M55M1 實現軟體同步信號 LCD 控制器

NuMicro® 32 位元系列微控制器範例代碼介紹

文件資訊

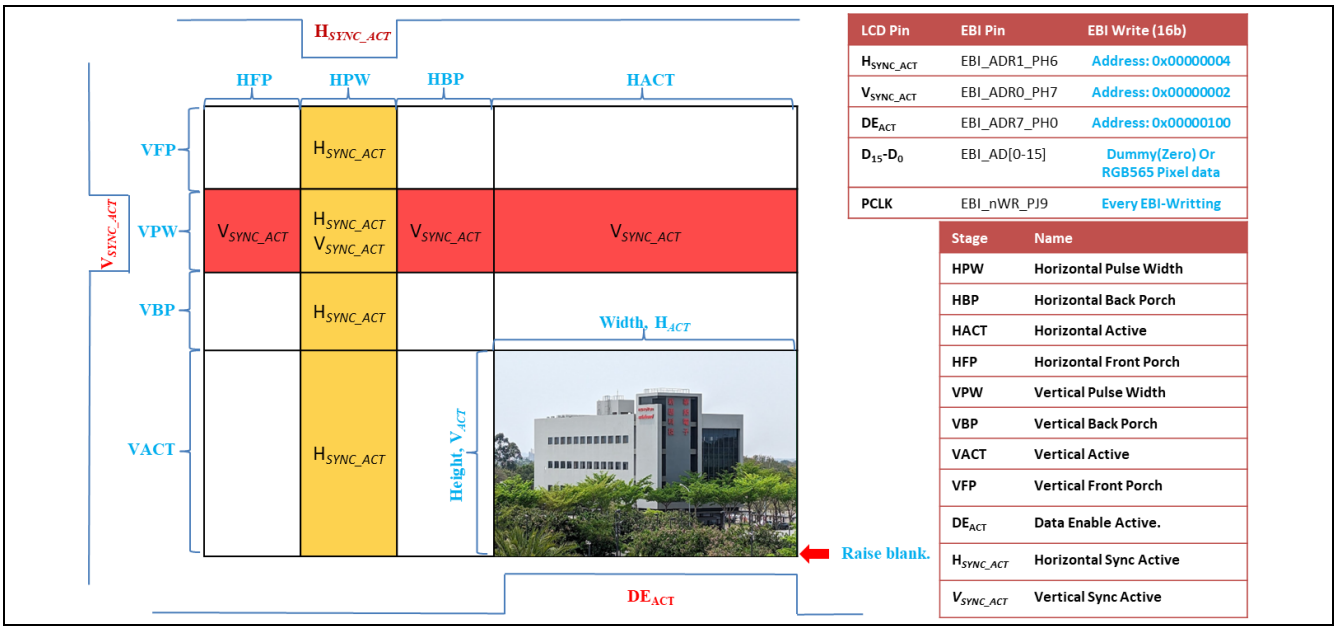
應用簡述	本範例代碼使用 M55M1 EBI 驅動同步信號 LCD 屏幕
BSP 版本	M55M1_Series_BSP_CMSIS_V3.01.001
開發平台	NuMaker-M55M1 V1.0

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

www.nuvoton.com

1. 概述

一般常用於微控制器的 LCD 屏幕介面為 MPU-type LCD，由於這類屏幕內建控制器與顯示記憶體，以致於價格高於同步信號 LCD 屏幕 (可參閱表 1-1 MPU-type 與 Sync-type LCD 屏幕比較表)。本範例代碼使用 M55M1 EBI 驅動同步信號(Sync-Type) LCD 屏幕，透過 EBI 介面並搭配 M55M1 系列支援的 PDMA 或是 GDMA 實現自我刷新畫面功能。



LCD Pin	EBI Pin	EBI Write (16b)
H_{SYNC_ACT}	EBI_ADR1_PH6	Address: 0x00000004
V_{SYNC_ACT}	EBI_ADR0_PH7	Address: 0x00000002
DE_{ACT}	EBI_ADR7_PH0	Address: 0x00000100
D_{15}-D_0	EBI_AD[0-15]	Dummy(Zero) Or RGB565 Pixel data
PCLK	EBI_nWR_PJ9	Every EBI-Writting

Stage	Name
HPW	Horizontal Pulse Width
HBP	Horizontal Back Porch
HACT	Horizontal Active
HFP	Horizontal Front Porch
VPW	Vertical Pulse Width
VBP	Vertical Back Porch
VACT	Vertical Active
VFP	Vertical Front Porch
DE_{ACT}	Data Enable Active.
H_{SYNC_ACT}	Horizontal Sync Active
V_{SYNC_ACT}	Vertical Sync Active

