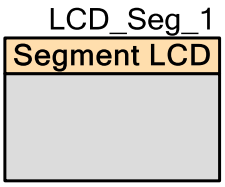
# 段 LCD (LCD_Seg)

3.0




## 特性

- 2 到 768 像素或符号
- 支持 1/3、1/4 和 1/5 偏置
- 10 Hz 到 150 Hz 刷新率
- 介于 2.0 V 和 5.2 V 之间的集成偏置生成，对于动态对比度控制，具有最高 128 级数字控制偏置电平
- 支持 A 型（标准）和 B 型（低功耗）波形
- 可反转显示屏的像素状态，从而显示反转图像
- 256 字节的显示存储器（帧缓冲区）
- 用户定义的像素或符号图，可使用 7 段、14 段或 16 段字符；5x7 或 5x8 点阵；条形图计算子程序。
- 支持 PSoC 3 ES3 芯片版本和之上版本

## 概述

段 LCD (LCD_Seg) 组件可在不同的电压下直接驱动各种 LCD 显示屏，复用率高达 16x。此组件提供一个简单的 PSoC 器件配置方法，以驱动您的自定义或标准显示屏。

内部偏置生成消除了任何外部硬件方面的需求，并允许进行基于软件的对比度调整。使用升压转换器时，显示屏偏置的电压可能比 PSoC 电源电压高。这样，便携应用中的显示屏灵活性就增加了。

各个 LCD 像素/符号处于打开或关闭状态。段 LCD 组件也提供高级支持，以简化显示屏中的以下显示结构类型：

- 7 段数字
- 14 段字母数字
- 16 段字母数字

- 5x7 和 5x8 点阵字母数字（5x7 和 5x8 使用的同一查找表。查找表中的所有符号的大小都是 5x7 像素。）
- 1 到 255 个元素的条形图

有关使用段 LCD 组件的更多信息，请参见应用笔记 *AN52927：PSoC® 3：段 LCD 直接驱动基础*。

### 何时使用段 LCD

需要在复用率高达 16x 的情况下，在不同的电压下直接驱动各种 LCD 显示屏时，使用直接段驱动 LCD 组件。直接段驱动 LCD 组件要求目标 PSoC 器件支持 LCD 直接驱动。

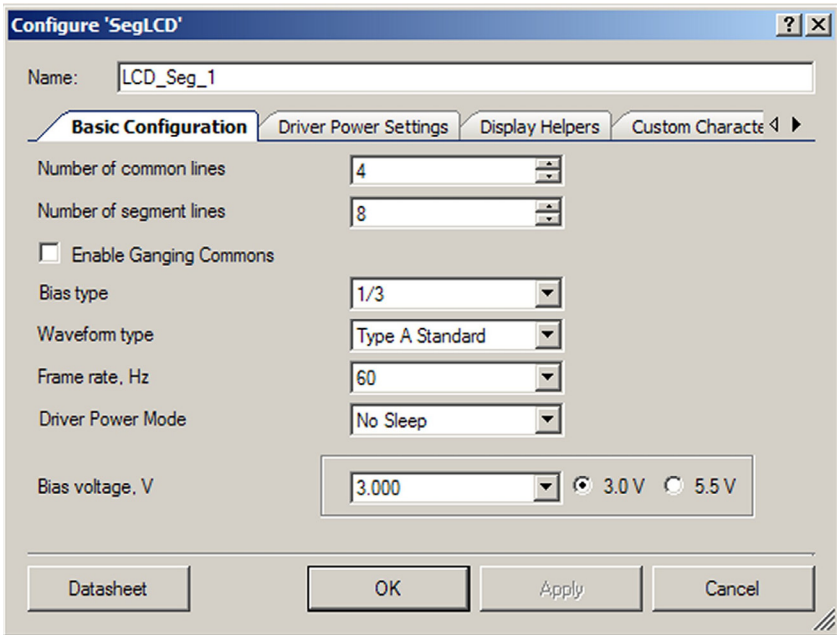
## 输入/输出连接

原理图上没有可见的组件连接；但可使用设计范围资源引脚编辑器将各种信号连接到引脚。

## 元件参数

将一个段 LCD 组件拖放到您的设计上，并双击以打开 **Configure**（配置）对话框。**Configure**（配置）对话框包含带不同参数类型的多个选项卡，用于设置段 LCD 组件。

### Basic Configuration（基本配置）选项卡



### Number of common lines（common线路数）

定义显示屏所需的 common 信号数（默认值为 4）。

### Number of segment lines（segment线路数）

定义显示屏所需的 segment 信号数。可能值的范围为从 2 到 62。默认值为 8。

### Enable Ganging Commons（启用组common信号）

选择此复选框以对 PSoC 引脚进行连接，以驱动 common 信号。针对各个 common 信号分配两个 PSoC 引脚。这用于驱动较大的显示屏。

### Bias type（偏置类型）

此值确定 common 线路和 segment 线路适当的偏置模式。

### Waveform type（波形类型）

此参数确定波形类型：A 型 — 单帧 0 VDC 平均值（默认）；B 型 — 双帧 0 VDC 平均值。

### Frame rate, Hz（帧率 (Hz)）

此参数确定显示屏的刷新率。在“无睡眠”模式中，可能值的范围为 10 Hz 到 150 Hz，以 10 为单位递增。默认值为 60 Hz。

在低功耗模式中，帧率选择对于每个配置都是有限制且唯一的。有关详细的说明，稍后请参见此数据手册中功能描述一节中的Driver Power Mode（驱动器功耗模式）。

### Driver Power Mode（驱动器功耗模式）

Driver Power Mode（驱动器功耗模式）参数定义组件的功耗模式。以下功耗模式设置可用：

- **No Sleep**（无睡眠）：LCD DAC 时钟处于打开状态，芯片将不会进入睡眠模式
- **Low Power using ILO**（使用 ILO 的低功耗）：LCD DAC 打开，但芯片将在电压操作之间进入睡眠状态。作为唤醒源，组件将使用 1 kHz 内部 ILO。
- **Low Power using Ext 32kHz crystal**（使用外部 32kHz 晶振的低功耗）：LCD DAC 打开，但芯片将在电压操作之间进入睡眠状态。作为唤醒源，组件将从 OPPS 定时器中使用 8 K 节拍。

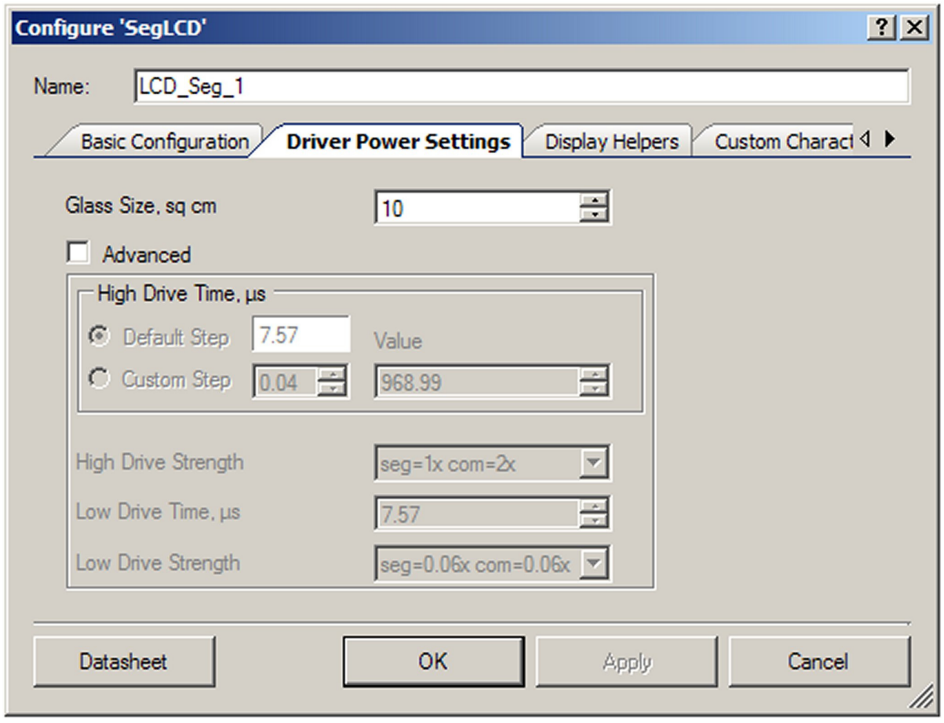
**注意**根据使用的低功耗模式，使用设置为 1 kHz 的 ILO 或已连接并启用的外部 32 kHz 晶振。可在设计范围资源时钟编辑器中启用 32 kHz 晶振或设置 ILO 频率。

稍后另请参见此数据手册中功能描述一节中的Driver Power Mode（驱动器功耗模式）。

### Bias voltage,V（偏置电压（V））

此参数定义 LCD DAC 的偏置电压电平。可能值的范围为 2.35 V 到 5.5 V 或 2.017 V 到 3 V，这取决于电源。

## Driver Power Settings（驱动器功耗设置）选项卡




### Glass Size, sq cm（显示屏大小（平方厘米））

使用此字段输入显示屏活动区域的正确大小（单位为平方厘米）。此值将与 **Number of common lines**（common 线路数）参数一同用于计算正确的电容负载。

