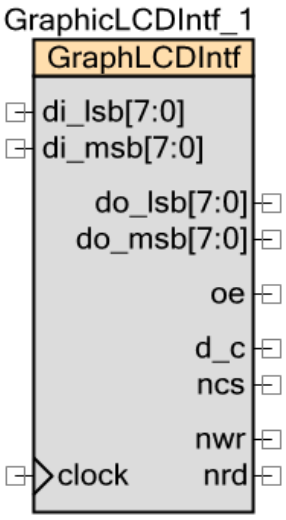
图形 LCD 接口（GraphicLCDIntf）

1.70

特性



- 8 位或 16 位接口到图形 LCD 控制器
- 兼容多种图形控制器器件
- 可与 SEGGER emWin 图形库接口
- 可执行读写操作
- 2 到 255 个周期的读取低脉冲宽度
- 1 到 255 个周期读取高脉冲宽度
- 实现典型的 i8080 接口

概述

图形 LCD 接口（GraphicLCDIntf）组件提供了连接到图形 LCD 控制器和驱动器件的接口。这些器件通常集成到 LCD 面板中。用于这些器件的接口通常称为 i8080 接口。这是来自于以前 Intel 8080 微处理器的并行总线接口协议。

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，用于绘制和表现文本和图像。

何时使用 GraphicLCDIntf

LCD 控制器和驱动器件通常集成到 LCD 面板中。它们包含或提供与显示帧缓冲器的接口和管理该缓冲器。GraphicLCDIntf 组件对此控制器进行读取和写入数据操作。这些数据操作具有以下参数：

- Read（读取）或 write（写入）
- Address：驱动 d_c pin 的一位地址
- Data（数据）（8 或 16 位）：针对写入发送“do”，针对读取读取“di”

GraphicLCDIntf 组件支持许多控制器。配置此组件时使用这三个参数。

- 时钟频率：驱动此组件的时钟频率通常受写入信号的最小低电平脉冲宽度所限（可在图形 LCD 控制器数据表中找到此值）。对于单时钟周期写入脉冲为低电平，因此要设置时钟频率以满足此要求。
- 读取高电平脉冲宽度：配置现象中的这个设置是以时钟周期计算的。时钟周期乘以为脉冲宽度高电平设置的周期数必须满足控制器的读取高电平脉冲宽度电平的要求。
- 读取低电平脉冲宽度：此参数与读取高电平脉冲宽度参数的设置方式相同。读取低电平脉冲宽度电平的时序必须满足控制器对读取脉冲宽度的要求以及对读取访问时间的要求。数据采样是在有效低电平脉冲结束之前的一个时钟周期进行的，因此，脉冲宽度必须足够长，以满足访问时间

以下列出了适用 LCD 控制器的设置：

Solomon Systech SSD1289

- 时钟频率：20 MHz （50 ns）
- 读取高电平脉冲宽度：10 个时钟周期（500 ns）
- 读取低电平脉冲宽度：10 个时钟周期（500 ns）

Solomon Systech SSD2119

- 时钟频率：25 MHz （40 ns）
- 读取高电平脉冲宽度：13 个时钟周期（500 ns）
- 读取低电平脉冲宽度：13 个时钟周期（500 ns）

Himax HX8347A

- 时钟频率：28.5 MHz （35 ns）
- 读取高电平脉冲宽度：3 个时钟周期（105 ns）
- 读取低电平脉冲宽度：11 个时钟周期（385 ns）

ILITEK ILI9325

- 时钟频率：20 MHz （50 ns）
- 读取高电平脉冲宽度：3 个时钟周期（150 ns）
- 读取低电平脉冲宽度：3 个时钟周期（150 ns）

Epson S1D13743

- 时钟频率：33 MHz （33.3 ns）
- 读取高电平脉冲宽度：2 个时钟周期（67 ns）
- 读取低电平脉冲宽度：5 个时钟周期（167 ns）

输入/输出连接

本节介绍了 GraphicLCDIntf 组件的输入和输出连接。I/O 列表中的星号 (*) 表示，在 I/O 说明中列出的情况下，该 I/O 可能不可见。

时钟

运行该组件的时钟。GraphicLCDIntf 完全由连接到组件的单个时钟运行。

di_lsb[7:0]

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，并禁用此引脚的“Input Synchronized”（同步输入）选项。信号自身就是同步的，因为它们是基于同步输出信号驱动。

di_msb[7:0] *

输入总线的高八位。在读取数据操作过程中，它们用于数据。它们仅在 16 位接口模式中存在。

将这些信号连接到器件的输入引脚上，并禁用此引脚的“Input Synchronized”（同步输入）选项。信号自身就是同步的，因为它们是基于同步输出信号驱动。

do_lsb[7:0]