圖 1-1 EBI 驅動同步信號 LCD 屏幕

	MPU-Type LCD (Memory Processor Unit)	Sync-Type LCD (RGB or SYNC Interface)
傳輸方式	EBI i80 介面或 SPI 進行指令操作和影像資料的傳輸	RGB 與同步信號來顯示影像
內存與頻寬使用量	低	高
適用場合	適合靜態畫面或小量變動畫面	適合較高解析度、動態畫面
硬體成本	自帶 LCD 控制器，由屏幕輸出 RGB 資料與同步訊號	無 LCD 控制器，Host 端直接輸出 RGB 資料與同步訊號至屏幕

表 1-1 MPU-type 與 Sync-type LCD 屏幕比較表

1.1 原理

本範例代碼使用 EBI 模擬同步信號 LCD 控制器，透過 PDMA 或是 GDMA 進行 Memory-to-memory 傳輸功能，將 RGB565 像素資料透過 EBI 介面傳送至同步信號 LCD 屏幕，其運作原理為屏幕畫面建立 V Lines * H Stages DMA 描述符，其中：

- V 代表 LCD 時序的 Vertical Pulse Width(VPW)、Vertical Back Porch(VBP)、Vertical Active(VA) 和 Vertical Front Porch(VFP) 的掃描線數量。
- H 代表 Horizontal Pulse Width(HPW)、Horizontal Back Porch(HBP)、Horizontal Active(HA)和 Horizontal Front Porch(HFP)的這四個階段。

如圖 所示，這些 V Lines × H Stages 的 DMA 描述符依次鏈接，將最後一個描述符的 **NEXT LINK** 設置為第一個描述符的地址，從而形成一個環狀描述符集，並將最後一個描述符啟用傳輸完成中斷，並在中斷服務程序（ISR）中更新 HACT 階段的描述符資料傳輸起始地址，以切換至新圖像的每條 LINE 記憶體起始地址，從而實現畫面動態切換，提供防止畫面撕裂（Anti-tearing）功能。

