恩智浦半导体
应用说明

文件编号：AN12302
修订. 0, 12/2018

i.MX RT eLCDIF RGB 模式用例

URL: https://www.nxp.com/docs/en/application-note/AN12302.pdf

1. 简介

增强型液晶显示接口（eLCDIF）是一种通用显示控制器。该模块支持红、绿、蓝（RGB）模式接口（DCTCLK）和可编程功能。包括

- 系统总线主控器到源帧缓冲区数据以进行显示刷新；
- 用于各种颜色格式的 8/16/18/24 位 LCD 数据总线；
- DOTCLK LCD 接口的可配置时序和参数。

本应用笔记中例程源代码是已发布 SDK 中的 sd_jpeg 工程。硬件环境是 MIMXRT1050-EVK 板卡。

目录

2. LCD 显示 RGB 接口模式

i.MX RT eLCDIF 控制器支持 RGB（DOTCLK）LCD 接口，它是一种通用的并行数据总线接口。

2.1. RGB 模式介绍

LCD（TFT）显示器在每个时钟周期驱动 1 个像素，同时需要背光来进行显示。RGB 模式有如下特点：

- 支持很多颜色。
- 每个时钟传输一个像素（红，绿，蓝三段）。
- 显示的颜色深度取决于在 LCD 屏幕的数据线数量和 LCD 控制器输出的数据信号量。每个像素（bpp）可能包含 24bit，18bit，16bit，15bit 或者 8bit。
- 支持并行数据接口。1 个时钟传输 1 个像素的 24bit（或其他格式）。

图 1 展示了 1 个像素包含的数据。

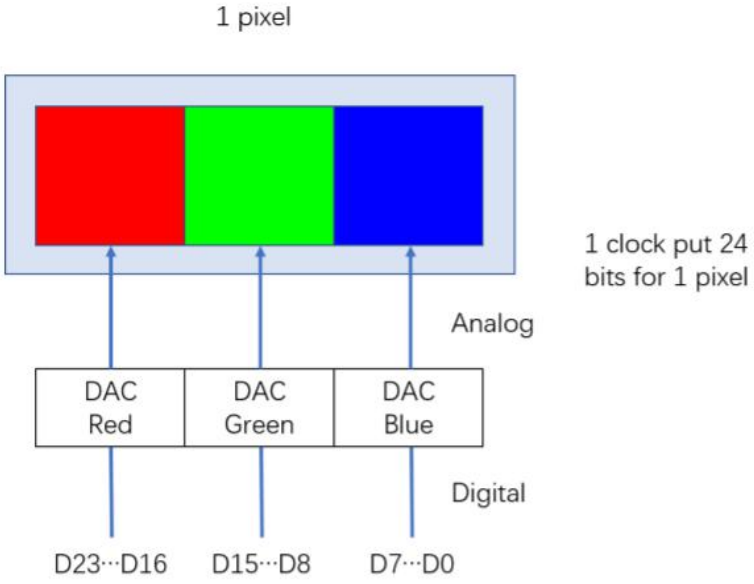

图 1. 一个像素数据

对于 RGB 模式接口，LCD 背光不是通过 LCD 控制总线控制。不同的 LCD 屏幕大小使用不同的背光技术：大的屏幕使用荧光灯而稍小的屏幕使用 LED。通常，背光由恒定电流源供电，数字信号用于开/关控制。

2.2. LCD RGB 总线时序

对于 LCD RGB 模式总线，控制器在每个像素时钟沿（上升沿或下降沿）从 FIFO 中获取一个像素数据，将其转化为 LCD 像素格式（RGB888 或其它格式编码）后放入信号发生器中，并输出到 RGB 接口。然后，这些像素数据就会显示在 LCD 屏幕的有效区域内。

