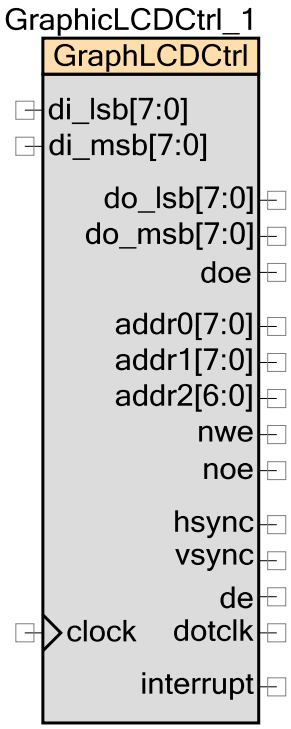
图形 LCD 控制器 (GraphicLCDCtrl)

1.61



特性

- 完全可编程的屏幕尺寸，支持最高达 HVGA 分辨率，包括：
  - QVGA (320x240) @ 60 Hz 16 bpp
  - WQVGA (480x272) @ 60 Hz 16 bpp
  - HVGA (480x320) @ 60 Hz 16 bpp
- 支持虚拟屏幕运行
- 配有 SEGGER emWin 图形库的接口
- 可在 blanking 间隔期间执行读写操作
- 生成发送到面板的连续时序信号，而无需 CPU 干预
- 最高支持 23 位地址、16 位数据的异步 SRAM 用作外部帧缓冲器
- 在横向和纵向 blanking 间隔的开始和结束时生成可选中断脉冲

概述

图形 LCD 控制器 (GraphicLCDCtrl) 组件提供了连接 LCD 面板的接口，此 LCD 应具有 LCD 驱动器但无 LCD 控制器。这种类型的面板不含帧缓冲器。帧缓冲器必须由外部提供。

此组件还可以连接至使用 16 位宽的异步 SRAM 器件实现的外部提供的帧缓冲器。

此组件旨在与 SEGGER emWin 图形库配合使用。此图形库由赛普拉斯提供，用于与赛普拉斯器件配合使用，可访问赛普拉斯网站 www.cypress.com/go/comp_emWin 获取此图形库。此图形库提供一组功能齐全的图形函数，用于绘制和渲染文本和图像。

何时使用 GraphicLCDCtrl

GraphicLCDCtrl 组件支持许多 LCD 面板。它可以直接驱动控制信号和管理外部 SRAM 中的帧缓冲器。此组件可生成 dotclk、hsync、vsync 和 de 输出以访问 SRAM 中的数据并使用 将数据显示在 LCD 上。

仅当帧缓冲器 SRAM 未刷新 LCD 面板时，才可访问帧缓冲器 SRAM 进行读取和写入操作。如果在刷新周期提出了读取或写入请求，提供的 API 函数将一直等待，直至刷新进入消隐信号周期。在消隐信号周期中，将完成读取或写入操作。

中断可用于指示进入或退出消隐信号周期。当与 RTOS 相结合时，这一点尤其有用，在进入和退出消隐信号周期后，可换入或换出需要访问帧缓冲器的任务。

输入/输出连接

本节介绍 GraphicLCDCtrl 组件的输入和输出连接。某些 I/O 在其描述中所列的条件下可能不在符号上显示。

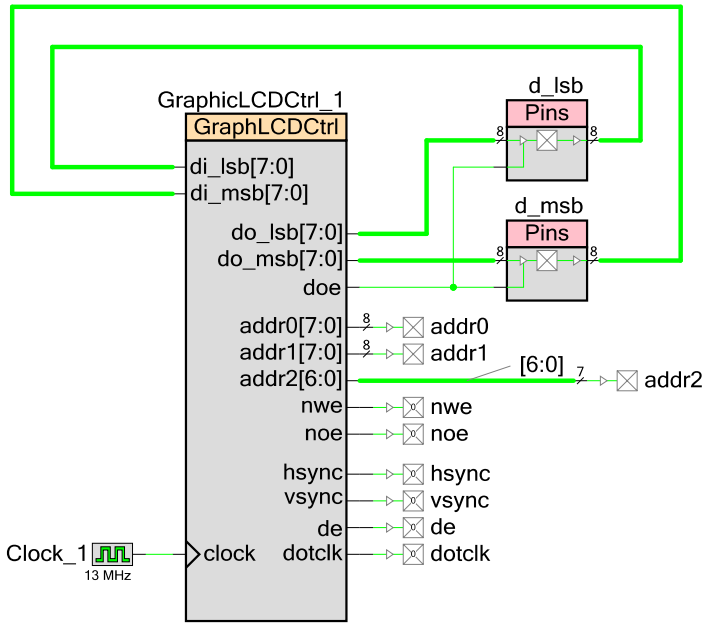
输入	可能已隐藏	说明
di_lsb[7:0]	N	输入总线的低八位。在读取数据操作过程中，它们用于数据。将这些信号连接到器件的输入引脚上，并禁用这些引脚的“Input Synchronized”（同步输入）选项。信号自身就是同步的，因为是基于同步输出信号驱动它们的。
di_msb[7:0]	N	输入总线的高八位。在读取数据操作过程中，它们用于数据。将这些信号连接到器件的输入引脚上，并禁用这些引脚的“Input Synchronized”（同步输入）选项。信号自身就是同步的，因为是基于同步输出信号驱动它们的。
clock	N	操作此组件的时钟。它的频率是 dotclk 的频率的两倍。

输出	可能已隐藏	说明
do_lsb[7:0]	N	输出数据总线的低八位。在写入数据操作过程中，它们用于数据。
do_msb[7:0]	N	输出总线的高八位。在写入数据操作过程中，它们用于数据。
doe	N	PSoC 中数据总线组件的输出使能。它通常连接到数据总线的输入/输出引脚组件的输出使能上。
addr0[7:0]	N	连接到帧缓冲器的地址总线的最低八位。
addr1[7:0]	N	连接到帧缓冲器的地址总线的中间八位。
addr2[6:0]	N	连接到帧缓冲器的地址总线的高八位。帧缓冲器所需的数据位数取决于所使用的 SRAM 器件。
nwe	N	帧缓冲器 SRAM 的低电平有效写入使能。
noe	N	帧缓冲器的低电平有效输出使能。
de	N	面板的数据使能。
hsync	N	面板的水平同步时序信号。

输出	可能已隐藏	说明
vsync	N	面板的垂直同步时序信号。
dotclk	N	驱动至面板的时钟。此时钟的频率是输入时钟的频率的一半。
中断	Y	边沿触发的中断信号。如果选择了不生成中断，此输出将处于隐藏状态。

原理图宏信息

宏使用默认设置配置，连接 Optrex T-55343GD035JU-LW-AEN 面板。宏中包括的时钟设置为 13 MHz，这使得向 Optrex QVGA 面板提供 6.5 MHz dotclk。



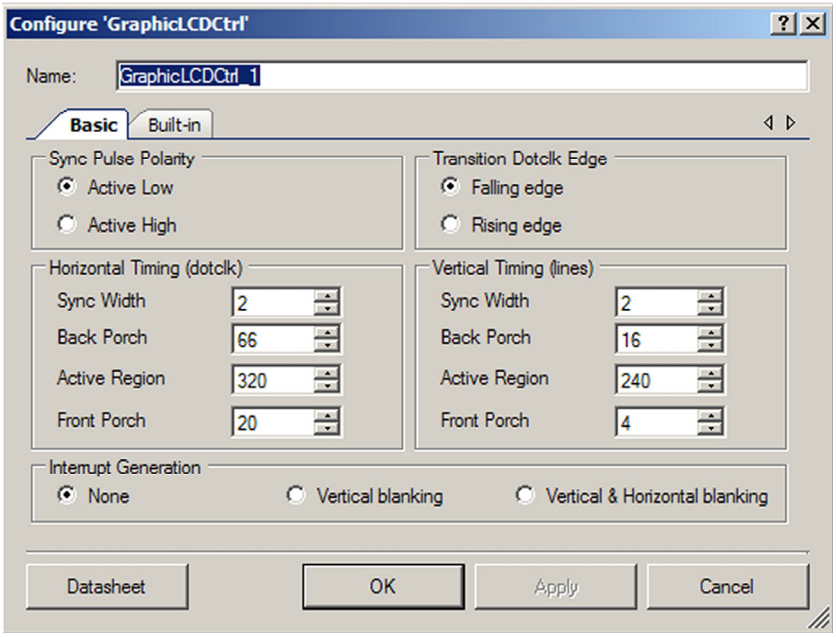
通常，高地址位 (addr2) 中仅有一些位用于连接到帧缓冲器。具体位数取决于所使用的 SRAM 的大小。根据需要的地址位数，可以使用以下步骤调整此总线的大小。