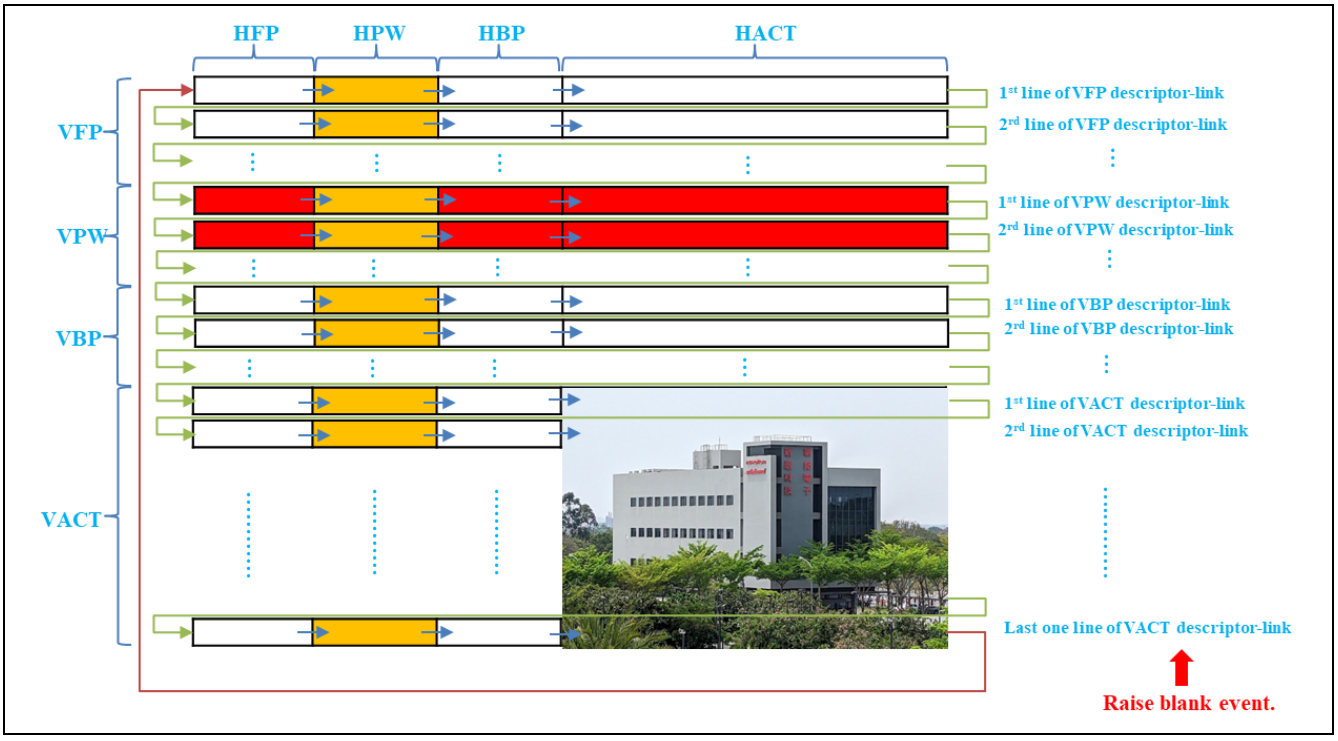


圖 1-2 V Lines * H Stages DMA 描述符

V \ H	HFP	HPW	HBP	HACT
VFP	Set H_{SYNC_ACT} to Inactive Set V_{SYNC_ACT} to Inactive Set DE_{ACT} to Inactive Set $D_{[15-0]}$ to Dummy	Set H_{SYNC_ACT} to Active Set V_{SYNC_ACT} to Inactive Set DE_{ACT} to Inactive Set $D_{[15-0]}$ to Dummy	Set H_{SYNC_ACT} to Inactive Set V_{SYNC_ACT} to Inactive Set DE_{ACT} to Inactive Set $D_{[15-0]}$ to Dummy	Set H_{SYNC_ACT} to Inactive Set V_{SYNC_ACT} to Inactive Set DE_{ACT} to Inactive Set $D_{[15-0]}$ to Dummy
VPW	Set H_{SYNC_ACT} to Inactive Set V_{SYNC_ACT} to Active Set DE_{ACT} to Inactive Set $D_{[15-0]}$ to Dummy	Set H_{SYNC_ACT} to Active Set V_{SYNC_ACT} to Active Set DE_{ACT} to Inactive Set $D_{[15-0]}$ to Dummy	Set H_{SYNC_ACT} to Inactive Set V_{SYNC_ACT} to Active Set DE_{ACT} to Inactive Set $D_{[15-0]}$ to Dummy	Set H_{SYNC_ACT} to Inactive Set V_{SYNC_ACT} to Active Set DE_{ACT} to Inactive Set $D_{[15-0]}$ to Dummy
VBP	Set H_{SYNC_ACT} to Inactive Set V_{SYNC_ACT} to Inactive Set DE_{ACT} to Inactive Set $D_{[15-0]}$ to Dummy	Set H_{SYNC_ACT} to Active Set V_{SYNC_ACT} to Inactive Set DE_{ACT} to Inactive Set $D_{[15-0]}$ to Dummy	Set H_{SYNC_ACT} to Inactive Set V_{SYNC_ACT} to Inactive Set DE_{ACT} to Inactive Set $D_{[15-0]}$ to Dummy	Set H_{SYNC_ACT} to Inactive Set V_{SYNC_ACT} to Inactive Set DE_{ACT} to Inactive Set $D_{[15-0]}$ to Dummy
VACT	Set H_{SYNC_ACT} to Inactive Set V_{SYNC_ACT} to Inactive Set DE_{ACT} to Inactive Set $D_{[15-0]}$ to Dummy	Set H_{SYNC_ACT} to Active Set V_{SYNC_ACT} to Inactive Set DE_{ACT} to Inactive Set $D_{[15-0]}$ to Dummy	Set H_{SYNC_ACT} to Inactive Set V_{SYNC_ACT} to Inactive Set DE_{ACT} to Inactive Set $D_{[15-0]}$ to Dummy	Set H_{SYNC_ACT} to Inactive Set V_{SYNC_ACT} to Inactive Set DE_{ACT} to Active Set $D_{[15-0]}$ to pixel data in every Line Raising in latest descriptor.

表 1-2 各 LCD 時序階段與 I/O 腳位對應表

如圖 1-3 EBI16-to-RGB24 各訊號腳位連接圖所示，LCD 時序的 **H_{SYNC_ACT}**、**V_{SYNC_ACT}**和 **DE_{ACT}** 三個同步訊號腳位分別透過 EBI 的 ADR1、ADR0 和 ADR7 位址腳位來模擬，透過指定不同位址解碼以產生 LCD 同步訊號、PCLK 訊號連接至 EBI-nWR 腳位和 EBI 的 16 位元資料腳位 $D_{[15-0]}$對應到 LCD 顏色信號：

- $R_{[7-3]}$(紅色)，$G_{[7-2]}$(綠色) 和 $B_{[7-3]}$(藍色)。
- 額外的 $R_{[2-0]}$、$G_{[1-0]}$ 和 $B_{[2-0]}$可以使用 $R_{[7-5]}$、$G_{[7-6]}$ 和 $B_{[7-5]}$的資料進行 RGB888 補色，以實現更高的顏色精度。

