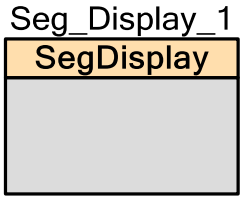
# 段显示器 (Seg_Display)

1.0




## 特点

- 2-768 像素或符号
- 支持 1/3、1/4 和 1/5 偏压
- 10-150-Hz 刷新率
- 集成的偏压生成介于 2.0 V-5.2 V 之间，具有高达 128 个数控偏压电平，用于动态对比度控制
- 支持类型 A（标准）和类型 B（低功耗）波形
- 显示像素状态可能被反转，从而显示负像
- 256 字节的显示存储器（帧缓冲区）
- 用户定义的像素或符号映射，可选 7、14 或 16 段字符；5x7 或 5x8 点阵；条形图计算子程序
- 支持 PSoC 5 芯片版本。
- 不支持 PSoC 3。段式 LCD 版本 3.0 用于 PSoC 3 Production 芯片版本或更高版本。

## 概述

段显示器 (Seg_Display) 组件可以在复用率达到 16x 时直接驱动 3.3-V 和 5.0-V LCD 显示屏。此组件提供用来配置 PSoC 器件的简易方法，以便驱动自定义或标准显示屏。生成内部偏压，这去除了任何外部硬件需求，从而支持基于软件的对比度调整。使用升压转换器，显示屏偏置电压可能高于 PSoC 供电电压。这样，便携式应用的显示灵活性大大升高。每个 LCD 像素/符号既可以打开，也可以关闭。此外，段显示器组件还提供高级支持，从而简化下列显示屏显示结构的类型：

- 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（对 5x7 和 5x8 使用相同的查找表。查找表中的所有符号大小均为 5x7 像素。）

- 1-255 个元件条形图

有关使用段显示器组件的详细信息，请参见应用笔记 *AN52927: PSoC® 3: Segment LCD Direct Drive Basics*。

### 何时使用段显示器

当需要在复用率为 2x-16x 下直接驱动 3.3-V 或 5.0- V LCD 显示屏时，使用直接段式驱动 LCD 组件。直接段式驱动 LCD 组件要求目标 PSoC 器件支持 LCD 直接驱动。

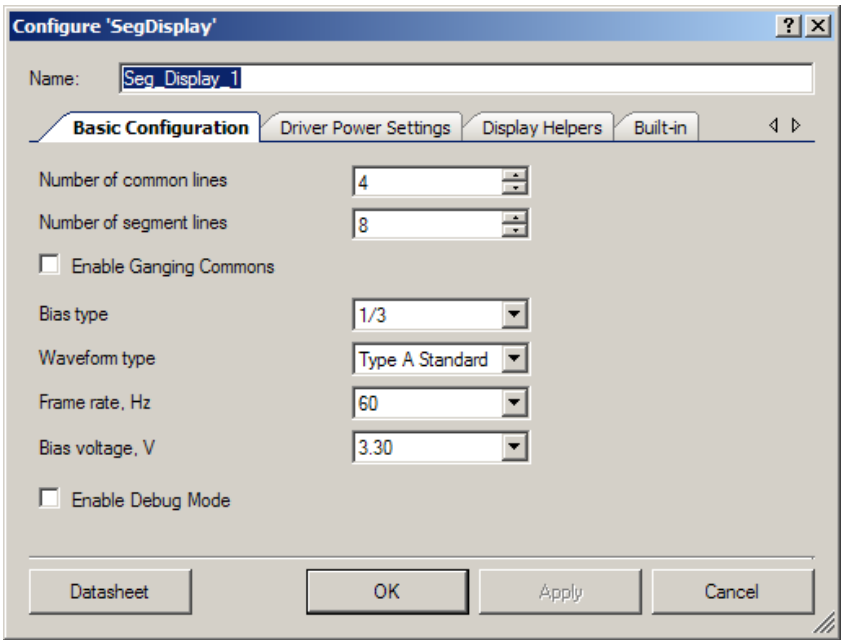
## 输入/输出连接

在原理图画布上没有可见的组件连接；然而，使用设计范围资源引脚编辑器，可以将各种信号连接至引脚。

## 元件参数

将段显示器组件拖入设计中，双击该组件，打开 **Configure**（配置）对话框。**Configure**（配置）对话框包含多个不同参数类型的选项卡，用于设置段显示器组件。