- 配置 addr2 输出引脚组件并设置“Number of Pins”（引脚数量）。
- 右键单击驱动输出引脚的信号，并选择“Edit Name And Width”（编辑名称和宽度）。然后，调整“Left Index”（左索引）以反映输出引脚的宽度。

取消选择所有数据引脚上的“Input Synchronized”（同步输入）选项，并关闭所有引脚的 API 函数的生成。

元件参数

将一个 GraphicLCDCtrl 组件拖放到您的设计上，并双击以打开 **Configure**（配置）对话框。GraphicLCDCtrl 的默认设置用于对 Optrex 面板的操作。



Configure GraphicLCDCtrl（配置 GraphicLCDCtrl）对话框包含以下参数。所有这些设置都是编译时选择；运行时，不需要更改这些设置。它们都是所使用的面板和帧缓冲器 SRAM 的特性。

Sync Pulse Polarity（同步脉冲极性）

基于此设置，hsync 信号和 vsync 信号是 **Active High**（高电平有效）（生成的脉冲是高脉冲）还是 **Active Low**（低电平信号）的。此选择可控制 hsync 极性和 vsync 极性。默认设置为 **Active Low**（低电平有效）。

切换 Dotclk 边缘

Dotclk 是发送到面板的时钟，面板由此时钟供电运转。当切换设置为 **Rising edge**（上升沿），所有关联的信号（例如 hsync、vsync 和 de）将在 dotclk 的上升沿上切换。当设置为 **Falling edge**（下降沿）时，他们将在 dotclk 的下降沿上切换。

通常，如果面板指定了到 dotclk 的一个边沿的建立和保持时间，应将此设置配置为此时钟的另一个边沿。这提供了大约设置和保持的时钟周期的一半周期。默认设置为 **Falling edge**（下降沿）。

Horizontal Timing(dotclk)（水平时序 (dotclk)）

- **Sync Width**（同步宽度）— 定义 dotclk 中的水平同步宽度。此值的范围为 1 到 256 个时钟周期。默认设置为 **2**。
- **Back Porch**（后沿）— 定义 dotclk 中的水平后沿宽度。此值的范围为 6 到 256 个时钟周期。最小值 6 向状态机提供了早期足够的指示，以防止在活动的屏幕区域开始之前存在无法完成的读取或写入访问。一些面板规范将后沿作为同步信号的结尾与活动区域的开头之间的区域进行衡量。另一些面板规范则将后沿作为从同步脉冲的开头到活动区域的开头之间的区域。此组件将后沿作为同步脉冲的结尾到活动区域的开头之间的周期进行衡量。默认设置为 **66**。
- **Active Region**（活动区域）— 定义 dotclk 中的水平活动区域。活动区域是使用为 4 的倍数的设置实现的。当仅使用 8 位计数器时，这允许区域最大为 1024 x 1024。所有常见的屏幕尺寸在两个方向上都是 4 的倍数。此值的范围为 4 到 1024（必须是 4 的倍数）个时钟周期。默认设置为 **320**。
- **Front Porch**（前沿）— 定义 dotclk 中的水平前沿宽度。此值的范围为 1 到 256 个时钟周期。默认设置为 **20**。

Vertical Timing (lines)（垂直时序（线路））

- **Sync Width**（同步宽度）— 定义线路中的垂直同步宽度。此值的范围为 1 到 256 个时钟周期。默认设置为 **2**。
- **Back Porch**（后沿）— 定义线路中的垂直后沿宽度。此值的范围为 1 到 256 个线路。一些面板规范将后沿作为同步信号的结尾与活动区域的开头之间的区域进行衡量。另一些面板规范则将后沿作为从同步脉冲的开头到活动区域的开头之间的区域。此组件将后沿作为同步脉冲的结尾到活动区域的开头之间的周期进行衡量。默认设置为 **16**。
- **Active Region**（活动区域）— 定义线路中的垂直活动区域宽度。活动区域是使用为 4 的倍数的设置实现的。当仅使用 8 位计数器时，这允许区域最大为 1024 x 1024。所有常见的屏幕尺寸在两个方向上都是 4 的倍数。此值的范围为 4 到 1024（必须是 4 的倍数）个线路。默认设置为 **240**。
- **Front Porch**（前沿）— 定义线路中的垂直前沿宽度。此值的范围为 1 到 256 个线路。默认设置为 **4**。

Interrupt Generation（中断生成）

定义中断生成的设置。默认设置为 **None**（无）。

- 如果设置为 **Vertical blanking**（纵向 blanking），在纵向 blanking 间隔的开始和结束时都将生成一个中断脉冲。

- 如果设置为 **Vertical & Horizontal blanking**（纵向和横向 blanking），在每个活动区域的开始和结束时都将生成一个中断脉冲。

  在活动纵向区域期间，每条线路的活动区域的开始和结束时都有一个中断。对于纵向 blanking 区域，最后一个活动线路结束时有一个中断，第一个活动线路开始时有另一个中断。

时钟选择

此组件中没有内部时钟。您必须附加时钟源。提供的时钟频率必须是输出 dotclk 时钟到面板的所需时钟频率的两倍。

放置

GraphicLCDCtrl 组件放置于整个 UDB 阵列中，并且所有放置信息通过 *cyfitter.h* 文件提供给 API。

资源

资源类型				API Memory（API 存储器）（字节）		Pins（引脚）（每个外部 I/O）
数据路径单元	PLD	状态单元	Control/Count7 单元	Flash（闪存）	RAM	
7	11	1	1	421	6	45

应用程序编程接口

应用程序编程接口 (API) 子程序允许您使用软件配置组件。下表列出了每个函数的接口，并进行了说明。以下各节将更详细地介绍每个函数。

默认情况下，PSoC Creator 将实例名称“GraphicLCDCtrl_1”分配给指定设计中组件的第一个实例。您可以将该实例重命名为符合标识符语法规则的任意唯一值。实例名称会成为每个全局函数名称、变量和常量符号的前缀。出于可读性考虑，下表中使用的实例名称为“GraphicLCDCtrl”。

函数	说明
GraphicLCDCtrl_Start()	启动 GraphicLCDCtrl 接口。
GraphicLCDCtrl_Stop()	禁用 GraphicLCDCtrl 接口。
GraphicLCDCtrl_Write()	启动对帧缓冲器的写入数据操作。
GraphicLCDCtrl_Read()	启动从帧缓冲器中的读取数据操作。

函数	说明
GraphicLCDCtrl_WriteFrameAddr()	设置在刷新屏幕时使用的启动帧缓冲器地址。
GraphicLCDCtrl_ReadFrameAddr()	读取在刷新屏幕时使用的启动帧缓冲器地址。
GraphicLCDCtrl_WriteLineIncr()	设置相邻线路之间的地址间距。
GraphicLCDCtrl_ReadLineIncr()	读取线路之间的地址增量。
GraphicLCDCtrl_Sleep()	保存配置，并禁用 GraphicLCDCtrl。
GraphicLCDCtrl_Wakeup()	恢复配置，并启用 GraphicLCDCtrl。
GraphicLCDCtrl_Init()	初始化或将组件参数恢复为随组件定制器提供的设置。
GraphicLCDCtrl_Enable()	启用 GraphicLCDCtrl。
GraphicLCDCtrl_SaveConfig()	保存 GraphicLCDCtrl 的配置。
GraphicLCDCtrl_RestoreConfig()	恢复 GraphicLCDCtrl 的配置。