需要对时序参数进行配置以匹配不同尺寸的 LCD 屏。

图 2 说明了可配置的时序参数和分辨率。表格 1 给出了 480×272 大小的 LCD 屏的时序参数配置。

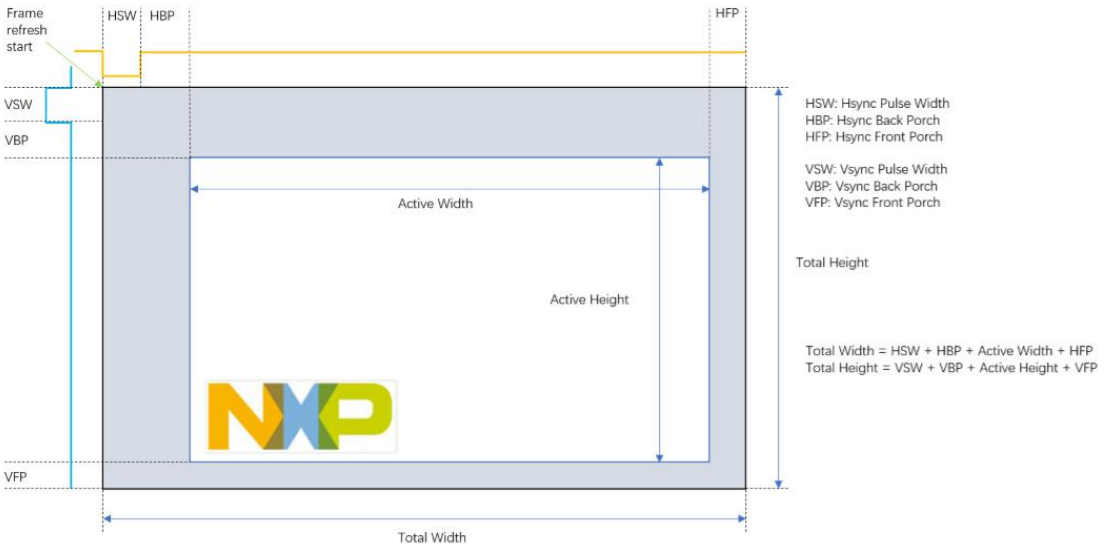

图 2. 可配置时序参数和分辨率

表格 1. 480×272LCD 屏时序参数

Item		Symbol	Min.	Type	Max.	Unit	Remarks
DCLK frequency		Fclk	5	9	12	MHz	—
DCLK period		Fclk	83	110	200	Ms	—
Hsync	Period time	Th	490	531	605	DCLK	—
	Display period	Thdisp	—	480	—	DCLK	—
	Back porch	Thbp	8	43	—	DCLK	By H_BLANKING setting
	Front porch	Thfp	2	8	—	DCLK	—
	Pulse width	Thw	1	—	—	DCLK	—
Vsync	Period time	Tv	275	288	335	H	—
	Display period	Tvdisp	—	272	—	H	—
	Back porch	Tvbp	2	12	—	H	By V_BLANKING setting
	Front porch	Tvfp	1	4	—	H	
	Pulse width	Tvw	1	10	—	H	

图 3 说明了一整个帧的处理流程。

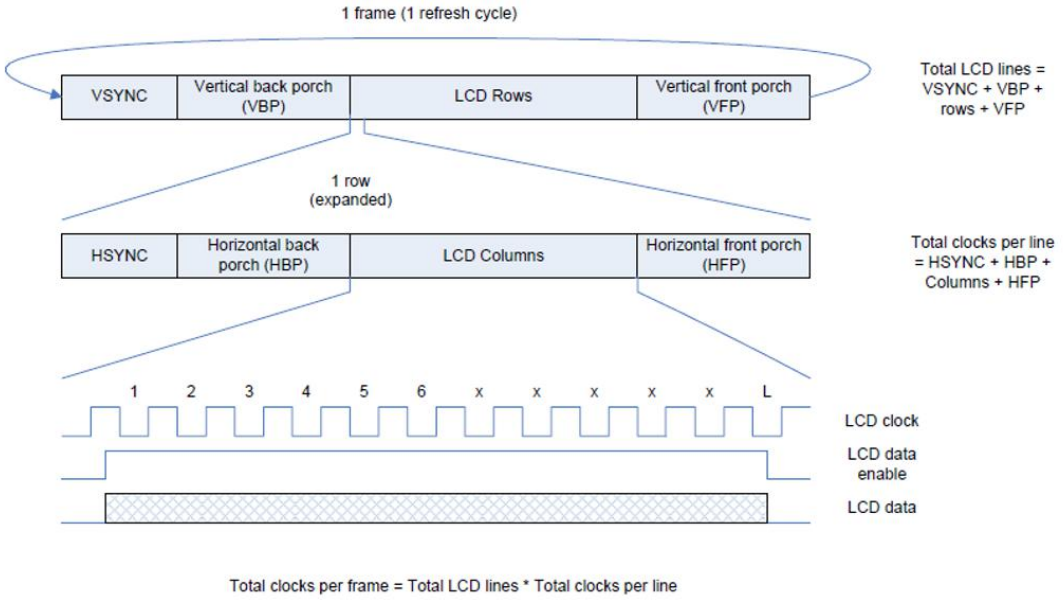

图 3. 帧和行时序参数

2.3. LCD 显示系统

一个基本的嵌入式显示系统由 MCU、帧缓冲区、显示控制器和显示屏组成。

- MCU 计算要在帧缓冲区中显示的图像/GUI。根据 MCU 计算能力，帧缓冲区图像速度越快，图像显示越流畅。
- 帧缓冲区是一个用于存储图像/GUI 像素数据的存储空间。这个存储空间通常被叫做帧缓冲区。帧缓冲区的大小取决于 LCD 分辨率和显示的颜色深度。使用双帧缓冲区来避免破坏当前的显示数据。
- 显示控制器不断刷新 LCD 屏，并将帧缓冲区的内容以每秒 60 次（60Hz）的频率传输到 LCD 屏上。显示控制器可以内嵌在显示模块或 MCU 中。I.MX RT MCU 包含一个名为 eLCDIF 的显示控制器。
- 显示屏由显示控制器驱动，并显示图像/GUI。显示屏的特征如下:
  - 显示屏尺寸
    显示屏的尺寸由水平（像素数）乘以垂直（行数）的像素数来定义的。
  - 颜色深度
    颜色深度决定了一个像素可以显示的颜色数量。它以每像素位数（bpp）表示。对于 24bpp 的色深(也可以表示为 RGB888)，一个像素可以显示 16777216 种颜色。
  - 刷新率（以 Hz 为单位）
    刷新率是每秒钟刷新显示屏的次数。因为较低的刷新率会导致不良的视觉效果，所以显示屏应每秒刷新 60 次（60Hz）。