### Basic Configuration（基本配置）选项卡



### 常用行数

定义显示屏所需的通用信号数（默认值为 **4**）。

### 段行数

定义显示屏所需的段信号数。可能值的范围为 (2-62) – **通用行数**. 默认值为 **8**。

### 启用通用机械连接

选中此复选框时，可以将 PSoC 引脚组合在一起以驱动通用信号。为每个通用信号分配两个 PSoC 引脚。这用来驱动较大显示屏。

### 偏压类型

此值用来确定一组通用及段线路的正确偏压模式。

### 波形类型

这用来确定波形类型：类型 A - 0 VDC 平均数超过单个帧（默认值）或类型 B - 0 VDC 平均数超过两个帧。

### 帧率

这用来确定显示屏的刷新率。可能值的范围是 10 Hz - 150 Hz。默认值为 **60 Hz**。

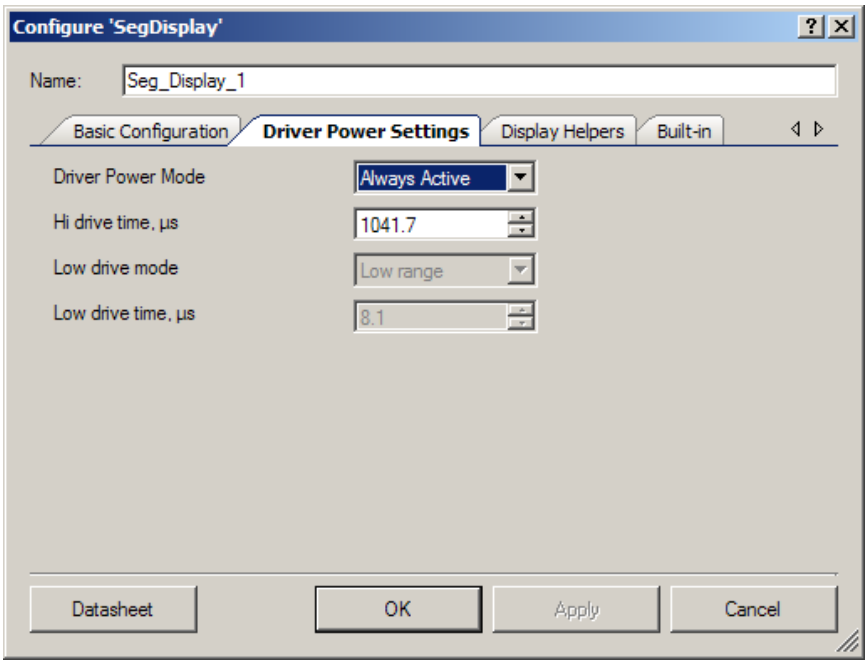
### 偏压

这用来确定 LCD DAC 的偏置电压电平。可能值的范围是 2 V - 5.2 V。默认值为 **3.3 V**。

### 启用调试模式

如果启用，则将调试 端口添加到组件，由此查看驱动 LCD 驱动程序的信号。例如，可以使用它来查看高电平和低电平驱动时间。

## Driver Power Settings（驱动程序功耗设置）选项卡




### 驱动程序功耗模式

该**驱动程序功耗模式**参数用来定义组件的功耗模式。有两种功耗模式可用：

- **始终有效**：LCD DAC 始终为打开状态
- **低功耗**：LCD DAC 在两次电压转变之间为关闭状态

请参见此数据手册后面章节功能描述中的驱动程序功耗模式。

### 高电平驱动时间

此参数用来定义高电平驱动模式在一次电压转变范围内有效的高电平驱动模式的时间期间。

**注意：** 如果更改**帧率**、**通用线路数**或**波形类型**参数，**高电平驱动时间**参数将设置为最小值。在此数据手册后面定义**高电平驱动时间**最小值的计算。当前配置中**高电平驱动时间**的最大值为：

HiDriveTimemax= 1/(FrameRate × (2 × NumCommonLines) × 256 ) × 253

### 低电平驱动模式

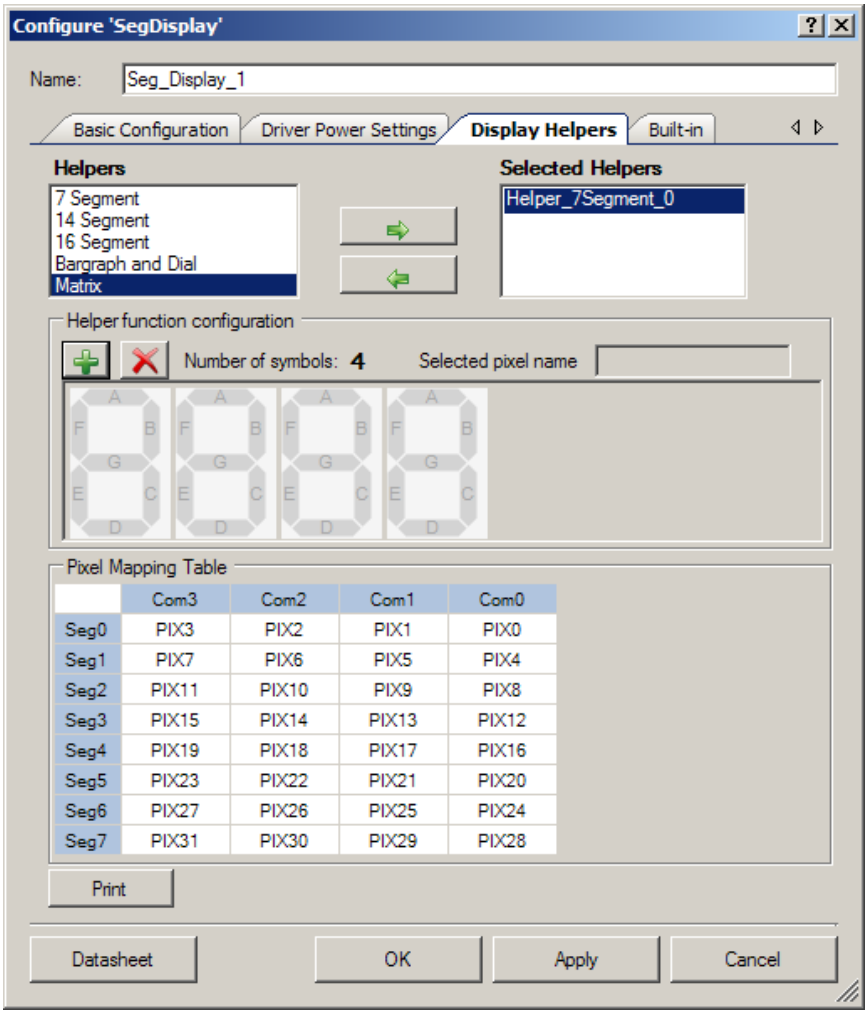
将**驱动程序功耗模式**设置为**低功耗**时，此参数可用。**低电平驱动模式**有两个可用值：

- **低电平范围** – 激活低电平驱动模式
- **高电平范围** – 激活第二个低电平驱动高电流模式 (Lo2)。

### 低电平驱动时间

该**低电平驱动时间**参数用于定义低电平驱动模式在电压转变过程中有效的时间期间。

## Display Helpers（显示助手）选项卡




通过显示助手，您可以配置一组显示段，将其同时用作其中一种预定义的显示元件类型：

- 7、14 或 16 段显示
- 点阵显示（5x7 或 5x8）
- 线性或循环条形图显示

基于字符的显示助手可用于组合多个显示符号，从而创建多字符显示元件。

### 助手(Helper)/选定助手(Selected helper)

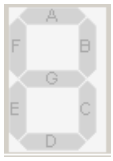
可以将一个或多个助手添加到**选定助手**列表中，方法是在**助手**列表中选择所需助手类型，然后单击右箭头按钮。如果没有足够的引脚来支持新助手，则无需添加该助手。要删除某个助手，在**选定助手**列表中选中该助手，然后单击左箭头按钮。

**注意：** 将显示助手添加到组件后，即可以更改通用（common）或段（segment）线路数。重要的一点是，定义任何显示助手前，一定要为组件设置通用（common）和段（segment）线路数。更改通用或段线路之前，必须删除已定义的所有显示助手。

在列表中，**选定助手**显示的顺序是有效的。默认情况下，在**选定助手**列表中添加的第一个特定类型的助手命名时使用 0 作为后缀，下一个相同类型的助手将使用 1 作为后缀等。如果从列表中移除**选定助手**，则重新命名剩余助手。添加助手时，名称将使用最小可用后缀。

为每个助手提供 API。有关详细信息，请参见应用程序编程接口。

- **7 段助手–** 此助手可以是 1-5 个数字长度，显示格式即可以是十六进制数字 0-F，也可以是十进制 16 位无符号整型 (uint16) 值。助手函数不支持小数点。




- **14 段助手 –** 此助手长度最多为 20 个字符。它可以显示为单个 ASCII 字符或空结尾的字符串。可能的值是标准的 ASCII 可打印字符（编码范围为 0-127）


- **16 段助手 –** 此助手长度最多为 20 个字符。它可以显示为单一的 ASCII 字符或空结尾的全字符串。可能值是标准的 ASCII 字符和扩展编码表（编码范围为 0-255）。不提供扩展编码表。


- **条形图和拨号助手–** 这些助手用于 1-255 段条形图和拨号指示符。条形图是单个选择的像素或选定像素及指定像素左边或右边的所有像素

- **阵列助手–** 此助手最多支持 8 个字符元素。该组件支持 5x7 或 5x8 行/列字符。通过配置 2 个或多个点阵助手，可以创建较长字符串，用来定义显示屏相邻点阵部分。该助手显示为单一 ASCII 字符或空结尾字符串。

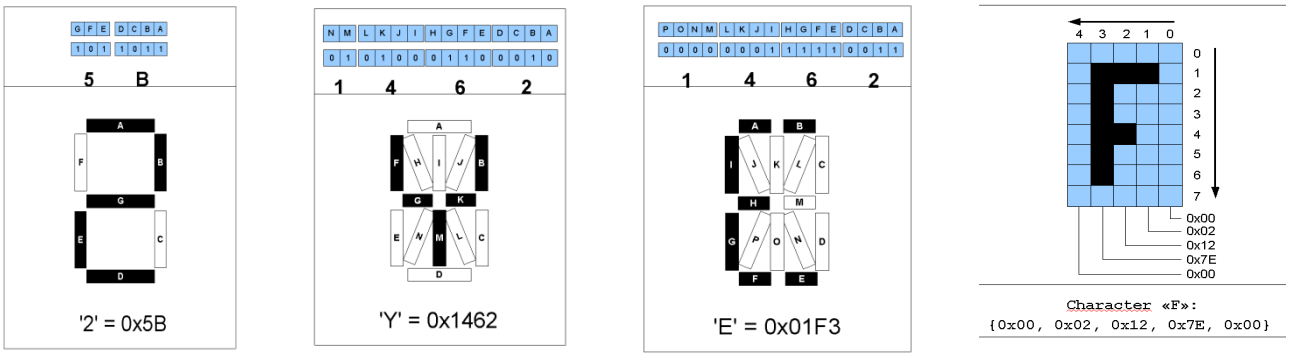


点阵助手具有打印输出限制。它必须对矩阵行使用 7 或 8 个连续通用驱动程序，对矩阵列使用 5-40 个连续段驱动程序。该组件支持标准的 Hitachi HD44780 字符集。

### 字符编码

每个高级助手 API 有其自己的查找表。该表包含一组编码像素状态，这构成特定的字符反射。下列示例说明如何对特定字符进行编码（段名称与自定义程序中显示的名称不同）。