全局变量

变量	说明
GraphicLCDCtrl_initVar	GraphicLCDCtrl_initVar 指示是否已初始化图形 LCD 控制器。变量将初始化为 0，并在第一次调用 GraphicLCDCtrl_Start() 时设置为 1。这样，第一次调用 GraphicLCDCtrl_Start() 子程序后，组件不用重新初始化即可重启。 如果需要重新初始化组件，则应在调用 GraphicLCDCtrl_Start() 或 GraphicLCDCtrl_Enable() 函数之前先调用 GraphicLCDCtrl_Init() 函数。

void GraphicLCDCtrl_Start(void)

说明： 根据需要，此函数启用活动模式电源模板位或时钟。这是开始执行组件操作的首选方法。GraphicLCDCtrl_Start() 将 initVar 设置为1，调用 GraphicLCDCtrl_Init() 函数，然后调用 GraphicLCDCtrl_Enable() 函数。

参数： None（无）

Return Value（返回值）： None（无）

Side Effects（副作用）： 如果已设置 initVar 变量，则该函数仅调用 GraphicLCDCtrl_Enable() 函数。

void GraphicLCDCtrl_Stop(void)

说明： 根据需要，此函数禁用活动模式电源模板位或时钟关断。

参数： None（无）

Return Value（返回值）： None（无）

Side Effects（副作用）： None（无）

void GraphicLCDCtrl_Write(uint32 addr, uint16 data)

说明： 此函数使用提供的地址和数据启动对帧缓冲器的写入数据操作。此命令将被列入命令队列，因此，在接口上实际完成写入之前，此函数将返回。如果命令队列已满，此函数不返回，直至有空间将此请求写入队列

参数： addr：要在组件的地址行上发送的地址（addr2[6:0]、addr1[7:0]、addr0[7:0]）。

data：在 do_msb[7:0]（最高有效位）和 do_lsb[7:0]（最低有效位）引脚上发送的数据

Return Value（返回值）： None（无）

Side Effects（副作用）： None（无）

uint16 GraphicLCDCtrl_Read(uint32 addr)

说明： 此函数启动从帧缓冲器中的读取数据操作。在完成所有当前发布的写入操作之后执行读取操作。此函数等待直至读取完成，然后返回读取值。

参数： addr：要在组件的地址行上发送的地址（addr2[6:0]、addr1[7:0]、addr0[7:0]）

Return Value（返回值）： di_msb[7:0]（最高有效位）和 di_lsb[7:0]（最低有效位）引脚中的读取值

Side Effects（副作用）： None（无）

void GraphicLCDCtrl_WriteFrameAddr(uint32 addr)

说明： 此函数设置在刷新屏幕时使用的启动帧缓冲器地址。在各个纵向 blanking 间隔期间读取此寄存器。要实现此寄存器的原子更新，应在活动刷新区域期间写入此寄存器。

参数： addr：帧缓冲器起始地址

Return Value（返回值）： None（无）

Side Effects（副作用）： None（无）

uint32 GraphicLCDCtrl_ReadFrameAddr(void)

说明： 此函数读取在刷新屏幕时使用的启动帧缓冲器地址。

参数： None（无）

Return Value（返回值）： 帧缓冲器起始地址

Side Effects（副作用）： None（无）

void GraphicLCDCtrl_WriteLineIncr(uint32 incr)

说明： 此函数设置相邻线路之间的地址间距。默认情况下，这是线路的显示大小。此设置可用于将线路与不同的字边界对齐，或实现比显示区域大的虚拟线路长度。

参数： incr：线路之间的地址增量。必须至少为线路的显示大小。

Return Value（返回值）： None（无）

Side Effects（副作用）： None（无）

uint32 GraphicLCDCtrl_ReadLineIncr(void)

说明： 此函数读取线路之间的地址增量。

参数： None（无）

Return Value（返回值）： 线路之间的地址增量

Side Effects（副作用）： None（无）

void GraphicLCDCtrl_Sleep(void)

说明： 这是准备组件睡眠的首选子程序。GraphicLCDCtrl_Sleep() 子程序保存当前组件的状态。然后它调用 GraphicLCDCtrl_Stop() 函数，并调用 GraphicLCDCtrl_SaveConfig() 函数以保存硬件配置。

在调用 CyPmSleep() 或 CyPmHibernate() 函数之前调用 GraphicLCDCtrl_Sleep() 函数。有关电源管理函数的更多信息，请参考 PSoC Creator *System Reference Guide*（《系统参考指南》）。

参数： None（无）

Return Value（返回值）： None（无）

Side Effects（副作用）： None（无）

void GraphicLCDCtrl_Wakeup(void)

说明： 该函数是将组件恢复到调用 GraphicLCDCtrl_Sleep() 时状态的首选子程序。GraphicLCDCtrl_Wakeup() 函数调用 GraphicLCDCtrl_RestoreConfig() 函数以恢复配置。如果组件在调用 GraphicLCDCtrl_Sleep() 函数前已启用，则 GraphicLCDCtrl_Wakeup() 函数还将重新启用组件。

参数： None（无）

Return Value（返回值）： None（无）

Side Effects（副作用）： 调用 GraphicLCDCtrl_Wakeup() 函数前未调用 GraphicLCDCtrl_Sleep() 或 GraphicLCDCtrl_SaveConfig() 函数可能会产生意外行为。

void GraphicLCDCtrl_Init(void)

说明： 此函数初始化或将组件参数恢复为随组件定制器提供的设置。将定义时序生成的编译时间配置恢复到随定制器提供的设置。帧缓冲器地址的运行时配置设为 0；对于线路增量，此配置设为显示线路大小。

参数： None（无）

Return Value（返回值）： None（无）

Side Effects（副作用）： 调用此函数之前，必须用 GraphicLCDCtrl_Stop() 函数禁用组件，否则组件的行为会出现异常。此函数将重新初始化组件，但以下部分将不被重新初始化：它不清除 FIFO 中的数据，且不复位组件的硬件状态。

void GraphicLCDCtrl_Enable(void)

说明： 此函数激活硬件并开始执行组件操作。无需调用 GraphicLCDCtrl_Enable()，因为 GraphicLCDCtrl_Start() 子程序会调用该函数，这是开始组件操作的首选方法。

参数： None（无）

Return Value（返回值）： None（无）

Side Effects（副作用）： None（无）

void GraphicLCDCtrl_SaveConfig(void)

说明： 此函数会保存组件配置和非保留寄存器。它还保存 Configure（配置）对话框中定义的或通过相应 API 修改的当前组件参数值。该函数由 GraphicLCDCtrl_Sleep() 函数调用。

参数： None（无）

Return Value（返回值）： None（无）

Side Effects（副作用）： None（无）

void GraphicLCDCtrl_RestoreConfig(void)

说明： 此函数会恢复组件配置和非保留寄存器。它还将组件参数值恢复为在调用 GraphicLCDCtrl_Sleep() 函数之前的值。

参数： None（无）

Return Value（返回值）： None（无）

Side Effects（副作用）： 如果在调用 GraphicLCDCtrl_SaveConfig() 之前调用 API，组件配置将恢复到其默认设置。帧缓冲器地址的运行时配置设为 0；对于线路增量，此配置设为显示线路大小。

固件源代码示例

PSoC Creator 在“查找示例项目”对话框中提供了很多包括原理图和代码示例的示例项目。要获取组件特定的示例，请打开组件目录中的对话框或原理图中的组件实例。要获取通用的示例，请打开 Start Page（开始页）或 **File**（文件）菜单中的对话框。根据需要，使用对话框中的 **Filter Options**（滤波器选项）可缩小可选项目的列表。

