NuMicro® 32位元系列微控制器範例程式碼介紹

使用 EBI 驅動 RGB LCD 屏幕

**grity**

|  |  |
| --- | --- |
| **文件資訊** | |
| 應用簡述 | 本範例程式碼使用M55M1 EBI 驅動RGB LCD 屏幕 |
| BSP版本 | M55M1\_Series\_BSP\_CMSIS\_V3.01.001 |
| 開發平台 | NuMaker-M55M1 V1.0 |

*The information described in this document is the exclusive intellectual property of  
 Nuvoton Technology Corporation and shall not be reproduced without permission from Nuvoton.*

*Nuvoton is providing this document only for reference purposes of NuMicro microcontroller and microprocessor based system design. Nuvoton assumes no responsibility for errors or omissions.*

*All data and specifications are subject to change without notice.*

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

[www.nuvoton.com](http://www.nuvoton.com)

# 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(Sync-Type) LCD屏幕 (可參閱表 1‑1)。本範例程式碼使用 M55M1 系列支援的 EBI 傳輸介面，並搭配 PDMA 或 GDMA 控制器，實現自我刷新畫面功能，以驅動同步信號 LCD 屏幕。

|  |
| --- |
|  |

圖 1‑1 EBI驅動同步信號LCD屏幕

|  |  |  |
| --- | --- | --- |
|  | **MPU-Type LCD** | **Sync-Type LCD** |
| **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，價位高。 | 無 LCD 控制器，Host 端直接輸出 RGB 資料與同步訊號至屏幕，價位相對低。 |

表 1‑1 MPU-type 與 Sync-type LCD屏幕比較表

## 原理

本範例程式碼實現同步信號LCD控制器，透過EBI傳輸介面與 PDMA或是GDMA控制器進行Memory-to-memory 傳輸，將圖片像素資料傳送至同步信號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。

|  |
| --- |
| A screenshot of a computer  Description automatically generated |

圖 1‑2 V Lines × H Stages DMA描述符

如圖 1‑2所示，將所有DMA 描述符依序鏈接，並把最後一個描述符的 **NEXT**參數設置為第一個描述符的地址，從而形成一個環狀描述符集。將最後一個描述符啟用傳輸完成中斷響應，並在中斷服務程序（ISR）中更新*VACT.HACT*階段的描述符資料傳輸起始地址，切換至新圖像的每條圖像記憶體起始地址，實現畫面動態切換功能，以防止畫面撕裂（Anti-tearing）。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **V \ H** | **HFP** | **HPW** | **HBP** | **HACT** |
| **VFP** | **Set *HSYNC\_ACT* to Inactive**  **Set *VSYNC\_ACT* to Inactive**  **Set *DEACT* to Inactive**  **Set *D[15-0]* to Dummy** | **Set *HSYNC\_ACT* to Active**  **Set *VSYNC\_ACT* to Inactive**  **Set *DEACT* to Inactive**  **Set *D[15-0]* to Dummy** | **Set *HSYNC\_ACT* to Inactive**  **Set *VSYNC\_ACT* to Inactive**  **Set *DEACT* to Inactive**  **Set *D[15-0]* to Dummy** | **Set *HSYNC\_ACT* to Inactive**  **Set *VSYNC\_ACT* to Inactive**  **Set *DEACT* to Inactive**  **Set *D[15-0]* to Dummy** |
| **VPW** | **Set *HSYNC\_ACT* to Inactive**  **Set *VSYNC\_ACT* to Active**  **Set *DEACT* to Inactive**  **Set *D[15-0]* to Dummy** | **Set *HSYNC\_ACT* to Active**  **Set *VSYNC\_ACT* to Active**  **Set *DEACT* to Inactive**  **Set *D[15-0]* to Dummy** | **Set *HSYNC\_ACT* to Inactive**  **Set *VSYNC\_ACT* to Active**  **Set *DEACT* to Inactive**  **Set *D[15-0]* to Dummy** | **Set *HSYNC\_ACT* to Inactive**  **Set *VSYNC\_ACT* to Active**  **Set *DEACT* to Inactive**  **Set *D[15-0]* to Dummy** |
| **VBP** | **Set *HSYNC\_ACT* to Inactive**  **Set *VSYNC\_ACT* to Inactive**  **Set *DEACT* to Inactive**  **Set *D[15-0]* to Dummy** | **Set *HSYNC\_ACT* to Active**  **Set *VSYNC\_ACT* to Inactive**  **Set *DEACT* to Inactive**  **Set *D[15-0]* to Dummy** | **Set *HSYNC\_ACT* to Inactive**  **Set *VSYNC\_ACT* to Inactive**  **Set *DEACT* to Inactive**  **Set *D[15-0]* to Dummy** | **Set *HSYNC\_ACT* to Inactive**  **Set *VSYNC\_ACT* to Inactive**  **Set *DEACT* to Inactive**  **Set *D[15-0]* to Dummy** |
| **VACT** | **Set *HSYNC\_ACT* to Inactive**  **Set *VSYNC\_ACT* to Inactive**  **Set *DEACT* to Inactive**  **Set *D[15-0]* to Dummy** | **Set *HSYNC\_ACT* to Active**  **Set *VSYNC\_ACT* to Inactive**  **Set *DEACT* to Inactive**  **Set *D[15-0]* to Dummy** | **Set *HSYNC\_ACT* to Inactive**  **Set *VSYNC\_ACT* to Inactive**  **Set *DEACT* to Inactive**  **Set *D[15-0]* to Dummy** | **Set *HSYNC\_ACT* to Inactive**  **Set *VSYNC\_ACT* to Inactive**  **Set *DEACT* to Active**  **Set *D[15-0]* to pixel data in every Line**  Raising blank event in latest descriptor. |