图 4 展示了基本的 LCD RGB 模式系统。

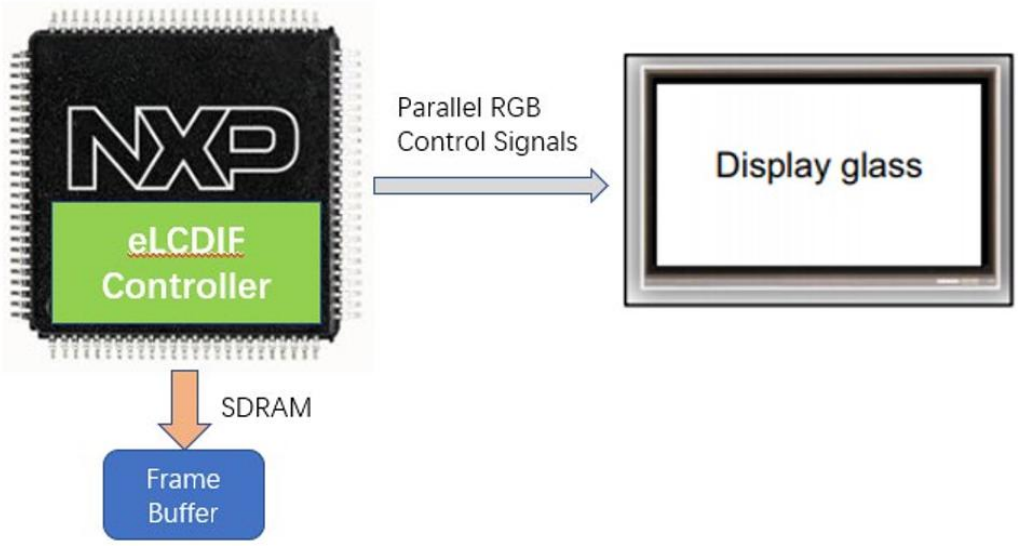

图 4. LCD RGB 基本系统

对于 LCD 显示系统，帧缓冲区是关键。它影响着系统的性能和视觉效果。

- 为帧缓冲区分配的内存空间通常与其他系统设备（CPU 内核，DMA，network 等）共享
- 帧缓冲区的数据被组织为位，字节，半字或字的数组，具体取决于选择的颜色深度和颜色位组织
- 使用（列×行×（色深）的大小）来计算缓冲区的大小
  - 例如
    800×600@16bpp（半字 RGB565）=800 列×600 行×2 字节/像素=96000 字节
  - 24 位数据存储在一个 32 位字段中（扔掉每个像素的高字节）。

为了实现显示兼容性，最重要的步骤就是确定帧缓冲区的内存大小及其位置。对于双帧缓冲区配置，所需

的帧缓冲区的大小也增加了一倍。通常使用双缓冲区配置，其中一个缓冲区用于存储当图像，而另一个用于准备存储下一个图像。

表 2 显示了具有不同像素颜色格式的标准屏幕分辨率的帧缓冲区大小。

表 2. 不同屏幕分辨率的帧缓冲区大小

Screen resolution	Number of pixels	Framebuffer size (Kbyte)			
		8 bpp	16 bpp	24 bpp	32 bpp
QVGA (320 × 240)	76800	75	150	225	300
Custom (480 × 272)	130560	128	255	383	510
HVGA (480 × 320)	153600	150	300	450	600
VGA (640 × 480)	307200	300	600	900	1200
WVGA (800 × 480)	384000	375	750	1125	1500
SVGA (800 × 600)	480000	469	938	1407	1875
XGA (1024 × 768)	786432	768	1536	2304	3072
HD (1280 × 720)	921600	900	1800	2700	3600

注意

i.MX RT eLCDIF 控制器可以支持 1280×800HD LCD 显示屏以 60fps 的速度运行

帧缓冲区通常以 RGB 格式存储像素数据。

- 一个像素数据由红，绿，蓝三色组成：
  - RGB332->8bpp(红色 3，绿色 3，蓝色 2)，在内存中以字节形式组织为（RRR GGG BB）
  - RGB555->16bpp(红色 5, 绿色 5, 蓝色 5), 在内存中以半字形式组织为(U RRRRR GGGGG BBBBB)
  - RGB565->16bpp(红色 5, 绿色 6, 蓝色 5), 在内存中以半字形式组织为(RRRRR GGGGGG BBBBB)
  - RGB888->24bpp(红色 8，绿色 8，蓝色 8)，在内存中以字形式组织为（UUUUUUUU RRRRRRRR GGGGGGGG BBBBBBB）(U=未使用)