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] *

输出总线的高八位。在写入数据操作过程中，它们用于数据。它们仅在 16 位接口模式中存在。

oe

数据总线的输出使能。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。有关此信号的使用方式，请参见原理图宏信息。

d_c

数据/命令信号。此信号在高电平状态下表示数据操作和低电平状态下表示命令操作。

ncs

有效低电平芯片选择。

nwr

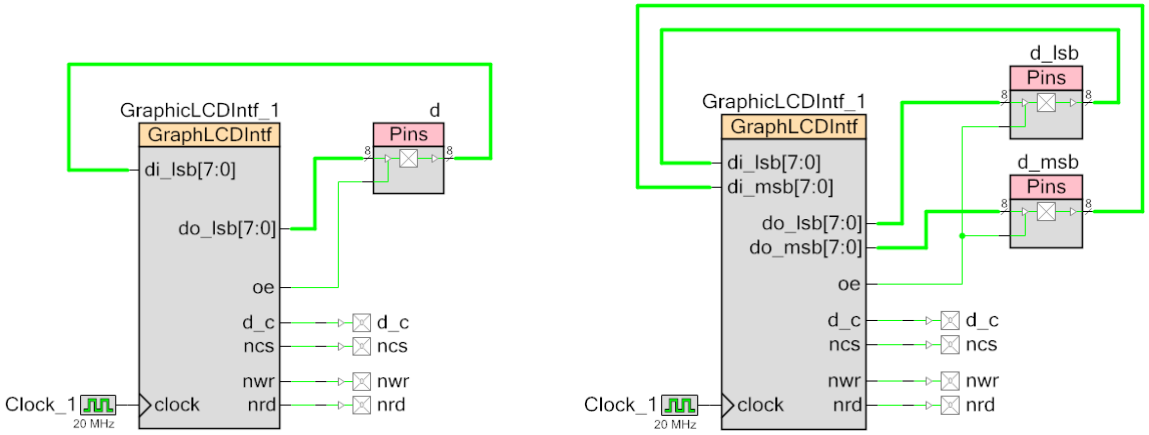
有效低电平写入控制信号。

nrd

有效低电平读取控制信号。

原理图宏信息

除了组件目录中的标准符号输入，PSoC Creator 还提供两个宏。一个宏用于实现 8 位连接到引脚和时钟。另一个宏用于实现 16 位连接到引脚和时钟。

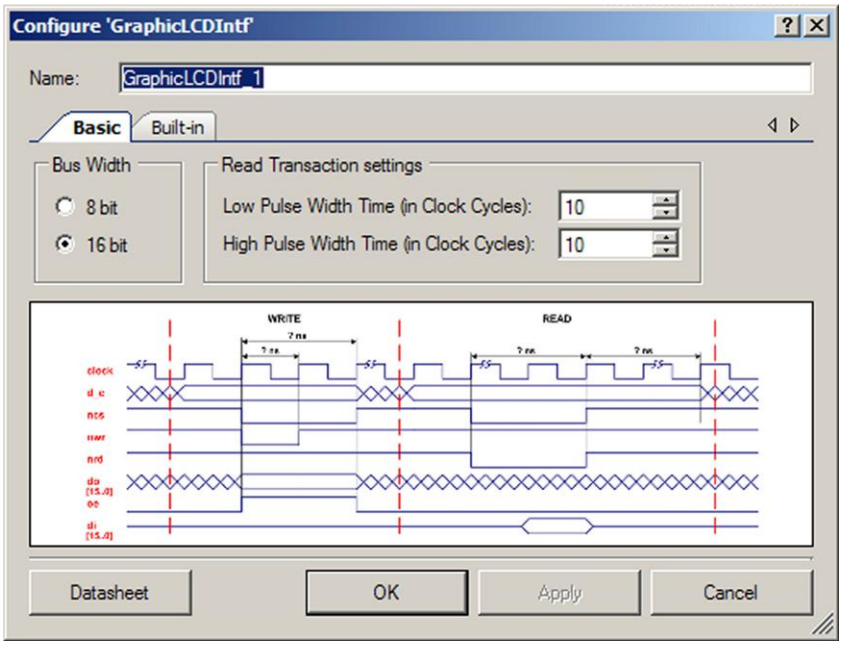


各个宏的时钟设置为 20 MHz，并将脉冲宽度设置为默认值。这是针对 SSD1289 控制器的正确设置。

所有数据引脚上的“Input Synchronized”（同步输入）选项为取消，并关闭所有引脚的 API 生成。

组件参数

将一个 GraphicLCDIntf 组件拖放到您的设计上，并双击以打开 **Configure**（配置）对话框。默认 GraphicLCDIntf 设置就是对 Solomon Systech SSD1289 控制器正确的设置。