有关更多信息，请参见 PSoC Creator 帮助中的“Find Example Project（查找示例项目）”主题。

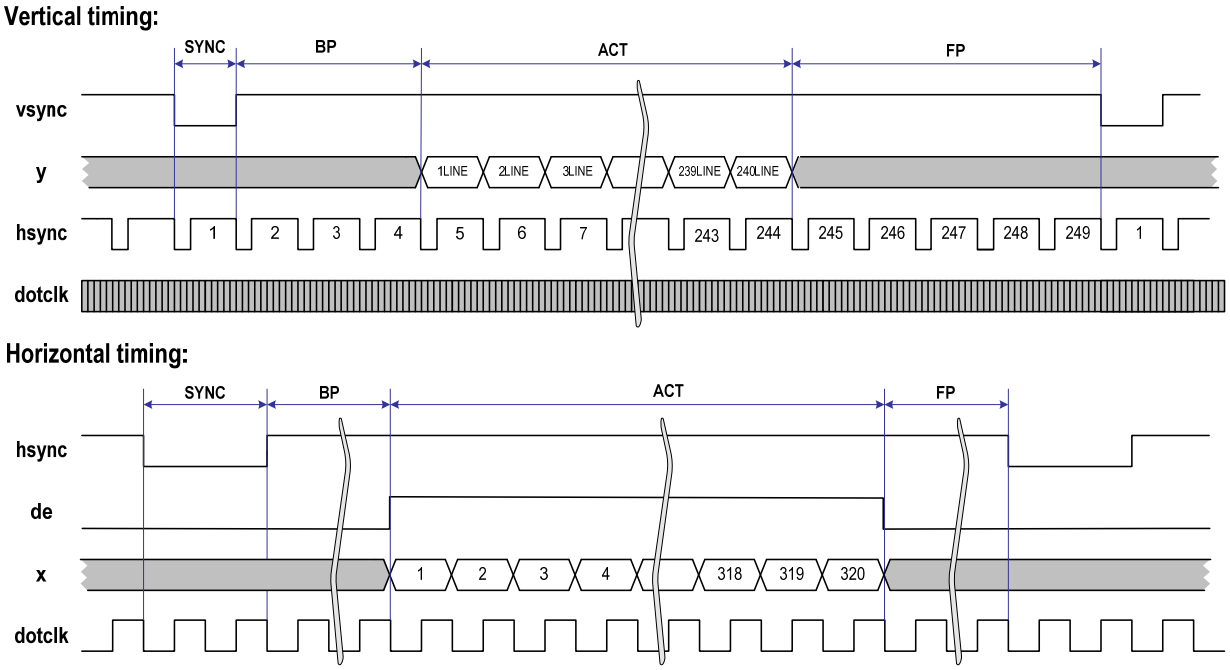
功能描述

此组件生成发送到面板的连续时序信号，而无需 CPU 干预。在刷新期间，组件还生成对帧缓冲器的读取请求，通过 16 位像素数据的帧进行扫描。在 blanking 间隔（横向和纵向）期间，组件可在帧缓冲器接口上生成读取或写入数据操作。

屏幕刷新和时序

在整个帧时间中，组件在 vsync 上生成已配置的纵向时序模式，而在帧的各个线路中，组件在 hsync 上生成已配置的横向模式。除了 hsync 和 vsync，一些面板需要 de（数据使能）信号，该信号在屏幕刷新的有效期间处于高电平有效状态。尽管刷新周期各个部分的时序不同，但所有面板都以同样的方式操作。图 1 显示典型面板的时序图。

图 1. 典型面板时序



纵向信号的各个帧的序列和横向信号的各线路的序列遵循此模式：

- Sync pulse（同步脉冲）：同步脉冲有效时期
- Back Porch（后沿）：从同步脉冲结束直至有效显示区域的时期
- Active（活动）：屏幕上的显示区域
- Front Porch（前沿）：从有效显示结束直至同步脉冲开始的时期

地址生成

由于屏幕已刷新，组件必须扫描帧缓冲器以生成屏幕上像素的地址。各个像素需要帧缓冲器中的一个 16 位读数。对于各个帧的开头，帧缓冲器中的索引复位为帧缓冲器的指定起点。此值初始设为 0，之后可用 API 函数进行更改。帧缓冲器地址不影响读取和写入 API 函数，它仅影响刷新操作。

帧缓冲器数据操作

控制器组件可执行读取或写入数据操作。这些数据操作具有以下参数：

- “Read”（读取）或“write”（写入）
- Address（地址）：最多达 23 位地址
- Data：16 位值。数据在“do”（数据输出）输出，在“di”（数据输入）输入。

用于此组件的实现将 23 位地址和 1 位读/写指示符进行组合。这允许用三个字节将地址和数据操作类型传输到组件。它还允许将数据操作类型和地址一同放在数据路径 FIFO 中。

在横向和纵向 blanking 间隔期间执行读取和写入数据操作。

空闲状态

在帧缓冲器接口上未发生读取和写入操作时，接口处于空闲状态。空闲状态控制信号的值与读状态时控制信号的值相同。在空闲情况下，输出引脚的值如下所示：

- do：无需关注（可保留其上一个状态）
- doe：0
- addr：无需关注（可保留其上一个状态）
- nwe：1
- noe：0

读取和写入数据操作的说明中未列出的任何信号都处于空闲状态。

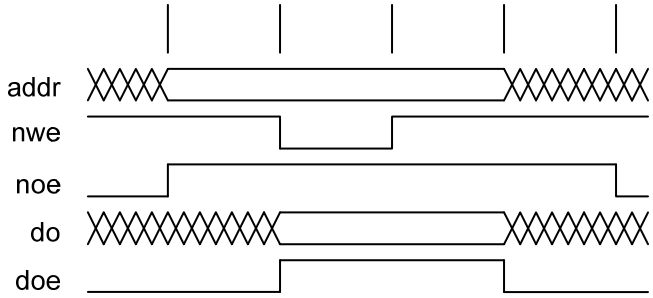
写入数据操作

组件实现图 2 中写入数据操作的时序图。此图显示写入数据操作需要四个 dotclk 周期（所有图都处于 dotclk 周期中）。此数据操作之前或之后可紧跟另一个读取或写入数据操作，或可在写入数据操作之前或之后处于空闲状态。

与 CPU 的接口允许 CPU 将写命令加入命令队列（请求一次写入并提供地址和数据，然后在数据操作实际在帧缓冲器中完成之前继续处理其他任务）。实现允许 CPU 有四个待处理写入请求，而不会停止。