### Advanced（高级）

如果选定了此复选框，您可手动更改 LCD 的 **Driver Power Settings**（驱动器功耗设置）。如果未选中此复选框，将基于 **Basic Configuration**（基本配置）选项卡和 **Glass Size**（显示屏大小）参数中的选择自动设置驱动器功耗设置。

### High Drive Time,µs（高驱动时间（µs））

此参数定义在一次电压数据操作中 High Drive（高驱动）的有效时间。

### Default Step（默认步骤）

此参数定义针对选定配置的自动计算的高驱动步骤。如果选择了此参数，则此步骤将递增/递减高驱动时间值。

默认高驱动步长是“无睡眠”模式的仅有可选项。

### Custom Step（自定义步长）

仅在低功耗模式下可用。定义用户可选择的高驱动步长。自定义高驱动步长的最小值取决于主时钟的频率。最大值由 **Default Step**（默认步长）的值限制。

**注意**如果更改了 **Frame rate**（帧率）、**Number of common lines**（common 线路数）或 **Waveform type**（波形类型）参数，**High Drive Time** （高驱动时间）参数将自动设为帧默认值的一半（仅在选定了 **Advanced**（高级）复选框时）：

HighDriveTimeValdef= 1.075/DefaultStep × 128 （A 型）

HighDriveTimeValdef= 1.075/DefaultStep × 128 （B 型）

**High Drive Time**（高驱动时间）最小值的计算将稍后在此数据手册中定义。当前配置中的 **High Drive Time**（高驱动时间）最大值为：

HighDriveTimeValmax= 1.075/DefaultStep × 253 （A 型）

HighDriveTimeValmax= 1.075/DefaultStep × 253 （B 型）

上述等式也可利用 **Custom Step**（自定义步长）应用于 **High Drive Time** （高驱动时间）值。

如果选定了某个低功耗模式，使用默认高驱动步长时，不建议将 **High Drive Time**（高驱动时间）设置为其最大值。使用低功耗模式效率很低，因为器件将仅进入睡眠模式一小段时间，在最差的情况下，可能根本不会进入睡眠模式。利用自定义步长，可更精确的调节 **High Drive Time**（高驱动时间）值，这是许多低功耗应用的要求。

### High Drive Strength（高驱动强度）

此参数选择 High Drive（高驱动）的驱动强度。

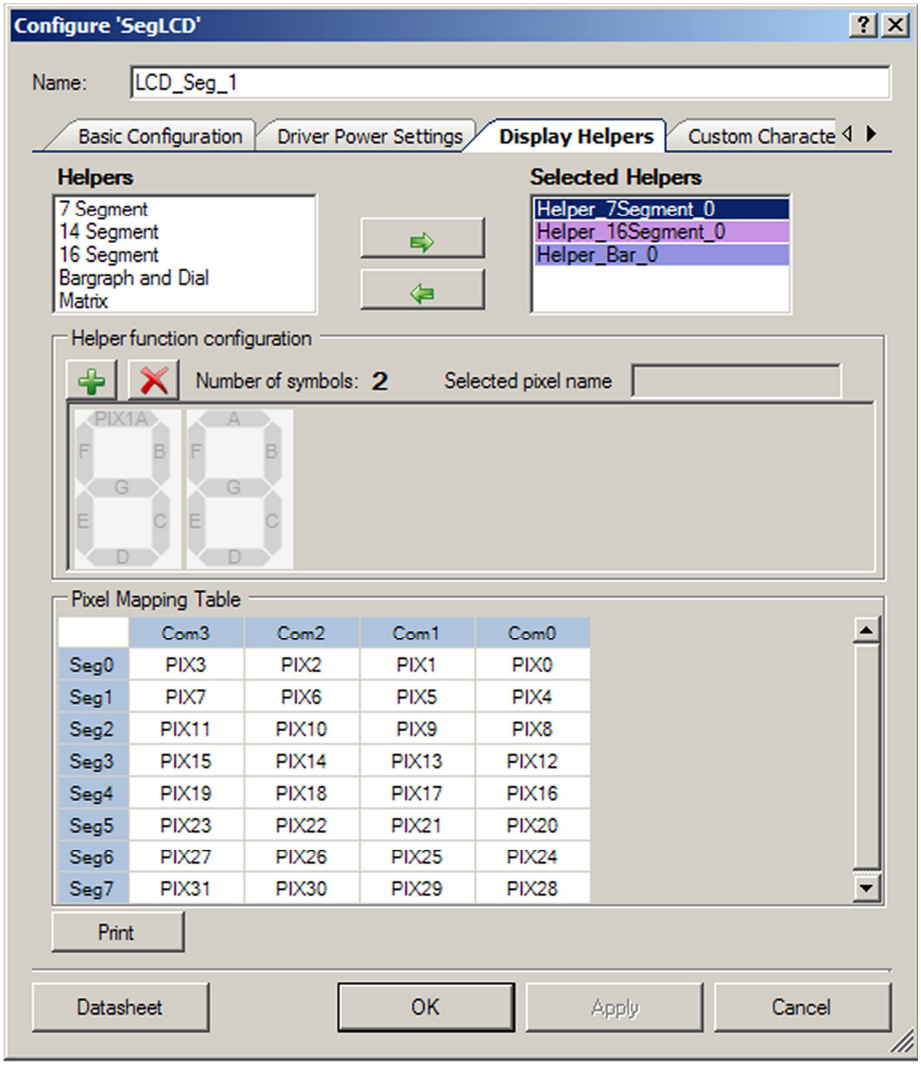
### Low Drive Time, µs（低驱动时间 (µs)）

低驱动时间参数定义在一次电压数据操作中低驱动模式的有效时间。

### Low Drive Strength （低驱动强度）

此参数选择 Low Drive（低驱动）的驱动强度。**Low Drive Strength**（低驱动强度）与 **High Drive Strength**（高驱动强度）相对，它是基于 High Drive（高驱动）而自动设置的。

## Display Helpers（显示助手）选项卡




显示助手允许您配置一组一同使用显示屏段，作为多个预定义显示元素类型之一：

- 7 段、14 段 或 16 段显示屏
- 点阵显示屏（5x7 或 5x8）
- 线性或圆形条形图显示屏

基于字符的显示助手可用于组合多个显示符号，以创建多字符显示元素。

### Helpers/Selected Helpers（助手/选定助手）

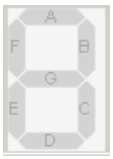
可通过在 **Helpers**（助手）列表中选择所需的助手类型并单击右箭头按钮，向 **Selected Helpers**（选定助手）添加一个或多个助手。如果没有足够引脚支持新助手，则将不添加助手。要删除助手，在 **Selected Helpers**（选定助手）列表中选择该助手，并单击左箭头按钮。

**注意：** 请务必注意，在定义任何显示助手之前，为组件设置 common 线路和 segment 线路的数量。更改 common 线路和 segment 线路的数量之前，必须移除任何已定义的显示助手，因为您会丢失助手配置信息。如果试图更改 common 线路和 segment 线路的数量，将显示警告，指示助手像素映射配置将丢失。

**Selected Helpers**（选定助手）在列表中显示的顺序是很重要的。默认情况下，添加到 **Selected Helper**（选定助手）列表的给定类型的第一个助手的名称带后缀 0，同类型的下一个助手名称带后缀 1，以此类推。如果已从列表中移除了 **Selected Helper**（选定助手），将不会重命名剩下的助手。添加了助手后，其名称将使用最低的可用后缀。

为各个助手提供了 API。有关更多信息，请参见应用程序编程接口一节。

- **7 段助手 —** 此助手长度可能为 1 到 5 位，可显示十六进制位 0 到 F，或十进制 16 位无符号整数 (uint16) 值。助手函数不支持小数点。



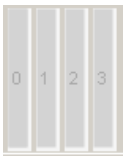

- **14 段助手 —** 此助手长度可高达 20 个字符。它会显示单个 ASCII 字符或以空字符结尾的字符串。可能的值为标准 ASCII 可打印字符（代码范围为 0 到 127）




- **16 段助手 —** 此助手长度可高达 20 个字符。它会显示单个 ASCII 字符或以空字符结尾的完整字符串。可能的值为标准 ASCII 字符和扩展码表（代码范围为 0 到 255）。不提供扩展码表。




- **条形图和拨号助手 —** 这些助手用于条形图和刻度盘（段数为 1 到 255）。条形图可能是单个选定像素或指定像素左侧或右侧的所有像素的选定像素




- **矩阵助手 —** 此助手支持高达八个字符元素。组件支持 x5x7 或 x5x8 行/列字符。可通过配置两个或更多点阵助手以创建更长的字符串，以定义显示屏的相邻点阵部分。助手会显示单个 ASCII 字符或以空字符结尾的字符串。

