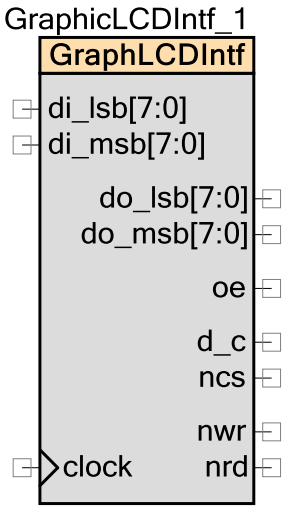
# 图形 LCD 接口 (GraphicLCDIntf)

1.61

## 特性




- 图形 LCD 控制器的 8 位或 16 位接口
- 与许多图形控制器器件兼容
- 配有 SEGGER emWin 图形库的接口
- 可执行读写操作
- 对于读取低脉冲宽度，2 到 255 个周期
- 对于读取高脉冲宽度，1 到 255 个周期
- 实现典型 i8080 接口

## 概述

图形 LCD 接口 (GraphicLCDIntf) 组件提供链接图形 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。这是来源于历史上 Intel 8080 微处理器的并行总线。

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 GraphicLCDIntf

LCD 控制器和驱动器件通常集成到 LCD 面板中。它们包含或提供与帧缓冲器的接口，以显示和管理该缓冲器。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（地址）： 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脉冲宽度偏低所限（可在图形 LCD 控制器数据表中找到此值）。写入脉冲对于单时钟周期偏低，因此要设置时钟频率以满足此要求。
- 读取脉冲宽度高电平：配置选项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脉冲宽度高电平的要求。
- 读取脉冲宽度低电平：此参数与读取脉冲宽度高参数的设置方式相同。读取脉冲宽度低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)
- 读取脉冲宽度高：10 个时钟周期 (500 ns)
- 读取脉冲宽度低：10 个时钟周期 (500 ns)

### Solomon Systech SSD2119

- 时钟频率：25 MHz (40 ns)
- 读取脉冲宽度高：13 个时钟周期 (500 ns)
- 读取脉冲宽度低：13 个时钟周期 (500 ns)

### Himax HX8347A

- 时钟频率：28.5 MHz (35 ns)
- 读取脉冲宽度高：3 个时钟周期 (105 ns)
- 读取脉冲宽度低：11 个时钟周期 (385 ns)

### ILITEK ILI9325

- 时钟频率：20 MHz (50 ns)
- 读取脉冲宽度高：3 个时钟周期 (150 ns)
- 读取脉冲宽度低：3 个时钟周期 (150 ns)

### Epson S1D13743

- 时钟频率：33 MHz (33.3 ns)
- 读取脉冲宽度高：2 个时钟周期 (67 ns)
- 读取脉冲宽度低：5 个时钟周期 (167 ns)