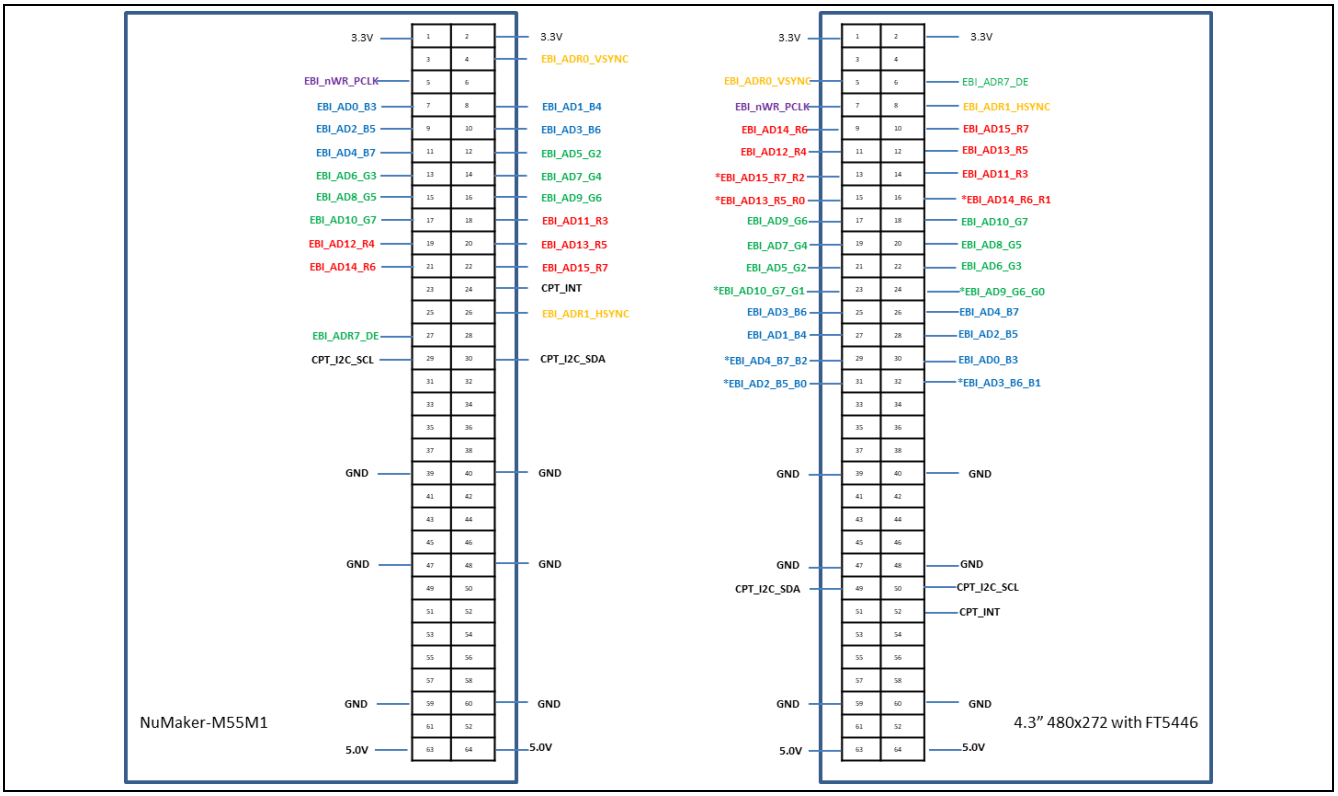


圖 1-3 EBI16-to-RGB24 各訊號腳位連接圖

本範例代碼需另接外部硬體模組，包含NuMaker-M55M1開發板、4.3” WQVGA 同步信號屏幕子板與 EBI16–RGB24 轉接板，如圖1-4 所示。

TBD

圖 1-4 外部硬體模組

1.2 執行結果

本範例代碼提供兩種目標編譯選項：WQVGA_GDMA 和 WQVGA_PDMA，分別對應使用 GDMA 或 PDMA 控制器進行自我刷新畫面功能，如圖 1-5 選擇編譯選項與執行編譯所示。

執行時，可透過連接 NULINK2ME 的 VCOM 功能，搭配電腦端終端機軟體進行觀測，預設通訊參數為 115200N81。如圖 1-6 執行畫面所示，終端機將顯示執行結果；使用者亦可直接從連接的屏幕上觀察圖像切換與更新效果。