总线宽度

确定组件支持 8 位还是 16 位并行接口到图形 LCD 控制器。默认设置为 **16 bit**（16 位）。

Low Pulse Width Time（低电平脉冲宽度时间）

确定控制器的读取低电平脉冲宽度所需的时钟周期数。此值的范围为 2 到 255 个时钟周期（最小值为 2，因为必须在脉冲结束之前的一个时钟周期内对读取值进行采样）。默认设置为 **10**。

High Pulse Width Time（高电平脉冲宽度时间）

确定控制器的读取高电平脉冲宽度所需的时钟周期数。此值的范围为 1 到 255 个时钟周期。默认设置为 **10**。

时钟选择

此器件中没有内部时钟。您必须添加时钟源。此器件根据连接到器件的单一时钟进行操作。

应用编程接口

通过应用编程接口（API），您可以使用软件进行配置组件。下表列出并说明每个函数的接口。以下各节将更详细地介绍了每个函数。

默认情况下，PSoC Creator 将实例名称“GraphicLCDIntf_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。该实例名称成为为组件生成的每个全局函数名称、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“GraphicLCDIntf”。

函数	说明
GraphicLCDIntf_Start()	启动GraphicLCDIntf接口。
GraphicLCDIntf_Stop()	禁用GraphicLCDIntf接口。
GraphicLCDIntf_Write8()	在8位并行接口上启动写入数据操作。
GraphicLCDIntf_Write16()	在16位并行接口上启动写入数据操作。
GraphicLCDIntf_WriteM8()	在8位并行接口上启动多个写入数据操作。
GraphicLCDIntf_WriteM16()	在16位并行接口上启动多个写入数据操作。
GraphicLCDIntf_Read8()	在8位并行接口上启动读取数据操作。
GraphicLCDIntf_Read16()	在16位并行接口上启动读取数据操作。
GraphicLCDIntf_Sleep()	保存配置，并禁用GraphicLCDIntf。
GraphicLCDIntf_Wakeup()	恢复配置，并启用GraphicLCDIntf。
GraphicLCDIntf_Init()	初始化或恢复GraphicLCDIntf默认配置。
GraphicLCDIntf_Enable()	使能GraphicLCDIntf。
GraphicLCDIntf_SaveConfig()	保存GraphicLCDIntf的配置。
GraphicLCDIntf_RestoreConfig()	恢复GraphicLCDIntf的配置。

全局变量

变量	说明
GraphicLCDIntf_initVar	指示是否已初始化Graphic LCD接口。变量将初始化为0，并在第一次调用GraphicLCDIntf_Start()时设置为1。这样，第一次调用GraphicLCDIntf_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GraphicLCDIntf_Start()或GraphicLCDIntf_Enable()函数之前先调用GraphicLCDIntf_Init()函数。

void GraphicLCDIntf_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。配置组件以进行操作。

参数： 无

返回值： 无

其他影响： 无

void GraphicLCDIntf_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： 无

返回值： 无

其他影响： 无

void GraphicLCDIntf_Write8(uint8 d_c、uint8 data)

说明： 此函数在8位并行接口上启动写入数据操作。此写入为已发布的写入，因此，此函数在接口上实际完成写入之前将返回。如果命令队列已满，此函数不返回直至有空间将此写入请求列入队伍。

参数： d_c：指示Data (1)或Command (0)。传递到d_c引脚上

data：在 do_lsb[7:0] 引脚上发送的数据

返回值： 无

其他影响： 无

void GraphicLCDIntf_Write16(uint8 d_c、uint16 data)

说明： 此函数在16位并行接口上启动写入数据操作。此写入为已发布的写入，因此，此函数在接口上实际完成写入之前将返回。如果命令队列已满，此函数不返回直至有空间将此写入请求列入队伍。

参数： d_c：指示Data (1)或Command (0)。传递到d_c引脚上

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

返回值： 无

其他影响： 无

void GraphicLCDIntf_WriteM8(uint8 d_c、uint8 * data、uint16 num)

说明： 此函数在8位并行接口上启动多个写入数据操作。通过一次（而不是多次）执行GraphicLCDIntf_WriteM8写入多个字节可以提高接口上的写性能。

参数： d_c：指示Data（1）或Command（0）。传递到d_c引脚上

data：写缓冲区的指针。在do_lsb[7:0]引脚上发送该缓冲区中的数据

num：要写入的字节数

返回值： 无

其他影响： 无

void GraphicLCDIntf_WriteM16(uint8 d_c、uint16 * data、uint16 num)

说明： 此函数在16位并行接口上启动多个写入数据操作。通过一次（而不是多次）执行GraphicLCDIntf_WriteM16写入多个字可以提高接口上的写性能。

参数： d_c：指示Data（1）或Command（0）。传递到d_c引脚上

data：写缓冲区的指针。在do_msb[7:0]（最高有效位）和do_lsb[7:0]（最低有效位）引脚上发送该缓冲区中的数据

num：要写入的字数

返回值： 无

其他影响： 无

uint8 GraphicLCDIntf_Read8(uint8 d_c)