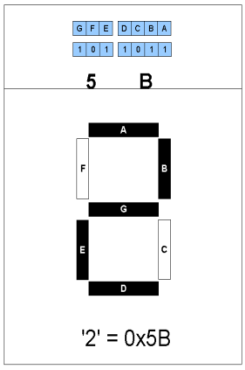


点阵助手具有引脚分布限制。针对矩阵行，点阵助手必须使用 7 到 8 个连续 common 驱动；针对矩阵列，则必须使用 5 到 40 个连续 segment 驱动。组件支持标准 Hitachi HD44780 字符集。

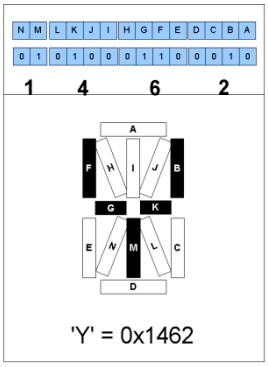
### Character Encoding（字符编码）

所有高级助手 API 都有自己的查找表。表中包括一个编码像素状态集合，它构建了特定字符映像。以下范例显示特定字符的编码方式（segment 名称可能与 **Configure**（配置）对话框中显示的不同）。

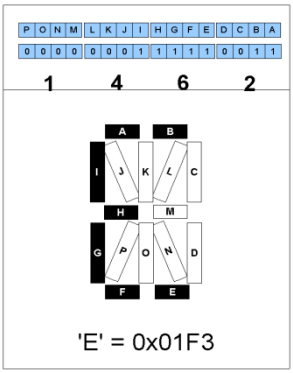
**7 段编码**



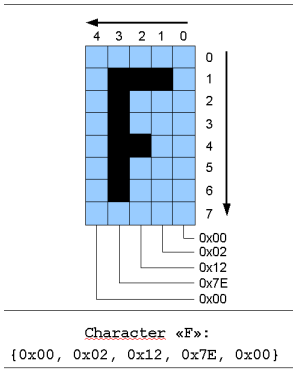

**14 段编码**




**16 段编码**




**点阵编码**




### Helper function configuration（助手函数配置）

对话框的此部分允许您配置助手，包括向/从助手中添加或移除符号，以及命名像素。

1. 从 **Selected Helpers**（选定助手）列表中选择助手。
2. 单击 **[+]** 或 **[x]** 按钮添加或移除选定助手的符号。

可添加的最大符号数取决于助手类型和组件支持的总像素数。如果可用引脚数量不足以支持新符号，则不添加新符号。

3. 重命名一个像素是助手功能的一部分，在 **Helper function configuration**（助手函数配置）显示屏中的符号图像上选择像素。当前名称将显示在选定像素名称字段中，并可按需要进行修改。

### Pixel Naming（像素命名）

默认像素名称的格式为“PIX#”，其中“#”是 **Pixel Mapping Table**（像素映射表）右上角以递增顺序排列的像素数。

与助手符号相关联的像素的默认命名具有不同的格式。默认名称包含前缀部分、符号中所有像素的通用部分，以及唯一的段标识符。默认前缀指示助手类型和符号实例。例如，7 段显示助手中某符号中的像素默认名称可能是“H7SEG4_A”，其中：

| | |
|---|---|
| H7 | 说明像素是 7 段助手的一部分 |
| SEG4 | 说明像素是被指定为项目中第四个 7 段符号的符号的一部分 |
| A | 指示 7 段符号中的唯一段 |

对于默认像素名称，只有像素名称的唯一部分显示在符号图像中。如果修改了像素名称，则即使修改前和修改后的名称都有共同的前缀，符号图像上也将显示整个名称。

**注意**所有像素名称必须是唯一的。

当将助手函数符号元素分配给 **Pixel Mapping Table**（像素映射表）（如下所述）中的一个像素时，此像素使用助手符号元素的名称。助手符号元素名称取代默认像素名称，但不会替换它。您可重新使用与助手函数相关联的像素的默认名称。

### Pixel Mapping Table（像素映射表）

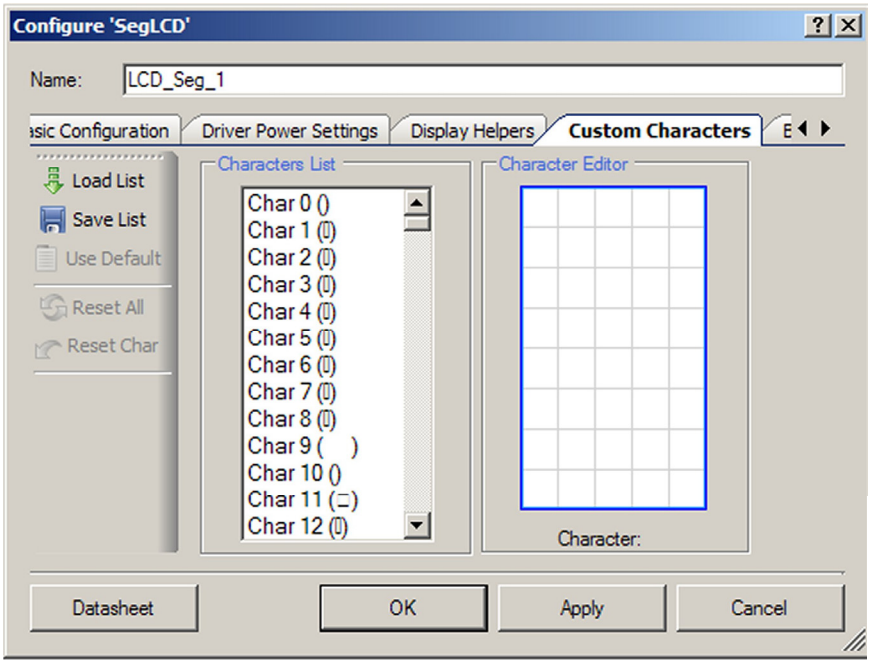
**Pixel Mapping Table**（像素映射表）是帧缓冲区的展示。要使 API 函数正常工作，**Helper function configuration**（助手函数配置）中的各个像素必须分配到 **Pixel Mapping Table**（像素映射表）中的像素位置。有关进行正确分配所需的信息，请参见 你的 LCD 显示屏的数据手册。

要分配像素，在 **Helper function configuration**（助手函数配置）面板上选择所需的像素，并将其拖到 **Pixel Mapping Table**（像素映射表）中正确的位置。

可通过双击表中的像素并输入所需名称，以在 **Pixel Mapping Table**（像素映射表）中重命名像素。此方法可用于对与某可用助手类型不关联的像素进行命名。

**Print**（打印）按钮用于打印 **Pixel Mapping Table**（像素映射表）。

### Custom Characters（自定义字符）选项卡




使用此选项卡，您可为 5x8 点阵显示屏创建自定义字符。也可用其将自定义字符查找表作为 XML 字符串进行存储。

默认情况下，**Characters List**（字符列表）字段包含 255 个 ASCII 字符，这些字符具有作为标准 Hitachi HD44780 字符集的映像。可使用 **Character Editor**（字符编辑器）访问和修改这些字符。

使用 **Reset Char**（复位字符）选项，可将未保存的字符复位到默认映像。使用 **Reset All** （全部复位）选项，可将所有未保存字符复位到标准映像。

保存您自己的字符集之后，可用 **Save List** （保存列表）命令将其另存为 XML 字符串。使用 **Load List**（加载列表）命令，可从 XML 字符串中加载您的字符列表。可使用 **Use Default**（使用默认值）选项返回到标准字符集。

## 时钟选择

LCD_Seg 组件使用内部时钟，无需外部时钟。一旦安装了组件，时钟自动专用于 LCD 组件。将自动计算其频率，这取决于 common 线路数、刷新率和波形类型。

## 放置

LCD_Seg 组件实现包含两个部分。LCDDAC 是此组件使用的 PSoC 中的固定功能硬件。驱动信号的其他时序逻辑在 UDB 中实现。在项目生成流程中，UDB 资源将自动放置在 UDB 阵列中。

**注**：一个项目中仅可使用一个组件实例。如果尝试使用更多实例，将在构建过程中生成放置错误。

默认引脚分配是在构建过程中进行的，可使用 PSoC Creator 设计范围资源工具中的引脚编辑器对其进行修改。

# 资源

下表显示静态段 LCD 组件的所有可能的配置。配置名称的含义如下：

**Basic**（基本）：低级 API 函数集，无任何高级助手 API

**Basic, 7-Segment helper**（基本，7 段助手）：低级 API 函数集 + 7 段助手 API

**Basic, 14-Segment helper**（基本，14 段助手）：低级 API 函数集 + 14 段助手 API

**Basic, 16-Segment helper**（基本，16 段助手）：低级 API 函数集 + 16 段助手 API

**Basic, Dot Matrix helper**（基本，点阵助手）：低级 API 函数集 + 点阵助手高级 API

**Basic, Bar Graph helper**（基本，条形图助手）：低级 API 函数集 + 条形图助手高级 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 | LCD 固定模块 | Control/Count7 单元 | 同步单元 | 中断 | Flash（闪存 | RAM | |
| 基本 | 1 | 3 | 1 | 1 | 0 | 1 | 2691 | 104 | 3 到 62 |
| 基本，7 段助手 | 1 | 3 | 1 | 1 | 0 | 1 | 2870 | 129 | 3 到 62 |
| 基本，14 段助手 | 1 | 3 | 1 | 1 | 0 | 1 | 3711 | 409 | 3 到 62 |
| 基本，16 段助手 | 1 | 3 | 1 | 1 | 0 | 1 | 3812 | 409 | 3 到 62 |
| 基本，点阵助手 | 1 | 3 | 1 | 1 | 0 | 1 | 4875 | 409 | 3 到 62 |
| 基本，条形图助手 | 1 | 3 | 1 | 1 | 0 | 1 | 3590 | 409 | 3 到 62 |