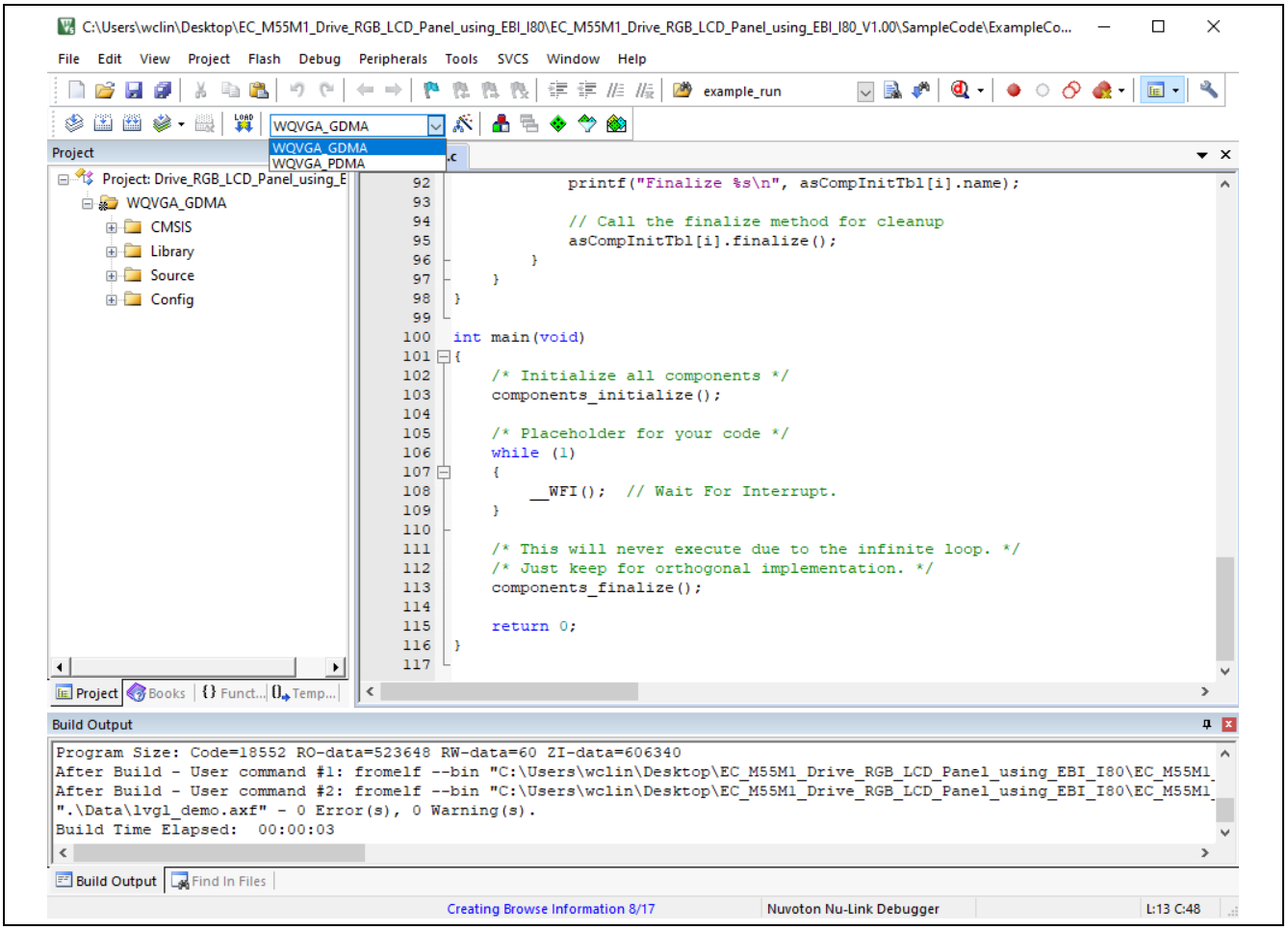


圖 1-5 選擇編譯選項與執行編譯

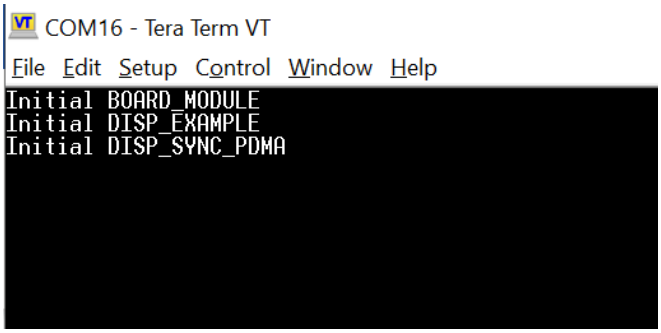


圖 1-6 執行畫面

2. 代碼介紹

本範例代碼執行各元件初始化與結束流程，程式碼位於 *main.c*。各組件初始化與結束函式位於 *disp_sync_gdma.c*，*disp_sync_pdma.c*，*board.c* 和 *disp_example.c* 實作內。各組件實作使用 *COMPONENT_EXPORT* 巨集註冊組件名稱(*.name*)、組件初始化回調函式(*.initialize*) 和組件結束回調函式(*.finalize*)，並將這些組件註冊資料結收錄至 *CompInitTab* 區域，供主程式呼叫執行。

```
/**
  * @brief Initializes the components.
  * @param[in]  None
  * @return     None
  */
static void components_initialize(void)
{
    int i;
    component_export_t asCompInitTbl = (component_export_t)&CompInitTab$$Base;
    uint32_t u32CompInitNum = (component_export_t)&CompInitTab$$Limit - asCompInitTbl;

    for (i = 0; i < u32CompInitNum; i++)
    {
        if (asCompInitTbl[i].initialize)
        {
            printf("Initial %s\n", asCompInitTbl[i].name);
            asCompInitTbl[i].initialize();
        }
    }
}

/**
  * @brief      Finalizes the components.
  * @param[in]  None
  * @return     None
  */
static void components_finalize(void)
{
    int i;
    component_export_t asCompInitTbl = (component_export_t)&CompInitTab$$Base;
    uint32_t u32CompInitNum = (component_export_t)&CompInitTab$$Limit - asCompInitTbl;

    for (i = 0; i < u32CompInitNum; i++)
    {
        if (asCompInitTbl[i].finalize)
        {
            printf("Finalize %s\n", asCompInitTbl[i].name);
            asCompInitTbl[i].finalize();
        }
    }
}

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

```
    /* Just keep here, or put your code here. */
    while (1)
    {
        __WFI();
    }

    components_finalize();

    return 0;
}
```

2.1 disp_sync_gdma 組件代碼介紹