说明： 此函数在8位并行接口上启动读取数据操作。在完成所有当前已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： d_c：指示Data（1）或Command（0）。传递到d_c引脚上。

返回值： di_lsb[7:0]引脚中的8位读取值

其他影响： 无

uint16 GraphicLCDIntf_Read16(uint8 d_c)

说明： 此函数在16位并行接口上启动读取数据操作。在完成所有当前已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： d_c：指示Data（1）或Command（0）。传递到d_c引脚上。

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16位读取值

其他影响： 无

void GraphicLCDIntf_Sleep(void)

说明： 这是组件准备进入睡眠模式的首选子程序。GraphicLCDIntf_Sleep()子程序保存当前组件的状态。然后它调用GraphicLCDIntf_Stop()函数，并调用GraphicLCDIntf_SaveConfig()函数以保存硬件配置。根据需要，禁用活动模式电源模板位或时钟关断。

在调用CyPmSleep()或CyPmHibernate()函数之前调用GraphicLCDIntf_Sleep()函数。有关功耗管理函数的详细信息，请参考*《系统参考指南》*。

参数： 无

返回值： 无

其他影响： 无

void GraphicLCDIntf_Wakeup（void）

说明： 该函数是将组件恢复到调用GraphicLCDIntf_Sleep()时状态的首选子程序。GraphicLCDIntf_Wakeup()函数调用GraphicLCDIntf_RestoreConfig()函数以恢复配置。如果组件在调用GraphicLCDIntf_Sleep()函数前已启用，则GraphicLCDIntf_Wakeup()函数还将重新启用组件。根据需要，启用活动模式电源模板位或时钟关断。

参数： 无

返回值： 无

其他影响： 调用GraphicLCDIntf_Wakeup()函数前未调用GraphicLCDIntf_Sleep()或GraphicLCDIntf_SaveConfig()函数可能会产生不可预知的行为。

void GraphicLCDIntf_Init(void)

说明： 此函数根据配置窗口“Configure”对话框设置来初始化或恢复器件。无需调用GraphicLCDIntf_Init()，因为GraphicLCDIntf_Start()子程序会调用该函数并且这是开始组件操作的首选方法。只有定义“读取低脉冲和高脉冲宽度”的静态组件配置将恢复到其初始值。

参数： 无

返回值： 无

其他影响： 此函数将重新初始化组件，但不清除FIFO中的数据，且不复位组件硬件状态机。当前数据操作在总线上执行。

void GraphicLCDIntf_Enable(void)

说明： 此函数激活硬件并开始执行器件操作。无需调用GraphicLCDIntf_Enable()，因为GraphicLCDIntf_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： 无

返回值： 无

其他影响： 无

void GraphicLCDIntf_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”对话框中定义的或通过相应API修改的当前器件参数值。该函数由GraphicLCDIntf_Sleep()函数调用。存储定义读取低脉冲和高脉冲宽度的编译时间组件配置。

参数： 无

返回值： 无

其他影响： 无

void GraphicLCDIntf_RestoreConfig(void)

说明： 此函数会恢复GraphicLCDIntf非保留寄存器的配置。API由GraphicLCDIntf_Wakeup函数调用，以恢复组件非保留寄存器。

参数： 无

返回值： 无

其他影响： 如果在调用GraphicLCDIntf_SaveConfig()之前调用此函数，针对读取低脉冲和高脉冲宽度的组件配置将恢复到定制器提供的值。

MISRA 合规性

本节介绍了MISRA-C:2004合规性和本器件的偏差情况。规定了两种类型的偏差：

- 项目偏差 — 适用于所有 PSoC Creator 器件的偏差
- 特定偏差 — 仅适用于本器件的偏差

本节提供了有关器件特定偏差的信息。*《系统参考指南》*的“MISRA 合规性”章节中介绍了项目偏差以及有关 MISRA 合规性验证环境的信息。

GraphicLCDIntf 组件尚未根据 MISRA-C:2004 编码准则合规性验证。

样例固件源代码

在 Find Example Project 对话框中，PSoC Creator 提供了多个样例项目，并且每个项目都包括了原理图和样例代码。要查看特定组件实例，请打开“Component Catalog”中的对话框或者原理图中的组件样例。要查看通用样例，请打开 Start Page 或 **File** 菜单中的对话框。根据要求，可以通过使用对话框中的 **Filter Options** 项来限定可选的项目列表。

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

总线数据操作

此接口可执行数据读取或写入操作。这些数据操作具有以下参数：

- Read（读取）或 write（写入）
- Address：在此情况下，它是驱动 d_c pin 上的一位地址
- Data（数据）（8 或 16 位）：针对写入发送“do”，针对读取读取“di”。