# 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“LCD_Seg_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 PSoC Creator 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“LCD_Seg”。

| 函数 | 说明 |
|---|---|
| LCD_Seg_Start() | 设置 initVar 变量，调用 LCD_Seg_Init() 函数，然后调用 LCD_Seg_Enable() 函数。 |
| LCD_Seg_Stop() | 禁用 LCD 组件、相关联的中断和 DMA 通道。 |
| LCD_Seg_EnableInt() | 启用 LCD 中断。如果调用了 LCD_Seg_Start()，则无需此函数 |
| LCD_Seg_DisableInt() | 禁用 LCD 中断。如果调用了 LCD_Seg_Stop()，则无需此函数 |
| LCD_Seg_SetBias() | 将 LCD 显示屏的偏置电平设为 128 个值中的其中一个。 |
| LCD_Seg_WriteInvertState() | 基于输入参数反转显示屏。 |
| LCD_Seg_ReadInvertState() | 返回显示屏反转状态的当前值：正常或反转 |
| LCD_Seg_ClearDisplay() | 清除显示屏和关联帧缓冲区 RAM。 |
| LCD_Seg_WritePixel() | 基于 PixelState（像素状态）设置或清除像素 |
| LCD_Seg_ReadPixel() | 读取帧缓冲区中像素的状态。 |
| LCD_Seg_Sleep() | 停止 LCD 并保存用户配置。 |
| LCD_Seg_Wakeup() | 恢复并启用用户配置。 |
| LCD_Seg_SaveConfig() | 保存 LCD 配置。 |
| LCD_Seg_RestoreConfig() | 恢复 LCD 配置。 |
| LCD_Seg_Init() | 根据“Configure”（配置）对话框设置初始化或存储 LCD。 |
| LCD_Seg_Enable() | 启用 LCD。 |

## 全局变量

| 变量 | 说明 |
|---|---|
| LCD_Seg_initVar | LCD_Seg_InitVar 说明段 LCD 是否已初始化。变量将初始化为 0，并在第一次调用 LCD_Seg_Start() 时设置为 1。这样，第一次调用 LCD_Seg_Start() 子程序后，组件不用重新初始化即可重启。<br>如需重新初始化组件，可在 LCD_Seg_Start() 或 LCD_Seg_Enable() 函数前调用 |

| | |
|---|---|
| | LCD_Seg_Init() 函数。 |

## uint8 LCD_Seg_Start(void)

**说明：** 此函数启动 LCD 组件并启用所需中断、DMA 通道、帧缓冲区和硬件。切勿清除帧缓冲区 RAM。

**参数：** None（无）

**Return Value（返回值）：** uint8 cystatus：标准 API 返回值。

| 返回值 | 说明 |
|---|---|
| CYRET_LOCKED | 一些 DMA TD 或某个通道已被使用。 |
| CYRET_SUCCESS | 函数成功完成 |

**Side Effects（副作用）：** 此 API 从时轮配置寄存器 2 中启用每秒一脉冲位。仅在低功耗 32kHz 外部 Xtal 组件模式下才会出现此情况。

## void LCD_Seg_Stop(void)

**说明：** 此函数禁用 LCD 组件、相关联的中断和 DMA 通道。自动清空显示屏，以避免因直流偏移而产生损害。切勿清除帧缓冲区。

**参数：** None（无）

**Return Value（返回值）：** None（无）

**Side Effects（副作用）：** 此 API 不从之前由 LCD_Seg_Init() API 启用的时轮配置寄存器 2 中清除每秒一脉冲位。仅在低功耗 32kHz 外部 Xtal 组件模式下才会出现此情况。

## void LCD_Seg_EnableInt(void)

**说明：** 此函数启用 LCD 中断。如果调用了 LCD_Seg_Start()，则无需此函数。每次 LCD 更新后都会出现中断（TD 完成）

**参数：** None（无）

**Return Value（返回值）：** None（无）

**Side Effects（副作用）：** None（无）

## void LCD_Seg_DisableInt(void)

**说明：** 此函数禁用 LCD 中断。如果调用了 LCD_Seg_Stop()，则无需此函数。

**参数：** None（无）

**Return Value（返回值）：** None（无）

**Side Effects（副作用）：** None（无）

## void LCD_Seg_SetBias(uint8 biasLevel)

**说明：** 此函数将 LCD 显示屏的偏置电平设为 128 个值中的其中一个。值的实际数量受模拟电源电压 ($V_{DDA}$) 的限制。偏置电压不可超过 $V_{DDA}$。更改偏置电压会影响 LCD 对比度。

**参数：** uint8 biasLevel：显示屏的偏置电平

**Return Value（返回值）：** None（无）

**Side Effects（副作用）：** None（无）

## uint8 LCD_Seg_WriteInvertState(uint8 invertState)

**说明：** 此函数基于输入参数反转显示屏。反转在硬件中发生，无需对帧缓冲区中的显示屏 RAM 进行更改。

**参数：** uint8 invertState：设置显示屏的反转状态

**Return Value（返回值）：** uint8 cystatus：标准 API 返回值

**Side Effects（副作用）：** None（无）

## uint8 LCD_Seg_ReadInvertState(void)

**说明：** 此函数返回显示屏反转状态的当前值：正常或反转。

**参数：** None（无）

**Return Value（返回值）：** uint8 invertState：显示屏的反转状态

**Side Effects（副作用）：** None（无）

## void LCD_Seg_ClearDisplay(void)

**说明：** 此函数清除显示屏和关联帧缓冲区 RAM。

**参数：** None（无）

**Return Value（返回值）：** None（无）

**Side Effects（副作用）：** None（无）

## uint8 LCD_Seg_WritePixel(uint16 pixelNumber, uint8 pixelState)

**说明：** 此函数基于输入参数“PixelState”设置或清除像素。通过 pixelNumber 对此像素进行寻址。

**参数：** uint16 pixelNumber：指向帧缓冲区中像素位置。LSB 低位半字节的最低三位是字节中的比特位置，LSB 高位半字节低（四位）是复用器行中的字节地址，MSB 低位半字节高（四位）是复用器行数。生成的组件 .h 文件包括各像素此格式的 #defines。

uint8 pixelState：指定的 pixelNumber 设为此像素状态。

**Return Value（返回值）：** uint8 status：基于字节地址和复用器行数的范围检查，为通过或失败。不对比特位置进行检查。

**Side Effects（副作用）：** None（无）

## uint8 LCD_Seg_ReadPixel(uint16 pixelNumber)

**说明：** 此函数读取帧缓冲区中像素的状态。通过 pixelNumber 对此像素进行寻址。

**参数：** uint16: pixelNumber：指向帧缓冲区中像素位置。LSB 低位半字节的最低三位是字节中的比特位置，LSB 高位半字节低（四位）是复用器行中的字节地址，MSB 低位半字节高（四位）是复用器行数。生成的组件 .h 文件包括各像素此格式的 #defines。

**Return Value（返回值）：** uint8 pixelState：返回指定 PixelNumber 的当前状态

| 值 | 说明 |
|---|---|
| 0x00 | 像素为关闭状态。 |
| 0x01 | 像素为打开状态。 |
| 0xFF | 像素未分配。 |

**Side Effects（副作用）：** None（无）

## void LCD_Seg_Sleep(void)

**说明：** 这是准备组件睡眠的首选子程序。LCD_Seg_Sleep() 子程序保存当前组件的状态。然后该函数调用 LCD_Seg_Stop() 函数和 LCD_Seg_SaveConfig() 来保存硬件配置。

在调用 CyPmSleep() 或 CyPmHibernate() 函数之前调用 LCD_Seg_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）：** 不更改组件引脚的驱动模式。

## void LCD_Seg_Wakeup(void)

**说明：** 该函数是将组件恢复到调用 LCD_Seg_Sleep() 时状态的首选子程序。LCD_Seg_Wakeup() 函数调用 LCD_Seg_RestoreConfig() 函数，以恢复配置。如果组件在调用 LCD_Seg_Sleep() 函数前已启用，则 LCD_Seg_Wakeup() 函数也将重新启用组件。

**参数：** None（无）

**Return Value（返回值）：** uint8 cystatus：标准 API 返回值

| 返回值 | 说明 |
|---|---|
| CYRET_LOCKED | 一些 DMA TD 或某个通道已被使用 |
| CYRET_SUCCESS | 函数成功完成 |