# 输入/输出连接

本节介绍 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。

## clock

操作此组件的时钟。GraphicLCDIntf 完全是根据连接到组件的单个时钟操作的。

## 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。此信号表示高电平状态下的数据操作和低电平状态下的命令操作。

### ncs

有效低电平芯片选择。

### nwr

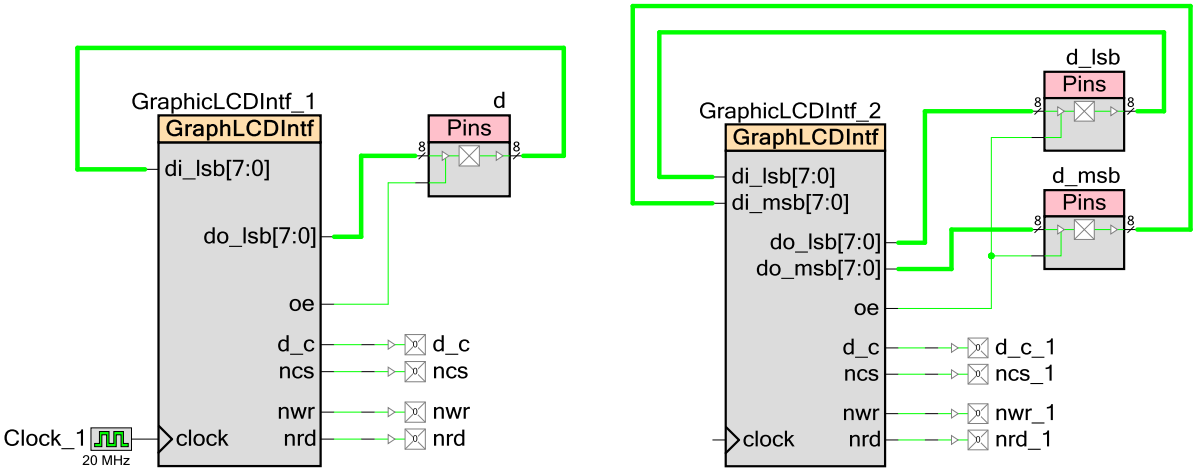
有效低电平写入控制信号。

### 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.

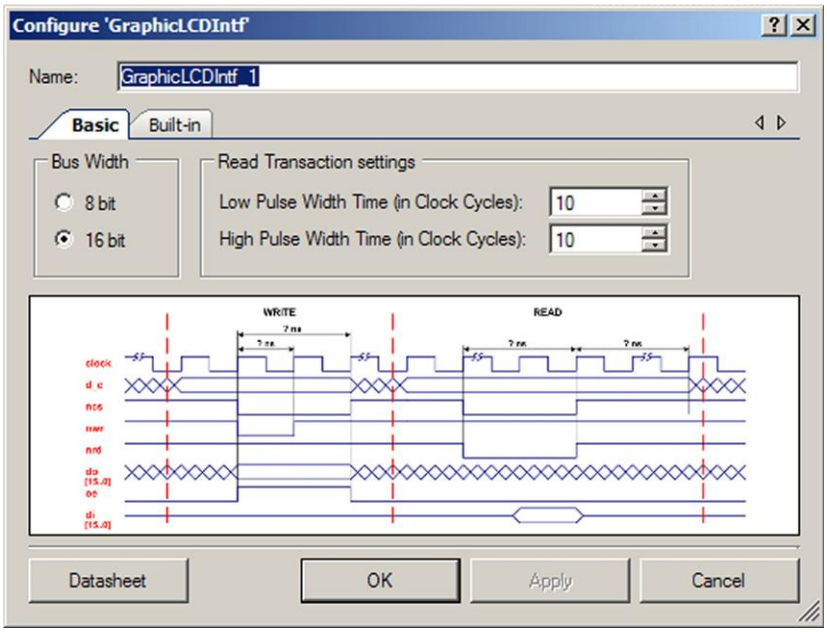



各个宏的时钟设置为 20 MHz，并将脉冲宽度设置为默认值。这是针对 SSD1289 控制器的正确设置。

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 生成。

# 元件参数

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。要使用 Solomon Systech SSD1289 控制器进行操作，默认 GraphicLCDIntf 设置就是正确的设置。




### Bus Width（总线宽度）

确定组件支持 8 位还是 16 位与图形 LCD 控制器的并行接口。默认设置为 **16 bit**（**16 位**）。

### Low Pulse Width Time（低脉冲宽度时间）

确定控制器的读取脉冲宽度低所需的时钟周期数。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（高脉冲宽度时间）

确定控制器的读取脉冲宽度偏高所需的时钟周期数。此值的范围为 1 到 255 个时钟周期。默认设置为 **10**。

# 时钟选择

此组件中没有内部时钟。您必须添加时钟源。此组件根据连接到组件的单时钟进行操作。

# 放置

GraphicLCDIntf 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 | 资源类型 | | | API Memory（API 存储器）（字节） | | Pins（引脚）（每个外部 I/O） |
|---|---|---|---|---|---|---|
| | Datapath 单元 | PLD | 状态单元 | Flash（闪存） | RAM | |
| 8 位接口 | 1 | 4 | 2 | 109 | 1 | 12 |
| 16 位接口 | 2 | 4 | 3 | 120 | 1 | 20 |