实现假设 CPU 使用 FIFO 将一个命令字节发送到组件。该命令字节指示读取或写入，并提供 d_c 位。

空闲状态

在接口上未发生读取和写入操作时，接口处于空闲状态。在此情况下，输出引脚的值为：

- d_c：无需关注（可能保留其上一个状态）
- ncs：1

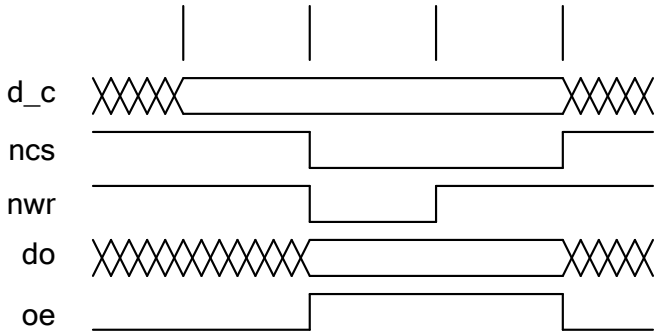
- nwr：1
- nrd：1
- do：无需关注（可能保留其上一个状态）
- oe：0

在读取和写入数据操作的说明中，任何未列出的信号都处于空闲状态。

写入数据操作

图 1 显示并行接口上数据写入操作的时序图。

图 1. 写入数据操作时序图



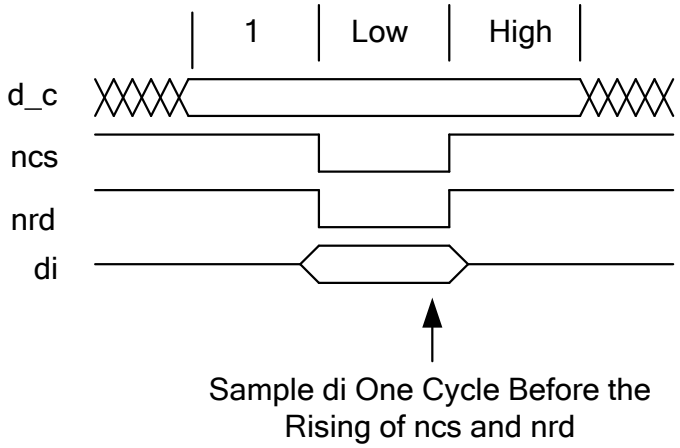
此图显示写入数据操作需要三个时钟循环。无论位宽如何，时序图都是相同的。此数据操作之前或之后可紧跟另一个读取或写入数据操作，或可在写入数据操作之前或之后处于空闲状态。

与 CPU 的接口允许 CPU 发出后写入的写入请求（请求写入以提供地址和数据，然后在数据操作实际在并行总线上完成之前继续处理）。实现允许 CPU 有四个待处理写入请求，而不会停止。

