NuMicro® 32位系列微控制器范例程序代码介绍

使用 EBI 驱动同步訊號 LCD 屏幕

**grity**

|  |  |
| --- | --- |
| **文件信息** | |
| 应用简述 | 本范例程序代码使用EBI 驱动同步讯号LCD 屏幕 |
| BSP版本 | M55M1\_Series\_BSP\_CMSIS\_V3.01.001 |
| 开发平台 | 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

一般常用于微控制器的LCD屏幕接口为MPU-type LCD，由于这类屏幕内建控制器与显示内存，以致于价格高于同步信号(Sync-Type) LCD屏幕 (可参阅表 1‑1)。本范例程序代码使用 M55M1 系列支持的 EBI 传输接口，并搭配 PDMA 或 GDMA 控制器，实现自我刷新画面功能，以驱动同步信号 LCD 屏幕，如图 1‑1所示。

|  |  |  |
| --- | --- | --- |
|  | **MPU-type LCD** | **Sync-type LCD** |
| **传输方式** | Host 透过传输接口将影像数据同步写入 LCD 显示内存，由 LCD 控制器负责直接输出 RGB 数据与同步信号至面板。 | Host 端直接输出 RGB 与同步信号至屏幕，用于影像显示。 |
| **内存与带宽使用量** | Host 可将少量图像数据分批同步至 LCD 显示内存，由 LCD 面板周期性地读取该内存并输出影像，达成画面显示。此架构可有效降低 Host 的内存与带宽使用量。 | Host 需将完整的图像数据储存在自身内存中，并由内建的显示控制器读取这些资料后输出至 LCD 屏幕。控制器需以较高带宽周期性地存取内存以维持影像输出，因此 Host 的内存与带宽使用量相对较高。 |
| **传输接口** | LCD 模块可透过 EBI i80（并行接口）或 SPI（串行接口）与 Host 连接。EBI 提供较高的数据传输速率，适合高分辨率或高更新率应用；而 SPI 则以低脚位数与简单连接为优势，适用于资源受限的系统设计。 | 24-bit RGB 接口（RGB888）采用 8-bit 的红、绿、蓝三信道数据进行显示输出，并搭配同步讯号（如 HSYNC、VSYNC、DE 及 PCLK）实现实时画面驱动。此接口可提供高画质与低延迟的影像显示，但需占用较多的 I/O 脚位与传输带宽。 |
| **硬件成本** | 具备内建控制器与显示内存的屏幕可减轻 Host 侧的负担，然而由于额外硬件设计，其模块成本相对较高。 | 此类屏幕未内建显示控制器与显示内存，须由 Host 端负责影像输出处理，但由于省略了相关硬件设计，模块成本相对较低。 |

表 1‑1 MPU-type 与 Sync-type LCD屏幕比较表

|  |
| --- |
|  |

图 1‑1 EBI驱动同步信号LCD屏幕

## 原理

本范例程序代码实现同步信号LCD控制器，透过EBI传输接口与 PDMA或是GDMA控制器进行Memory-to-memory 传输，将图片像素数据传送至同步信号LCD屏幕，其运作原理为屏幕画面建立V Lines × H Stages DMA描述符，其中：

* V代表 LCD时序的 Vertical Pulse Width (VPW)、Vertical Back Porch (VBP)、Vertical Active (VA) 和 Vertical Front Porch (VFP) 的扫描线数量。
* H代表Horizontal Pulse Width (HPW)、Horizontal Back Porch (HBP)、Horizontal Active (HA)和Horizontal Front Porch (HFP)的这四个阶段。

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

图 1‑2 V Lines × H Stages DMA描述符

如图 1‑2所示，将所有DMA 描述符依序链接，并把最后一个描述符的 NEXT参数设置为第一个描述符的地址，从而形成一个环状描述符集，并且将最后一个描述符启用传输完成中断响应，如表 1‑2所示。驱动程序码在中断服务程序（ISR）中更新VACT.HACT阶段的描述符数据传输起始地址，切换至新图像的每条图像内存起始地址，实现画面动态切换功能，以防止画面撕裂（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. |

表 1‑2 各LCD 时序阶段与HSYNC\_ACT、VSYNC\_ACT 和 DEACT 脚位对应表

由于不同类型屏幕对于同步信号支持程度不同，有些屏幕可支持 DEACT-ONLY 同步模式，表示它们仅需要 DEACT 信号输入；而无需 HSYNC\_ACT 和 VSYNC\_ACT 信号输入。在操作 DEACT-ONLY屏幕时，可大幅度地减少 DMA 描述符的使用量，以节省系统内存空间与使用带宽。如表 1‑2与表 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. |

