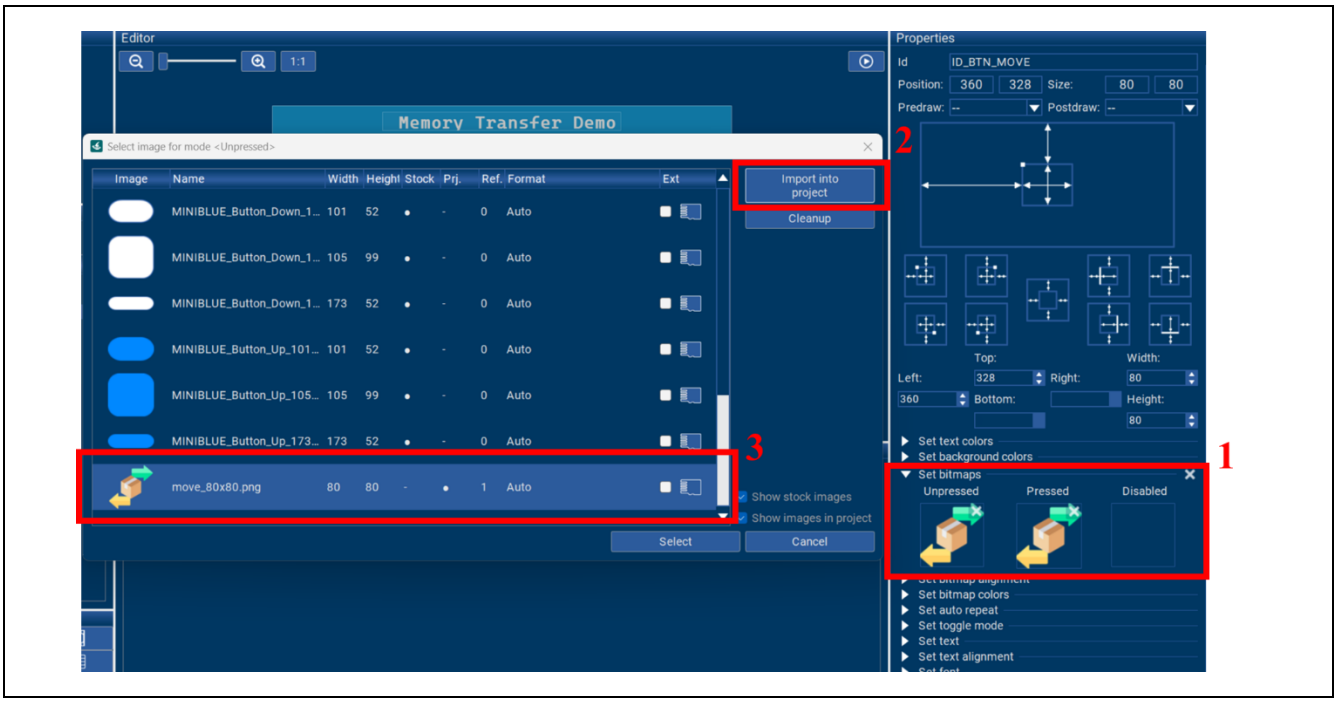


圖 1-12 Button 物件屬性設置

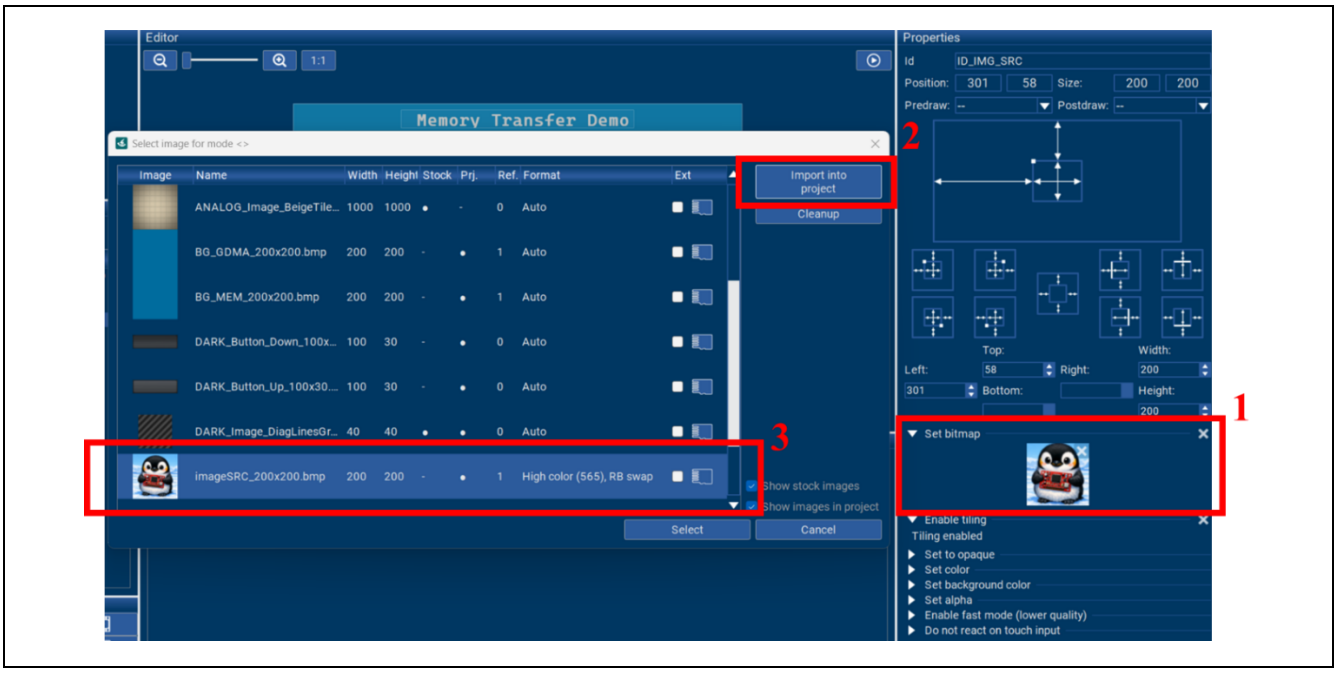


圖 1-13 Image 物件屬性設置

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，如下圖 1-14~圖 1-15 所示。

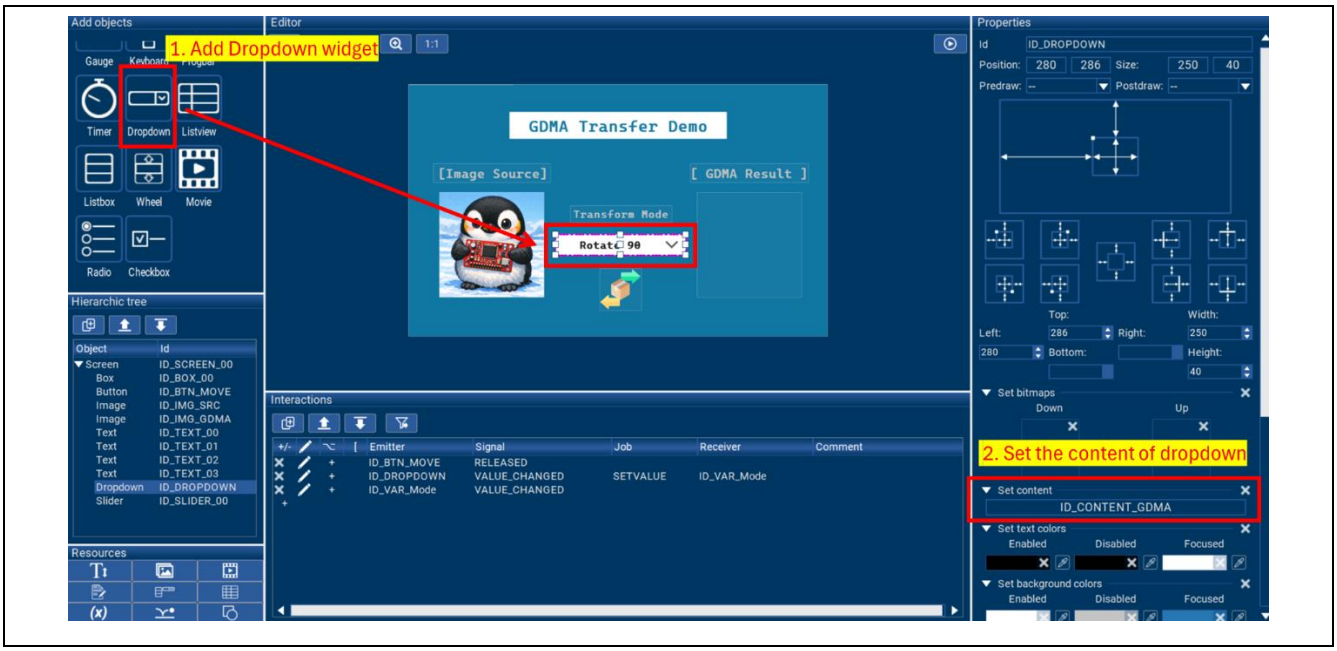


圖 1-14 Dropdown 設置