**7 段编码** **14 段编码** **16 段编码** **点阵编码**




### 助手功能配置

通过本节对话框，可以配置助手；这包括为助手添加符号或从助手中移除符号及命名像素。

1. 从**选定助手**列表中选择助手。
2. 单击 **[+]** 或 **[x]** 按钮，为选定助手添加或移除符号。可以添加符号的最大数取决于助手类型，像素总数由组件支持。如果没有足够的可用引脚数支持新符号，则无需再添加符号。
3. 要重新命名作为助手功能组成部分的像素，在**助手功能配置**显示屏中，选择符号图像上的像素。当前名称显示在选定像素名称字段中，可以根据需要进行修改。

### 像素命名

默认像素名称有 “PIX#”，其中“#” 表示像素从**像素映射表**右上角开始的增量顺序号。

像素（与助手符号相关）的默认命名方式具有不同格式。默认名称由前缀部分、所有像素的通用的符号部分和唯一段标识符组成。默认像素表示助手类型和符号实例。例如，在 7 段助手中，其中一个符号的默认像素名称可能是“H7SEG4_A”，其中：

| | |
|---|---|
| H7 | 表示 7 段助手的像素部分 |
| SEG4 | 表示指定为项目 7 段符号第 4 个符号的部分像素 |
| A | 标识 7 段符号中的唯一段 |

仅有默认像素名称的唯一部分显示在符号图像中。如果修改像素名称，则整个名称显示在符号图像中，即使与其他像素共同占用通用前缀也如此。

**注意**：所有像素名称必须是唯一的。

当助手功能符号元素分配到**像素映射表**（如下所述）中的某个像素时，该像素采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，但不替换该名称。无法重新使用与助手功能相关默认像素名称。

### 像素映射表

该**像素映射表**是帧缓冲区的表示法。为使 API 函数正确工作，**助手函数配置**的每一个像素必须分配到**像素映射表**中的像素位置。有关正确分配位置的信息，请参见 LCD 显示屏数据手册。

要分配像素，在**助手函数配置**面板中选择所需像素，然后将其拖入**像素映射表**中的正确位置。

通过在表显示屏上双击像素，然后输入所需名称，可以重新命名**像素映射表**中的像素。可以使用此方法命名与其中某个可用助手类型关联的像素。

**Print**（打印）按钮可以打印**像素映射表**。

## 时钟选择

Seg_Display 组件使用两个内部时钟，而无需使用外部时钟。放置组件之后，LCD 自动专门使用这两个时钟。第一个时钟生成刷新频率，第二个时钟为低驱动缓冲区生成 100-kHz 频率的时钟。

## 放置

Seg_Display 组件实现包含两个部件。LCDDAC 是在 PSoC 中由此组件使用的固定功能的硬件模块。驱动信号的其他时序逻辑是在 UDB 中实现的。项目生成过程中，UDB 资源自动放置在 UDB 阵列中。

**注意**：仅有一个组件实例可以用于某个项目。如果某个项目使用一个以上的实例，则在构建过程中产生位置错误。

构建过程中执行默认引脚分配，在 PSoC Creator 设计范围资源工具中，使用引脚编辑器，可以修改该默认引脚分配。

# 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） |
|---|---|---|---|---|---|---|---|---|---|
| | 数据路径单元 | PLD | LCD 固定模块 | 控制/ Count7 单元 | 同步单元 | 中断 | Flash（闪存） | RAM | |
| 基本 | 1-2 | 3-5 | 1 | 2 | 0 | 1 | 2068 | 77 | 3-62 |
| 基本 7 段助手 | 1-2 | 3-5 | 1 | 2 | 0 | 1 | 2441 | 95 | 3-62 |
| 基本的 14 段助手 | 1-2 | 3-5 | 1 | 2 | 0 | 1 | 3319 | 293 | 3-62 |
| 基本的 16 段助手 | 1-2 | 3-5 | 1 | 2 | 0 | 1 | 3399 | 293 | 3-62 |
| 基本的点阵助手 | 1-2 | 3-5 | 1 | 2 | 0 | 1 | 3182 | 293 | 3-62 |
| 基本的条形图助手 | 1-2 | 3-5 | 1 | 2 | 0 | 1 | 4462 | 293 | 3-62 |

# 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“Seg_Display_1” 分配给指定设计中组件的第一个实例。可以将实例重新命名为唯一值，遵循 PSoC Creator 标识符语法规则。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 “Seg_Display”。

| 函数 | 说明 |
|---|---|
| Seg_Display_Start() | 启动 LCD 组件，启用所需中断 |
| Seg_Display_Stop() | 禁用 LCD 组件和关联的中断和 DMA 通道。 |
| Seg_Display_EnableInt() | 启用 LCD 中断。如果调用 Seg_Display_Start()，则无需启用 |
| Seg_Display_DisableInt() | 禁用 LCD 中断。如果调用 Seg_Display_Stop()，则无需禁用 |
| Seg_Display_SetBias() | 将 LCD 显示屏的偏压电平设置为 128 个值的其中一个值 |

| 函数 | 说明 |
|---|---|
| Seg_Display_WriteInvertState() | 根据输入参数反转显示屏。 |
| Seg_Display_ReadInvertState() | 返回显示屏反转状态的当前值：正常或反转 |
| Seg_Display_ClearDisplay() | 清除显示屏和关联帧缓冲区 RAM。 |
| Seg_Display_WritePixel() | 根据 **PixelState** 设置或清除像素 |
| Seg_Display_ReadPixel() | 读取帧缓冲区中的像素状态。 |
| Seg_Display_SetAwakeMode() | 当处于低功耗模式时，将 LCD 驱动程序缓冲输出设置为高阻抗。 |
| Seg_Display_Sleep() | 停止 LCD，然后保存用户配置。 |
| Seg_Display_Wakeup() | 恢复并启用用户配置。 |
| Seg_Display_Init() | 根据 Configure（配置）对话框设置初始化或恢复 LCD。 |
| Seg_Display_Enable() | 。启用组件 |
| Seg_Display_SaveConfig() | 保存 LCD 配置。 |
| Seg_Display_RestoreConfig() | 恢复 LCD 配置。 |

## 全局变量

| 变量 | 说明 |
|---|---|
| Seg_Display_initVar | 表示 Seg_Display 是否完成初始化。变量将初始化为 0，并在第一次调用 Seg_Display_Start() 时设置为 1。这样，第一次调用 Seg_Display_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，则在调用 Seg_Display_Init() 函数后，可以调用 Seg_Display_Start() or Seg_Display_Enable() 函数。 |

## uint8 Seg_Display_Start(void)

**说明：** 启动 LCD 组件，然后启用所需中断、DMA 通道、帧缓冲区和硬件。请勿清除帧缓冲区 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 | 某些 DMA TD 或通道已经被占用。 |
| CYRET_SUCCESS | 成功完成的功能 |

**Side Effects（副作用）：** None（无）

## void Seg_Display_Stop(void)

**说明：** 禁用 LCD 组件和关联的中断及 DMA 通道。自动清空显示屏以避免直流偏移。请勿清除帧缓冲区。

**参数：** None（无）

**Return Value（返回值）：** None（无）

**Side Effects（副作用）：** None（无）

## void Seg_Display_EnableInt(void)

**说明：** 启用 LCD 中断。如果调用 Seg_Display_Start()，则无需启用。每次更新 LCD 后（TD 完成）发生中断。

**参数：** None（无）

**Return Value（返回值）：** None（无）

**Side Effects（副作用）：** None（无）

## void Seg_Display_DisableInt(void)

**说明：** 禁用 LCD 中断。如果调用 Seg_Display_Stop()，则无需禁用。

**参数：** None（无）

**Return Value（返回值）：** None（无）

**Side Effects（副作用）：** None（无）

## uint8 Seg_Display_SetBias(uint8 biasLevel)

**说明：** 此函数用于将 LCD 显示屏的偏压电平设置为 128 个值的其中一个值。实际值数由模拟供电电压 $V_{DDA}$ 来限制。偏置电压不能超过 $V_{DDA}$。更改偏置电压会影响 LCD 对比度。

**参数：** uint8 biasLevel: 显示屏的偏压电平

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

| 返回值 | 说明 |
|---|---|
| CYRET_BAD_PARAM | **biasLevel**参数赋值失败 |
| CYRET_SUCCESS | 成功完成的功能 |