在 *disp_sync_gdma.c* 的實作中，在 *disp_gdma_dsc_init* 函式逐一地建立 V Lines * H Stages DMA 描述符，依不同 H/VLINE Stage 給予不同的 Memory-to-memory 操作屬性和寫出位址，以完成 LCD 屏幕 **H*SYNC_ACT***、**V*SYNC_ACT***、**DE*ACT*** 與 VRAM 緩衝區內的像素輸出。

```
// Function to initialize EBI sync GDMA
static int disp_ebi_sync_gdma_init(void)
{
    enum dma350_lib_error_t lib_err;

    /* Set the VRAM address by default. */
    s_pu16BufAddr = (uint16_t*)g_au8FrameBuf;

    /* Enable GDMA module clock and un-mask interrupt. */
    gdma_init();

    /* Initial all Lines descriptor-link. */
    disp_gdma_dsc_init();

    /* Link to external command */
    dma350_ch_enable_linkaddr(GDMA_CH_DEV_S[1]);
    dma350_ch_set_linkaddr32(GDMA_CH_DEV_S[1], (uint32_t) s_head);
    dma350_ch_disable_intr(GDMA_CH_DEV_S[1], DMA350_CH_INTREN_DONE);
    dma350_ch_cmd(GDMA_CH_DEV_S[1], DMA350_CH_CMD_ENABLECMD);

    return 0;
}

static int disp_ebi_sync_gdma_fini(void)
{
    /* Disable GDMA module clock and mask interrupt. */
    gdma_fini();

    return 0;
}
```

2.2 disp_sync_pdma 組件代碼介紹

在 *disp_sync_pdma.c* 的實作中，在 *disp_pdma_dsc_init* 函式逐一地建立 V Lines * H Stages DMA 描述符，依不同 H/VLINE Stage 給予不同的 Memory-to-memory 操作屬性和寫出位址，以完成 LCD 屏幕 $\mathbf{H}_{SYNC\_ACT}$、$\mathbf{V}_{SYNC\_ACT}$、$\mathbf{DE}_{ACT}$ 各時序與 VRAM 緩衝區內的像素輸出。

```
// Function to initialize the EBI sync PDMA
static int disp_sync_pdma_init(void)
{
    struct nu_pdma_chn_cb sChnCB;

    /* Set the VRAM address by default. */
    s_pu16BufAddr = (uint16_t*)g_au8FrameBuf;

    pdma_init();

    if (s_i32Channel < 0)
    {
        /* Allocate a PDMA channel resource. */
        s_i32Channel = nu_pdma_channel_allocate(PDMA_MEM);
        if (s_i32Channel < 0)
            return -1;
    }

    /* Initial all Lines descriptor-link. */
    disp_pdma_dsc_init();