3. i.MX RT eLCDIF 控制器

i.MX RT 包含增强液晶显示接口（eLCDIF）控制器。该模块支持 RGB（DOTCLK）模式接口和和可编程功能，包括：

- 系统总线主控器，用于获取帧缓冲区的数据以进行显示刷新；
- 用于不同颜色格式的 8/16/18/24 位 LCD 数据总线；
- 用于各种 LCD 分辨率的可配置时序和参数。

3.1. 基本特征和硬件连接

eLCDIF 控制器使用三个时钟域，如图 5 所示。

- AHB 时钟域
  AHB 时钟域用于将数据从内存传输到 FIFO。
- APB 时钟域
  APB 时钟域用于访问配置和状态寄存器。
- 像素时钟域
  像素时钟域用于生成 LCD 接口信号。

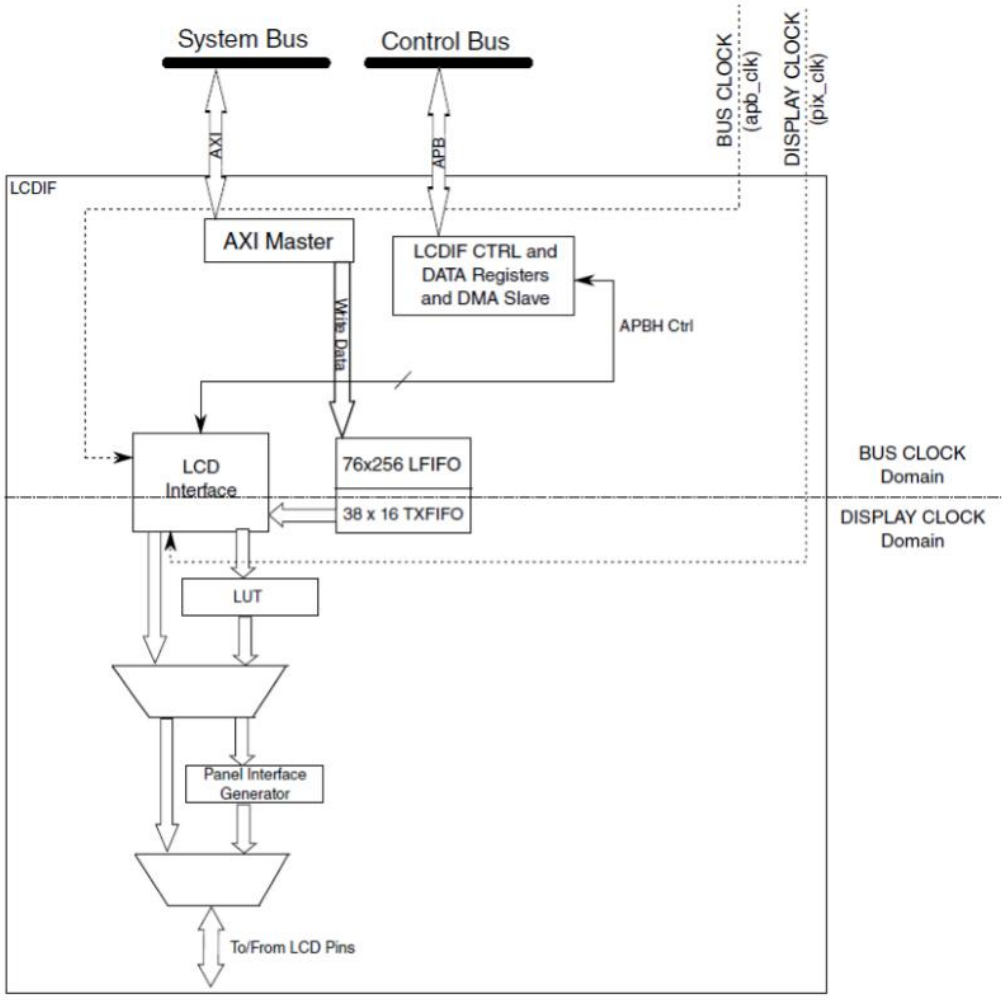

图 5.eLCDIF 控制器顶层模块

根据时序和硬件接口的灵活性，eLCDIF 控制器能驱动具有不同分辨率和信号极性的显示器。
为了驱动 LCD 显示器，eLCDIF 使用简单的 3.3V 信号提供了高达 29 个信号。这些信号包括：

- 像素时钟：LCD_DOTCLK。
- 数据使能：LCD_ENABLE。
- 重置：LCD_RESET。
- 同步信号：LCD_HSYNG 和 LCD_VSYNC。
- 像素数据：RGB888。