**Side Effects（副作用）：** None（无）

## uint8 Seg_Display_WriteInvertState(uint8 invertState)

**说明：** 此函数用来根据输入参数反转显示屏。在硬件中反转显示屏，无需在帧缓冲区中对显示 RAM 进行更改

**参数：** uint8 invertState: 设置显示屏的反转状态

| 值 | 说明 |
|---|---|
| Seg_Display_NORMAL_STATE | 设置显示屏的正常状态 |
| Seg_Display_INVERTED_STATE | 设置显示屏的反转状态 |

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

| 返回值 | 说明 |
|---|---|
| CYRET_BAD_PARAM | **InvertState**参数赋值失败 |
| CYRET_SUCCESS | 成功完成的功能 |

**Side Effects（副作用）：** None（无）

## uint8 Seg_Display_ReadInvertState(void)

**说明：** 此函数用于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

| 返回值 | 说明 |
|---|---|
| Seg_Display_NORMAL_STATE | 显示屏的正常状态 |
| Seg_Display_INVERTED_STATE | 显示屏的反转状态 |

**Side Effects（副作用）：** None（无）

## void Seg_Display_ClearDisplay(void)

**说明：** 此函数用于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 Seg_Display_WritePixel(uint16 pixelNumber, uint8 pixelState)

**说明：** 此函数用来根据输入参数 **PixelState** 设置或清除某个像素。通过压缩数寻址像素。

**参数：** uint16 pixelNumber: 帧缓冲区中指向像素位置的压缩数。LSB 低效半字节中 3 个最低效位是字节中的数位位置，LSB 高效半字节（4 位）是复用行中的字节地址，而 MSB 低效半字节（4 位）是复用行编号。生成组件 .h 文件，其中包括各个像素此格式的 #定义。

uint8 pixelState: 将指定的 **PixelNumber** 设置为此像素状态。

| 值 | 说明 |
|---|---|
| Seg_Display _PIXEL_STATE_OFF | 将像素设置为关闭。 |
| Seg_Display_PIXEL_STATE_ON | 将像素设置为打开。 |
| Seg_Display_PIXEL_STATE_INVERT | 反转像素的当前值。 |

**Return Value（返回值）：** uint8 Status: 根据字节地址和复用行编号范围检查传递或失败。未对数位位置执行检查。

| 返回值 | 说明 |
|---|---|
| CYRET_BAD_PARAM | 压缩字节地址或行值无效 |
| CYRET_SUCCESS | 成功完成的功能 |

**Side Effects（副作用）：** None（无）

## uint8 Seg_Display_ReadPixel(uint16 pixelNumber)

**说明：** 此函数用来读取帧缓冲区中的像素状态。通过压缩数寻址像素。

**参数：** uint16: pixelNumber: 帧缓冲区中指向像素位置的压缩数。LSB 低效半字节中 3 个最低效位是字节中的数位位置，LSB 高效半字节（4 位）是复用行中的字节地址，而 MSB 低效半字节（4 位）是复用行编号。生成组件 .h 文件，其中包括各个像素此格式的 #定义。

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

| 值 | 说明 |
|---|---|
| 0x00 | 像素关闭。 |
| 0x01 | 像素打开。 |
| 0xFF | 像素未连接 |

**Side Effects（副作用）：** None（无）

## void Seg_Display_SetAwakeMode(void)

**说明：** 当处于低功耗模式时，将 LCD 驱动程序缓冲输出设置为高阻抗状态。

**参数：** None（无）

**Return Value（返回值）：** None（无）

**Side Effects（副作用）：** None（无）

## void Seg_Display_SetSleepMode(void)

**说明：** 当处于低功耗模式时，将 LCD 驱动程序缓冲输出设置为接地状态。

**参数：** None（无）

**Return Value（返回值）：** None（无）

**Side Effects（副作用）：** None（无）

## void Seg_Display_Sleep(void)

**说明：** Seg_Display_Sleep() 函数检查组件是否为启用状态，并保存该状态。然后调用 Seg_Display_Stop() 函数和 Seg_Display_SaveConfig() 函数以保存用户配置。

调用 Seg_Display_Sleep() 函数后，调用 CyPmSleep() 或 CyPmHibernate() 函数。有关功耗管理函数的详细信息，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）：** 请勿更改组件引脚驱动模式。

## uint8 Seg_Display_Wakeup(void)

**说明：** Seg_Display_Wakeup() 函数调用 Seg_Display_RestoreConfig() 函数以用来恢复用户配置。如果在调用 Seg_Display_Sleep() 函数之前已启用组件，则 Seg_Display_Wakeup() 函数将重新启用该组件。

**参数：** None（无）

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

| 返回值 | 说明 |
|---|---|
| CYRET_LOCKED | 某些 DMA TD 或通道已经被占用。 |
| CYRET_SUCCESS | 成功完成的功能 |

**Side Effects（副作用）：** 如果在调用 Seg_Display_Wakeup() 函数之前未调用 Seg_Display_Sleep() 或 Seg_Display_SaveConfig() 函数，可能产生意外行为。

## void Seg_Display_Init(void)

**说明：** 根据 Configure（配置）对话框设置初始化或恢复组件参数。配置并启用所有必要的硬件模块，清除帧缓冲区。

**参数：** None（无）

**Return Value（返回值）：** None（无）

**Side Effects（副作用）：** None（无）

## uint8 Seg_Display_Enable(void)

**说明：** 。并执行组件复位，设置寄存器初始值，启用所有必要的时钟。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 | 某些 DMA TD 或通道已经被占用。 |
| CYRET_SUCCESS | 成功完成的功能 |

**Side Effects（副作用）：** None（无）

## void Seg_Display_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。通过 Seg_Display_Sleep() 函数调用此函数。

**参数：** None（无）

**Return Value（返回值）：** None（无）

**Side Effects（副作用）：** None（无）

## void Seg_Display_RestoreConfig(void)

**说明：** 此函数会恢复组件配置和非保留寄存器。此外，该函数还用于将组件参数值恢复至调用 Seg_Display_Sleep() 函数之前的状态。

**参数：** None（无）

**Return Value（返回值）：** None（无）

**Side Effects（副作用）：** 如果在调用此函数之前未调用 Seg_Display_Sleep() 或 Seg_Display_SaveConfig() 函数，可能产生意外行为。

## 可选助手 API

只有在 Configure（配置）对话框中选中个别助手时，才提供下列 API。

| 函数 | 说明 |
|---|---|
| Seg_Display_Write7SegDigit_n | 显示 7 段显示元素阵列中的十六进制数字。 |
| Seg_Display_Write7SegNumber_n | 显示 7 段显示元素 1-5 位数字的整型值。 |
| Seg_Display_WriteBargraph_n | 显示线或圆条形图中的整数位置。 |
| Seg_Display_PutChar14Seg_n | 显示 14 段字母数字字符显示元素阵列中的字符。 |
| Seg_Display_WriteString14Seg_n | 显示 14 段字母数字字符显示元素阵列中的空字符结束的字符串。 |
| Seg_Display_PutChar16Seg_n | 显示 16 段字母数字字符显示元素阵列中的字符。 |
| Seg_Display_WriteString16Seg_n | 显示 16 段字母数字字符显示元素阵列中的空字符结束的字符串。 |
| Seg_Display_PutCharDotMatrix_n | 显示点阵字母数字字符显示元素阵列中的字符。 |
| Seg_Display_WriteStringDotMatrix_n | 显示点阵字母数字字符显示元素阵列中的空字符结束的字符串。 |

**注意**：包含后缀 “n” 的函数名称表示在组件自定义程序中创建多个相同符号类型的显示助手。特定显示助手元素由函数名称中包含各自后缀 “n” 的 API 函数来控制。

### void Seg_Display_Write7SegDigit_n(uint8 digit, uint8 position)

**说明：** 此函数用来显示 7 段显示元素阵列中的十六进制数字。这些数字可以是十六进制值，取值范围为 0-9 和 A-F。自定义程序显示助手工厂必须用于定义与 7 段显示元素关联的像素集。在帧缓冲区中，可以定义多个 7 段显示元素，并通过函数名称中的后缀 (n) 来寻址。只有在组件自定义程序中定义 7 段显示元素，才包含此函数。