读取数据操作

图 2 显示并行接口上读取数据操作的时序图。

图 2. 读取数据操作时序图



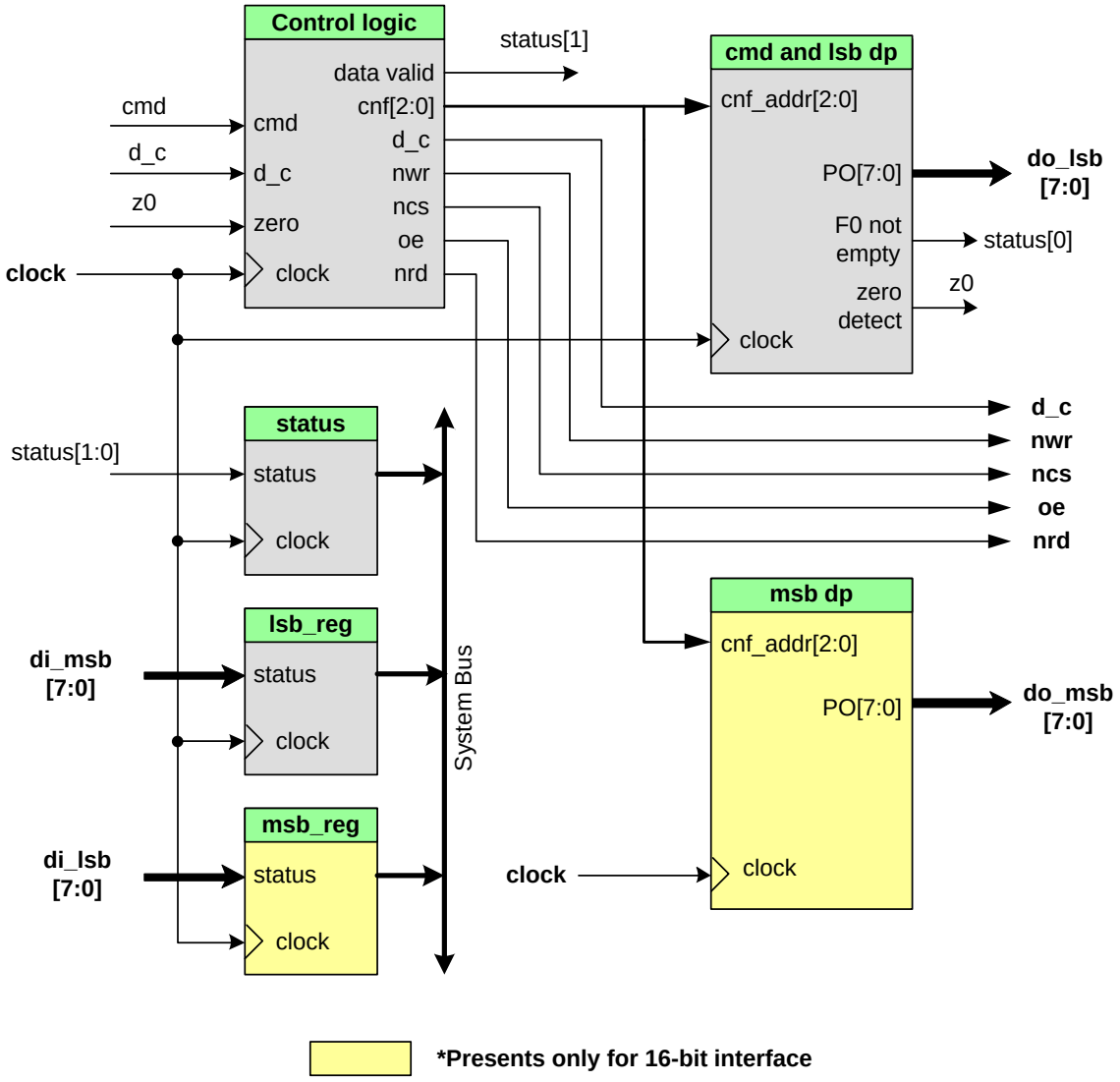
此图显示读取数据操作需要可变的时钟循环，取决于高和低读取脉冲宽度的设置。无论位宽如何，时序图都是相同的。注意，在 ncs 和 nrd 低电平脉冲结束之前的一个时钟周期内对输入数据进行采样。此数据操作之前或之后可紧跟另一个读取或写入数据操作，或可在读取数据操作之前或之后处于空闲状态。

读取和写入的顺序被保持（读取操作在后写入写入操作前发生）。读取操作要求 CPU 在继续处理之前等待读取数据操作完成。

框图和配置

GraphicLCDIntf 组件通过一组已配置的 UDB 实现。图 3 显示此实现。

图 3. 框图



寄存器

GraphicLCDIntf_STATUS_REG

位	7	6	5	4	3	2	1	0
值	保留						data_valid	F0_half_empty

- F0_half_empty：如果设置了此参数，命令/数据 FIFO 中至少有两个字节的空间。
- data_valid：设置读取数据对于 CPU 是否有效。当 CPU 读取寄存器时，清除此位。

GraphicLCDIntf_DIN_LSB_DATA_REG

位	7	6	5	4	3	2	1	0
值	di_lsb[7:0]							

- 读取数据操作的输入总线的低八位

针对 8 位接口，使用 GraphicLCDIntf_Read8() API 函数读取寄存器值。此值是 16 位接口的 GraphicLCDIntf_Read16() API 函数中返回值的最低有效字节。

GraphicLCDIntf_DIN_MSB_DATA_REG

位	7	6	5	4	3	2	1	0
值	di_msb[7:0]							

- 读取数据操作的输入总线的高八位

此寄存器值是 16 位接口的 GraphicLCDIntf_Read16() API 函数中返回值的最高有效位。

注意：当 CPU 固件读取这些寄存器时，DIN_LSB_DATA_REG 和 DIN_MSB_DATA_REG 位被清除。

资源

图形 LCD 接口组件放置在整个 UDB 阵列中。该器件利用以下资源。

配置	资源类型					
	数据路径单元	宏单元	状态单元	控制单元	DMA通道	中断
8位接口	1	11	2	–	–	–
16位接口	2	11	3	–	–	–

API 存储器大小

根据编译程序、器件、所使用的 API 数量以及组件的配置情况，组件的内存大小也不一样。下表提供了已给组件配置中的所有 API 存储器大小。

通过使用发表模式中的相应编译器，可以进行测量操作。在该模式中，存储器大小得到优化。对于特定的设计，分析编译程序生成的映射文件后可以确定存储器的大小。

配置	PSoC 3（Keil_PK51）		PSoC 5（GCC）		PSoC 5LP（GCC）	
	闪存 字节	SRAM 字节	闪存 字节	SRAM 字节	闪存 字节	SRAM 字节
8位接口	158	1	256	5	184	1
16位接口	196	1	264	5	192	1