表 3. eLCDIF 接口输出信号

eLCDIF signal	Description
LCD_DOTCLK	The LCD_DOTCLK acts as the data valid signal for the LCD. The data is considered by the display only on the LCD_DOTCLK rising or falling edge.
LCD_HSYNC	The line synchronization signal (LCD_HSYNC) manages horizontal line scanning and acts as line display strobe.
LCD_VSYNC	The frame synchronization signal (LCD_VSYNC) manages vertical scanning and acts as a frame update strobe.
LCD_ENABLE	The LCD_ENABLE signal indicates to the LCD that the data in the RGB bus is valid and must be latched drawn.
LCD_RESET	The LCD_RESET signal is used to reset the LCD or touch panel.
Pixel RGB data	The eLCDIF interface can be configured to output more than one color depth. It can use up to 24 data lines (RGB888) as display interface bus.

eLCDIF 控制信号的极性是可配置的，从而使得 i.MX RT 可以驱动任意 RGB 并行显示屏。可以通过 eLCDIF 寄存器将控制信号（Hsync，Vsync 和数据使能 LCD_ENABLE）以及像素时钟（LCD_DOTCLK）定义为高电平有效或低电平有效。
LCD 屏数据信号并非与 eLCDIF 接口信号一一匹配：

- LCD 屏未使用的信号可以接地，从而使接口信号线少于 LCD 屏；
- 对于具有比 LCD 屏更多信号的接口，eLCDIF 接口信号保持未使用状态；
- 使用了非标准的颜色映射。

图 6 展示了一个 18bpp LCD 屏的 24 位 RGB888 接口的示例。

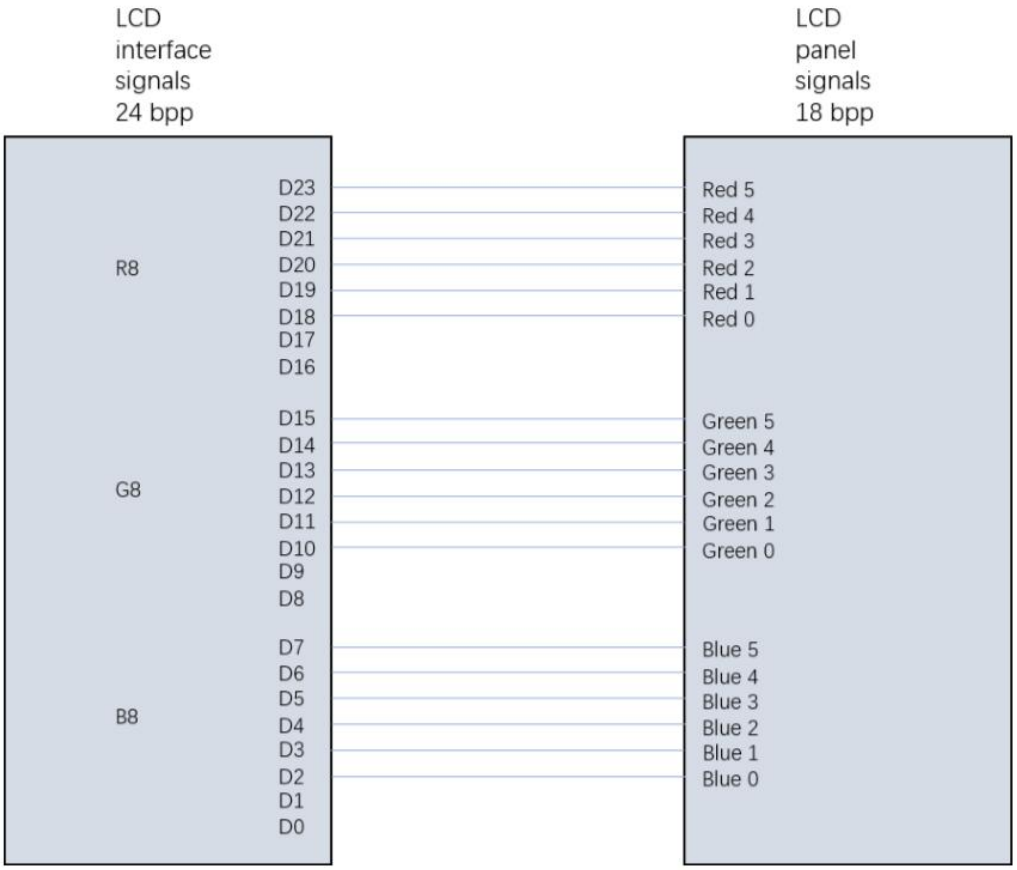

图 6. 18bpp LCD 的 24 位 RGB888 接口

表 4 列出了最佳性能的时序参数。

表 4. eLCDIF 时序参数

Parameter	Symbol	Min.	Max.	Unit
LCD pixel clock frequency	tCLK (LCD)	—	75	MHz
LCD pixel clock high (falling edge capture)	tCLKH (LCD)	3	—	ns
LCD pixel clock low (rising edge capture)	tCLKL (LCD)	3	—	ns
LCD pixel clock high to data valid (falling edge capture)	td (CLKH-DV)	-1	1	ns
LCD pixel clock low to data valid (rising edge capture)	td (CLKL-DV)	-1	1	ns
LCD pixel clock high to control signal valid (falling edge capture)	td (CLKH-CTRLV)	-1	1	ns
LCD pixel clock low to control signal valid (rising edge capture)	td (CLKL-CTRLV)	-1	1	ns