表 1‑3 各LCD 时序阶段与**DEACT-Only** 脚位对应表

如图1‑3所示，LCD时序的**HSYNC\_ACT**、**VSYNC\_ACT**和**DEACT** 三个同步讯号脚位分别透过EBI的**ADR1**、**ADR0**和**ADR7**地址脚位来仿真，透过指定不同地址译码以产生LCD同步讯号、PCLK讯号连接至 **EBI-nWR** 脚位和EBI 的16位数据脚位**D[15-0]** 对应到LCD颜色信号：

* **R**[7-3](红色)，**G**[7-2](绿色) 和 **B**[7-3](篮色)。
* 额外的**R**[2-0]、**G**[1-0] 和 **B**[2-0] 可以使用**R**[7-5]、**G**[7-6] 和 **B**[7-5] 的数据进行**RGB888补色**，以呈现更高的颜色精度。

|  |
| --- |
|  |

图1‑3 EBI16-to-RGB24各讯号脚位连接图

## 执行结果

本范例程序代码提供两种目标编译选项：**WQVGA\_GDMA** 和 **WQVGA\_PDMA**，分别对应使用 GDMA 或 PDMA 控制器进行自我刷新画面功能，如图1‑4所示。执行时，可透过连接 Nu-Link2-Me 的 VCOM 功能，搭配计算机端终端机软件进行观测，默认通讯参数为 115200N81。如图1‑5与图1‑6所示，呈现终端机执行结果和图像于屏幕呈现图片效果。

|  |
| --- |
|  |

图1‑4 选择编译选项与执行编译

|  |
| --- |
|  |

图1‑5执行画面

|  |  |
| --- | --- |
|  |  |

图1‑6屏幕呈现图像

# 程序代码介绍

本范例程序代码执行各组件初始化与结束流程，程序代码位于 **main.c**。各组件初始化、结束函式程序代码和配置文件位于 **disp\_sync\_gdma.c**，**disp\_sync\_pdma.c**，**disp\_example.c** 和 **disp.h**实作内。各组件实作使用**COMPONENT\_EXPORT** 宏注册组件名称(.**name**)、组件初始化回调函式(.**initialize**) 和组件结束回调函式(.**finalize**)，并将这些组件注册数据结收录至 **CompInitTab** 区域，供主程序呼叫执行。

/\*\*

\* @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);

if ( asCompInitTbl[i].initialize() < 0 )

{

// Print message if return of initialize function is small than zero.

printf("Initialize %s failure.\n", asCompInitTbl[i].name);

}

}

}

}

/\*\*

\* @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);

if (asCompInitTbl[i].finalize() < 0)

{

// Print message if return of finalize function is small than zero.

printf("Initialize %s failure.\n", asCompInitTbl[i].name);

}

}

}

}

int main(void)

{

// Module clocks and function pin setting initialization.

board\_init();

/\* Initialize all components \*/

components\_initialize();

/\* Just keep here, or put your code here. \*/

while (1)

{

\_\_WFI();

}

/\* This will never execute due to the infinite loop. \*/

/\* Just keep for orthogonal implementation. \*/

components\_finalize();

// Module clocks and function pin setting finalization.

board\_fini();

return 0;

}

## disp\_sync\_gdma.c

在 **disp\_sync\_gdma.c** 的实作中，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**给予不同的Memory-to-memory 操作属性和写出地址，输出LCD 屏幕**HSYNC\_ACT**、**VSYNC\_ACT**、**DEACT** 讯号与VRAM缓冲区内的像素。这些描述符会储存在一块不可快取的内存区域，程序代码变量名称为**s\_sDecLCD**，属性宣告为**NVT\_NONCACHEABLE**，意味着当 CPU 向这些内存地址写入数据时，数据不会进入 CPU 的数据快取空间，而是直接写入内存。在中断服务函式内处理 **Blank** 事件时，程序会更新所有 **VACT.HACT** 区DMA 描述符的起始地址参数，从而切换不同的 VRAM 缓冲区。这样的做法避免了重复操作**SCB\_CleanDCache\_by\_Addr** 函式来刷新起始地址参数，从而提高切换不同 VRAM 缓冲区的执行效率。

