Example Code Introduction for 32-bit NuMicro® Family

A Software-based LCD Controller

**grity**

|  |  |
| --- | --- |
| **Document Information** | |
| Application | This example code uses the M55M1 EBI to drive a synchronous signal LCD screen. |
| BSP Version | M55M1\_Series\_BSP\_CMSIS\_V3.01.001 |
| Hardware | 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)

# Overview

The commonly used LCD screen interface for microcontrollers is the MPU-type LCD. Since this type of screen has a built-in controller and display memory, its cost is higher than that of a synchronous signal LCD screen (refer to Table 1‑1 Comparison Table of MPU-type and Sync-type LCD). This example code uses the M55M1 EBI to drive a synchronous signal (Sync-Type) LCD screen. It leverages the EBI interface along with the PDMA or GDMA supported by the M55M1 series to achieve self-refresh functionality.

|  |
| --- |
|  |

Figure 1‑1 EBI drives a synchronous signal LCD Panel

|  |  |  |
| --- | --- | --- |
|  | **MPU-Type LCD** | **Sync-Type LCD** |
| **Transmission Method** | EBI i80 interface or SPI for command operations and image data transmission. | Display images using RGB and synchronous signals. |
| **Memory and Bandwidth Usage** | Low | High |
| **Applicable Scenarios** | Suitable for static images or minor dynamic image updates. | Suitable for higher resolution and dynamic images. |
| **Hardware Cost** | Equipped with a built-in LCD controller, outputs RGB data and synchronous signals from the screen, with a higher price. | No built-in LCD controller; the host directly outputs RGB data and synchronous signals to the screen, resulting in a relatively lower cost. |

Table 1‑1 Comparison Table of MPU-type and Sync-type LCD Panel

## Principle

This example code simulates a synchronous signal LCD controller using EBI and utilizes PDMA or GDMA for memory-to-memory transmission. It transfers RGB565 pixel data to the synchronous signal LCD screen through the EBI interface. The operating principle involves establishing DMA descriptors for V Lines × H Stages based on the screen layout, where:

* **V (Vertical)** represents the number of scan lines for **Vertical Pulse Width (VPW), Vertical Back Porch (VBP), Vertical Active (VA),** and **Vertical Front Porch (VFP)** in the LCD timing.
* **H (Horizontal)** represents the four stages: **Horizontal Pulse Width (HPW), Horizontal Back Porch (HBP), Horizontal Active (HA),** and **Horizontal Front Porch (HFP).**

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

Figure 1‑2 V Lines × H Stages DMA Descriptors

As shown in Figure 1‑2 V Lines × H Stages DMA Descriptors, these **V Lines × H Stages** DMA descriptors are linked sequentially. The **NEXT LINK** of the last descriptor is set to the address of the first descriptor, forming a circular descriptor set. The last descriptor enables a transfer completion interrupt, and within the Interrupt Service Routine (ISR), the starting address for **HACT** phase descriptor data transmission is updated. This ensures that each lines switches to the memory start address of the new image, achieving dynamic screen updates while providing an anti-tearing function.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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. |

Table 1‑2 Correspondence of LCD Timing Stages and ***HSYNC\_ACT, VSYNC\_ACT*** and ***DEACT*** Pins

Due to the varying levels of support for synchronization signals across different types of screens, some screens can support ***DEACT-Only*** synchronization mode, meaning they only require the ***DEACT*** signal input and do not need ***HSYNC\_ACT*** or ***VSYNC\_ACT*** signal inputs. When operating such ***DEACT-Only*** Panel, the number of DMA descriptors can be significantly reduced, saving system memory space and bandwidth usage. This is illustrated in Table 1‑3 Correspondence of LCD Timing Stages and ***DEACT-Only*** Pin, as well as.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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 in latest descriptor. |

Table 1‑3 Correspondence of LCD Timing Stages and ***DEACT-Only*** Pin

As shown in Figure 1‑3 EBI16-to-RGB24 Signal Pin Connection Diagram, the ***HSYNC\_ACT***, ***VSYNC\_ACT***, and ***DEACT*** synchronization signal pins in the LCD timing are simulated through the EBI's ***ADR1***, ***ADR0***, and ***ADR7*** address pins. By specifying different address decoding, LCD synchronization signals are generated, and the PCLK signal is connected to the ***EBI-nWR*** pin. The EBI's 16-bit data pins *D[15-0]* correspond to the LCD color signals as follows:

* R[7-3] (Red), G[7-2] (Green), and B[7-3] (Blue).
* Additional R[2-0], G[1-0], and B[2-0] can be generated using the data from R[7-5], G[7-6], and B[7-5] for RGB888 color complement, enabling higher color precision.

|  |
| --- |
|  |

Figure 1‑3 EBI16-to-RGB24 Signal Pin Connection Diagram

This example code requires an expansion sub-board, including the **NuMaker-M55M1 development board**, a **4.3” WQVGA synchronous signal panel**, and an **EBI16–RGB24 glue board**, as shown in Figure 1‑4 External Hardware Modules.

|  |
| --- |
| A close up of a device  Description automatically generated |

Figure 1‑4 External Hardware Modules

## Execution Result

This example code provides two target compilation options: ***WQVGA\_GDMA*** and ***WQVGA\_PDMA***, corresponding to the use of the GDMA or PDMA controllers for the self-refresh screen functionality, as shown in Figure 1‑5 Compile Option Selection and Execution. During execution, you can observe the results by connecting the NULINK2ME through the VCOM feature and using terminal software on the computer. The default communication parameters are ***115200N81***. As shown in Figure 1‑6 Execution Screen, the terminal will display the execution results. Users can also directly observe the image switching and updating effects on the connected screen.