**参数：** uint8 digit: 取值范围 0-16 中未分配的整型值显示为十六进制数字。此外，十六进制字符的 ASCII 数字也有效。如果**数字**无效，则在指定位置显示 0 **值**。将此参数设置为 16，可以清除指定**位置**上的数字。

uint8 position: 从右至左计算数字位置，右侧以 0 开头。如果数字位置伴于定义显示区域之后，则字符将不显示。

**Return Value（返回值）：** None（无）

**Side Effects（副作用）：** None（无）

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

**说明：** 此函数用来显示 7 段显示元素 1-5 位数字阵列的 16 位整型值。自定义程序显示助手工厂必须用于定义与 7 段显示元素关联的像素集。在帧缓冲区中，可以定义多个 7 段显示元素组，并通过函数名称中的后缀 (n) 来寻址。符号转换、符号显示、小数点和其他自定义特性必须由应用特定的用户代码来进行处理。只有在组件自定义程序中定义 7 段显示元素，才包含此函数。

**参数：** uint16 value：显示无符号的整型值。

uint8 position: 由右至左计数最低有效数字的位置，右侧以 0 开头。如果定义的显示区域包含值所需的较少数字，则不会显示最高有效数字或所有数字

uint8 mode：设置显示模式。可能为 0 或 1。

**Return Value（返回值）：** None（无）

**Side Effects（副作用）：** None（无）

## void Seg_Display_WriteBargraph_n(uint8 location, uint8 mode)

**说明：** 此函数用来显示 1-255 段条形图中的 8 位整数位置（编号顺序从左到右）。条形图可能是用户定义的 1-255 段之间的任意大小。此外，条形图还可以创建为圆形，用来显示旋转位置。自定义程序显示助手工厂必须用于定义与条形图显示元素相关联的像素集。在帧缓冲区中，可以创建多个条形图显示，并通过函数名称中的后缀 (n) 来寻址。只有在组件自定义程序中定义条形图显示元素，才可以包含此函数。

**参数：** uint8 location: 显示无符号的整数位置。有效值为从 0 到条形图的段数。0 值关闭所有条形图元素。大于条形图段数的值导致所有元素为打开状态。

uint8 mode：设置条形图显示模式。

| 值 | 说明 |
|---|---|
| 0 | 打开特定位置的段。 |
| 1 | 位置段和左侧所有段均为打开状态。 |
| –1 | 位置段和右侧所有段均为打开状态。 |
| 2-10 | 显示位置段和右侧 2-10。此模式可用于创建宽位指示符。 |

**Return Value（返回值）：** None（无）

**Side Effects（副作用）：** None（无）

## void Seg_Display_PutChar14Seg_n(uint8 character, uint8 position)

**说明：** 此函数用来显示 14 段字母数字字符显示元素阵列中的 8 位字符。自定义程序显示助手工厂必须用于定义与 14 段显示元素相关联的像素集。在帧缓冲区中，可以定义多个 14 段字母数字显示元素组，并通过函数名称中的后缀 (n) 来寻址。只有在组件自定义程序中定义 14 段显示元素，才包含此函数。

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

uint8 position: 由左至右计数的字符位置，左侧以 0 开头。如果数字位置位于定义的显示区以外，则将不显示该字符。

**Return Value（返回值）：** None（无）

**Side Effects（副作用）：** None（无）

## void Seg_Display_WriteString14Seg_n(*uint8 character, uint8 position)

**说明：** 此函数用来显示 14 段字母数字字符显示元素阵列中的空字符结束的字符串。自定义程序显示助手工厂必须用于定义与 14 段显示元素相关联的像素集。在帧缓冲区中，可以定义多个 14 段字母数字显示元素组，并通过函数名称中的后缀 (n) 来寻址。只有在组件自定义程序中定义 14 段显示元素，才包含此函数。

**参数：** *uint8 character: 指针指向空字符结束的字符串。

uint8 position: 由左至右计数的第一个字符的位置，在左侧以 0 开头。如果字符串的长度超出定义的显示区大小，则不将显示额外字符。

**Return Value（返回值）：** None（无）

**Side Effects（副作用）：** None（无）

## void Seg_Display_PutChar16Seg_n(uint8 character, uint8 position)

**说明：** 此函数用来显示 16 段字母数字字符显示元素阵列中的 8 位字符。自定义显示助手工具必须用于定义与 16 段显示元素相关联的像素集。在帧缓冲区中，可以定义多个 16 段字母数字显示元素组，并通过函数名称中的后缀 (n) 来寻址。只有在组件自定义程序中定义 16 段显示元素，才包含此函数。

**参数：** uint8 character: 要显示字符的 ASCII 值（取值范围为 0-255 的可打印 ASCII 和表扩展字符）。

uint8 position: 由左至右计数的字符位置，左侧以 0 开头。如果数字位置位于定义的显示区以外，则将不显示该字符。

**Return Value（返回值）：** None（无）

**Side Effects（副作用）：** None（无）

## (void) Seg_Display_WriteString16Seg_n(*uint8 character, uint8 position)

**说明：** 此函数用来显示 16 段字母数字字符显示元素阵列中的空字符结束的字符串。自定义程序显示助手工厂必须用于定义与 16 段显示元素相关联的像素集。在帧缓冲区中，可以定义多个 16 段字母数字显示元素组，并通过函数名称中的后缀 (n) 来寻址。只有在组件自定义程序中定义 16 段显示元素，才包含此函数。

**参数：** *uint8 character: 指针指向空字符结束的字符串。

uint8 position: 由左至右计数的第一个字符的位置，在左侧以 0 开头。如果字符串的长度超出定义的显示区大小，则不将显示额外字符。

**Return Value（返回值）：** None（无）

**Side Effects（副作用）：** None（无）

### void Seg_Display_PutCharDotMatrix_n(uint8 character, uint8 position)

**说明：** 此函数用来显示点阵字母数字字符显示元素阵列中的 8 位字符。自定义程序显示助手工厂必须用于定义与点阵显示元素相关联的像素集。在帧缓冲区中，可以定义多个点阵字母数字显示元素组，并通过函数名称中的后缀 (n) 来寻址。只有在组件自定义程序中定义点阵显示元素，才包含此函数。

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

uint8 position: 由左至右计数的字符位置，左侧以 0 开头。如果数字位置位于定义的显示区以外，则将不显示该字符。

**Return Value（返回值）：** None（无）

**Side Effects（副作用）：** None（无）

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

**说明：** 此函数有来显示点阵字母数字字符显示元素阵列中的空字符结束的字符串。自定义程序显示助手工厂必须用于定义与点阵显示元素相关联的像素集。在帧缓冲区中，可以定义多个点阵字母数字显示元素组，并通过函数名称中的后缀 (n) 来寻址。只有在组件自定义程序中定义点阵显示元素，才包含此函数。

**参数：** *uint8 character: 指针指向空字符结束的字符串。

uint8 position: 由左至右计数的第一个字符的位置，在左侧以 0 开头。如果字符串的长度超出定义的显示区大小，则不将显示额外字符。

**Return Value（返回值）：** None（无）

**Side Effects（副作用）：** None（无）

## 引脚 API

这些 API 函数用于更改由段显示器组件使用的引脚驱动模式。

| 函数 | 说明 |
|---|---|
| Seg_Display_ComPort_SetDriveMode | 设置由段显示器组件通用线路占用的所有引脚的驱动模式。 |
| Seg_Display_SegPort_SetDriveMode | 设置由段显示器组件段线路占用的所有引脚的驱动模式。 |