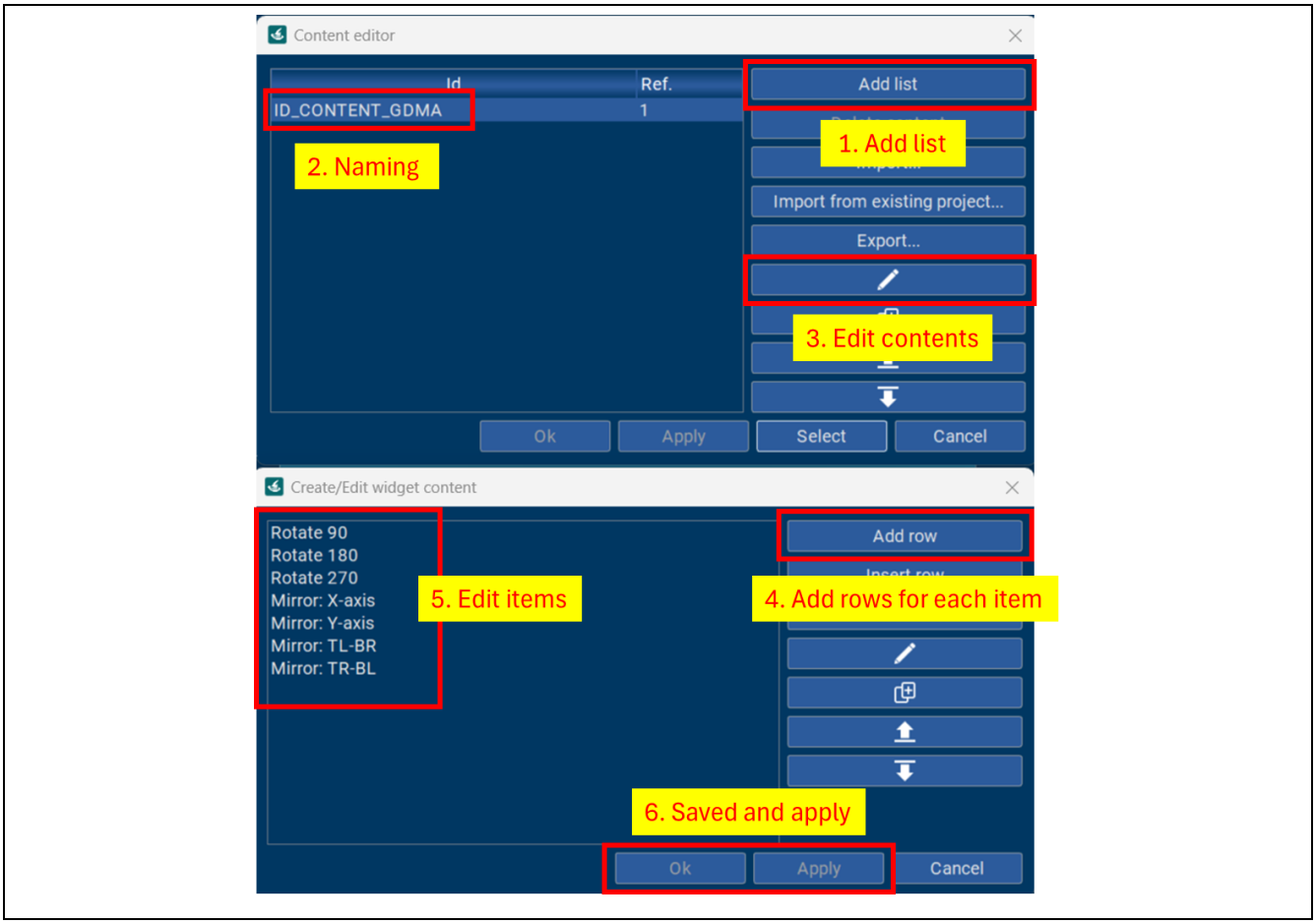


圖 1-15 Dropdown 選單內容設置

所有物件設置完成，在 Interaction 欄位新增 Button 物件的交互功能，設定項為「Emitter (ID_BTN_MOVE)」、「Signal (RELEASED)」、「Job (NULL)」及「Comment (NULL)」，以此產生命令函式 ***(ID_SCREEN_00__ID_BTN_MOVE__WM_NOTIFICATION_RELEASED)***，使用者也可於「Edit code」內新增程式(Optional)。