注意 noe 和 doe 信号的模式，此模式防止数据总线同时被组件和帧缓冲器驱动（与信号时滞无关）。

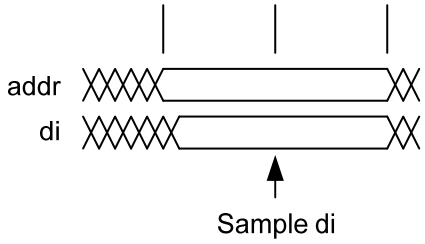
图 2. 写入数据操作时序图



读取数据操作

此组件实现图 3 中读取数据操作的时序图。此数据操作之前或之后可紧跟另一个读取或写入数据操作，或可在写入数据操作之前或之后处于空闲状态。

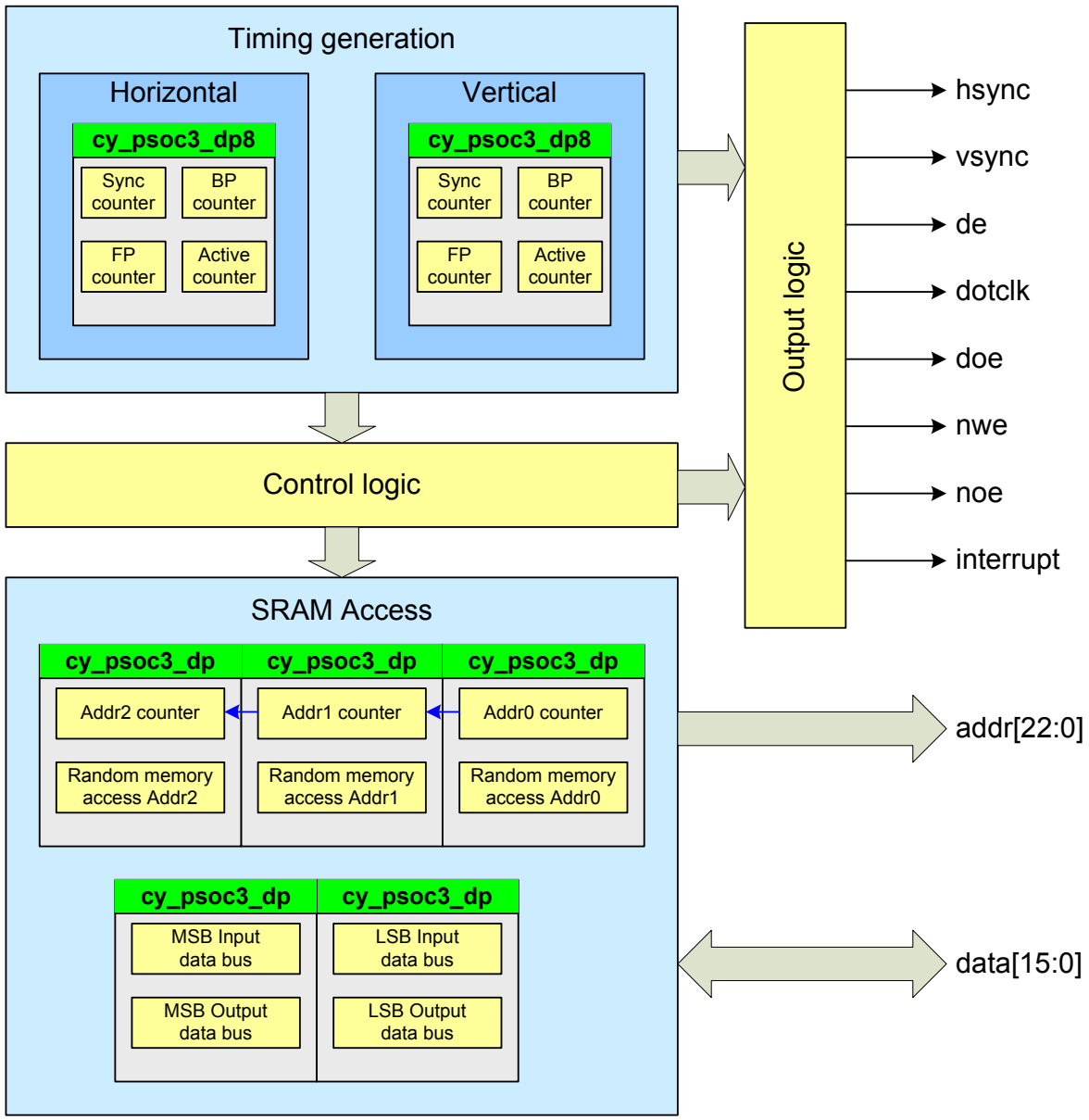
图 3. 读取数据操作时序图



框图和配置

将 GraphicLCDCtrl 组件作为一组已配置的 UDB 实现。图 4 显示此实现。

图 4. 框图



寄存器

GraphicLCDCtrl_STATUS_REG

位	7	6	5	4	3	2	1	0
值	保留				v_blanking	h_blanking	Avail（可用）	full（满）

- full（满）：设置命令/数据 FIFO 是否已满
- avail（可用）：设置读取数据对于 CPU 是否有效
- h_blanking：在横向 blanking 间隔期间设置
- v_blanking：在纵向 blanking 间隔期间设置

直流和交流电气特性

下面的值表示了预计性能，它们基于初始特性数据。

时序特性“额定路由的最大值”

参数	说明	最小值	典型值	最大值 [1]	单位
f_{DOTCLK}	Dotclk 频率	–	–	20	MHz
f_{CLOCK}	组件时钟频率	$2*f_{DOTCLK}$	–	–	MHz
t_{DOTCLK}	Dotclk 周期	$1/f_{DOTCLK}$	–	–	ns
t_{CKL}	Dotclk 低时间	–	0.5	–	$1/f_{DOTCLK}$
t_{CKH}	Dotclk 高时间	–	0.5	–	$1/f_{DOTCLK}$
屏幕刷新和数据操作时序					
t_{HSYNC}	水平同步脉冲周期	1	–	256	t_{DOTCLK}
t_{HBP}	水平后沿周期	6	–	256	t_{DOTCLK}
$t_{HACTIVE}$	水平活动区域周期	4	–	1024	t_{DOTCLK}
t_{HFP}	水平前沿周期	1	–	256	t_{DOTCLK}

[1] 这些“额定”数字提供了额定路由条件下组件的最大安全运行频率。可以在更高的时钟频率运行组件，但是将需要使用 STA 结果验证时序要求。

参数	说明	最小值	典型值	最大值 [1]	单位
t_{HBLANK}	水平消隐期	–	t_{HSYNC} + t_{HBP} + t_{HFP}	–	t_{DOTCLK}
H_{CYCLE}	水平循环	–	t_{HBLANK} + $t_{HACTIVE}$	–	t_{DOTCLK}
t_{VSYNC}	垂直同步脉冲周期	1	–	256	H_{CYCLE}
t_{VBP}	垂直后沿周期	1	–	256	H_{CYCLE}
$t_{VACTIVE}$	垂直活动区域周期	4	–	1024	H_{CYCLE}
t_{VFP}	垂直前沿周期	1	–	256	H_{CYCLE}
t_{VBLANK}	水平循环	–	t_{VSYNC} + t_{VBP} + t_{VFP}	–	H_{CYCLE}
V_{CYCLE}	垂直循环	–	t_{VBLANK} + $t_{VACTIVE}$	–	H_{CYCLE}
像素时序					
t_{HV}	同步信号下降沿的相差	–	t_{HFP}	–	t_{DOTCLK}
t_{VSYS}	垂直同步设置时间	–	0.5	–	t_{DOTCLK}
t_{VSYH}	垂直同步保持时间	–	0.5	–	t_{DOTCLK}
t_{HSYS}	水平同步建立时间	–	0.5	–	t_{DOTCLK}
t_{HSYH}	水平同步保持时间	–	0.5	–	t_{DOTCLK}
t_{DS}	到 LCD 面板的数据建立时间	–	0.5	–	t_{DOTCLK}
t_{DH}	到 LCD 面板的数据保持时间	–	0.5	–	t_{DOTCLK}
帧缓冲器操作时序					
t_{AS}	地址建立时间	1	–	–	t_{DOTCLK}
t_{AH}	地址保持时间	–	2	–	t_{DOTCLK}
t_{PWE}	NWE 脉冲宽度	–	1	–	t_{DOTCLK}
t_{DSW}	到帧缓冲器的数据建立时间	–	1	–	t_{DOTCLK}
t_{DHW}	到帧缓冲器的数据保持时间	–	1	–	t_{DOTCLK}
t_{CYCLE}	时钟周期时间				
	写周期	4	–	–	t_{DOTCLK}
	读周期	2	–	–	t_{DOTCLK}
t_{ACC}	数据访问时间	–	1	–	t_{DOTCLK}

参数	说明	最小值	典型值	最大值 [1]	单位
t_{OH}	输出保持时间	–	0	–	t_{DOTCLK}

时序特性“所有路由的最大值”