### void Seg_Display_ComPort_SetDriveMode(uint8 mode)

**说明：** 设置由段显示器组件通用线路占用的所有引脚的驱动模式。

**参数：** uint8 mode：所需驱动模式。有关驱动模式的信息，请参考引脚组件数据手册。

**Return Value（返回值）：** None（无）

**Side Effects（副作用）：** None（无）

### Seg_Display_SegPort_SetDriveMode(uint8 mode)

**说明：** 设置由段显示器组件段线路占用的所有引脚的驱动模式。

**参数：** uint8 mode：所需驱动模式。有关驱动模式的信息，请参考引脚组件数据手册。

**Return Value（返回值）：** None（无）

**Side Effects（副作用）：** None（无）

## 宏

### Seg_Display_COMM_NUM

为组件当前配置定义用户定义的显示中的通用线路数。

### Seg_Display_SEG_NUM

为组件当前配置定义用户定义的显示的段线路数。

### Seg_Display_BIAS_TYPE

为组件当前配置定义用户定义的显示的偏压类型。

### Seg_Display_BIAS_VOLTAGE

定义用户定义的显示的偏置电压电平。初始化过程中，将在 LCDDAC 控制寄存器中设置此值。

### Seg_Display_FRAME_RATE

为组件当前配置定义用户定义的显示的刷新率。

### Seg_Display_WRITE_PIXEL

这是包含无效类型的 Seg_Display_WritePixel() 函数的宏定义。

### Seg_Display_READ_PIXEL

这是 Seg_Display_ReadPixel() 函数的宏定义。

### Seg_Display_FIND_PIXEL

此宏用来计算帧缓冲区中的像素位置。它采用自定义程序像素表和专用于 LCD 的物理引脚的信息。此宏是像素映射机制的基础。在帧缓冲区中，使用计算后的像素位置来定义像素表中的每一个像素名称。API 使用像素名称访问各自的像素。

# 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

## 默认配置

Seg_Display 组件的默认配置提供通用的 LCD 直接段式驱动控制器。默认 Seg_Display 配置为：

- 4 个通用线路
- 8 个段线路
- 60-Hz 刷新率
- 始终有效的功耗模式
- 未定义显示助手。默认的 API 生成程序不包含所支持的任意显示元素的功能。