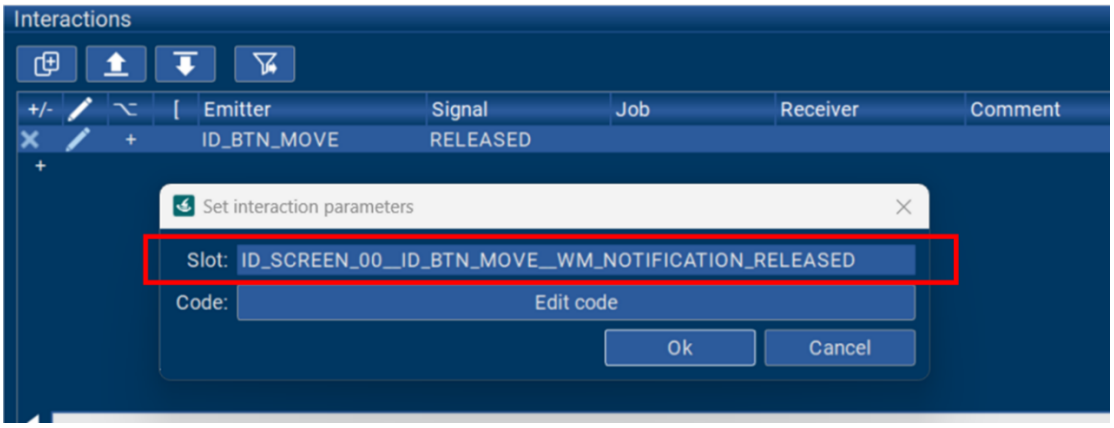


圖 1-16 Interactions 設置與生成 Slot

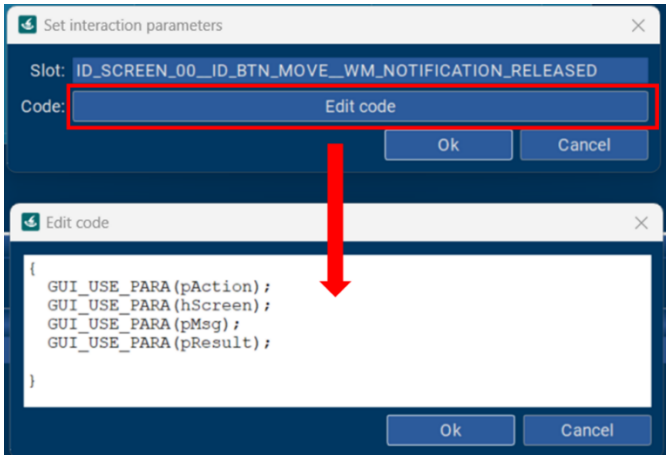


圖 1-17 Slot 程式編輯

本範例會透過選單項目改變做 GDMA 處理時的呼叫參數(*dma350_lib_transform_t*)，故新增變數(ID_VAR_Mode)記錄下拉式選單的資料。

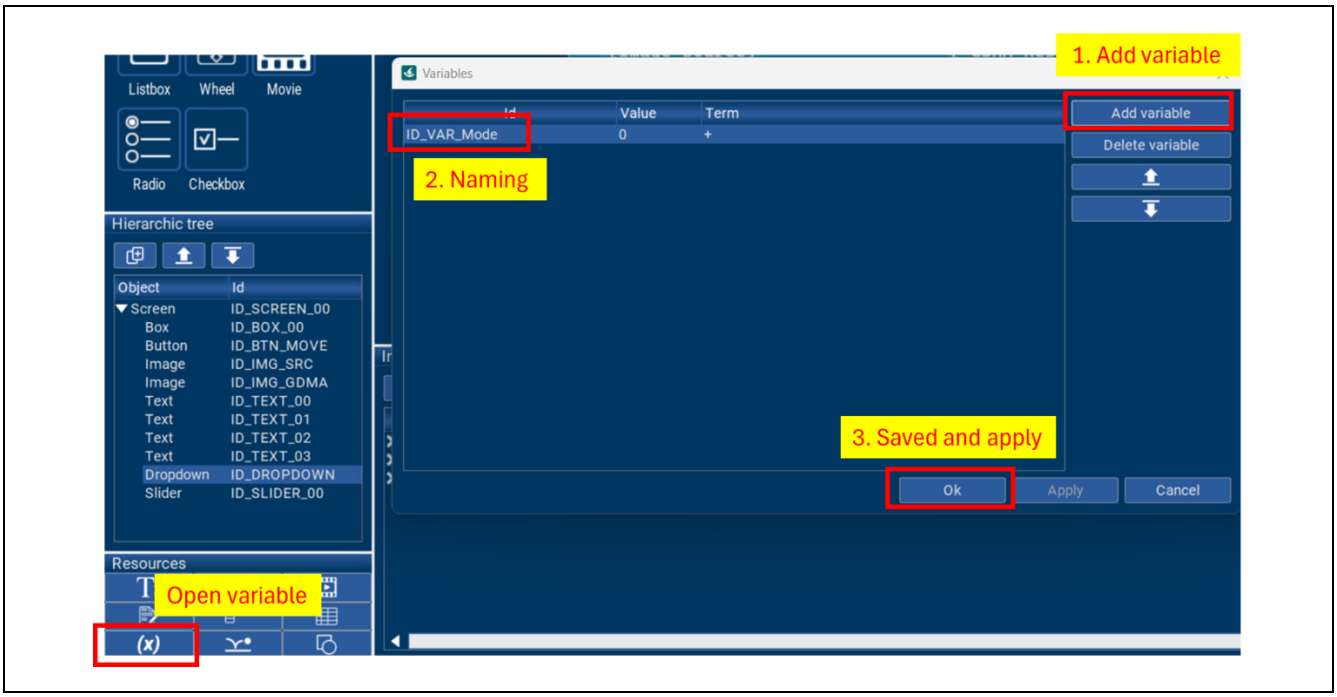


圖 1-18 新增變數

新增變數後，透過設置 dropdown 物件與變數 ID_VAR_Mode 間的交互關係(如下表所示)記錄 dropdown 更新狀態，並於"**Edit code**"欄位新增程式碼，使得程式中可透過 *APPW_GetVarDara()*函式讀取 Dropdown 物件的 index 值(起始值為 0)識別使用者所選擇的項目。