    /* Register ISR callback function */
    sChnCB.m_eCBType = eCBType_Event;
    sChnCB.m_pfnCBHandler = nu_pdma_memfun_cb;
    sChnCB.m_pvUserData = (void *)NULL;

    nu_pdma_filtering_set(s_i32Channel, NU_PDMA_EVENT_TRANSFER_DONE);
    nu_pdma_callback_register(s_i32Channel, &sChnCB);

    /* Trigger scatter-gather transferring. */
    return nu_pdma_sg_transfer(s_i32Channel, s_head, 0);
}

// Function to deinitialize the EBI sync PDMA
static int disp_sync_pdma_fini(void)
{
    if (s_i32Channel >= 0)
    {
        /* Free allocated PDMA channel resource. */
        nu_pdma_channel_free(s_i32Channel);

        s_i32Channel = -1;
    }

    pdma_fini();

    return 0;
}
```

2.3 disp_example 組件代碼介紹

在 *disp_example.c* 的實作中，透過 *.incbin* 組合語言指令將兩張 RGB565 格式的影像嵌入程式中，直接存入 FLASH，以縮短編譯時間。在 *disp_example_init* 函式中，首先註冊 Blank 事件的回調函式 (*disp_example_blankcb*)。最後，將兩張 RGB565 影像資料拷貝至影像緩衝區。由於 Cortex-M55 支援 Data-Cache，拷貝完成後需呼叫 *SCB_CleanDCache_by_Addr* 函式，以同步影像資料至 SRAM。在 *disp_example_blankcb* 中，透過 *disp_set_vrambufaddr* 函式並傳入不同的 VRAM 位址，交替更新畫面，以防止畫面撕裂。然而，由於屏示範例更新速率過快導致顯示重影，因此設定條件為收到兩次 Blank 事件後再進行畫面切換。

```
#define STR2(x) #x
#define STR(x) STR2(x)
#define INCBIN(name, file) \
    __asm__(".section .rodata\n" \
            ".global incbin_" STR(name) "_start\n" \
            ".balign 16\n" \
            "incbin_" STR(name) "_start:\n" \
            ".incbin \"" file "\"\n" \
            \
            ".global incbin_" STR(name) "_end\n" \
            ".balign 1\n" \
            "incbin_" STR(name) "_end:\n" \
            ".byte 0\n" \
    ); \
    extern const __attribute__((aligned(32))) void* incbin_ ## name ## _start; \
    extern const void* incbin_ ## name ## _end; \

static uint8_t s_au8FrameBuf[CONFIG_VRAM_TOTAL_ALLOCATED_SIZE]
__attribute__((aligned(DCACHE_LINE_SIZE))); // Declare VRAM instance.
INCBIN(image1, PATH_IMAGE1_BIN);
INCBIN(image2, PATH_IMAGE2_BIN);

// Blank event callback function
void disp_example_blankcb(void *p)
{
    static uint32_t u32Counter = 0;

    /* Toggle different image showing after getting 2 event, */
    /* Just for avoid visual persistence ghosting. */
    #define DEF_TOGGLE_COND     (u32Counter & 0x10u)

    /* Toggle between image1 and image2 display based on u32Counter's value. */
    if (DEF_TOGGLE_COND)
    {
        /* If the condition is true, set VRAM buffer to image2 buffer address. */
        disp_set_vrambufaddr((void *)&s_au8FrameBuf[CONFIG_VRAM_BUF_SIZE]);
    }
    else
    {
        /* If the condition is false, set VRAM buffer to image1 buffer address. */
        disp_set_vrambufaddr((void *)s_au8FrameBuf);
```

```
    }

    // Increment the counter to alternate the display in the next callback
    u32Counter++;
}

// Initialize the display example
static int disp_example_init(void)
{
    /* Set VRAM buffer address. */
    disp_set_vrambufaddr((void *)s_au8FrameBuf);

    /* Set blank event callback function. */
    disp_set_blankcb(disp_example_blankcb);

    /* Copy image1 and image2 pixel data to VRAM buffer. */
    memcpy(s_au8FrameBuf, (const uint8_t *)&incbin_image1_start, CONFIG_VRAM_BUF_SIZE);
    memcpy(&s_au8FrameBuf[CONFIG_VRAM_BUF_SIZE], (const uint8_t *)&incbin_image2_start,
CONFIG_VRAM_BUF_SIZE);