# 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“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_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。 |

| 函数 | 说明 |
|---|---|
| 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。<br>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，此函数使能活动模式电源模板位或关断时钟。配置组件以进行操作。

**参数：** None（无）

**Return Value（返回值）：** None（无）

**Side Effects（副作用）：** None（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关断时钟。

**参数：** None（无）

**Return Value（返回值）：** None（无）

**Side Effects（副作用）：** None（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。此写入是后写入写操作，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，直至有空间将此写入请求列入队伍。

**参数：** d_c：指示 Data (1) 或 Command (0)。传递到 d_c pin

data：在 do_lsb[7:0] 引脚上发送的数据

**Return Value（返回值）：** None（无）

**Side Effects（副作用）：** None（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。此写入是后写入写操作，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，直至有空间将此写入请求列入队伍。

**参数：** d_c：指示 Data (1) 或 Command (0)。传递到 d_c pin

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）

## 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 pin。

**Return Value（返回值）：** di_lsb[7:0] 引脚中的 8 位读取值

**Side Effects（副作用）：** None（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 pin。

**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 16 位读取值

**Side Effects（副作用）：** None（无）

## void GraphicLCDIntf_Sleep(void)

**说明：** 这是设置组件睡眠的首选子程序。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关断时钟。

在调用 GraphicLCDIntf_Sleep() 函数之前调用 CyPmSleep() 或 CyPmHibernate() 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 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，此函数启用活动模式电源模板位或关断时钟。

**参数：** None（无）

**Return Value（返回值）：** None（无）

**Side Effects（副作用）：** 调用 GraphicLCDIntf_Wakeup() 函数前未调用 GraphicLCDIntf_Sleep() 或 GraphicLCDIntf_SaveConfig() 函数可能会产生不可知的行为。

## void GraphicLCDIntf_Init(void)

**说明：** 此函数根据配置窗口 Configure（配置）对话框设置来初始化或恢复组件。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：** None（无）

**Return Value（返回值）：** None（无）

**Side Effects（副作用）：** 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组件操作。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：** None（无）

**Return Value（返回值）：** None（无）

**Side Effects（副作用）：** None（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（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 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：** None（无）

**Return Value（返回值）：** None（无）

**Side Effects（副作用）：** None（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：** | None（无） |
| **Return Value（返回值）：** | None（无） |
| **Side Effects（副作用）：** | 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。 |

# 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**（筛选选项）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

## 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（地址）：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（用于数据的同一个 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（可保留其上一个状态）
- ncs：1

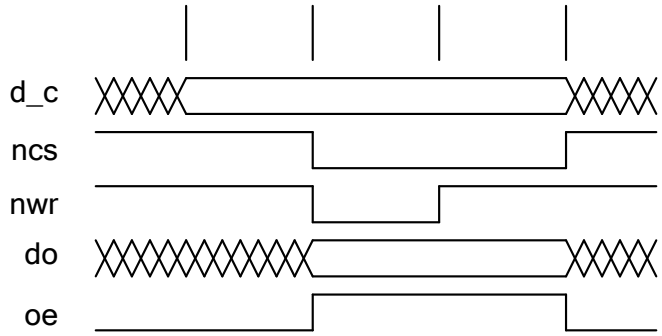
- nwr：1
- nrd：1
- do: 无关项（可保留其上一个状态）
- oe: 0