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 變數操作之 Interaction 設置

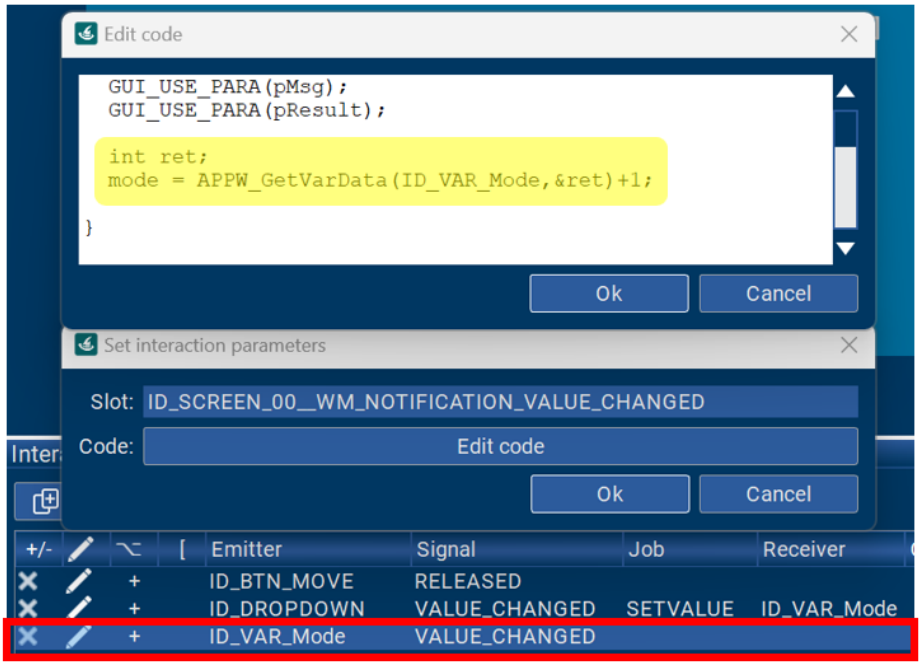


圖 1-19 設定交互關係

1.4.3 導出 emWin appWizard 專案

完成所有 GUI 設置後，將專案導出 Tool bar ***File→ Export & Save***，並將導出的 Resource 與 Source 資料夾複製並取代至 Keil 專案的 Application 資料夾內。（建議先將預設檔案刪除後，再貼上要使用的新檔案）。

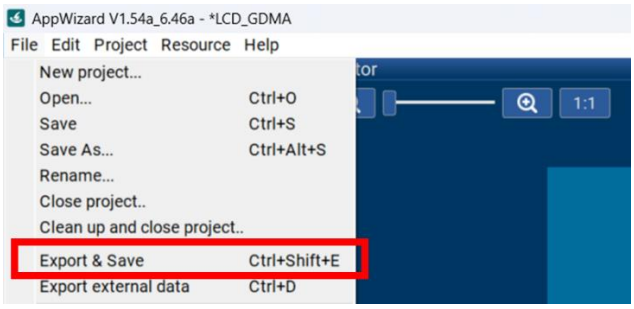


圖 1-20 導出 GUI 設置檔案

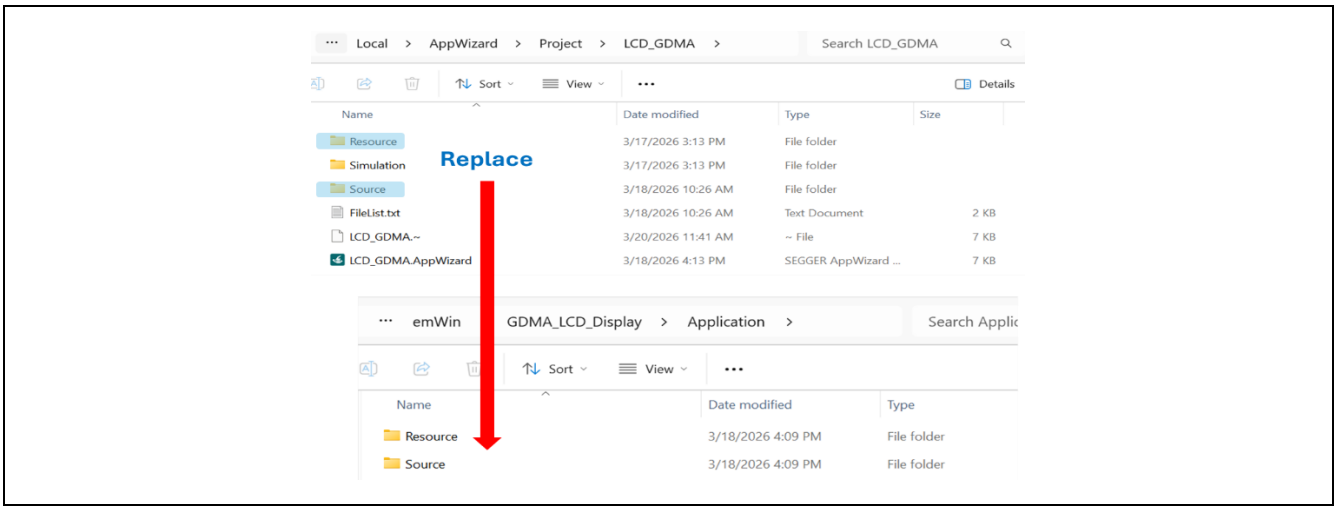


圖 1-21 將 AppWizard 檔案替換至 Keil 專案

1.4.4 將應用程式佈署至開發板

開啟 Keil 專案後，透過右鍵點選「*Add Existing Files to Group "Application"*」將 AppWizard 生成的檔案全部放入專案，並確認 **Includes Path** 欄位已新增所有檔案的路徑，最後執行編譯。