**Side Effects（副作用）：** 调用 LCD_Seg_Wakeup() 函数前未调用 LCD_Seg_Sleep() 或 LCD_Seg_SaveConfig() 函数可能会产生意外行为。

## void LCD_Seg_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 LCD_Seg_Sleep() 函数调用。

**参数：** None（无）

**Return Value（返回值）：** None（无）

**Side Effects（副作用）：** None（无）

### void LCD_Seg_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 LCD_Seg_Sleep() 函数之前的值。

**参数：** None（无）

**Return Value（返回值）：** None（无）

**Side Effects（副作用）：** 调用该函数前未调用 LCD_Seg_Sleep() 或 LCD_Seg_SaveConfig() 函数可能会产生意外行为。

### void LCD_Seg_Init(void)

**说明：** 根据“Configure”（配置）对话框设置来初始化或恢复组件参数。无需调用 LCD_Seg_Init()，因为 LCD_Seg_Start() 子程序会调用该函数并是开始组件操作的首选方法。配置和启用所有必要硬件模块，并清除帧缓冲区。

**参数：** None（无）

**Return Value（返回值）：** None（无）

**Side Effects（副作用）：** 所有寄存器将根据“Configure”（配置）对话框设置为相应的值。此 API 从时轮配置寄存器 2 中启用每秒一脉冲位。仅在低功耗 32 -kHz 外部 Xtal 组件模式下才会出现此情况。

### void LCD_Seg_Enable(void)

**说明：** 启用 LCD 固定硬件的电源，并启用 UDB 信号生成。

**参数：** None（无）

**Return Value（返回值）：** uint8 cystatus：标准 API 返回值

| 返回值 | 说明 |
|---|---|
| CYRET_LOCKED | 一些 DMA TD 或某个通道已被使用。 |
| CYRET_SUCCESS | 函数成功完成 |

**Side Effects（副作用）：** None（无）

## 可选助手 API

只有在“Configure”（配置）对话框中选定了相应助手后，才会显示以下 API。

| 函数 | 说明 |
|---|---|
| LCD_Seg_Write7SegDigit_n | 显示在 7 段显示元素阵列上的十六进制数字。 |
| LCD_Seg_Write7SegNumber_n | 显示 7 段显示元素的 1 位到 5 位阵列上的整数。 |
| LCD_Seg_WriteBargraph_n | 显示线性或圆形条形图上的整数位置。 |
| LCD_Seg_PutChar14Seg_n | 显示 14 段字母数字字符显示元素阵列上的字符。 |
| LCD_Seg_WriteString14Seg_n | 显示 14 段字母数字字符显示元素阵列上以空字符作为结尾的字符串。 |
| LCD_Seg_PutChar16Seg_n | 显示 16 段字母数字字符显示元素阵列上的字符。 |
| LCD_Seg_WriteString16Seg_n | 显示 16 段字母数字字符显示元素阵列上以空字符作为结尾的字符串。 |
| LCD_Seg_PutCharDotMatrix_n | 显示点阵字母数字字符显示元素阵列上的字符。 |
| LCD_Seg_WriteStringDotMatrix_n | 显示点阵字母数字字符显示元素阵列上以空字符作为结尾的字符串。 |

**注意**包含后缀“n”的函数名称表示在组件定制器中创建了多个相同符号类型的显示助手。特定显示助手元素由 API 函数控制，函数名称各自带有“n”后缀。

### void LCD_Seg_Write7SegDigit_n(uint8 digit, uint8 position)

**说明：** 此函数显示在 7 段显示元素阵列上的十六进制数字。数字可以为 0 到 9 和 A 到 F 范围内的十六进制值。必须使用定制器显示助手工具定义与 7 段显示元素相关联的像素集。可在帧缓冲区中定义多个 7 段显示元素，并可通过函数名称中的后缀 (n) 对这些元素进行寻址。只有在组件定制器中定义了 7 段显示元素时，此函数才可用。

**参数：** uint8 digit：要显示为十六进制数字的未分配整数值（范围为 0 到 15）

uint8 position：从右侧的 0 开始从右到左计算的数字位置。如果此位置不在定义的显示区域内，将不显示此字符。

**Return Value（返回值）：** None（无）

**Side Effects（副作用）：** None（无）

### void LCD_Seg Write7SegNumber_n(uint16 value, uint8 position, uint8 mode)

**说明：** 此函数显示 7 段显示元素的 1 位到 5 位阵列上的 16 位整数。必须使用定制器显示助手工具定义与 7 段显示元素相关联的像素集。可在帧缓冲区中定义多个 7 段显示元素组，并可通过函数名称中的后缀 (n) 对这些元素组进行寻址。必须由应用程序特定的用户代码处理符号转换、符号显示、小数点和其他自定义功能。只有在组件定制器中定义了 7 段显示元素时，此函数才可用。

**参数：** uint16 value：要显示的未分配整数值。

uint8 position：从右侧的 0 开始从右到左计算的最低有效位位置。如果定义的显示区域包含的位数少于值需要的位数，则将不显示最高有效位或多个位。

uint8 mode：设置显示模式。可为 0 或 1。

| 值 | 说明 |
|---|---|
| 0 | 不显示前导零。 |
| 1 | 显示前导零 |

**Return Value（返回值）：** None（无）

**Side Effects（副作用）：** None（无）

## void LCD_Seg_WriteBargraph_n(uint8 location, uint8 mode)

**说明：** 此函数显示 1 到 255 段条形图（从左到右编号）上的 8 位整数位置。条形图可以是任意用户定义的大小（1 到 255 段）。也可在圆圈中创建条形图，以显示旋转位置。必须使用定制器显示助手工具定义与条形图显示元素相关联的像素集。可在帧缓冲区中定义多个条形图显示，并可通过函数名称中的后缀 (n) 对这些显示进行寻址。只有在组件定制器中定义了条形图显示元素时，此函数才可用

**参数：** uint8 location：要显示的未分配整数位置。有效值范围为 0 到条形图中的段数。如是 0 值，则关闭所有条形图元素。如果值大于条形图中的段数，将打开所有元素。

uint8 mode：设置条形图显示模式。

| 值 | 说明 |
|---|---|
| 0 | 打开指定“位置”段。 |
| 1 | 打开“位置”段及其左侧的所有段。 |
| –1 | 打开“位置”段及其右侧的所有段。 |
| 2 到 10 | 显示“位置”段及其右侧的 2 到 10 个段。此模式可用于创建各种指示符。 |

**Return Value（返回值）：** None（无）

**Side Effects（副作用）：** None（无）

## void LCD_Seg_PutChar14Seg_n(uint8 character, uint8 position)

**说明：** 此函数显示 14 段字母数字字符显示元素阵列上的 8 位字符。必须使用定制器显示助手工具定义与 14 段显示元素相关联的像素集。可在帧缓冲区中定义多个 14 段字母数字显示元素组，并可通过函数名称中的后缀 (n) 对这些元素组进行寻址。只有在组件定制器中定义了 14 段元素时，此函数才可用。

**参数：** uint8 character：要显示的字符的 ASCII 值（ASCII 值为 0 到 127 的可打印字符）

uint8 position：从左侧的 0 开始从左到右计算的字符位置。如果此位置不在定义的显示区域内，将不显示此字符。

**Return Value（返回值）：** None（无）

**Side Effects（副作用）：** None（无）

## void LCD_Seg_WriteString14Seg_n(*uint8 character, uint8 position)

**说明：** 此函数显示 14 段字母数字字符显示元素阵列上以空字符作为结尾的字符串。必须使用定制器显示助手工具定义与 14 段显示元素相关联的像素集。可在帧缓冲区中定义多个 14 段字母数字显示元素组，并可通过函数名称中的后缀 (n) 对这些元素组进行寻址。只有在组件定制器中定义了 14 段显示元素时，此函数才可用。

**参数：** *uint8 character：指向以空字符作为结尾的字符串。

uint8 position：从左侧的 0 开始从左到右计算的第一个字符的位置。如果字符串的长度超出已定义显示区域的大小，将不显示额外的字符。

**Return Value（返回值）：** None（无）

**Side Effects（副作用）：** None（无）

## void LCD_Seg_PutChar16Seg_n(uint8 character, uint8 position)

**说明：** 此函数显示 16 段字母数字字符显示元素阵列上的 8 位字符。必须使用定制器显示助手工具定义与 16 段显示元素相关联的像素集。可在帧缓冲区中定义多个 16 段字母数字显示元素组，并可通过函数名称中的后缀 (n) 对这些元素组进行寻址。只有在组件定制器中定义了 16 段显示元素时，此函数才可用

**参数：** uint8 character：要显示的字符的 ASCII 值（值为 0 到 255 的可打印 ASCII 和表扩展字符）。

uint8 position：从左侧的 0 开始从左到右计算的字符位置。如果此位置不在定义的显示区域内，将不显示此字符。

**Return Value（返回值）：** None（无）

**Side Effects（副作用）：** None（无）

## (void) LCD_Seg_WriteString16Seg_n(*uint8 character, uint8 position)

**说明：** 此函数显示 16 段字母数字字符显示元素阵列上以空字符作为结尾的字符串。必须使用定制器显示助手工具定义与 16 段显示元素相关联的像素集。可在帧缓冲区中定义多个 16 段字母数字显示元素组，并可通过函数名称中的后缀 (n) 对这些元素组进行寻址。只有在组件定制器中定义了 16 段显示元素时，此函数才可用。