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. 数据写入操作时序图**



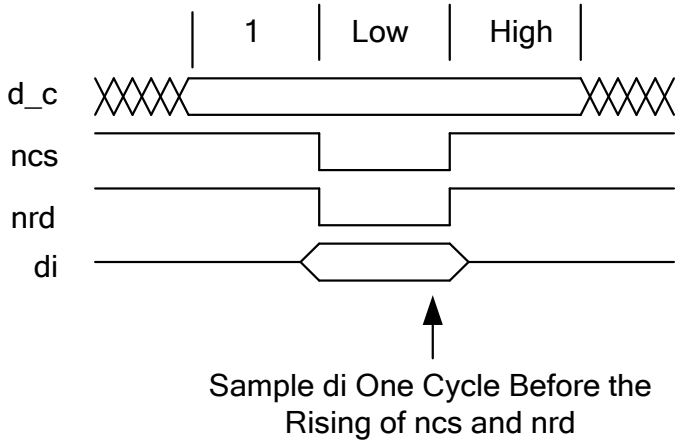

此图显示写入数据操作需要三个时钟循环。无论位宽如何，时序图都是相同的。此数据操作之前或之后可紧跟另一个读取或写入数据操作，或可在写入数据操作之前或之后处于空闲状态。

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）。这个实现方法允许 CPU 有两个待处理写入请求，而不会锁死。