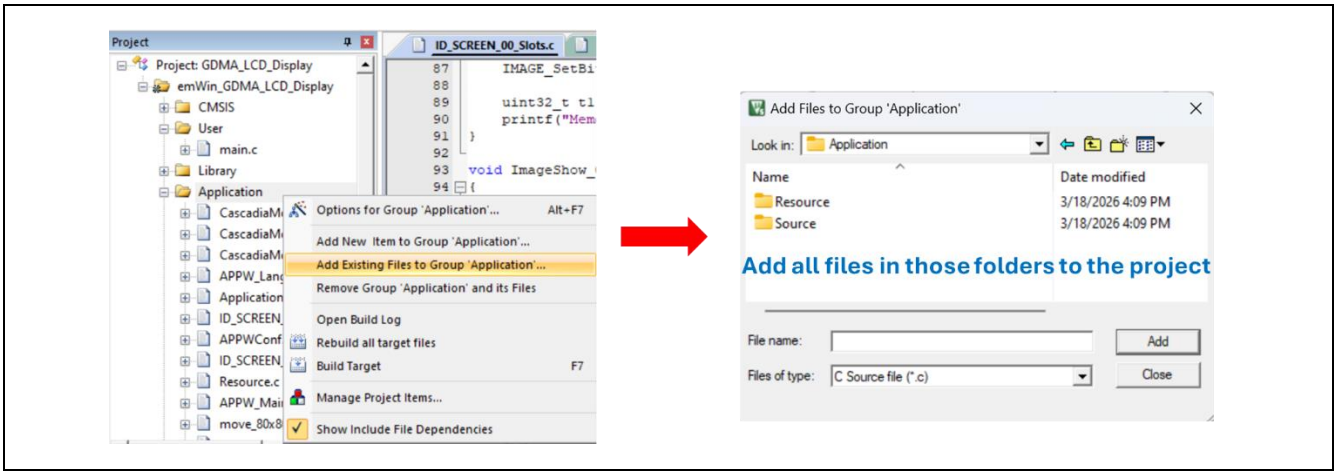


圖 1-22 將 AppWizard 檔案加入專案

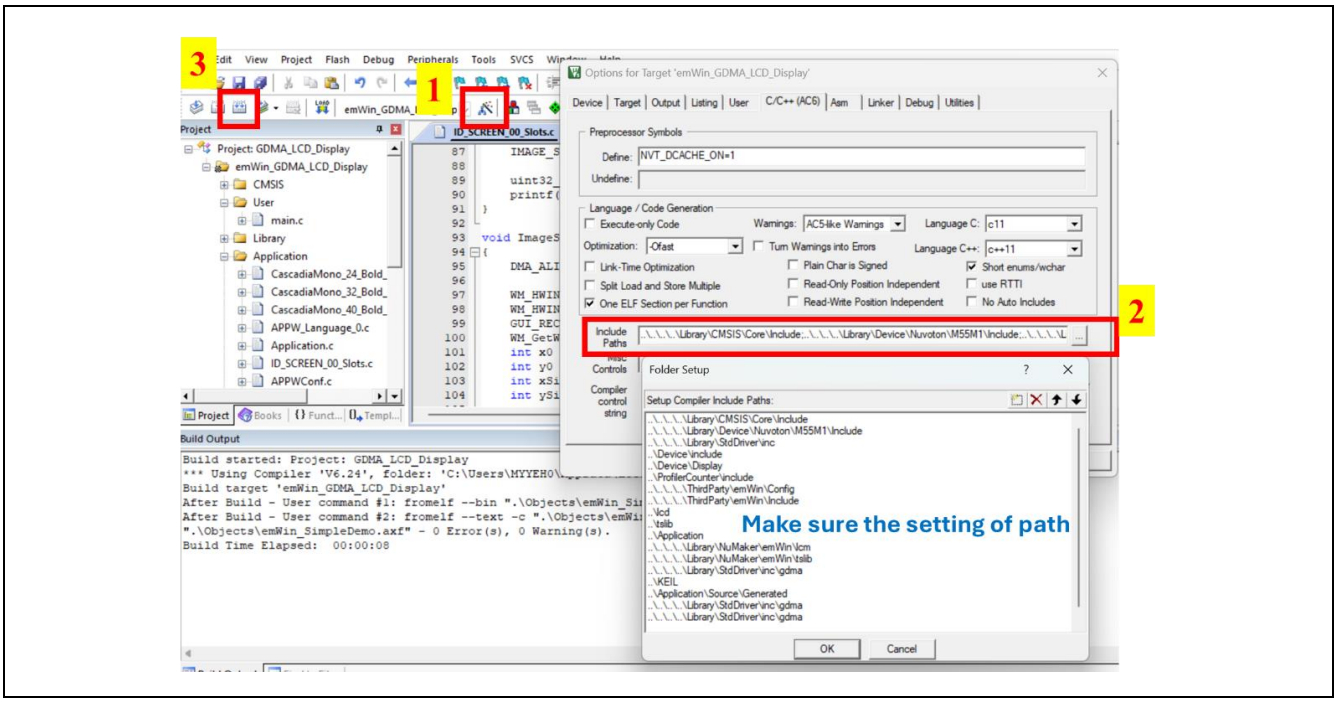


圖 1-23 檔案路徑設定與編譯

最後透過 type-C 線將電腦連接至 M55M1 開發板，上電後執行程式下載(Download)即可。

1.4.5 將 Keil 專案轉換成 VS code 專案

若使用者有 VS Code 開發需求，可將已匯入 GUI 檔案的 Keil 專案進行轉換。首先開啟 VS Code IDE，於左側功能列選取”**Explorer”**分頁，開啟 Keil 專案所在的資料夾。

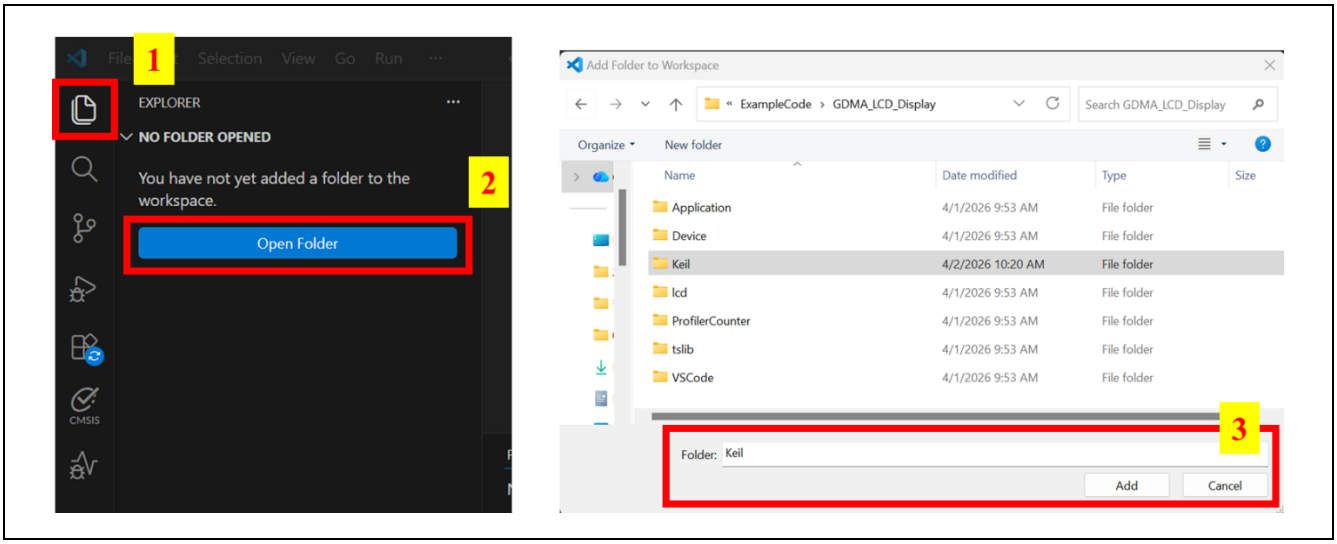


圖 1-24 在 VS Code IDE 開啟 Keil 資料夾

匯入資料夾後，選擇 Keil 專案**的*.uvprojx** 檔並點選右鍵 ”*Convert µVision project to CMSIS solution*” 開始轉換。