## 自定义配置

段显示器组件的主要特性为具有不同特征和布局的 LCD 提供灵活支持。

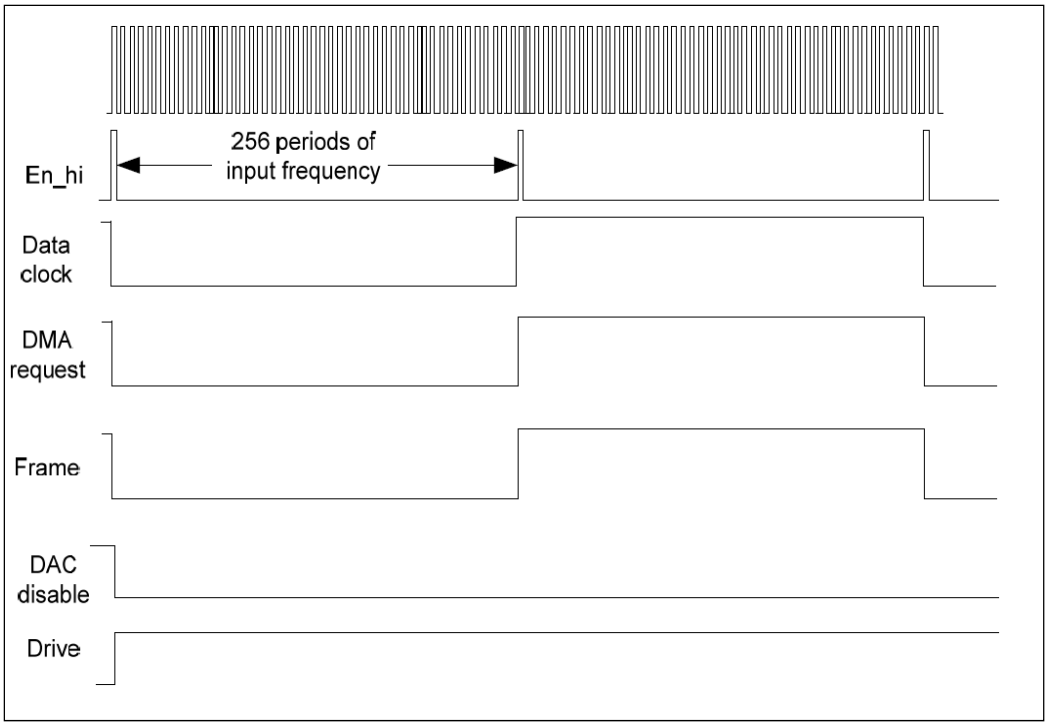
# 驱动程序功耗模式

## 始终有效的模式

在此使用模式下，LCD 可以在整个帧内获得驱动。这意味着该模式可以为 LCD DAC 供电，无论何时启用组件时，均可以将内部信号驱动置为高电平。

图 1 是 Seg_Display 组件（始终为有效模式）UDB 生成的（内部）信号的波形（类型 A）：

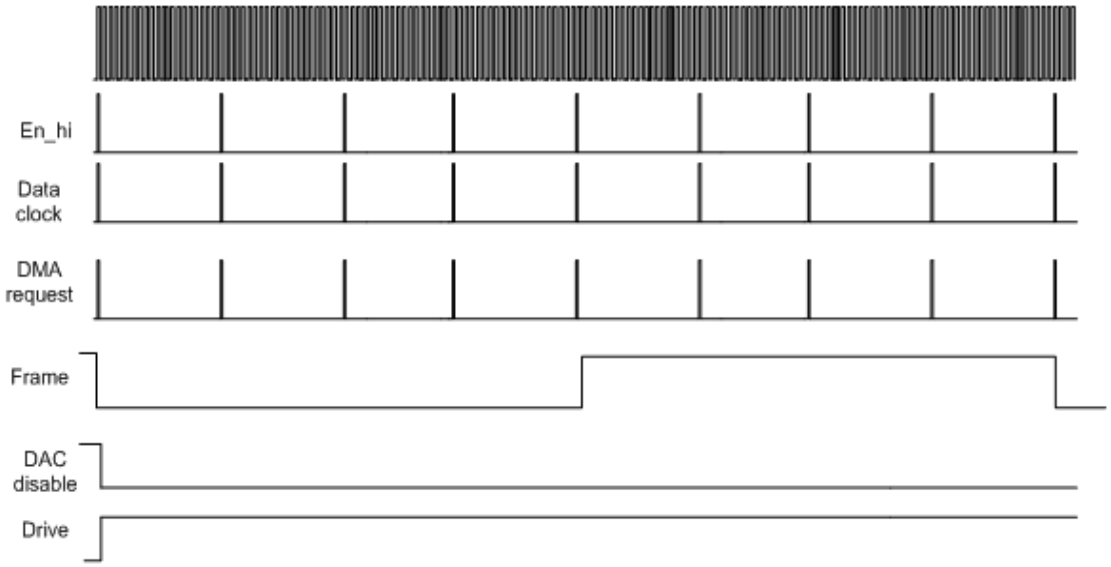
**图 1. 段显示器控制信号类型 A 波形（始终为有效模式）**




**注意**，有关详细信息，请参考时序计算。

图 2 是 Seg_Display 组件（始终为有效模式）UDB 生成的（内部）信号的波形（类型 B）：

**图 2. 段显示器控制信号类型 B 波形（始终为有效模式）**




显示的信号针对 1/4 复用率情况。

### 低功耗模式

在此使用模式下，仅在电压转变时才有效驱动 LCD，并在每次电压转变间隔内为 LCD 系统模拟组件供电。

图 3 是 Seg_Display 组件（低功耗模式）UDB 生成的信号的波形（类型 A）：

**图 3. 段显示器控制信号类型 A 波形（低功耗模式）**

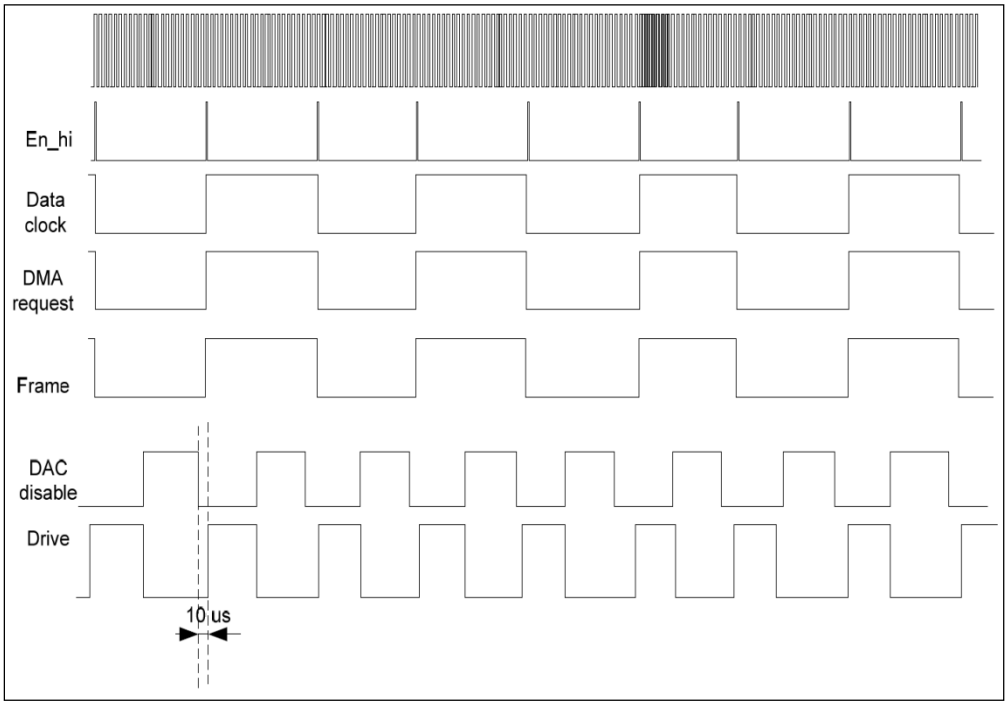
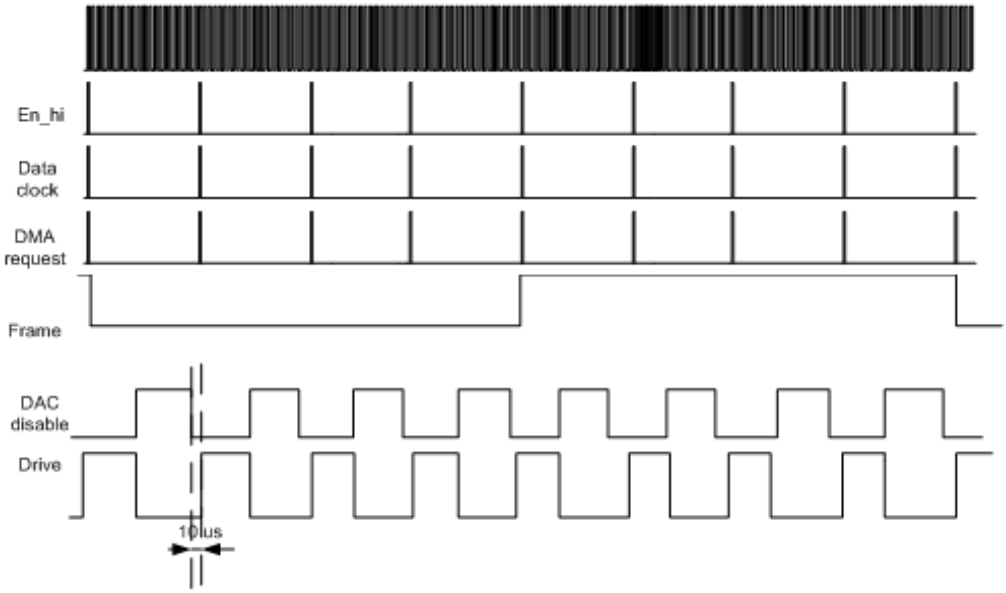


图 4 是 Seg_Display 组件（低功耗模式）UDB 生成的信号的波形（类型 B）：

**图 4. 段显示器控制信号类型 B 波形（低功耗模式）**



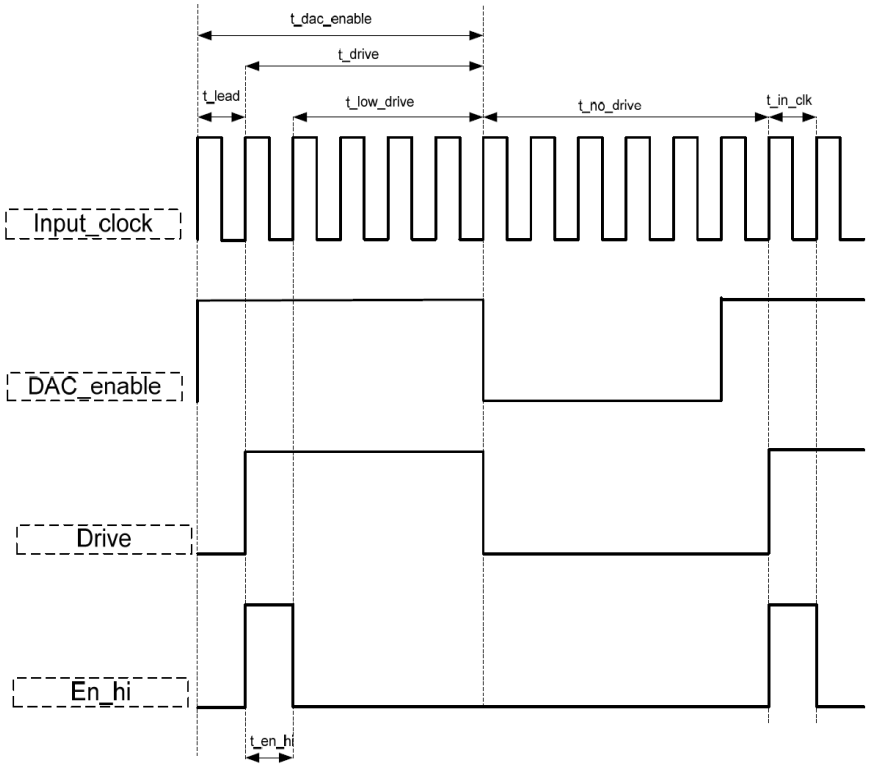

显示的信号针对 j 复用率情况。

# 时序计算

图 5 和下表说明 UDB 生成信号的时序信息。在图 5 中，仅表示 3 种内部信号。其他信号可以衍生于前几个图表。图 5 基于低功耗模式和类型 B 波形。

DAC_disable 信号是通过反转 DAC_enable 内部信号所生成的。

**图 5. 控制信号时序图示例**




| 参数 | 时间 | 说明 | 计算 |
|---|---|---|---|
| 输入时钟频率 | - | 输入时钟频率值（通过自定义程序自动设置） | f_in = f_frame_rate × N_common_lines × 256 × 2 |
| 输入时钟周期 | t_in_clk | 指定输入时钟周期 | t_in_clk = 1/(f_in) |
| 高电平驱动时间 | t_en_hi | 表示时间 HiDrive 的周期有效 | t_en_hi = t_in_clk |
| | t_lead | 指定 LCD DAC 设置时间。 | ~10 µs |
| | - | HiDrive 无效时间 | t_low_drive（始终有效）<br>t_low_drive + t_no_drive（低功耗） |
| | t_drive | 指定驱动时间 | t_drive = t_en_hi + t_low_drive |
| | t_drive | 指定驱动时间 | t_drive = t_en_hi + t_low_duty_cycle |

| 参数 | 时间 | 说明 | 计算 |
|---|---|---|---|
| | t_low_drive | 指定低电平驱动时间 | t_low_drive = t_drive - t_en_hi |
| | t_no_drive | 指定驱动信号置为低电平时的时间 | t_no_drive = t_in_clk * 256 - t_drive |
| | t_dac_enable | 指定 LCD DAC 打开过程中的时间（仅针对低功耗模式） | t_dac_enable = t_drive + t_lead |