**参数：** *uint8 character：指向以空字符作为结尾的字符串。

uint8 position：从左侧的 0 开始从左到右计算的第一个字符的位置。如果字符串的长度超出已定义显示区域的大小，将不显示额外的字符。

**Return Value（返回值）：** None（无）

**Side Effects（副作用）：** None（无）

## void LCD_Seg_PutCharDotMatrix_n(uint8 character, uint8 position)

**说明：** 此函数显示点阵字母数字字符显示元素阵列上的 8 位字符。必须使用定制器显示助手工具定义与点阵显示元素相关联的像素集。可在帧缓冲区中定义多个点阵字母数字显示元素组，并可通过函数名称中的后缀 (n) 对这些元素组进行寻址。只有在组件定制器中定义了点阵显示元素时，此函数才可用

**参数：** uint8 character：要显示的字符的 ASCII 值。

uint8 position：从左侧的 0 开始从左到右计算的字符位置。如果此位置不在定义的显示区域内，将不显示此字符。

**Return Value（返回值）：** None（无）

**Side Effects（副作用）：** None（无）

### void LCD_Seg_WriteStringDotMatrix_n(*uint8 character, uint8 position)

**说明：** 此函数显示点阵字母数字字符显示元素阵列上以空字符作为结尾的字符串。必须使用定制器显示助手工具定义与点阵显示元素相关联的像素集。可在帧缓冲区中定义多个点阵字母数字显示元素组，并可通过函数名称中的后缀 (n) 对这些元素组进行寻址。只有在组件定制器中定义了点阵显示元素时，此函数才可用

**参数：** *uint8 character：指向以空字符作为结尾的字符串。

uint8 position：从左侧的 0 开始从左到右计算的第一个字符的位置。如果字符串的长度超出已定义显示区域的大小，将不显示额外的字符。

**Return Value（返回值）：** None（无）

**Side Effects（副作用）：** None（无）

## 引脚 API

这些 API 函数用于更改段 LCD 组件所用引脚的驱动模式。

| 函数 | 说明 |
|---|---|
| LCD_Seg_ComPort_SetDriveMode | 设置段 LCD 组件的 common 线路所用的所有引脚的驱动模式 |
| LCD_Seg_SegPort_SetDriveMode | 设置段 LCD 组件的 segment 线路所用的所有引脚的驱动模式。 |

### void LCD_Seg_ComPort_SetDriveMode(uint8 mode)

**说明：** 设置段 LCD 组件的 common 线路所用的所有引脚的驱动模式。

**参数：** uint8 mode：所需的驱动模式。有关驱动模式的信息，请参见引脚组件数据表。

**Return Value（返回值）：** None（无）

**Side Effects（副作用）：** None（无）

### LCD_Seg_SegPort_SetDriveMode(uint8 mode)

**说明：** 设置段 LCD 组件的 segment 线路所用的所有引脚的驱动模式。

**参数：** uint8 mode：所需的驱动模式。有关驱动模式的信息，请参见引脚组件数据表。

**Return Value（返回值）：** None（无）

**Side Effects（副作用）：** None（无）

### 宏

- LCD_Seg_COMM_NUM — 针对组件目前的配置定义用户定义显示屏的 common 线路数。
- LCD_Seg_SEG_NUM — 针对组件目前的配置定义用户定义显示屏的 segment 线路数。
- LCD_Seg_BIAS_TYPE — 针对组件目前的配置定义用户定义显示屏的偏置类型。
- LCD_Seg_BIAS_VOLTAGE — 为用户定义的显示屏定义默认偏置电压。将在初始化过程中在 LCDDAC 控制寄存器中设置此值。
- LCD_Seg_FRAME_RATE — 针对组件目前的配置定义用户定义显示屏的刷新率。
- LCD_Seg_EXTRACT_ROW — 计算帧缓冲区中特定像素的行。
- LCD_Seg_EXTRACT_PORT — 计算帧缓冲区中特定像素的字节偏移。
- LCD_Seg_EXTRACT_PIN — 计算帧缓冲区中特定像素的比特位置。
- LCD_Seg_WRITE_PIXEL — 这是 viod 类型的 LCD_Seg_WritePixel() 函数的宏定义。
- LCD_Seg_READ_PIXEL — 这是 LCD_Seg_ReadPixel() 函数的宏定义。
- LCD_Seg_FIND_PIXEL — 此宏计算帧缓冲区中的像素位置。它使用定制器像素表中的信息和专用于 LCD 的物理引脚的相关信息。此宏是像素映射机制的基础。像素表中的每个像素名称都是用帧缓冲区中计算出的像素位置定义的。API 使用像素名称以访问各自的像素。

## 固件源代码示例

PSoC Creator 在“查找示例项目”对话框中提供了大量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。

**注意**由于特定的组件实现，当组件处于低功耗模式下时，您不可使用带 CyPmSaveClocks() 和 CyPmRestoreClocks() API 的组件。在其他情况下，可使用 CyPmSaveClocks() 和 CyPmRestoreClocks() API。

# 功能描述

段 LCD 组件提供强大而灵活的机制，以驱动不同类型的 LCD 显示屏。使用配置对话框，可访问可用于定制组件功能的参数。可使用一组标准的 API 子程序控制显示屏和特定像素。其他显示屏 API 是基于定义的显示助手的类型和数量而生成的。

## 默认配置

LCD_Seg 组件的默认配置提供普通的 LCD 直接段驱动控制器。默认 LCD_Seg 配置为：

- 四条 common 线路
- 八条 segment 线路
- 1/3 偏置类型
- 60-Hz 刷新率
- “无睡眠”功耗模式
- 3-V 偏置电压
- 显示屏大小：10 平方厘米
- 968.99-µs 高驱动时间
- seg - 1x、com - 2x 高驱动强度
- 7.57-µs 低驱动时间
- seg - 0.06x、com 0.12x 低驱动强度
- 未定义显示助手。默认 API 生成不包括任何支持的显示元素的函数。

## 自定义配置

段 LCD 组件的一个关键功能是灵活的支持不同特定和布局的 LCD。

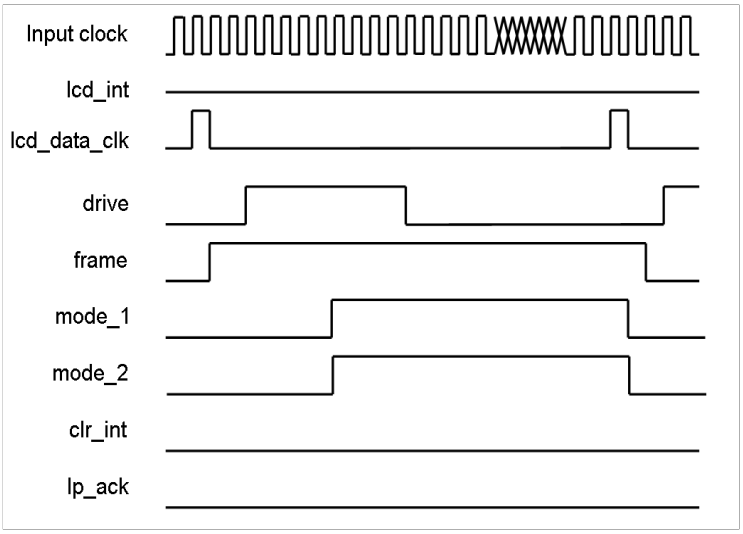
## Driver Power Mode（驱动器功耗模式）

段 LCD 可在两种功耗模式下操作：无睡眠模式和低功耗模式。无睡眠是默认操作模式。

### No Sleep Mode（无睡眠模式）

在此模式下，段 LCD 绝不会进入睡眠模式，在整个帧流程中都将驱动 LCD。图 1 显示无睡眠模式下 LCD_Seg 组件的由 UDB 生成（内部）的信号的波形。

**图 1. 段 LCD 控制信号（无睡眠模式）**




### Low Power Mode（低功耗模式）

在此模式下，仅在电压转换时才主动驱动 LCD，且 LCD 系统模拟组件在电压转换之间进入睡眠模式。内部 LCD 定时器将把器件从睡眠模式中唤醒一小段时间，以更新 LCD 屏幕。之后，内部逻辑使整个器件返回到睡眠模式，直至下一次刷新序列。

在低功耗模式下，“Frame Rate”（帧率）参数受到限制，并取决于其他组件参数（例如 LCD 定时器（1 kHz [ILO] 或 8 kHz [32kHz 晶振]）的时钟源、common 线路数和波形类型）。可用 LCD 定时器的有限输入频率说明帧率限制。以下是针对八条 common 线路、低功耗 ILO 模式和 A 型波形配置的最大帧率计算示例：