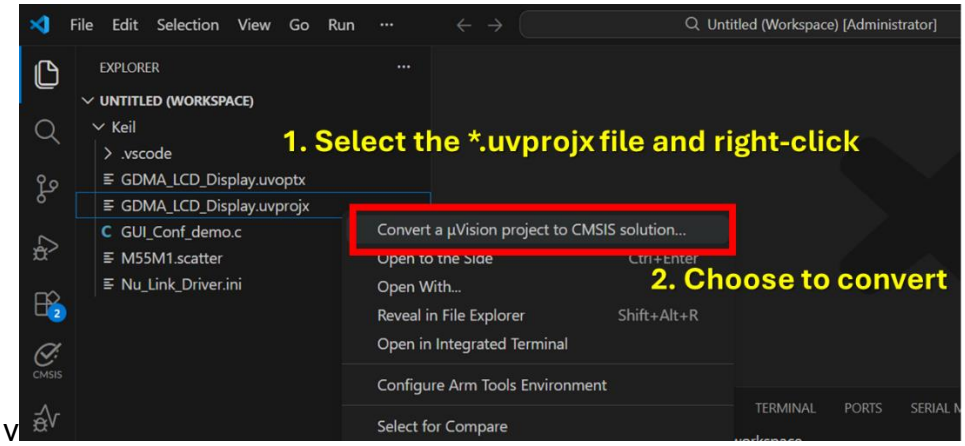


圖 1-25 透過 VS Code 內建功能進行轉換

轉換後將自動生成 *launch.json* 、*tasks.json* 、*vcpkg-configuration.json* 、**.cproject.yml* 與 **.csolution.yml* 等，開發 VS Code 專案所需的檔案。

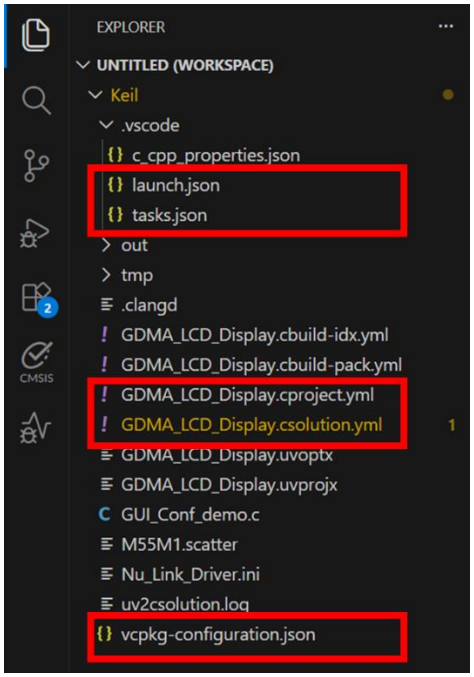


圖 1-26 轉換後的 VS Code Files

1.5 執行結果

本範例代碼執行編譯和燒錄的結果如圖 1-27~圖 1-28 所示，點選畫面中間的按鈕即透過 GDMA 複製目標影像(Image Source)，並依使用者選擇的方式將處理結果繪至影像物件中。

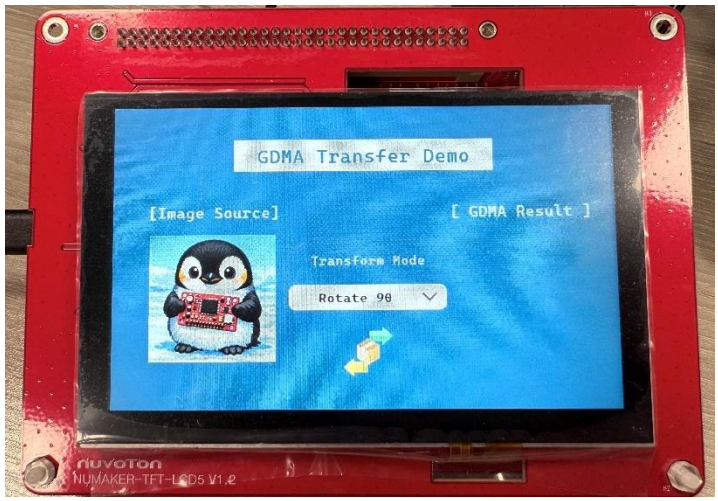
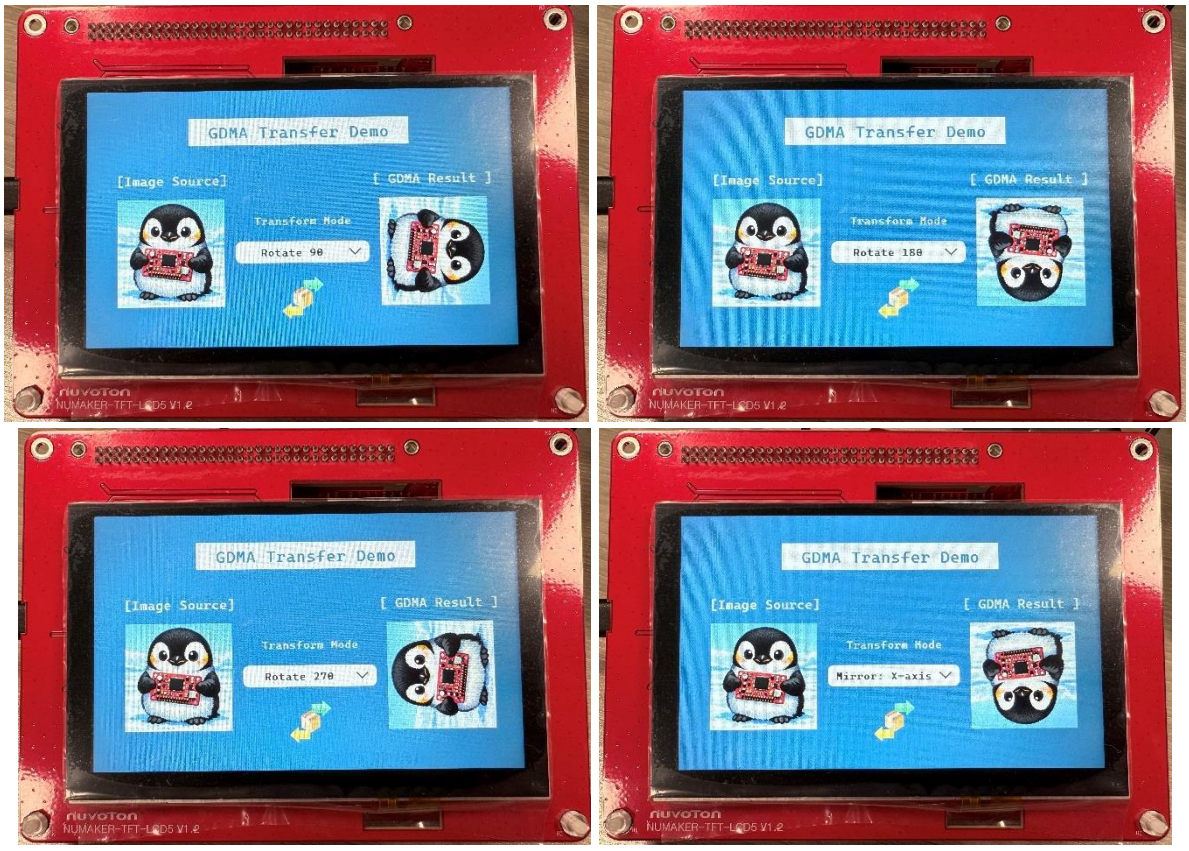