## 用户特定配置

通过在组件自定义程序中更改时序参数，可以调整信号时序。

### 高电平驱动时间

默认情况下，t_en_hi = 128 × t_in_clk（始终有效模式），或 t_en_hi = 64 × t_in_clk（低功耗模式）。此值由自定义程序自动计算。可以将此时间增加到最大值：

HiDriveTimemax = HiDriveTimemin × 253（始终有效模式）

HiDriveTimemax = HiDriveTimemin × 247（低功耗模式）

HiDriveTime 值增量对应输入时钟的一个周期的增量。这导致 en_hi signal 的有效时间被延长。

# 框图和配置

**图 6. 段显示器组件原理图**

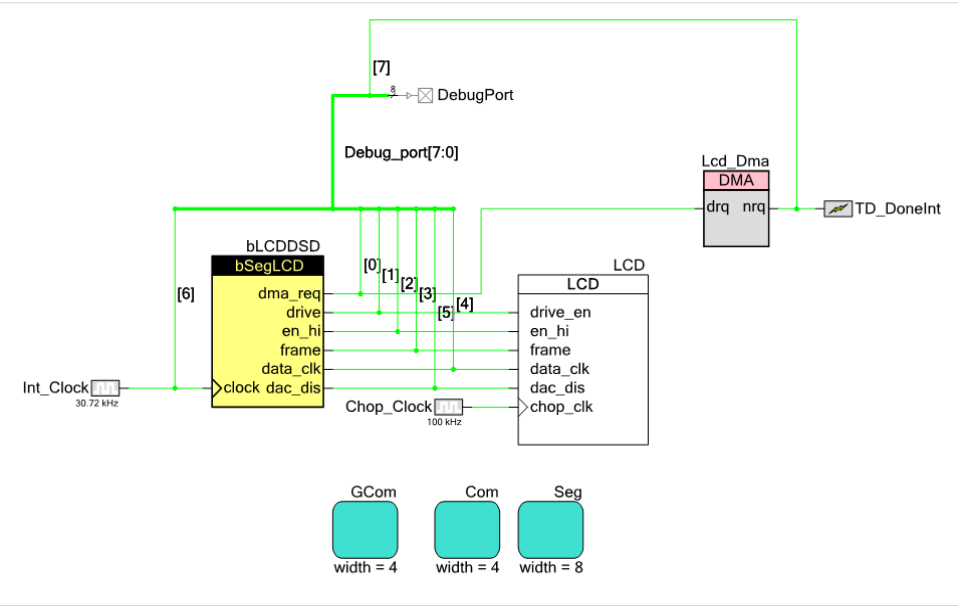
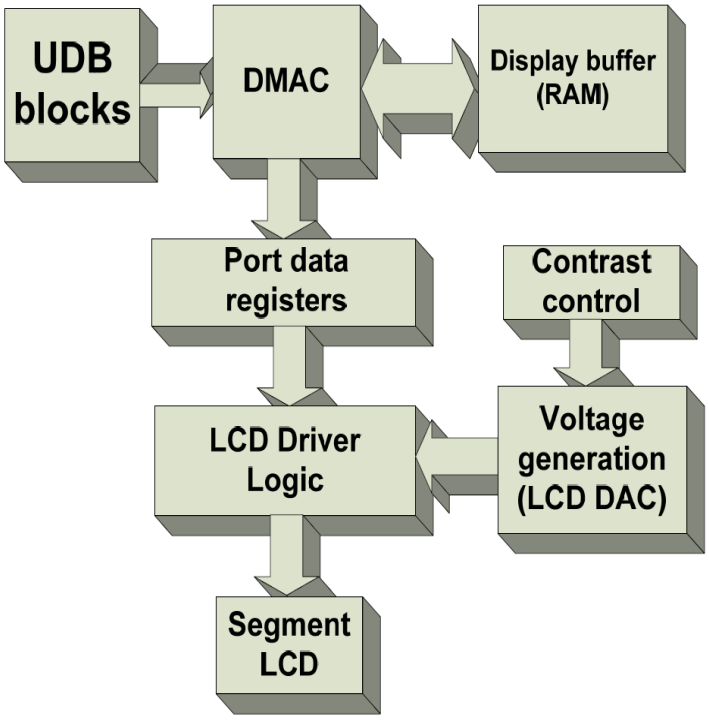


图 6说明段显示器组件的内部原理图。该组件由以下器件组成：基本的段显示器组件、LCD 控制模块 (LCD) 组件、DMA 组件、2 个 LCD 端口，1 个数字端口、ISR 组件和 2 个时钟。

- 基本段显示器负责为 LCD 端口和 DMA 组件生成适当的时序信号。
- DMA 组件用于将数据从帧缓冲区通过伪信号存储区传输到 LCD 数据寄存器。
- LCD 组件用于处理所需的 DSI 路由。此外，此模块还提供 *cyfitter.h* 中定义的所需寄存器名称。
- LCD 端口（GCom、Com 和 Seg）用于将逻辑信号映射到物理引脚。共有 2 个 LCD 端口实例；一个用于通用线路，另一个用于段线路。通用信号的 LCD 端口限定为 16 引脚宽；段信号的 LCD 端口限定为 48 引脚宽。
- DebugPort 端口组件仅用于调试。默认情况下，此组件被移除。

## 高级架构

**图 7. 高级段显示器**



# 寄存器

## Seg_Display_CONTRAST_CONTROL

保持 LCD DAC 使用的偏置电压电平，用以生成正确的偏置电压。提供 API 以用来更改偏置电压电平。

| 位 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 值 | 保留 | 对比度 | | | | | | |

- 对比度：如上所述的偏置电压电平。

## Seg_Display_LCDDAC_CONTROL

| 位 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 值 | 保留 | | | | | 禁用 DAC | 偏压选择 | |

- 偏压选择：偏压选择。
- DAC 禁用：如果不需要对比度控制，则禁用 LCD DAC。

## Seg_Display_DRIVER_CONTROL

| 位 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 值 | 保留 | | | | | 反转 | Io2 | 睡眠模式 |

- 睡眠模式 设置段显示器的睡眠模式。
- Io2: 启用或禁用 LCD 驱动程序模块 IoDrive 模式的高电流模式
- 反转：若设置，则反转段显示器中的所有数据。

## Seg_Display_CONTROL

| 位 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 值 | 保留 | | | | | | 控制 reset | 启用时钟 |

- 启用时钟：启用前面几节所述的所有内部信号的生成程序。
- 控制复位：执行组件数字部分的初始复位。

### Seg_Display_LCDDAC_SWITCH_REG[0..4]

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

- 开关控制 [0..4]：此位字段集选择 LCD 驱动程序的电压源。

# 直流和交流电气特性

## 5.0-V/3.3-V 直流和交流电气特性

| 参数 | 说明 | 条件 | 最小值 | 典型值 | 最大值 | 单位 |
|---|---|---|---|---|---|---|
| 输入电压范围 | LCD 工作电流 | | – | – | – | µA |
| | LCD 偏压范围 | | 2 | – | 5.2 | V |
| | LCD 偏压步长大小 | | – | 25.2 | – | mV |
| | 各个段/通用驱动程序的 LCD 电容 | 驱动程序可能相互组合 | – | 500 | 5000 | pF |
| | 每个段驱动器的 Iout | | | | | |
| | 强驱动 (Strong drive) | | 120 | 160 | 200 | µA |
| | 弱驱动 | | – | 0.5 | – | µA |
| | 弱驱动 2 | | – | 1 | – | µA |
| | 无驱动 | | – | 2 | – | µA |
| | 每个通用驱动程序的 Iout | | | | | |
| | 强驱动 (Strong drive) | | 160 | 220 | 300 | µA |
| | 弱驱动 | | – | 11 | – | µA |
| | 弱驱动 2 | | – | 22 | – | µA |
| | 无驱动 | | – | <25 | – | µA |

# 组件更改

| 版本 | 更改说明 | 更改/影响原因 |
|---|---|---|
| 1.0.a | 数据手册纠正 | |

版本 1.0 是段显示器组件的首次发行版本。

**注意：**段显示器仅支持 PSoC 5 芯片修订版。段式 LCD 版本 3.0 用于 PSoC 3 Production 芯片版本或更高版本。