最大帧率 = 1000/(2 × common 线路数) = 62.5 Hz

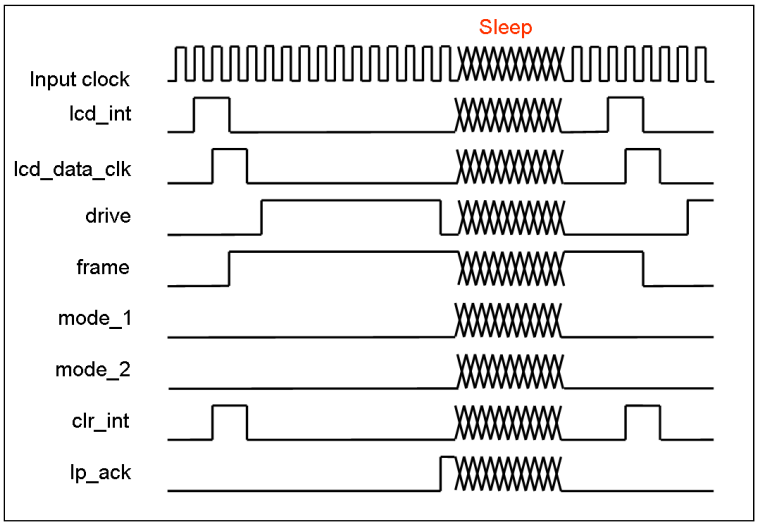
62.5 Hz 是 LCD 定时器可提供的最大帧率，在此帧率下，定时器在输入频率的每个时钟周期中生成一个唤醒信号。

在现实中，LCD 定时器需要一个周期以设置唤醒事件，还需一个周期清除它。这意味着最大帧率应被一分为二，即 31.25 Hz。根据此值，只取整数得到 31 Hz，作为实际的最大帧率。

对于 B 型波形，实际帧率比 A 型波形大两倍，因为 B 型波形需要的唤醒事件少两倍。

图 2 显示低功耗模式下 LCD_Seg 组件的由 UDB 生成的信号的波形。

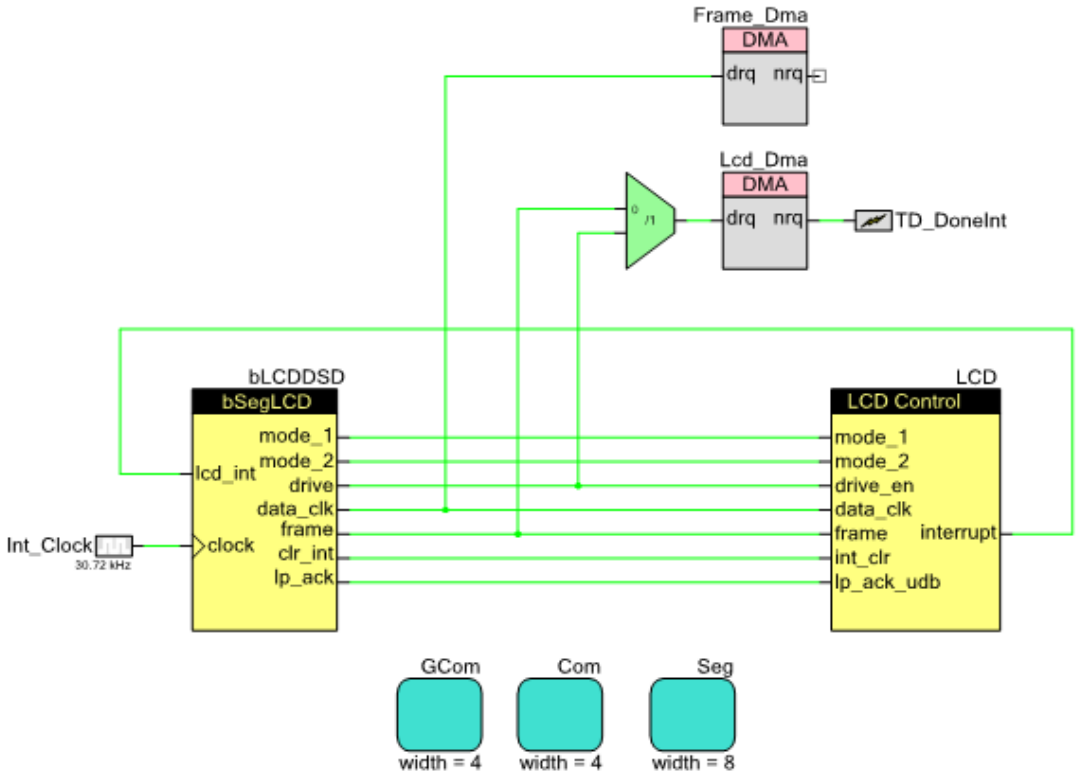
**图 2. 段 LCD 控制信号（低功耗模式）**



# 框图和配置

图 3 显示段 LCD 组件的内部原理图。它包含基本段 LCD 组件、LCD 控制模块 (LCD) 组件、DMA 组件、三个 LCD 端口、一个数字端口、一个 ISR 组件和一个时钟组件。

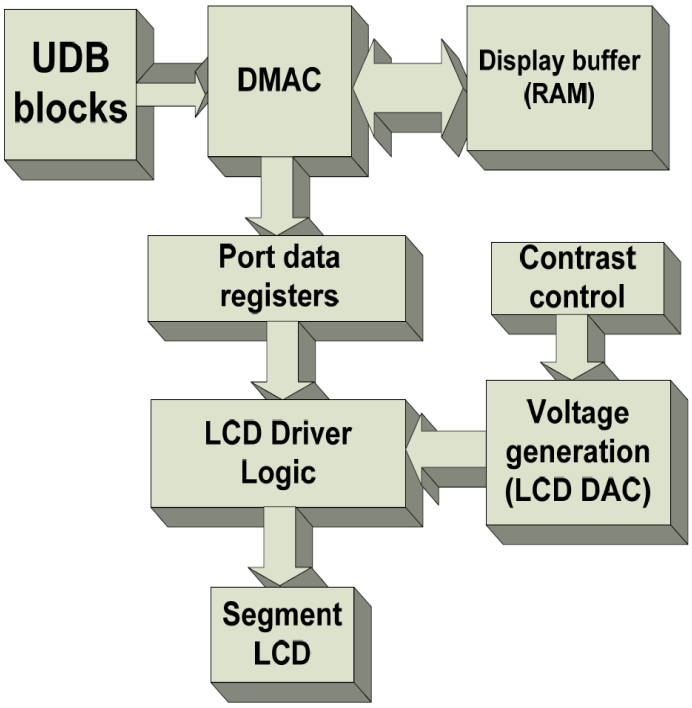
**图 3. 段 LCD 组件原理图**




- 基本段 LCD 组件负责为 LCD 端口和 DMA 组件生成正确的时序信号。
- DMA 组件用于将数据通过伪信号存储器区域从帧缓冲区传输到 LCD 数据寄存器。
- LCD 组件处理必要的 DSI 路由。此模块还提供如 *cyfitter.h* 中所定义必要的寄存器名称。
- LCD 端口（GCom、Com 和 Seg）用于将逻辑信号映射到物理引脚。有两个 LCD 端口实例：一个是 common 线路实例，一个是 segment 线路实例。common 信号的 LCD 端口限制在 16 引脚宽，而 segment 信号的 LCD 端口限制在 48 引脚宽。还有一个额外的 Common 端口 GCom，仅在启用了“Ganging”（组连接）选项后才包括此端口。

## 顶层架构

**图 4. 段 LCD 顶层**




## 寄存器

### LCD_Seg_CONTROL_REG

| 位 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 值 | Frame（帧） | 保留 | | | frame done（帧完成） | mode 2（模式 2） | mode 1（模式 1） | clock enable（时钟使能） |

- clock enable（时钟使能）：启用之前章节中所述的所有内部信号的生成。
- mode 1（模式 1）：mode[2:0] 位域的中间位，用于定义高驱动强度和低驱动强度。
- mode 2（模式 2）：mode[2:0] 位域的高位，用于定义高驱动强度和低驱动强度。
- frame done（帧完成）：完成 DMA 帧数据操作之后生成同步脉冲。
- Frame（帧）：生成 LCD 驱动器的帧信号。

### LCD_Seg_CONTRAST_CONTROL_REG

保留偏置电压，由 LCD DAC 用于生成正确偏置电压。提供 API 以更改偏置电压。

| 位 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 值 | 保留 | 保留 | contrast level（对比度） | | | | | |

- contrast level（对比度）：偏置电压如上所述。

### LCD_Seg_LCDDAC_CONTROL_REG

| 位 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 值 | lp enable（lp 使能） | 保留 | | | continuous drive（持续驱动） | 保留 | bias select（偏置选择） | |

- bias select（偏置选择）：选择偏置。
- continuous drive（持续驱动）：当芯片进入睡眠状态时，允许 LCDDAC 保持活动状态。
- lp enable（lp 使能）：允许 UDB 输入 LCD 子系统的低功耗 Ack。

### LCD_Seg_TIMER_CONTROL_REG

| 位 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 值 | period（周期） | | | | | | clk select（clk 选择） | enable timer（启用定时器） |

- enable timer（启用定时器）：启用 LCD 定时器。
- clk select（clk 选择）：LCD 定时器源选择时钟。
- period（周期）：LCD 定时器周期。