## 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**



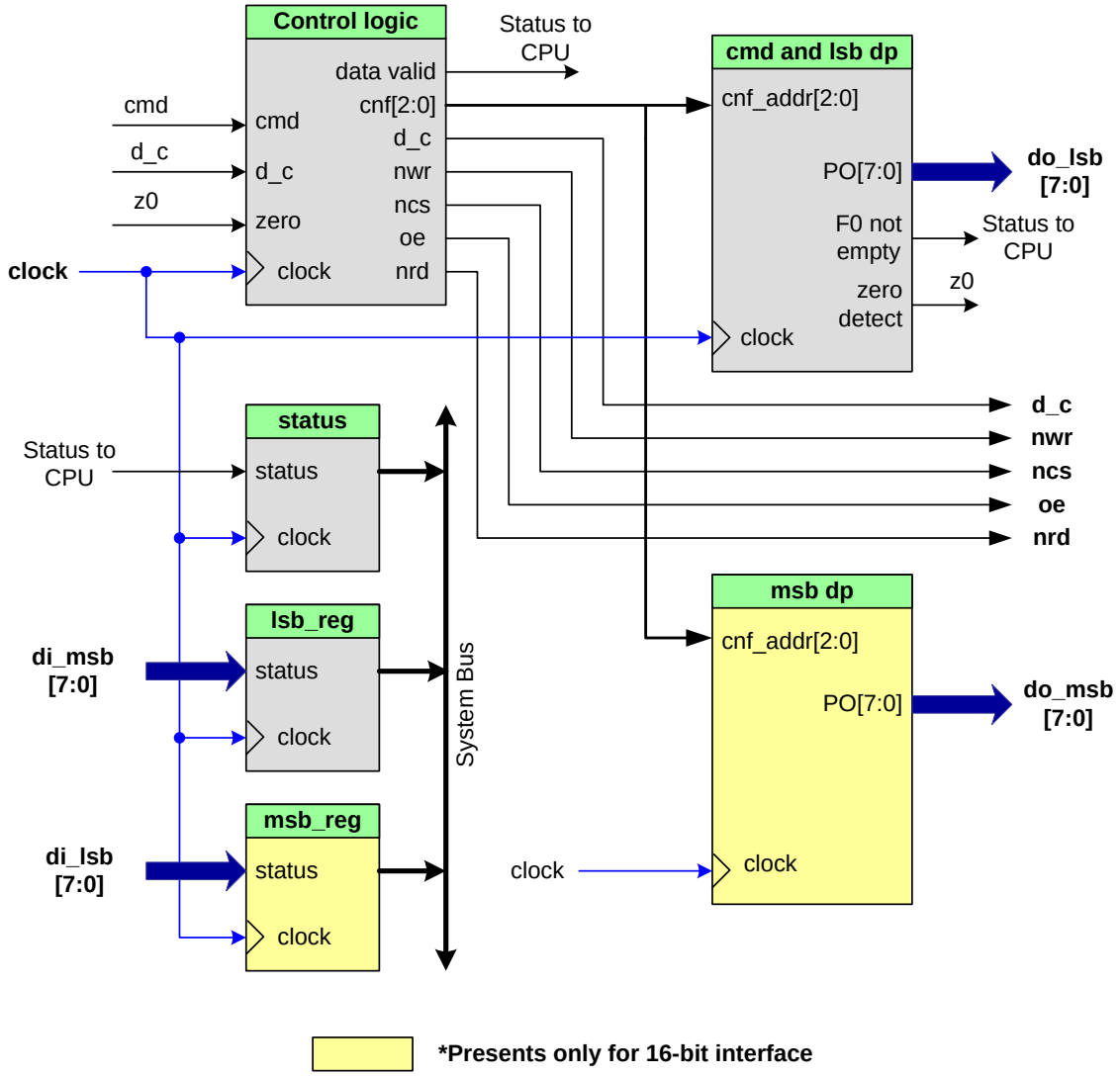

此图显示读取数据操作需要时钟循环的变量，这变量值取决于高和低读取脉冲宽度的设置。无论位宽如何，时序图都是相同的。注意，在 ncs 和 nrd 低电平脉冲结束之前的一个时钟周期内对输入数据进行采样。此数据操作之前或之后可紧跟另一个读取或写入数据操作，或可在读取数据操作之前或之后处于空闲状态。

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。

# 直流和交流电气特性

下面的值显示了预计性能，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 |
|---|---|---|---|---|---|
| $f_{CLOCK}$ | 组件时钟频率 | – | – | 33 | MHz |
| $t_{AS}$ | 地址设置时间 | 1 | – | – | $t_{CY\_clock}$ [2] |
| $t_{PWLW}$ | 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 | | | | |
| | 写循环 | 4 | – | – | $t_{CY\_clock}$ |
| | 读循环 | $t_{PWLR} + t_{PWRH} + 1$ | – | – | $t_{CY\_clock}$ |
| $t_{DSW}$ | 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 | – | $t_{CY\_clock}$ |

[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，但需要使用 STA 结果验证时序要求。

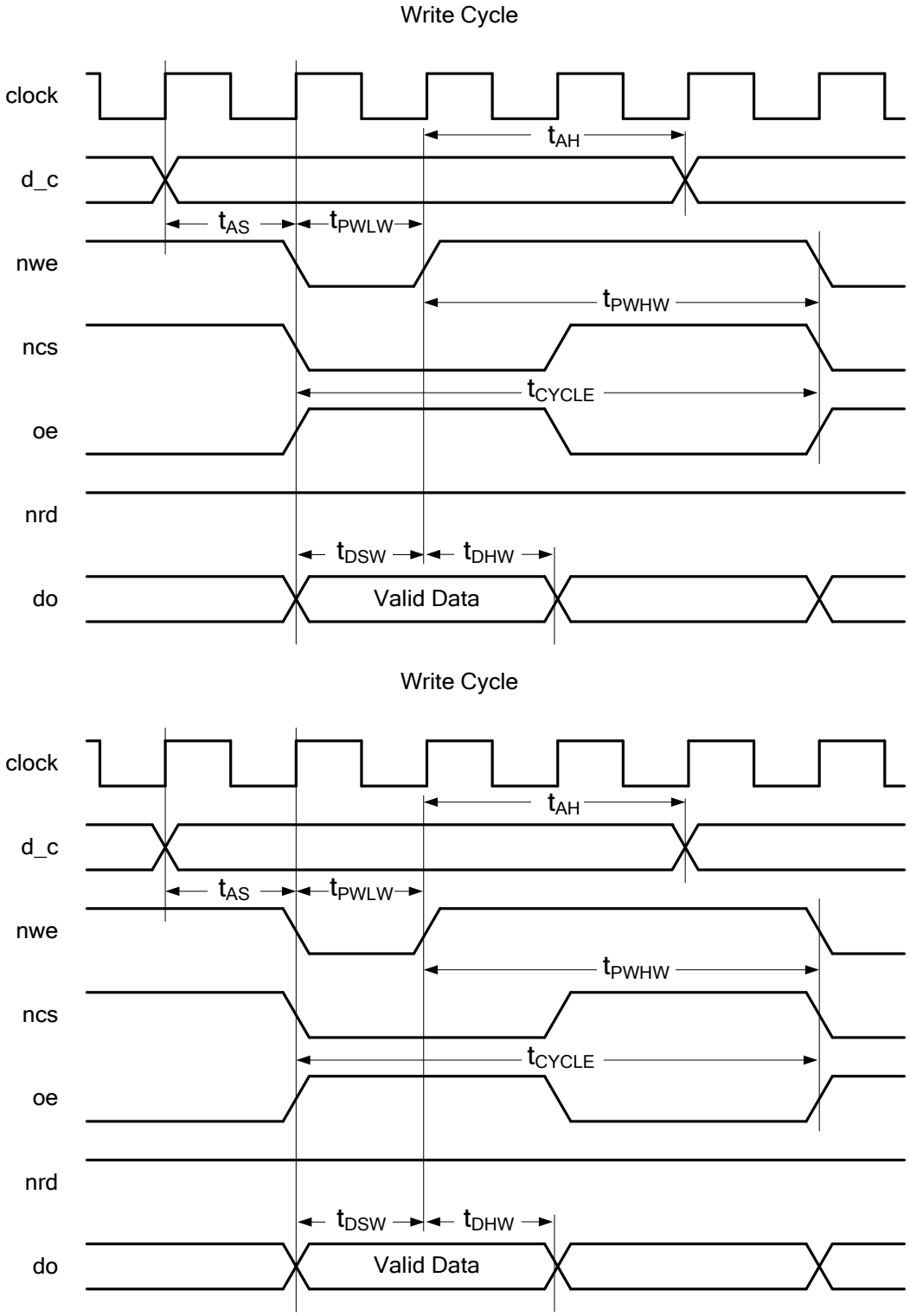
[2] $t_{CY\_clock} = 1/f_{CLOCK}$. 这是一个时钟周期的时间长度