直流和交流电气特性

除非另有说明，否则这些规范的适用条件是：-40 °C ≤ T_A ≤ 85 °C 且 T_J ≤ 100 °C。除非另有说明，否则这些规范的适用范围为 1.71 V 到 5.5 V。

直流电特性

参数	说明	最小值	典型值[1]	最大值	单位
$I_{DD(8\text{-}bit)}$	组件电流消耗				
	闲置电流[2]	–	5	–	µA/MHz
	写电流[3]	–	10	–	µA
$I_{DD(16\text{-}bit)}$	组件电流消耗				
	闲置电流[2]	–	8	–	µA/MHz
	写电流[3]	–	20	–	µA

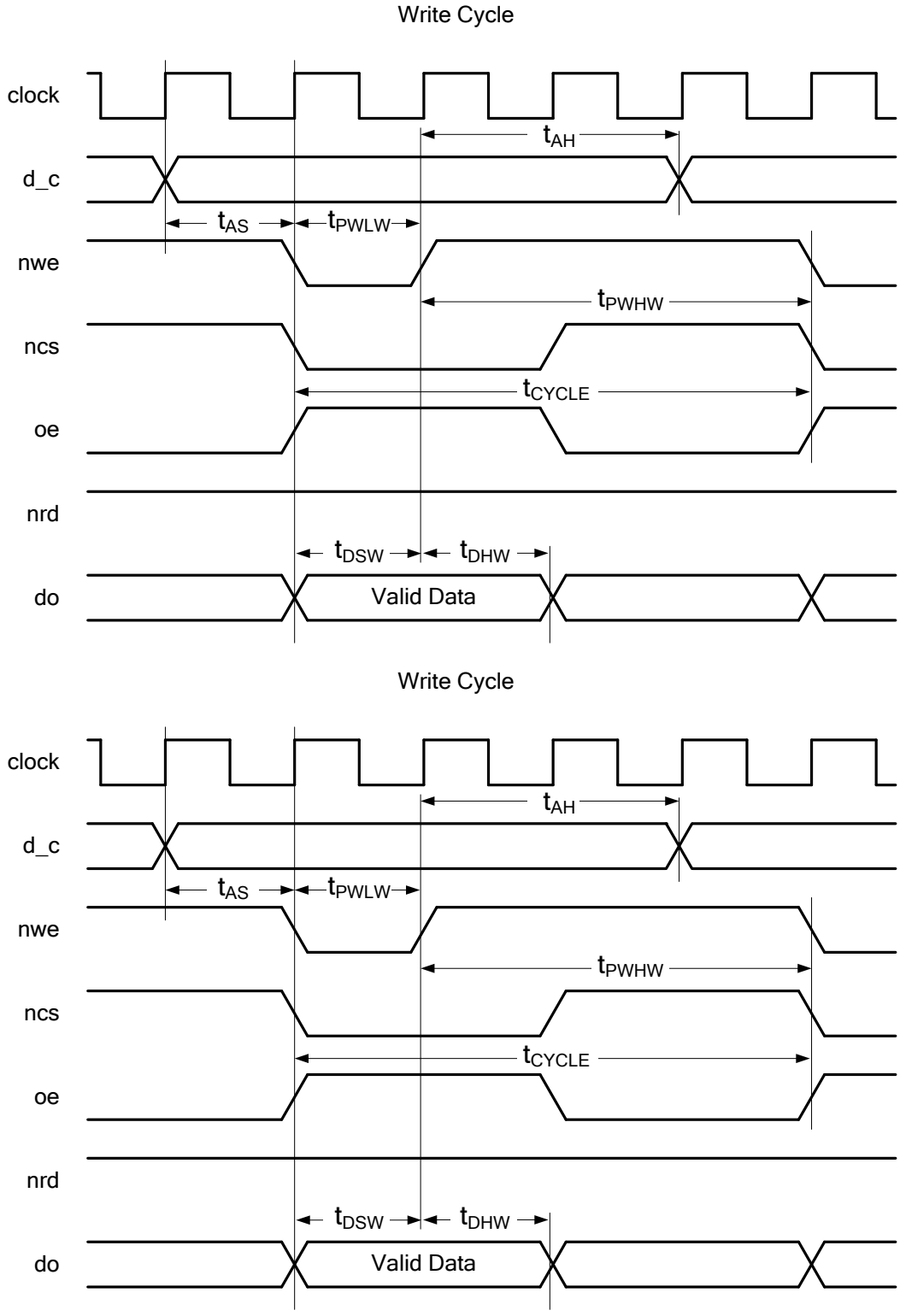
1. 未包括器件的 IO 和时钟分配的电流。这些值是在 25 °C 时的值。
2. 接口上没有进行转换数据时组件的消耗电流。
3. 接口上执行写转换时组件的额外电流消耗。该值表示每秒进行 100K 个写操作时的电流消耗。该值将被添加上闲置电流。