圖 1-27 程式執行結果 (GUI 畫面)




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

圖 1-28 操作結果

2. 代碼介紹

本範例透過 emWin 設計 GUI 並將檔案匯入 Keil 專案中。其中 *main.c* 會對程式進行初始化，設定觸控螢幕、frame buffer 與 Multiple Buffering 等，接著導向應用主程式 *MainTask()*開始執行專案。下列程式分別位於 *main.c* 與 *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);
  }
}
```

透過 emWin appWizard 製作的物件，將自動定義所有物件在程式中的名稱與資訊，並為每個畫面(Screen Widget)生成對應程式檔供編輯使用，其中物件相關定義將位於 *ID_SCREEN_00.c*，可於此查詢各物件在程式中的資料定義名稱，例如此範例中 ID_IMG_SRC 內容被定義為 ***acimageSRC_200x200***(名稱會因 emWin appWizard 匯入圖檔檔名變動)。

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 中所使用的圖片會在匯出檔案時，自動生成 C 檔，例如本例的 *imageSRC_200x200.c*。而為避免資料不對齊導致 GDMA 處理資料時發生影像偏移現象，故需在此新增對齊資料的定義 ***#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，本範例即使用函式 ***dma350_draw_from_canvas()***執行資料搬移，使用者可於程式頂端透過定義 *include* ***"dma350_lib.h"***及***#include "gdma.h"***使用，並可於 *dma350_lib.h* 內參閱其他相關 library 的使用說明。

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);
```

本範例相關操作皆位於 *ID_SCREEN_00_slots.c*，emWin appWizard 設置的 interaction 觸發函式也定義於此(*ID_SCREEN_00__ID_BTN_MOVE__WM_NOTIFICATION_RELEASED*)，使用者可於此新增欲執行的程式及相關定義宣告。若想將資料匯入形成 Widget 外觀，需定義格式為 GUI_BITMAP 的資料緩衝區，格式定義位於 *GUI_Type.h* 中。本例中宣告型態為 unsigned short 的 buffer 並指定給特定 Image 物件，透過更新 buffer 資料更新影像內容。