参数	说明	最小值	典型值	最大值 [2]	单位
f_{DOTCLK}	Dotclk 频率	–	–	10	MHz
f_{CLOCK}	组件时钟频率	$2*f_{DOTCLK}$	–	–	MHz
t_{DOTCLK}	Dotclk 周期	$1/f_{DOTCLK}$	–	–	ns
t_{CKL}	Dotclk 低时间	–	0.5	–	$1/f_{DOTCLK}$
t_{CKH}	Dotclk 高时间	–	0.5	–	$1/f_{DOTCLK}$
屏幕刷新和数据操作时序					
t_{HSYNC}	水平同步脉冲周期	1	–	256	t_{DOTCLK}
t_{HBP}	水平后沿周期	6	–	256	t_{DOTCLK}
$t_{HACTIVE}$	水平活动区域周期	4	–	1024	t_{DOTCLK}
t_{HFP}	水平前沿周期	1	–	256	t_{DOTCLK}
t_{HBLANK}	水平消隐期	–	$t_{HSYNC} + t_{HBP} + t_{HFP}$	–	t_{DOTCLK}
H_{CYCLE}	水平循环	–	$t_{HBLANK} + t_{HACTIVE}$	–	t_{DOTCLK}
t_{VSYNC}	垂直同步脉冲周期	1	–	256	H_{CYCLE}
t_{VBP}	垂直后沿周期	1	–	256	H_{CYCLE}
$t_{VACTIVE}$	垂直活动区域周期	4	–	1024	H_{CYCLE}
t_{VFP}	垂直前沿周期	1	–	256	H_{CYCLE}
t_{VBLANK}	水平循环	–	$t_{VSYNC} + t_{VBP} + t_{VFP}$	–	H_{CYCLE}
V_{CYCLE}	垂直循环	–	$t_{VBLANK} + t_{VACTIVE}$	–	H_{CYCLE}
像素时序					
t_{HV}	同步信号下降沿的相差	–	t_{HFP}	–	t_{DOTCLK}

[2] “所有路由”的最大值的计算方法是：<额定值>/2，然后取整到最近的整数。如果您的组件实例的运行速率不超过这些速度，则此组件应不会出现时序问题。

参数	说明	最小值	典型值	最大值 [2]	单位
t_{VSYS}	垂直同步建立时间	–	0.5	–	t_{DOTCLK}
t_{VSYH}	垂直同步保持时间	–	0.5	–	t_{DOTCLK}
t_{HSYS}	水平同步建立时间	–	0.5	–	t_{DOTCLK}
t_{HSYH}	水平同步保持时间	–	0.5	–	t_{DOTCLK}
t_{DS}	到 LCD 面板的数据建立时间	–	0.5	–	t_{DOTCLK}
t_{DH}	到 LCD 面板的数据保持时间	–	0.5	–	t_{DOTCLK}
帧缓冲器操作时序					
t_{AS}	地址建立时间	1	–	–	t_{DOTCLK}
t_{AH}	地址保持时间	–	2	–	t_{DOTCLK}
t_{PWE}	NWE 脉冲宽度	–	1	–	t_{DOTCLK}
t_{DSW}	到帧缓冲器的数据建立时间	–	1	–	t_{DOTCLK}
t_{DHW}	到帧缓冲器的数据保持时间	–	1	–	t_{DOTCLK}
t_{CYCLE}	时钟周期时间				
	写周期	4	–	–	t_{DOTCLK}
	读周期	2	–	–	t_{DOTCLK}
t_{ACC}	数据访问时间	–	1	–	t_{DOTCLK}
t_{OH}	输出保持时间	–	0	–	t_{DOTCLK}