交流特性

参数	说明	最小值	典型值	最大值	单位
f_{CLOCK}	组件时钟频率	–	–	33	MHz
t_{AS}	地址建立时间	1	–	–	t_{CY_clock}[4]
t_{PWLW}	脉冲宽度低电平写入	–	1	–	t_{CY_clock}
t_{PWHW}	脉冲宽度高电平写入	3	–	–	t_{CY_clock}
t_{PWLR}	脉冲宽度低电平读取	2	–	255	t_{CY_clock}
t_{PWHR}	脉冲宽度高电平读取	1	–	255	t_{CY_clock}
t_{AH}	地址保持时间				
	写	2	–	–	t_{CY_clock}
	读	t_{PWHR}	–	–	t_{CY_clock}
t_{CYCLE}	时钟周期时间				
	写周期	4	–	–	t_{CY_clock}
	读周期	t_{PWLR} + t_{PWRH} + 1	_	_	t_{CY_clock}
t_{DSW}	数据建立时间		1	–	t_{CY_clock}
t_{DHW}	数据保留时间	–	1	–	t_{CY_clock}
t_{ACC}	数据访问时间	–	t_{PWHR} – 1	–	t_{CY_clock}
t_{DHR}	输出保持时间	–	0	–	t_{CY_clock}

4. $t_{CY_clock} = 1/f_{CLOCK}$。这是一个时钟周期的时间长度

图 4. 数据转换时序图


Write Cycle
clock
d_c
nwe
ncs
oe
nrd
do
tAH
tAS
tPWLW
tPWHW
tCYCLE
tDSW
tDHW
Valid Data
Write Cycle
clock
d_c
nwe
ncs
oe
nrd
do
tAH
tAS
tPWLW
tPWHW
tCYCLE
tDSW
tDHW
Valid Data

如何将 STA 结果用于特性数据

您可以用下列方法，使用静态时序分析（STA）结果计算设计的最大值：

f_{CLOCK} 命名为组件时钟的最大组件时钟频率，该频率出现在时钟汇总的时序结果中（在本实例中为 CLK）。下图显示了时钟限制的示例。

- Clock Summary Section

Clock	Domain	Nominal Frequency	Required Frequency	Maximum Frequency	Violation
CyILO	CyILO	1.000 kHz	1.000 kHz	N/A	
CyIMO	CyIMO	3.000 MHz	3.000 MHz	N/A	
CyMASTER_CLK	CyMASTER_CLK	60.000 MHz	60.000 MHz	N/A	
CLK	CyMASTER_CLK	20.000 MHz	20.000 MHz	59.492 MHz	
CyBUS_CLK	CyMASTER_CLK	60.000 MHz	60.000 MHz	N/A	
CyPLL_OUT	CyPLL_OUT	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：

  t_{PWLW} 写入信号的最小低电平脉冲宽度时间

  t_{PWLR} 读取信号的最小低电平脉冲宽度时间

  t_{PWHR} 读取信号的最小高电平脉冲宽度时间

  您可在第 1 页上的何时使用 GraphicLCDIntf 一节中找到有关如何在配置组件时使用这些参数的具体说明。

- 基于组件实现固定的参数：

  t_{PWHW} 写入信号的最小高电平脉冲宽度时间

  t_{AS} 在 nwr/nrd 信号的下降沿之前，地址信号的最小有效时间

  t_{AH} 在 nwr/nrd 信号的上升沿之后，地址信号的最小有效时间

  t_{CYCLE} 在接口上执行单个数据操作（写入/读取）时所处的时间周期

  t_{DSW} 在写入信号的上升沿之前，数据的最小有效时间

  t_{DHW} 在写信号的上升沿之后数据有效的最小时间

  t_{ACC} 在读取信号的下降沿之后对数据进行采样的最小时间

  t_{DHR} 在 nrd 信号的上升沿之后，数据的最小有效时间

组件更改

本节列出了各版本的主要组件更改内容。

版本	更改内容	更改原因/影响
1.70.a	已添加了MISRA合规性章节。	此器件未进行MISRA合规性验证。
1.70	将GraphicLCDIntf_WriteM8/16 API添加到该组件。	提高emWin图形库的图像绘制操作性能。
	向数据手册中添加了特性部分。	
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。 对于采用了安全方式并且是不可重入的函数，则需要该项变更，这样可以消除编译器警告：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	从数据规格书中移除了相关联工具包的参考。	
1.60	重新采样发送到DP时钟的FIFO模块状态信号。	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。
	向数据规格书添加了特性数据	
	对数据规格书进行了较小程度的编辑和更新	

© 赛普拉斯半导体公司，2013。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器件。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器件。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。