    /* Flush all pixel data in DCache to memory. */
    SCB_CleanDCache_by_Addr(s_au8FrameBuf, 2 * CONFIG_VRAM_BUF_SIZE);

    return 0;
}
```

3. 軟體與硬體需求

3.1 軟體需求

- BSP 版本
  - M55M1_Series_BSP_CMSIS_V3.01.001。
- IDE 版本
  - Keil uVersion 5.40。

3.2 硬體需求

- 電路元件
  - NuMaker-M55M1 V1.0。
  - EBI16-RGB24 連接板。
  - 同步信號 LCD 屏幕，解析度為 WQVGA。
- 線路示意圖
  - 將 NuMaker-M55M1 V1.0、EBI16-RGB24 連接板和同步信號 LCD 屏幕進行組裝。
  - 使用 USB Type-C to Type-A 連接線，將 NuMaker-M55M1 V1.0 開發板連接至電腦。

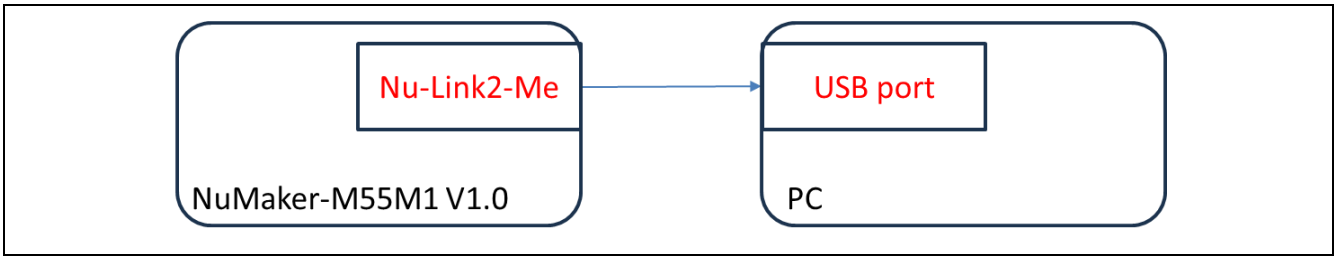


圖 3-1 線路示意圖

4. 目錄資訊

目錄結構如下圖所示。

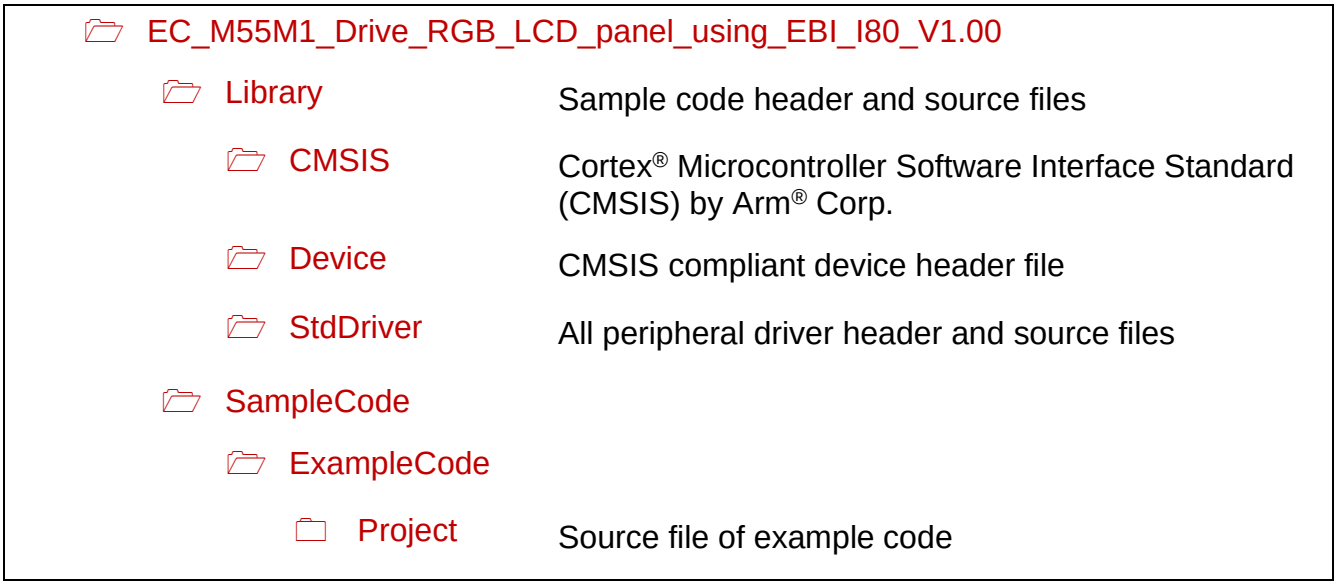


圖 4-1 目錄資訊

5. 範例程式執行

1. 根據目錄資訊章節進入 ExampleCode 路徑中的 KEIL 資料夾，雙擊 *Drive_RGB_LCD_Panel_using_EBI_I80.uvprojx*。
2. 進入編譯模式介面
   - 編譯
   - 下載代碼至記憶體
   - 執行代碼

6. 修訂紀錄

Date	Revision	Description
2025.04.01	1.00	初始發佈。

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.