## 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[3] | 单位 |
|---|---|---|---|---|---|
| $f_{CLOCK}$ | 组件时钟频率 | – | – | 25 | MHz |
| $t_{AS}$ | 地址设置时间 | 1 | – | – | $t_{CY\_clock}$ |
| $t_{PWLW}$ | 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 | 4 | – | – | $t_{CY\_clock}$ |
| | 读 | $t_{PWLR} + t_{PWRH} + 1$ | – | – | $t_{CY\_clock}$ |
| $t_{DSW}$ | 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 | – | $t_{CY\_clock}$ |

[3] “All Routing”（所有路由）的最大值意味着如果您的组件实例的运行速度等于或低于这些速度，则此组件无需考虑符合时序。

**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

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：

在命名了组件时钟的时钟汇总中的时序结果中显示了 **$f_{CLOCK}$** 最大组件时钟频率。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 | 57.019 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。这些参数可分为两个类别。

- 用于配置组件的参数：

  **$t_{PWLW}$** 写入信号的最小脉冲宽度低电平时间

  **$t_{PWLR}$** 读取信号的最小脉冲宽度低电平时间

  **$t_{PWHR}$** 读取信号的最小脉冲宽度高电平时间

  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。

- 这些参数是基于组件实现而修正的：

  **$t_{PWHW}$** 写入信号的最小脉冲宽度高电平时间

  **$t_{AS}$** 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

  **$t_{DHW}$** 在写入信号的上升沿之后，数据的最小有效时间

  **$t_{ACC}$** 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 | 添加了.cyre 文件中包括的所有带 CYREENTRANT 关键字的组件 API。 | 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候选函数。<br>需要此更改为采用安全方式使用（通过标志或关键节防止并发调用）并且不是可重入函数消除编译器警告。 |
| | 添加了时序限制，以对组件中的错误时序路径进行标记。 | 从时序分析中移除未使用的路径。这避免了错误时序冲突消息。 |
| 1.60.a | 从数据表中移除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。 | 允许组件使用对于所有 PSoC 3 和 PSoC 5 芯片都相同的时序结果进行操作。 |
| | 向数据表添加了特性数据 | |
| | 对数据表进行了少量编辑和更新 | |