### LCD_Seg_DRIVER_CONTROL_REG

| 位 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 值 | 保留 | | | bypass enable（旁路使能） | pts | Invert（反相） | mode 0（模式 0） | Sleep mode（睡眠模式） |

- Sleep mode（睡眠模式）：在低功耗模式下，如果此参数设为“1”，则将 LCD 驱动器中的输出缓冲区设为接地，否则将 LCD 驱动器设为 HI-Z。

- mode 0（模式 0）： mode[2:0] 位域的低位，用于定义高驱动强度和低驱动强度。
- invert（反相）：如果设置了此参数，将反转段引脚上的数据。
- pts：如果将此参数清除为“0”— 正常操作，VOUT = VIO-0.5V。如果设为“1”，VOUT = VIO。
- bypass_enable（旁路使能）：如果将此参数设为“1”，绕过 LCD 驱动器中的驱动缓冲区，并直接连接到选定的电压输入。如果将此参数清除为“0”，即正常操作，电压经过驱动缓冲区。

**LCD_Seg_LCDDAC_SWITCH_REG[0..4]**

| 位 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 值 | 保留 | | | | | | switch control[0..4]（开关控制[0..4]） | |

- switch control[0..4]（开关控制[0..4]）：此位域集为 LCD 驱动器选择电压源。

# 直流和交流电气特性

## LCD 直接驱动直流规范

| 参数 | 说明 | 条件 | 最小值 | 典型值 | 最大值 | 单位 |
|---|---|---|---|---|---|---|
| $I_{CC}$ | LCD 系统工作电流 | 器件睡眠模式在 400 Hz 速率下唤醒以刷新 LCD，总线时钟 = 3 MHz，$V_{DDIO}$ = $V_{DDA}$ = 3 V，4 个common，16 个segment，1/4 占空比，50 Hz 帧率，未连接显示屏 | – | 38 | – | µA |
| $I_{CC_SEG}$ | 每个segment驱动器的电流 | 强驱动模式 | – | 260 | – | µA |
| $V_{BIAS}$ | LCD 偏压范围（$V_{BIAS}$ 指 LCD DAC 的主要输出电压 (V0)） | $V_{DDA}$ ≥ 3 V 且 $V_{DDA}$ ≥ $V_{BIAS}$ | 2 | – | 5 | V |
| | LCD 偏压步长大小 | $V_{DDA}$ ≥ 3 V 且 $V_{DDA}$ ≥ $V_{BIAS}$ | – | 9.1 × $V_{DDA}$ | – | mV |
| | 每个 segment/common 驱动器的 LCD 电容 | 驱动器可以组合使用 | – | 500 | 5000 | pF |
| | 长期段偏移 | | – | – | 20 | mV |
| $I_{OUT}$ | 每个 segment 驱动器的输出驱动电流 | $V_{DDIO}$ = 5.5 V，强驱动模式 | 355 | – | 710 | µA |

### LCD 直接驱动交流规范

| 参数 | 说明 | 条件 | 最小值 | 典型值 | 最大值 | 单位 |
|---|---|---|---|---|---|---|
| $f_{LCD}$ | LCD 帧率 | | 10 | 50 | 150 | Hz |

## 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 |
|---|---|---|
| 3.0 | 已将 WRITE_PIXEL() 宏定义从<br>#define LCD_Seg_WRITE_PIXEL(pixelNumber, pixelState) LCD_Seg_WritePixel(pixelNumber, pixelState)<br>更改为<br>#define LCD_Seg_WRITE_PIXEL(pixelNumber, pixelState) (void) LCD_Seg_WritePixel(pixelNumber, pixelState) | 无需在组件实现过程中分析 LCD_Seg_WritePixel() API 的返回值，因此更改宏以处理此情况。 |
| | 已更改组件的 verilog 实现，以避免在使用低功耗模式下的组件时出现 STA 组合周期警告。 | 异步 SR 锁存器替换为同步逻辑。 |
| | 已在 LCD_Seg_ReadPixel() API 中添加了针对未分配助手像素的验证。 | 已将 0xFF (LCD_Seg_Stat_PIXEL_UNKNOWN_STATE) 的新返回状态添加到 LCD_Seg_ReadPixel()。当像素存在于助手中但未分配时，返回此状态。 |
| | 已优化函数 LCD_Seg_WriteBargraph()。 | 代码重构节省了一些闪存空间。 |
| | 优化了函数 LCD_Seg_WriteString14Seg() 和 LCD_Seg_WriteString16Seg()。 | 已将数组索引更改为指针递增，从而节省了 RAM 空间，并有更多时间执行这些函数。 |
| | 已优化 LCD_Seg_Write7SegNumber() 的 API。 | LCD_Seg_Write7SegNumber() API 中有一些在此 API 的较长执行时间内出现的无效代码。 |
| | LCD_Seg_Wakeup() API 的补丁。 | LCD_Seg_Wakeup() API 应返回但未返回任何状态，因此修复了 API 以返回正确的状态。 |
| 2.10 | 已修复 LCD_Seg_Start() 子程序 | 如果之前启用了全局中断，由于全局中断实施过程中出错，所以禁用全局中断而无任何通知。 |
| | 已更改定制器的“Driver Power Settings”（驱动器功耗设置）选项卡。已添加两个新字段“Custom Step”（自定义步骤）和“Default Step”（默认步骤）。 | 在低功耗模式中，“High Drive Time”（高驱动时间）参数的新选项可用。除了递增的默认步长，还可选择更精确的自定义步长。它是以主时钟的频率为基础的。 |

| 版本 | 更改说明 | 更改/影响原因 |
|---|---|---|
| | 解决了在完成 LCD 刷新序列之前使 SegLCD 进入睡眠状态的问题。 | 内部组件信号 lp_ack 在刷新序列开始时转变为高电平状态，但它本应该在刷新序列结束时才转变为高电平状态。 |
| | 解决了在“High Drive Strength”（高驱动强度）设为“seg=2x com=2x”或“seg=1x com=4x” 时反转 LCD 显示屏图像时出现的问题。 | 问题是将错误的值写入到 LCD 驱动器控制寄存器。此值通过 AND 运算和掩码生成，掩码不正确。改成正确的掩码以解决问题。 |
| | LCD_Seg_WriteBargraph() API 补丁。 | 解决了当模式参数设为 10 时，LCD 的条形图像素的错误输出导致的 LCD_Seg_WriteBargraph() API 问题。在“Position”（位置）参数满足了条件“位置 + 10 > 条形图像素”时，出现此情况。在此情况下，条形图的最后一个像素未设置，但实际应该设置。 |
| | 向数据手册中添加了特性数据 | |
| | 向组件中添加了信息，以说明它与芯片修订版的兼容性。 | 如果组件在不兼容的芯片上使用，该工具将返回错误。如果发生此情况，请更新到支持您的目标器件的版本。 |
| | 对数据手册进行了少量编辑和更新 | |
| 2.0.a | 已向数据表中的 LCD_Seg_Init()、LCD_Seg_Start() 和 LCD_Set_Stop() API 添加注解 | |
| | 向组件中添加了信息，以说明它与芯片修订版的兼容性。 | 如果组件在不兼容的芯片上使用，该工具将返回错误。如果发生此情况，请更新到支持您的目标器件的修订版。 |
| 2.0 | 添加了睡眠/唤醒和初始化/启用 API。 | 为支持低功耗模式并提供常用接口，以单独控制大多数组件的初始化和启用。 |
| | 添加了新 API 文件 — *SegLCD_PM.c*，该文件包含睡眠模式 API 的声明。 | 新增了支持低功耗模式的要求。 |

| 版本 | 更改说明 | 更改/影响原因 |
|---|---|---|
| | 更新了组件以支持 PSoC 3 ES3 及更高版本。更新了“Configure”（配置）对话框：<br>■ 已移除废弃控制：启用调试模式、低驱动模式<br>■ 已添加新控制：显示屏大小、高驱动强度、低驱动强度<br>■ 已通过将旧模式“始终活动”和“低功耗”更改为新模式“无睡眠”、“低功耗 ILO”和“低功耗 32XTAL”更改了“驱动器功耗模式”的选项<br>■ 已添加了自定义字符“Tab”，以允许用户为点阵助手创建自定义字符集<br>■ 使偏置类型控制可编辑，并更改了偏置电压的选择值 | 新增了支持 PSoC 3 ES3 器件和 LCD HW 架构的要求，因此创建了新的 2.0 版。<br>版本 1.xx 支持 PSoC 3 ES2 和 PSoC 5 芯片版本 |
| | 已将 `=ReentrantKeil(LCD_Seg_ . "_...")` 添加到下列函数中：<br>LCD_Seg_Stop()<br>LCD_Seg_EnableInt()<br>LCD_Seg_DisableInt()<br>LCD_Seg_SetBias()<br>LCD_Seg_WriteInvertState()<br>LCD_Seg_ReadInvertState()<br>LCD_Seg_RedPixel()<br>LCD_Seg_SaveConfig()<br>LCD_Seg_RestoreConfig() | 如需重入，请允许用户将这些 API 设置为重新进入。 |

© 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。