表 1‑2 各LCD 時序階段與***HSYNC\_ACT、VSYNC\_ACT*** 和 ***DEACT*** 腳位對應表

由於不同類型屏幕對於同步信號支援程度不同，有些屏幕可支援 ***DEACT-ONLY*** 同步模式，表示它們僅需要 ***DEACT*** 信號輸入；而無需 ***HSYNC\_ACT*** 和 ***VSYNC\_ACT*** 信號輸入。在操作 ***DEACT-ONLY***屏幕時，可大幅度地減少 DMA 描述符的使用量，以節省系統記憶體空間與使用頻寬。如表 1‑2 各LCD 時序階段與***HSYNC\_ACT、VSYNC\_ACT*** 和 ***DEACT*** 腳位對應表與表 1‑3所列。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **V \ H** | **HFP** | **HPW** | **HBP** | **HACT** |
| **VFP** | **Set *DEACT* to Inactive**  **Set *D[15-0]* to Dummy** | | | |
| **VPW** |
| **VBP** |
| **VACT** | **Set *DEACT* to Inactive**  **Set *D[15-0]* to Dummy** | | | **Set *DEACT* to Active**  **Set *D[15-0]* to pixel data in every Line**  Raising blank event in latest descriptor. |

表 1‑3 各LCD 時序階段與***DEACT-Only*** 腳位對應表

如圖1‑3所示，LCD時序的***HSYNC\_ACT***、***VSYNC\_ACT***和***DEACT*** 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](籃色)。
* 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

## 執行結果

本範例程式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1‑4 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1‑5與圖1‑6所示，呈現終端機執行結果和圖像於屏幕呈現與兩張圖片切換效果。

|  |
| --- |
|  |

圖1‑4 選擇編譯選項與執行編譯

|  |
| --- |
|  |

圖1‑5執行畫面

|  |
| --- |
| TBD…..  Due date: 3/21 |

圖1‑6屏幕呈現圖像

# 程式碼介紹

本範例程式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、結束函式程式碼和設定檔位於 *disp\_sync\_gdma.c*，*disp\_sync\_pdma.c*，*board.c*、*disp\_example.c* 和 *disp.h*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;

}

## disp\_sync\_gdma組件程式碼介紹

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，輸出LCD 屏幕***HSYNC\_ACT***、***VSYNC\_ACT***、***DEACT*** 訊號與VRAM緩衝區內的像素。這些描述符會儲存在一塊不可快取的記憶體區域*s\_sDecLCD*，程式碼變數屬性宣告為*NVT\_NONCACHEABLE*，意味著當 CPU 向這些記憶體位址寫入資料時，資料不會進入 CPU 的資料快取空間，而是直接寫入記憶體。在中斷服務函式內處理 Blank 事件時，程式會更新所有 *VACT.HACT* 區DMA 描述符的起始位址參數，從而切換不同的 VRAM 緩衝區。這樣的做法避免了重複操作*SCB\_CleanDCache\_by\_Addr* 函式來刷新起始位址參數，從而提高切換不同 VRAM 緩衝區的執行效率。

#if defined(NVT\_NONCACHEABLE)

/\* V × H DMA Descriptors, stored in a non-cacheable memory region. \*/

NVT\_NONCACHEABLE static S\_DSC\_LCD s\_sDscLCD;

#else

static S\_DSC\_LCD s\_sDscLCD;

#endif

// 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;

}

## disp\_sync\_pdma組件程式碼介紹

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，輸出LCD 屏幕***HSYNC\_ACT***、***VSYNC\_ACT***、***DEACT***訊號與VRAM緩衝區內的像素。這些描述符會儲存在一塊不可快取的記憶體區域*s\_sDecLCD*，程式碼變數屬性宣告為*NVT\_NONCACHEABLE*，意味著當 CPU 向這些記憶體位址寫入資料時，資料不會進入 CPU 的資料快取空間，而是直接寫入記憶體。在中斷服務函式內處理 Blank 事件時，程式會更新所有 *VACT.HACT* 區DMA 描述符的起始位址參數，從而切換不同的 VRAM 緩衝區。這樣的做法避免了重複操作*SCB\_CleanDCache\_by\_Addr* 函式來刷新起始位址參數，從而提高切換不同 VRAM 緩衝區的執行效率。