|  |
| --- |
|  |

Figure 1‑5 Compile Option Selection and Execution

|  |
| --- |
|  |

Figure 1‑6 Execution Screen

# Code Description

This example code executes the initialization and termination processes for each component, with the code located in ***main.c***. The initialization and termination functions for each component are implemented in ***disp\_sync\_gdma.c***, ***disp\_sync\_pdma.c***, ***board.c***, ***disp\_example.c***, and ***disp.h***. The component implementations use the ***COMPONENT\_EXPORT*** macro to register the component name (.*name*), initialization callback function (.*initialize*), and termination callback function (.*finalize*). These component registration details are stored in the ***CompInitTab*** section, which is then called and executed by the main program.

/\*\*

\* @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 Component Description

In the implementation of ***disp\_sync\_gdma.c***, the function ***disp\_gdma\_dsc\_init*** sequentially creates the **V Lines × H Stages DMA descriptors**. For each different **H/V LINE Stage**, it assigns the appropriate **Memory-to-memory** operation attributes and write addresses to output the pixel data from the **VRAM buffer** to the LCD screen's ***HSYNC\_ACT***, ***VSYNC\_ACT***, and ***DEACT*** signals. This process ensures that the synchronization signals and pixel data are correctly transferred to the screen.

// 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 Component Description

In the implementation of ***disp\_sync\_pdma.c***, the function ***disp\_pdma\_dsc\_init*** sequentially creates the **V Lines × H Stages DMA descriptors**. For each different **H/V LINE Stage**, it assigns the appropriate **Memory-to-memory** operation attributes and write addresses to output the pixel data from the **VRAM buffer** to the LCD screen. This process ensures that the ***HSYNC\_ACT***, ***VSYNC\_ACT***, and ***DEACT*** signals, along with the corresponding timing sequences, are correctly transferred to the screen, enabling proper image output.

// 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組件代碼介紹

In the implementation of ***disp\_example.c***, the ***.incbin*** assembly directive is used to embed two **RGB565** format images directly into the program and store them in **FLASH** to reduce compilation time. In the ***disp\_example\_init*** function, the callback function for the **Blank** event (***disp\_example\_blankcb***) is first registered. Then, the two **RGB565** image data are copied into the image buffer. Since the **Cortex-M55** supports **Data-Cache**, after the copy operation, the function ***SCB\_CleanDCache\_by\_Addr*** is called to synchronize the image data to **SRAM**. In the ***disp\_example\_blankcb*** function, the ***disp\_set\_vrambufaddr*** function is used to alternately update the screen by passing different **VRAM** addresses, which helps prevent screen tearing. However, because the update rate in the demo example is too fast and causes image ghosting, the condition is set to switch the screen only after receiving the **Blank** event twice.

#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

The ***disp.h*** header file is used to define the synchronization signal pins and timing drive parameters for the synchronous signal LCD screen. These parameters must be configured according to the actual board's pin layout, such as:

* ***CONFIG\_LCD\_PANEL\_USE\_DE\_ONLY***: When undefined, the system will enable the output of ***VSYNC\_ACT***, ***HSYNC\_ACT***, and ***DEACT*** signals to the screen; otherwise, only the ***DEACT*** signal is output.
* ***CONFIG\_DISP\_\*\_ACTIVE\_LOW***: When set to 0, it indicates the corresponding signal is **Active-High**; otherwise, it is **Active-Low**.
* ***CONFIG\_DISP\_\*\_BITIDX***: The setting value is the ***EBI\_ADRx*** pin number plus 1.
* ***CONFIG\_TIMING\_\****: Used to configure the timing parameters for the synchronization signal screen, which need to be adjusted according to the screen's datasheet.

#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) \*/

# Software and Hardware Requirements

## Software Requirements

BSP version

M55M1\_Series\_BSP\_CMSIS\_V3.01.001。

IDE version

Keil uVision 5.40。

## Hardware Requirements

Circuit components

NuMaker-M55M1 V1.0

EBI16-RGB24 Glue board

Sync-type LCD Panel, Resolution: WQVGA

Schematic diagram of the route

Assemble the NuMaker-M55M1 V1.0, EBI16-RGB24 adapter board, and the synchronous signal LCD screen.

Use an USB Type-C to Type-A cable to connect the NuMaker-M55M1 V1.0 development board to the computer.

|  |
| --- |
|  |

Figure 3‑1 Pin Connection Diagram

# 目錄資訊

The directory structure is shown in the diagram below.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| * EC\_M55M1\_Drive\_RGB\_LCD\_panel\_using\_EBI\_I80\_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 | Source file of example code | |

Figure 4‑1 Directory Structure

# Example Code Execution

* + - * 1. Navigate to the ***ExampleCode*** section and enter the **KEIL** folder. Double-click on ***Drive\_RGB\_LCD\_Panel\_using\_EBI\_I80.uvprojx***.
        2. Enter the compilation mode interface:

Compile the code.

Download the code to flash memory.

Run the code.

# Revision History

|  |  |  |
| --- | --- | --- |
| **Date** | **Revision** | **Description** |
| 2025.04.01 | 1.00 | Initial version。 |

**A notice with text and black text

Description automatically generated with medium confidence**