#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.c

在 **disp\_sync\_pdma.c** 的实作中，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**给予不同的Memory-to-memory 操作属性和写出地址，输出LCD 屏幕**HSYNC\_ACT**、**VSYNC\_ACT**、**DEACT**讯号与VRAM缓冲区内的像素。这些描述符会储存在一块不可快取的内存区域，程序代码变量名称为**s\_sDecLCD**，属性宣告为**NVT\_NONCACHEABLE**，意味着当 CPU 向这些内存地址写入数据时，数据不会进入 CPU 的数据快取空间，而是直接写入内存。在中断服务函式内处理 **Blank** 事件时，程序会更新所有 **VACT.HACT** 区DMA 描述符的起始地址参数，从而切换不同的 VRAM 缓冲区。这样的做法避免了重复操作**SCB\_CleanDCache\_by\_Addr** 函式来刷新起始地址参数，从而提高切换不同 VRAM 缓冲区的执行效率。

#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.c

在 **disp\_example.c** 的实作中，使用 .**incbin** 汇编语言指令将两张 RGB565 格式的影像嵌入程序中，直接存入 FLASH，以缩短编译时间。程序代码中，**PATH\_IMAGE1\_BIN**与**PATH\_IMAGE2\_BIN**分别为两张图片档案路径定义值，存于**disp**.h头文件中。图像文件案可透过在线转换工具 **Image to RGB565 Converter** (<https://longfangsong.github.io/en/image-to-rgb565/>) 进行汇入全屏分辨率的PNG图文件并下载，取得 RGB565 格式档案。在 **disp\_example\_init** 函式中，首先注册 Blank 事件的回调函式 (**disp\_example\_blankcb**)。最后，将两张 RGB565 影像数据拷贝至影像缓冲区。由于 **Cortex-M55** 支持 Data-Cache，拷贝完成后需呼叫 **SCB\_CleanDCache\_by\_Addr** 函式，以同步影像数据至 SRAM。在 **disp\_example\_blankcb** 中，透过 **disp\_set\_vrambufaddr** 函式并传入不同的 VRAM 地址，交替更新画面，以防止画面撕裂。然而，由于展示范例更新速率过快导致显示重影，因此设定条件为收到16次 **Blank** 事件后再进行画面切换。

#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 16 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.h

**disp.h** 头文件用于定义同步信号 LCD 屏幕的同步讯号脚位与时间驱动参数。这些参数需依据实际板子的引脚配置进行设置，例如：

**CONFIG\_LCD\_PANEL\_USE\_DE\_ONLY** 取消定义后，系统将启用 **VSYNC\_ACT**、**HSYNC\_ACT** 和 **DEACT** 三条讯号输出至屏幕；反之，仅输出 **DEACT** 讯号。

**CONFIG\_DISP\_\*\_ACTIVE\_LOW**设为 0 时，表示对应讯号为 **Active-High** 动作；反之，则为 **Active-Low** 动作。

**CONFIG\_DISP\_\*\_BITIDX**设定值为各**EBI\_ADRx**脚位编号加1的值。

**CONFIG\_TIMING\_\*** 用于设定同步讯号屏幕的时序参数，需与屏幕规格书对应调整。

#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 Width \*/

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

#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 Pulse 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) \*/

# 软件与硬件需求

## 软件需求

BSP版本

M55M1\_Series\_BSP\_CMSIS\_V3.01.001。

IDE版本

Keil uVision 5.40。

## 硬件需求

电路组件

NuMaker-M55M1 V1.0。

EBI16-RGB24连接板。

RGB LCD Panel，分辨率为WQVGA。

线路示意图

将NuMaker-M55M1 V1.0、EBI16-RGB24连接板和RGB LCD Panel进行组装。

使用USB Type-C to Type-A 连接线，将NuMaker-M55M1 V1.0开发板连接至计算机。

|  |
| --- |
|  |

图 3‑1 线路示意图

# 目录信息

目录结构如下图所示。

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| * EC\_M55M1\_Driving\_Sync\_Signal\_LCD\_Panel\_via\_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 | |

图 4‑1目录信息

# 范例程序执行

* + - * 1. 根据目录信息章节进入ExampleCode 路径中的KEIL文件夹，双击*Driving\_Sync\_Signal\_LCD\_Panel\_via\_EBI.uvprojx*。
        2. 进入编译模式接口

1. 编译
2. 下载代码至内存
3. 进入 / 离开除错模式
   * + - 1. 进入除错模式接口
4. 执行代码

# 修订纪录

|  |  |  |
| --- | --- | --- |
| **Date** | **Revision** | **Description** |
| 2025.03.21 | 1.00 | 初始发布。 |

**A notice with text and black text

Description automatically generated with medium confidence**