除上述信号外，通常显示屏接口还包含其他信号，这些信号不属于表 3 中所述的 eLCDIF 信号。这些附加信号是显示模块要发挥全部功能所必需的。eLCDIF 控制器只能驱动表 3 中所述的信号。不属于 eLCDIF 的信号可以使用 GPIO 进行控制，而其他外设则需要特定的电路。显示屏通常嵌入一个背光单元，该背光单元需要一个额外的背光控制电路和一个 GPIO 进行控制。某些显示屏需要串行接口，例如 I2C 或 SPI 用于实现触摸功能。

eLCDIF 支持多种中断，以帮助控制和报告模块状态：

- 模块处于活动时，当时钟域交叉 FIFO（TXFIFO）变空时，将触发下溢中断。 软件应采取纠正措施以防止该事件发生。
- Cur_frame_done 中断发生在每个帧的结束。

此外，eLCDIF 提供了两个 256×24 位的查找表（LUT），以匹配更多类型的 LCD。帧缓冲区的颜色深度和 LCD 屏的颜色深度不需要相同。查找表（LUT）可用于将小的子集映射到 LCD 屏的颜色值。

图 7 是一个查找表示例，该示例将 8 色 RGB 映射到 RGB565。该示例表假定红色，绿色和蓝色只能被关闭或者变为全强度。

Frame buffer value	Lookup table value (16-bit RGB565 output)	System color	
R0, G0, B0 (0)	0x0000	black	
R0, G0, B1 (1)	0x001F	blue	
R0, G1, B0 (2)	0x07E0	green	
R0, G1, B1 (3)	0x07FF	cyan	
R1, G0, B0 (4)	0xF800	red	
R1, G0, B1 (5)	0xF81F	magenta	
R1, G1, B0 (6)	0xFFE0	yellow	
R1, G1, B1 (7)	0xFFFF	white	

图 7. 3bpp 颜色（8 种颜色）映射到 16 位 RGB565 输出

3.2. NXP LCD 扩展板

NXP LCD 扩展板是为 i.MX RT EVK 板创建的。它可以支持连接到 i.MX RT EVK 板的各种 LCD 屏。EVK 板导出 i.MX RT 的 eLCDIF 端口用于特定演示，并且由于不同的 LCD 屏的端口设计不同，该端口不能用于所有 LCD 屏。

LCD 扩展板可以支持:

- 外部电源和不同的电源开关;
- i.MX RT EVK 板的标准接口;
- 灵活切换 LCD RGB 模式信号;
- 触摸屏;
- MIPI 端口。

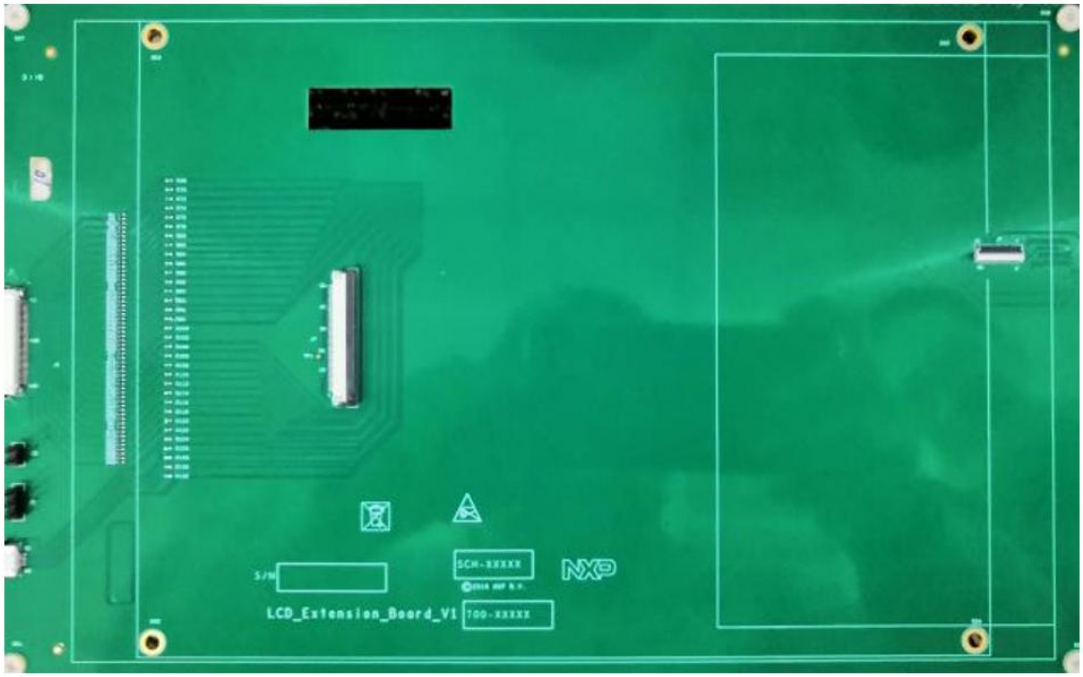

图 8. NXP LCD 扩展板

4. eLCDIF 例程和性能调整

在 i.MX RT SDK 软件包中有一个名为 sd_jpeg 的工程。该工程可以演示 eLCDIF 控制器的功能和性能调整。请根据您的开发环境从 NXP 官方网站下载 SDK 软件包。

4.1. sd_jpeg 工程

sd_jpeg 例程可以在 i.MX RT1050 SDK 的 boards \ evkbimxrt1050 \ demo_apps \ sd_jpeg 目录中找到。该例程从 SD 卡中读取 JPEG 图片，对其进行解码，然后在 LCD 屏中逐一显示。有关运行例程的详细步骤，请参考目录中的 readme.txt。

为了理解 eLCDIF 例程并修改源代码以驱动 1280×800 HD LCD 屏，以下各节介绍了 **i.MX RT1050EVKB + NXP LCD 扩展板+ 1280×800 LCD** 屏环境。工具链以 IAR 为例。

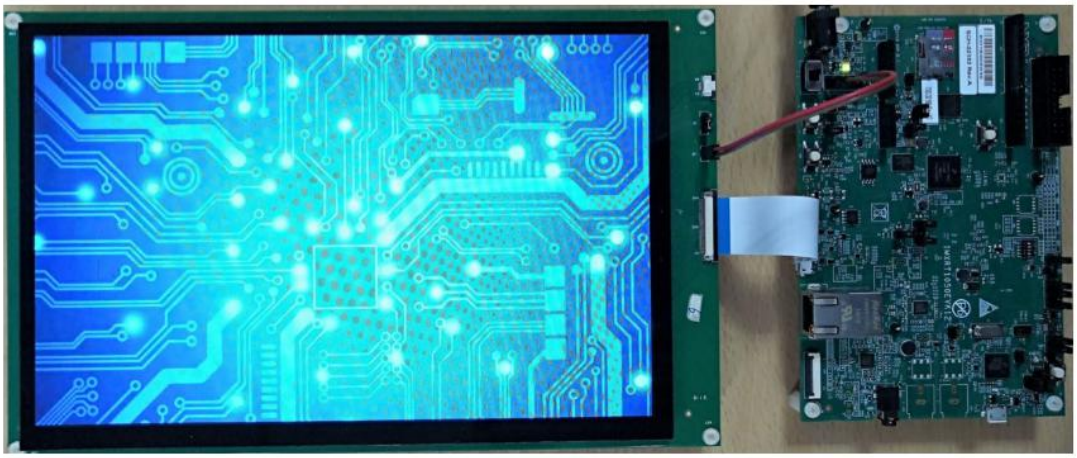

图 9. i.MX RT1500EVKB + NXP LCD 扩展板 + 1280 × 800 LCD 屏

原始 sd_jpeg 代码是适配 480×272 LCD 屏的。要修改以适配 1280×800 LCD 屏，需进行以下更改。

- 在 sd_jpeg.c 中配置 1280×800 LCD 时序参数。
  这些更改定义了 LCD 尺寸，HSW / HBP / HFP / VSW / VBP / VFP 时序参数和信号极性。

```
#define APP_IMG_WIDTH  1280
#define APP_IMG_HEIGHT  800
#define APP_HSW 10
#define APP_HBP 80
#define APP_HFP 70
#define APP_VSW 3
#define APP_VBP 10
#define APP_VFP 10
#define APP_POL_FLAGS \
  (kELCDIF_DataEnableActiveHigh | kELCDIF_VsyncActiveLow | kELCDIF_HsyncActiveLow |
kELCDIF_DriveDataOnRisingClkEdge)
```

- 在 sd_jpeg.c 中配置正确的复位和背光 GPIO 信号。

```
/* Display. */
#define LCD_DISP_GPIO GPIO1
#define LCD_DISP_GPIO_PIN 2
/* Back light. */
#define LCD_BL_GPIO GPIO2
#define LCD_BL_GPIO_PIN 31
```

- 利用 sd_jpeg.c 中的 RGB 模式结构体初始化时序参数，帧缓冲区，颜色格式和数据总线格式等。

```
const elcdif_rgb_mode_config_t config = {
        .panelWidth = APP_IMG_WIDTH,
        .panelHeight = APP_IMG_HEIGHT,
        .hsw = APP_HSW,
        .hfp = APP_HFP,
        .hbp = APP_HBP,
        .vsw = APP_VSW,
        .vfp = APP_VFP,
        .vbp = APP_VBP,
        .polarityFlags = APP_POL_FLAGS,
        .bufferAddr = (uint32_t)g_frameBuffer[0],
        .pixelFormat = kELCDIF_PixelFormatRGB888,
        .dataBus = APP_LCDIF_DATA_BUS,
};
```