图 5. 屏幕刷新和数据操作时序

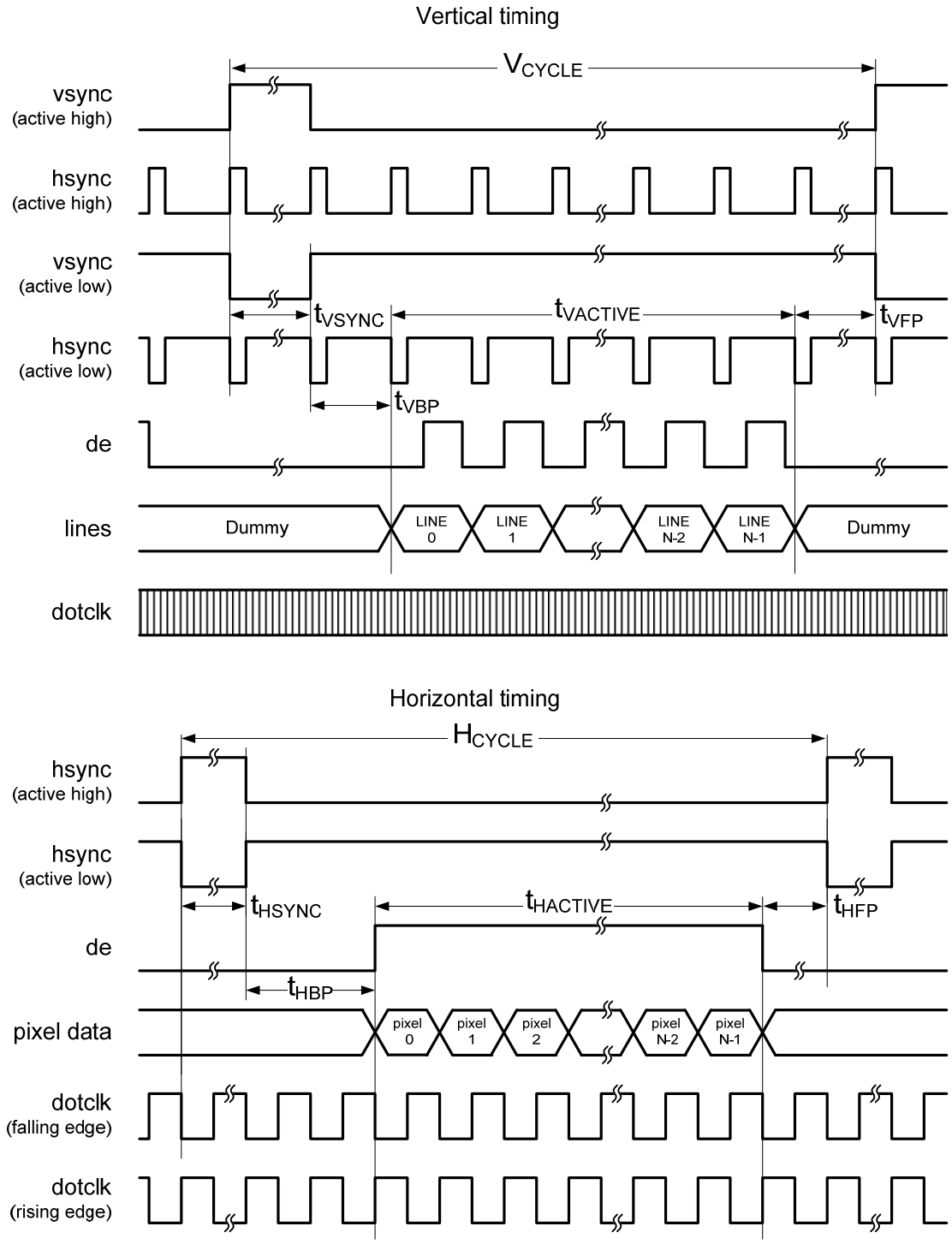


图 6. 像素时序图

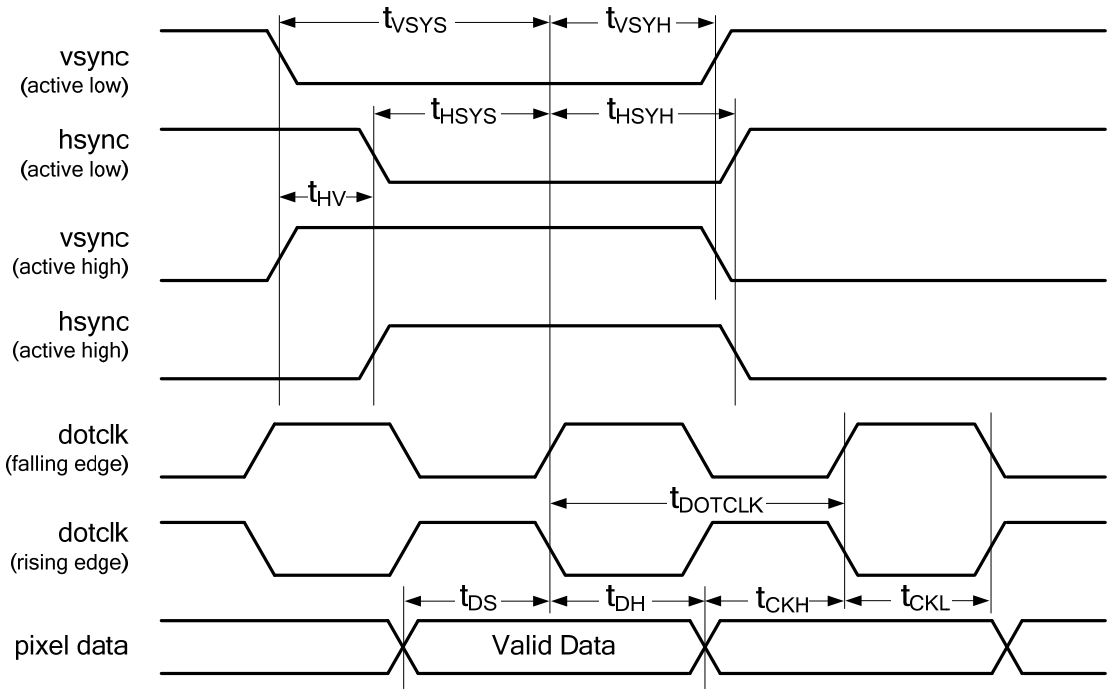
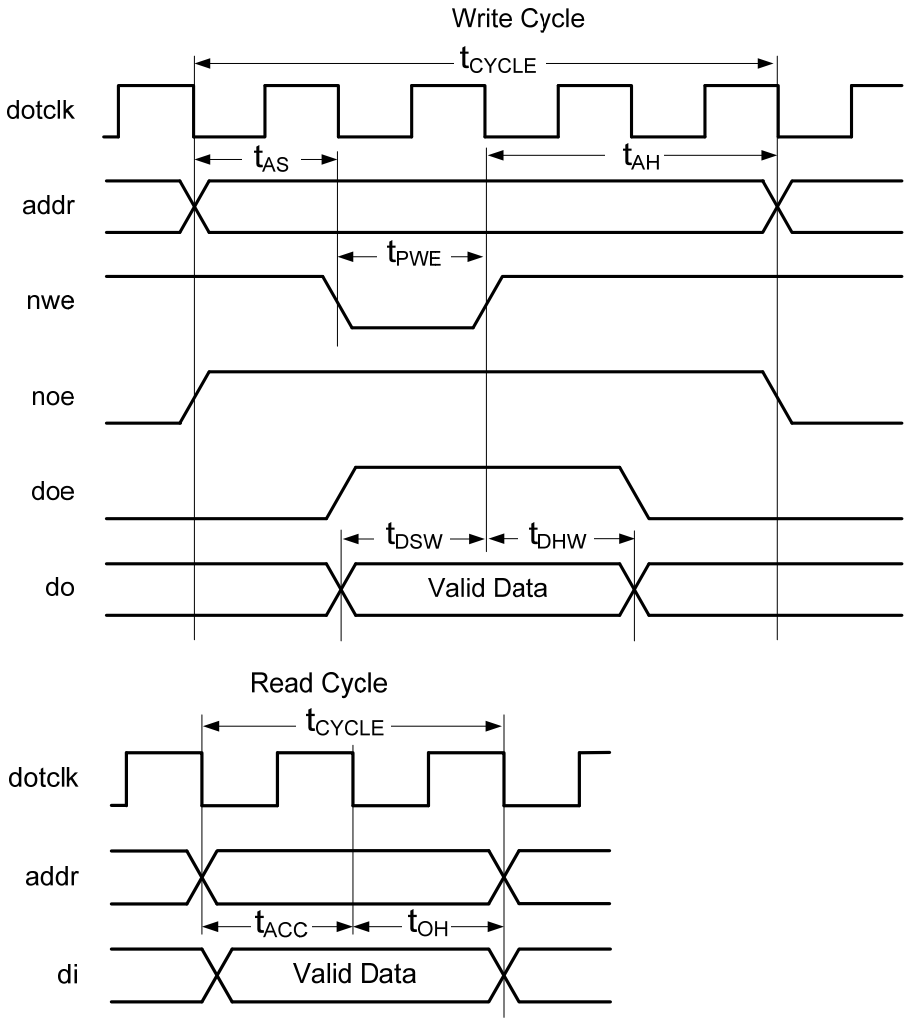

tVSYS
tVSYH
vsync
(active low)
tHSYS
tHSYH
hsync
(active low)
tHV
vsync
(active high)
hsync
(active high)
dotclk
(falling edge)
tDOTCLK
dotclk
(rising edge)
tDS
tDH
tCKH
tCKL
pixel data
Valid Data

图 7. 帧缓冲器操作时序图



如何将 STA 结果用于特性数据

额定路由最大值是通过使用静态时序分析 (STA) 进行多次测试而收集的。您可以用下列方法，使用 STA 结果计算设计的最大值：

f_{clock} 最大组件时钟频率显示在命名外部时钟的时钟汇总中的时序结果中。下图显示了时钟限制的示例。

- **Clock Summary Section**

Clock	Type	Nominal Frequency	Required Frequency	Maximum Frequency	Violation
CLK	Sync	20.000 MHz	20.000 MHz	41.545 MHz	
ClockBlock/clk_bus	Async	60.000 MHz	60.000 MHz	N/A	
ClockBlock/dclk_0	Async	20.000 MHz	20.000 MHz	N/A	
CyBUS_CLK	Sync	60.000 MHz	60.000 MHz	N/A	
CyILO	Async	1.000 kHz	1.000 kHz	N/A	
CyIMO	Async	3.000 MHz	3.000 MHz	N/A	
CyMASTER_CLK	Sync	60.000 MHz	60.000 MHz	N/A	
CyPLL_OUT	Async	60.000 MHz	60.000 MHz	N/A	

其余参数特定于实现，在时钟循环中进行计算。它们可以分为两类。

- 用于配置此组件的参数：

  屏幕刷新和数据操作时序参数

  f_{DOTCLK} 驱动至面板的时钟。此时钟的频率是输入时钟的频率的一半。此组件允许您改变数据传输到面板的dotclk边沿。此参数可设置为“上升沿”或“下降沿”。如果将其设置为上升沿，所有输出信号将在 dotclk 的上升沿上进行更改。如果将其设置为下降沿，将在 dotclk 的下降沿的同时进行输出跃变。从而，面板将在 dotclk 的对立边沿上对这些信号进行采样，从而满足建立和保持时间。

  t_{HSYNC} 在 dotclk 中水平 hsync 脉冲处于活动状态的周期。信号可以是高电平有效（生成的脉冲是高脉冲），也可以是低电平有效（生成的脉冲是低脉冲）。信号的极性在组件自定义程序中设置。

  t_{HBP} 在 dotclk 中从 hsync 脉冲的结尾到活动区域的开头的周期。

  $t_{HACTIVE}$ dotclk 中的水平活动区域周期（显示区域）。

  t_{HFP} 在 dotclk 中从活动显示的结尾直至 hsync 脉冲的开头的周期。

  t_{VSYNC} 在 H_{CYCLE} 中垂直同步脉冲处于活动状态的周期。信号可以是高电平有效（生成的脉冲是高脉冲），也可以是低电平有效（生成的脉冲是低脉冲）。信号的极性在组件自定义程序中设置。

  t_{VBP} 在 H_{CYCLE} 中从 vsync 脉冲的结尾到活动区域的开头的周期。

  $t_{VACTIVE}$ CYCLE 中的垂直活动区域周期（显示区域）。

  t_{VFP} 在 H_{CYCLE} 中从活动显示的结尾直至 vsync 脉冲的开头的周期。

  V_{CYCLE} 更新了一个完整的帧的周期。这定义为 t_{VSYNC} 周期、t_{VBP} 周期、$t_{VACTIVE}$ 周期与 t_{VFP} 周期之和。

  t_{VBLANK} 帧周期的消隐信号线路的数量。在此周期期间，可以更新帧缓冲器（此组件启动了到帧缓冲器的写/读操作）。在消隐信号期间，没有到 LCD 面板的数据流。此周期是 t_{VSYNC} 间隔、t_{VBP} 间隔与 t_{VFP} 间隔之和。