注意：每次更新時，需以 *IMAGE_SetBitmap()*做畫面刷新。

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. 軟體與硬體需求

3.1 軟體需求

- 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 硬體需求

- 電路元件
  - NuMaker-X-M55M1
- 線路示意圖
  - 透過 Type-C USB 連接 NuMaker-X-M55M1 與 PC。

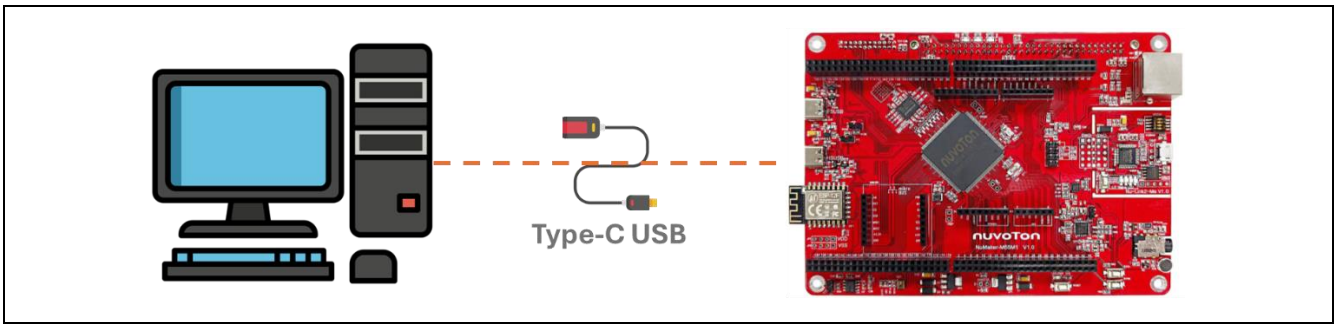


圖 3-1 M55M1 接線示意圖

4. 目錄資訊

目錄結構如下圖所示。

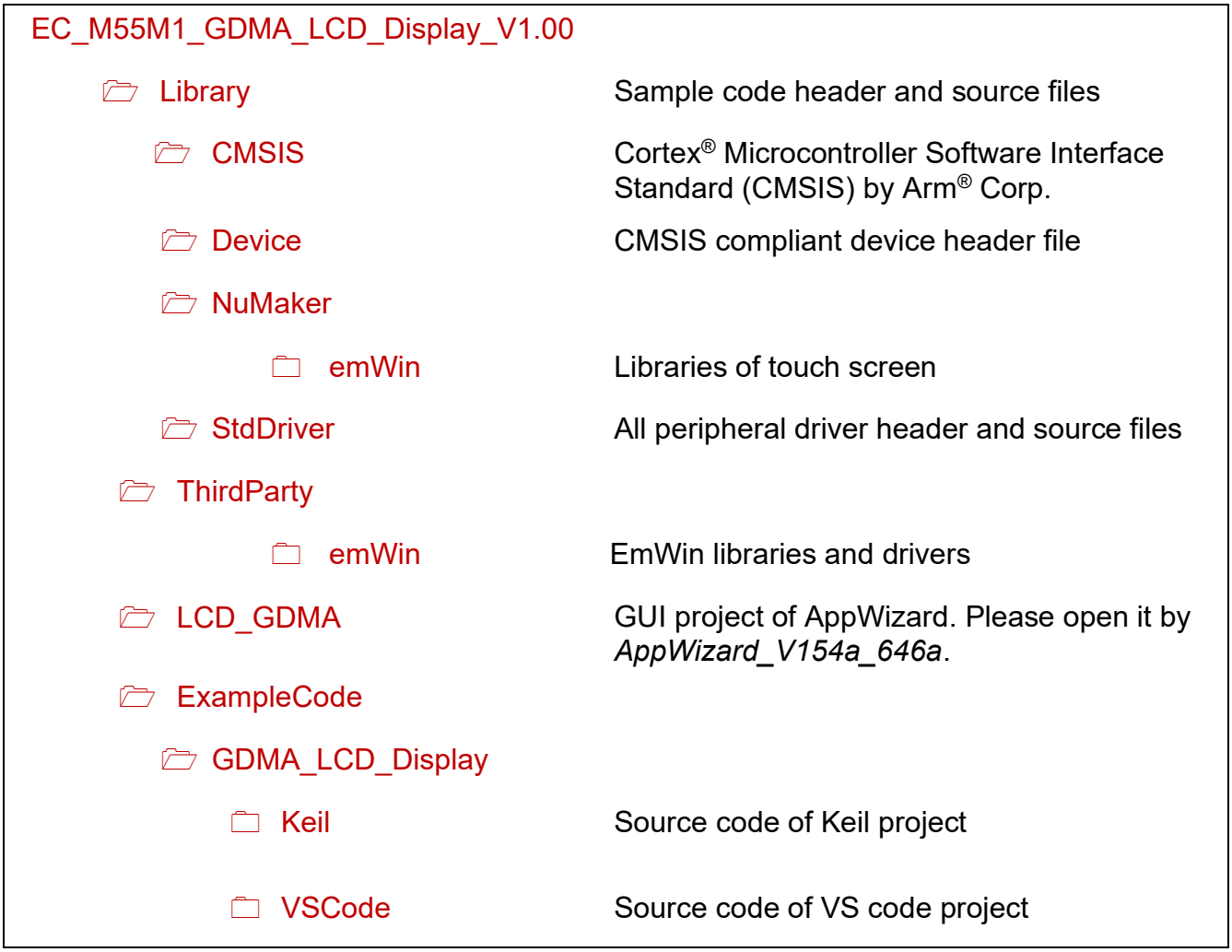


圖 4-1 目錄資訊

5. 範例程式執行

5.1 Keil 專案：

1. 根據目錄資訊章節進入 ExampleCode 路徑中的 Keil 資料夾，雙擊 *GDMA_LCD_Display.uvprojx*。
2. 進入編譯模式介面
   - 編譯
   - 下載代碼至記憶體
   - 進入 / 離開除錯模式
3. 進入除錯模式介面
   - 執行代碼

5.2 VS Code 專案：

1. 開啟 VS code，匯入本範例的 VSCode 資料夾。

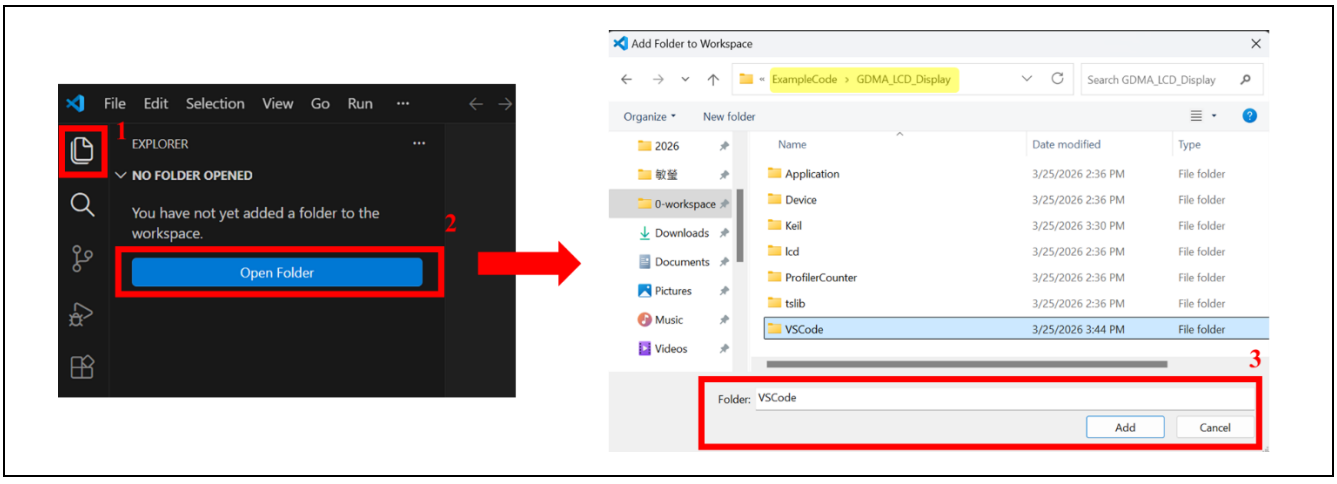


圖 5-1 開啟 VS code 專案資料夾

2. 點選***"CMSIS"***頁面，依序執行編譯(***Build solution***)與下載(***Load and run application***)按鈕，完成後即可於 LCD 上看見執行結果。

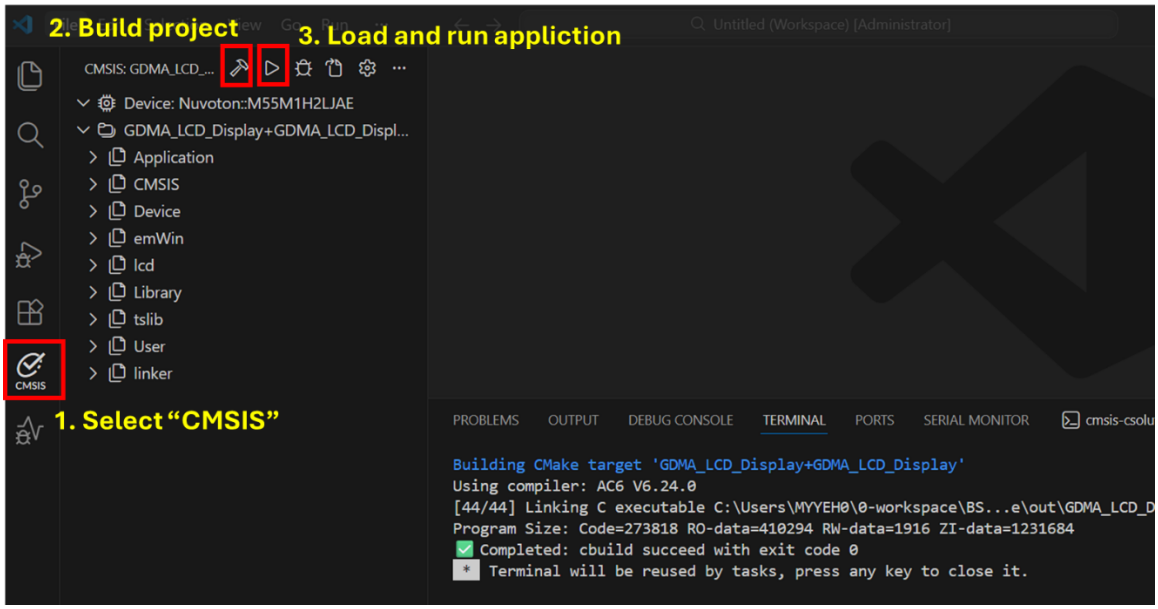


圖 5-2 編譯並下載程式至開發板

6. 修訂紀錄

Date	Revision	Description
2026.06.08	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.