- 根据 1280×800 LCD 时序参数计算和配置 sd_jpeg.c 中的像素时钟（DOTCLK）。

```
void BOARD_InitLcdifPixelClock(void)
{
    /*
     * The desired output frame rate is 60Hz. So the pixel clock frequency is:
     * (1280 + 10 + 80 + 70) * (800 + 3 + 10 + 10) * 60 = 71M.
     * Here set the LCDIF pixel clock to 70.5M.
     */


    /*
     * Initialize the Video PLL.
     * Video PLL output clock is OSC24M * (loopDivider + (denominator / numerator)) /
postDivider = 70.5MHz.
     */
    clock_video_pll_config_t config = {
        .loopDivider = 47, .postDivider = 16, .numerator = 0, .denominator = 0,
    };

    CLOCK_InitVideoPll(&config);

    /*
     * 000 derive clock from PLL2
```

```
     * 001 derive clock from PLL3 PFD3
     * 010 derive clock from PLL5
     * 011 derive clock from PLL2 PFD0
     * 100 derive clock from PLL2 PFD1
     * 101derive clock from PLL3 PFD1
     */
    CLOCK_SetMux(kCLOCK_LcdifPreMux, 2);

    CLOCK_SetDiv(kCLOCK_LcdifPreDiv, 0);

    CLOCK_SetDiv(kCLOCK_LcdifDiv, 0);
}
```

- 在 sd_jpeg.c 中配置 eLCDIF AXI 寄存器以提高 1280×800 LCD 的性能。

```
# In the bottom of function "void APP_ELCDIF_Init(void)", add the line:

APP_ELCDIF->CTRL2 = 0x00700000;
```

- 修改 IAR icf 文件，以在 *MIMXRT1052xxxxx_sdram.icf* 中获得更大的 1280×800 LCD 帧缓冲区。对于具有 RGB888 彩色格式的 1280×800 LCD，可以将帧缓冲区放入 i.MX RT1050 的外部 SDRAM 存储空间，从而扩展图像的栈和堆大小，以便对 1280×800 图片进行软件解码。

```
--- a/boards/evkbimxrt1050/demo_apps/sd_jpeg/iar/MIMXRT1052xxxxx_sdram.icf
+++ b/boards/evkbimxrt1050/demo_apps/sd_jpeg/iar/MIMXRT1052xxxxx_sdram.icf
@@ -73,13 +73,13 @@ define symbol m_ncache_end           = 0x81FFFFFF;
  if (isdefinedsymbol(__stack_size__)) {
define symbol __size_cstack__       = __stack_size__;
} else {
-  define symbol __size_cstack__       = 0x1000;
+  define symbol __size_cstack__       = 0x8000;
}

if (isdefinedsymbol(__heap_size__)) {
 define symbol __size_heap__         = __heap_size__;
} else {
-  define symbol __size_heap__         = 0x20000;
+  define symbol __size_heap__         = 0x800000;
}
```

在对源代码进行上述修改之后，在正确的硬件环境下，SD 卡中的 1280×800 图片可以在 1280×800 液晶显示屏上流畅显示。

4.2. 提高 eLCDIF 性能

较大的 LCD 屏需要较大的帧缓冲区，因此对于相同的刷新率（例如 60 fps），eLCDIF 需要提供更快的像素时钟和更好的系统内存总线带宽。

例如，一个 1280×800 像素的显示器使用 24 位颜色并以 60 Hz 刷新，因此：

Byte/sec for LCD = 60×1280×800×3 bytes/pixel = 184.32Mbytes/sec

eLCDIF 模块的性能可以从以下几点进行调整：

- 在 LCDIF_CTRL2 寄存器中配置突发长度和连续请求个数。
  - LCDIF_CTRL2_OUTSTANDING_REQS 字段控制在任何给定的时钟周期内 eLCDIF 可以处理多少个请求。
  - LCDIF_CTRL2_BURST_LEN_8 位将每个 eLCDIF 系统总线请求的 64 位字请求数设置为 8 或 16 个 QWORDS。
- 更改 LCDIF_THRES 寄存器中的 LCDIF_THRES_PANIC 值，该值可用于优化总线吞吐量和功耗。LCDIF_THRES_PANIC 值可用于提高 eLCDIF 发起的请求的优先级，以更改系统总线基础结构对 eLCDIF 请求进行仲裁的方式。LCDIF_THRES 寄存器提供的功能需要系统时钟和动态优先级控制的支持。要评估系统对这些功能的支持，请参考相应的模块文档。

5. 结论

本应用笔记描述了 LCD 显示屏的背景知识以及 i.MX RT eLCDIF 的使用用例。更多信息请参考：

- i.MX RT1050 Processor Reference Manual
- The `readme.txt` in the SDK *`sd_jpeg`* demo