H_{CYCLE} 更新了一个水平线路的周期。定义为 t_{HSYNC} 周期、t_{HBP}、$t_{HACTIVE}$ 周期与 t_{HFP} 周期之和。

t_{HBLANK} 一个水平线路的消隐信号像素的数量。在此周期期间，可以更新帧缓冲器（此组件启动了到帧缓冲器的写/读操作）。在消隐信号期间，没有到 LCD 面板的数据流。此周期是 t_{HSYNC} 周期、t_{HBP} 周期与 t_{HFP} 周期之和。

- 基于组件实现固定的参数：

像素时序参数

t_{DOTCLK} dotclk 信号的周期。

t_{CKL} 此组件生成了 50% 占空比 dotclk。

t_{CKH} 此组件生成了 50% 占空比 dotclk。

t_{VSYS} 在 dotclk 信号的活动边沿之前 vsync 信号有效的最小时间量。

t_{VSYH} 在 dotclk 信号的活动边沿之后 vsync 信号有效的最小时间量。

t_{HSYS} 在 dotclk 信号的活动边沿之前 hsync 信号有效的最小时间量。

t_{HSYH} 在 dotclk 信号的活动边沿之后 hsync 信号有效的最小时间量。

请注意，t_{VSYS}、t_{VSYH}、t_{HSYS}、t_{HSYH} 参数取决于垂直时序的 dotclk 和 vsync 与水平时序的 dotclk 和 hsync 之间关系。您可以针对信号跃变到面板更改 dotclk 边沿。从而，面板将在 dotclk 的对立边沿上对这些信号进行重新采样，并满足设置和保持时间。从而，您几乎可以将建立和保持时间（即 t_{VSYS}、t_{VSYH}、t_{HSYS}、t_{HSYH} 信号）延长为整个 dotclk 周期的一半。

t_{HV} 同步信号活动边沿的相差。在组件的实现过程中，在水平前沿的第一个循环内执行垂直计数，因此 hsync 之前的 vsync 的相差等于水平前沿周期 (t_{HFP})。

t_{DS} 在 dotclk 信号之前数据在面板的输入上有效的最小时间量。

t_{DH} 在 dotclk 信号之后数据在面板的输入上有效的最小时间量。

要确定这些参数，用作帧缓冲器的 SRAM 的时序必须 GraphicLCDCtrl 组件的实现一起考虑。当屏幕刷新时，此组件将扫描帧缓冲器，同时生成屏幕上的像素的地址。帧缓冲器的地址在有效的 dotclk 边沿上进行更改。dotclk 信号与地址信号之间的延迟接近于零，因为这两种信号都是在内部组件时钟上生成并传送到输出引脚上的。从而几乎允许保持时间为整个 dotclk 周期的一半。设置时间可以计算为 dotclk 的整个周期的一半周期减去 SRAM 帧缓冲器的 t_{AA}。t_{AA} 可以在相应的 SRAM 数据表中找到。

帧缓冲器数据操作参数

t_{AS} 在 nwe 信号的下降沿之前地址信号有效的最小时间量。

t_{AH} 在 nwe 信号的上升沿之后地址信号有效的最小时间量。

t_{PWE} 写入信号的最小脉冲宽度低时间。

t_{CYCLE} 在帧缓冲器的接口上执行单一操作（写/读）的时间段。

t_{DSW} 在写信号的下降沿之前数据有效的最小时间量。

t_{DHW} 在写信号的上升沿之后数据有效的最小时间量。

t_{ACC} 在读操作的地址有效之后对数据进行重新采样的最小时间量。

t_{OH} 在数据采样的 dotclk 信号的有效边沿之后数据应有效的最小时间量。

组件更改

本节介绍组件与以前版本相比的主要更改。

版本	更改说明	更改/影响原因
1.61	向.cyre 文件中包括的所有组件 API 添加了 CYREENTRANT 关键词。	并非所有 API 都是真正可重入的。组件 API 源文件中的注释指出了适用的函数。 需要此更改为采用安全方式使用并且不是可重入函数消除编译器警告：通过标志或关键节防止并发调用。
	添加了时序限制以标记此组件中的错误时序路径。	从时序分析中删除了未使用的路径。这避免了错误的时序冲突消息。
1.60.a	从数据表中删除了对关联套件的引用。	
1.60	将 FIFO 模块状态信号重新采样到 DP 时钟中。	允许组件针对所有 PSoC 3 芯片和 PSoC 5 芯片产生相同的时序结果。
	对数据表进行了少量编辑和更新	

© 赛普拉斯半导体公司，2012。此处所包含的信息可能会随时更改，恕不另行通知。除赛普拉斯产品的内嵌电路之外，赛普拉斯半导体公司不对任何其他电路的使用承担任何责任。也不根据专利或其他权利以明示或暗示的方式授予任何许可。除非与赛普拉斯签订明确的书面协议，否则赛普拉斯产品不保证能够用于或适用于医疗、生命支持、救生、关键控制或安全应用领域。此外，对于可能发生运转异常和故障并对用户造成严重伤害的生命支持系统，赛普拉斯不授权将其产品用作此类系统的关键组件。若将赛普拉斯产品用于生命支持系统中，则表示制造商将承担因此类使用而招致的所有风险，并确保赛普拉斯免于因此而受到任何指控。

PSoC® 是赛普拉斯半导体公司的注册商标，PSoC Creator™ 和 Programmable System-on-Chip™ 是赛普拉斯半导体公司的商标。此处引用的所有其他商标或注册商标归其各自所有者所有。

所有源代码（软件和/或固件）均归赛普拉斯半导体公司（赛普拉斯）所有，并受全球专利法规（美国和美国以外的专利法规）、美国版权法以及国际条约规定的保护和约束。赛普拉斯据此向获许可者授予适用于个人的、非独占性、不可转让的许可，用以复制、使用、修改、创建赛普拉斯源代码的派生作品、编译赛普拉斯源代码和派生作品，并且其目的只能是创建自定义软件和/或固件，以支持获许可者仅将其获得的产品依照适用协议规定的方式与赛普拉斯集成电路配合使用。除上述指定的用途之外，未经赛普拉斯的明确书面许可，不得对此类源代码进行任何复制、修改、转换、编译或演示。

免责声明：赛普拉斯不针对此材料提供任何类型的明示或暗示保证，包括（但不仅限于）针对特定用途的适销性和适用性的暗示保证。赛普拉斯保留在不做出通知的情况下对此处所述材料进行更改的权利。赛普拉斯不对此处所述之任何产品或电路的应用或使用承担任何责任。对于可能发生运转异常和故障并对用户造成严重伤害的生命支持系统，赛普拉斯不授权将其产品用作此类系统的关键组件。若将赛普拉斯产品用于生命支持系统中，则表示制造商将承担因此类使用而招致的所有风险，并确保赛普拉斯免于因此而受到任何指控。

产品使用可能受适用的赛普拉斯软件许可协议限制。