#if defined(NVT\_NONCACHEABLE)

/\* V × H DMA Descriptors, stored in a non-cacheable memory region. \*/

NVT\_NONCACHEABLE static S\_DSC\_LCD s\_sDscLCD;

#else

static S\_DSC\_LCD s\_sDscLCD;

#endif

// 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;

}

## disp\_example組件程式碼介紹

在 *disp\_example.c* 的實作中，使用 *.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。程式碼中，*PATH\_IMAGE1\_BIN*與*PATH\_IMAGE2\_BIN*分別為兩張圖片檔案路徑定義值，存於*disp.h*標頭檔中。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，由於展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;

}

## disp驅動參數介紹

*disp.h* 標頭檔用於定義同步信號 LCD 屏幕的同步訊號腳位與時間驅動參數。這些參數需依據實際板子的引腳配置進行設置，例如：

*CONFIG\_LCD\_PANEL\_USE\_DE\_ONLY* 取消定義後，系統將啟用 *VSYNC\_ACT*、*HSYNC\_ACT* 和 *DEACT* 三條訊號輸出至屏幕；反之，僅輸出 *DEACT* 訊號*。*

*CONFIG\_DISP\_\*\_ACTIVE\_LOW*設為 0 時，表示對應訊號為 Active-High 動作；反之，則為 Active-Low 動作。

*CONFIG\_DISP\_\*\_BITIDX*設定值為各*EBI\_ADRx*腳位編號加1的值。

*CONFIG\_TIMING\_\** 用於設定同步訊號屏幕的時序參數，需與屏幕規格書對應調整。

#define CONFIG\_LCD\_PANEL\_USE\_DE\_ONLY /\*!< DE-only mode, W/O H/VSync. \*/

#define CONFIG\_DISP\_DE\_ACTIVE\_LOW 0 /\*!< Disable DE active low \*/

#define CONFIG\_DISP\_VPW\_ACTIVE\_LOW 1 /\*!< Enable VPW active low \*/

#define CONFIG\_DISP\_HPW\_ACTIVE\_LOW 1 /\*!< Enable HPW active low \*/

#define CONFIG\_DISP\_DE\_BITIDX 8 /\*!< Implies SET\_EBI\_ADR7\_PH0 \*/

#define CONFIG\_DISP\_VSYNC\_BITIDX 1 /\*!< Implies SET\_EBI\_ADR0\_PH7 \*/

#define CONFIG\_DISP\_HSYNC\_BITIDX 2 /\*!< Implies SET\_EBI\_ADR1\_PH6 \*/

#define CONFIG\_TIMING\_HACT 480 /\*!< Specify XRES \*/

#define CONFIG\_TIMING\_VACT 272 /\*!< Specify YRES \*/

#define CONFIG\_TIMING\_HBP 30 /\*!< Specify HBP (Horizontal Back Porch) \*/

#define CONFIG\_TIMING\_HFP 5 /\*!< Specify HFP (Horizontal Front Porch) \*/

#define CONFIG\_TIMING\_HPW 41 /\*!< Specify HPW (HSYNC plus width) \*/

#define CONFIG\_TIMING\_VBP 2 /\*!< Specify VBP (Vertical Back Porch) \*/

#define CONFIG\_TIMING\_VFP 27 /\*!< Specify VFP (Vertical Front Porch) \*/

#define CONFIG\_TIMING\_VPW 10 /\*!< Specify VPW (VSYNC width) \*/

# 軟體與硬體需求

## 軟體需求

BSP版本

M55M1\_Series\_BSP\_CMSIS\_V3.01.001。

IDE版本

Keil uVision 5.40。

## 硬體需求

電路元件

NuMaker-M55M1 V1.0。

EBI16-RGB24連接板。

RGB LCD Panel，解析度為WQVGA。

線路示意圖

將NuMaker-M55M1 V1.0、EBI16-RGB24連接板和RGB LCD Panel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

# 目錄資訊

目錄結構如下圖所示。

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| * EC\_M55M1\_Drive\_RGB\_LCD\_Panel\_using\_EBI\_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 | | * + StdDriver | All peripheral driver header and source files | | * + SampleCode |  | | * + ExampleCode |  | |
| |  |  | | --- | --- | | * + - Project | Sources, extension and libraries | |

圖 4‑1目錄資訊

# 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.uvprojx*。
        2. 進入編譯模式介面

編譯

下載程式碼至記憶體

執行程式碼

# 修訂紀錄

|  |  |  |
| --- | --- | --- |
| **Date** | **Revision** | **Description** |
| 2025.03.05 | 1.00 | 初始發佈。 |

**A notice with text and black text

Description automatically generated with medium confidence**