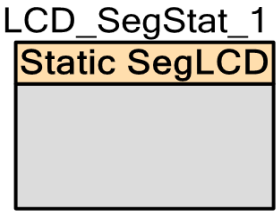
静态段 LCD (LCD_SegStat)

2.0



特性

- 1 到 61 像素或符号
- 10 Hz 到 150 Hz 刷新率
- 用户定义的像素或符号图，可使用 7 段、14 段、16 段和条形图计算子程序
- 直接驱动静态（一条通用线路）LCD

概述

静态段 LCD (LCD_SegStat) 组件可直接驱动 3.3-V 和 5.0-V LCD 显示屏。此组件为您的自定义或标准显示屏提供一个简单的 PSoC 器件配置方法。

各个 LCD 像素/符号处于打开或关闭状态。静态段 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
- 1 到 255 个元素的条形图

何时使用静态段 LCD

在静态模式下（对于仅有一条通用线路的显示屏配置），如果需要直接驱动 3.3-V 或 5.0-V LCD 显示屏，使用静态段驱动 LCD 组件。静态段 LCD 组件不需要目标 PSoC 器件以提供 LCD 驱动硬件资源。组件可用于任何有足够引脚支持所需通用线路和段线路数的目标芯片。

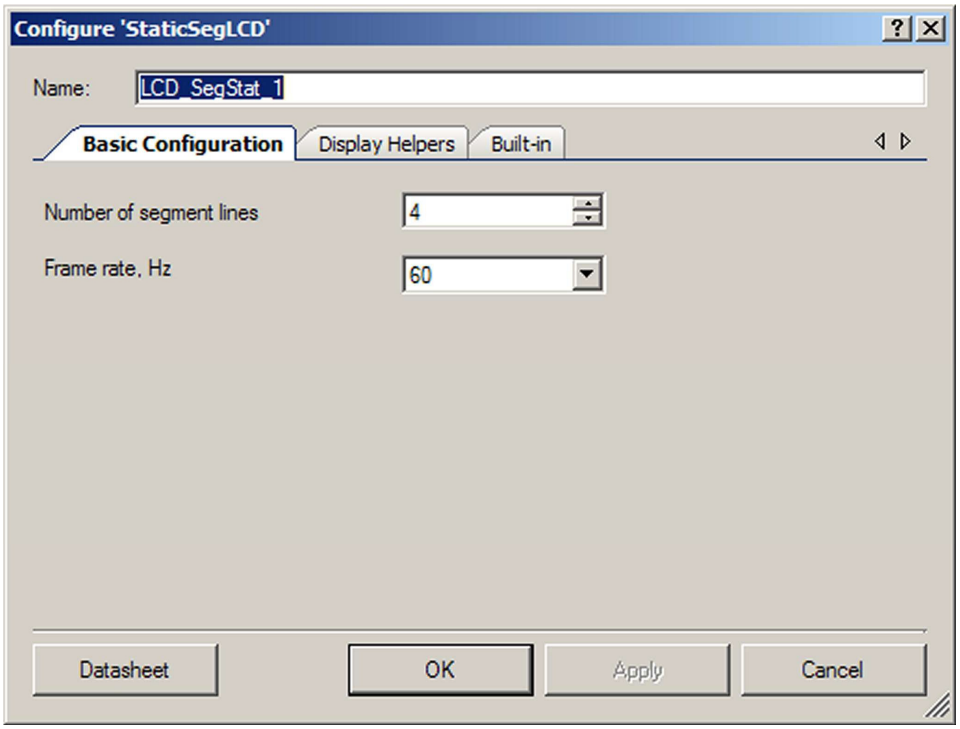
输入/输出连接

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

将一个静态段 LCD 组件拖放到您的设计上，并双击以打开 **Configure**（配置）对话框。

Basic Configuration（基本配置）选项卡



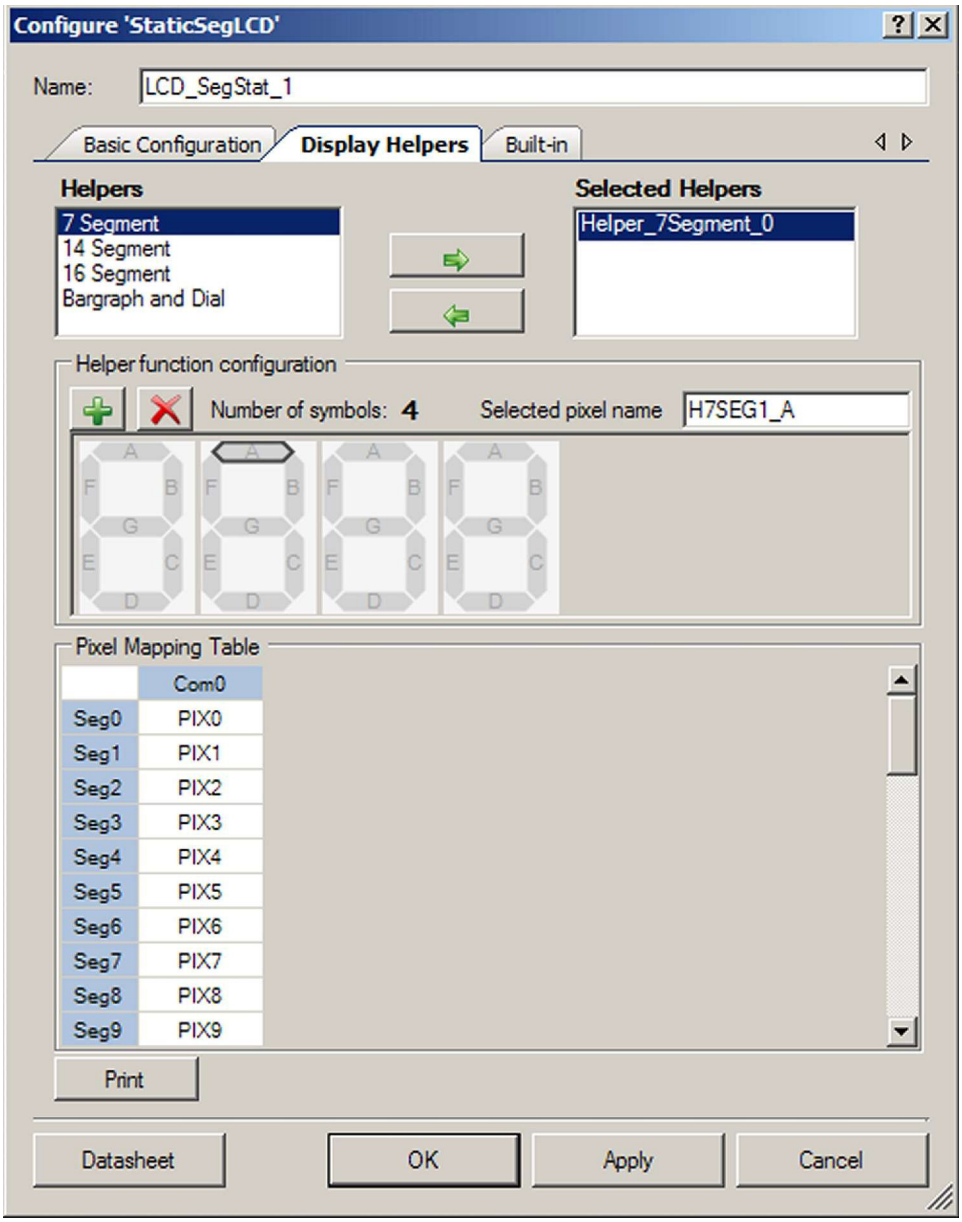
Number of segment lines（段线路数）

定义用户定义的显示屏中的段线路数。默认值为 4。

Frame rate, Hz（帧率 (Hz)）

此参数确定显示屏的刷新率。可能值为 10 Hz 到 150 Hz。默认值为 30 Hz。

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屏元素类型之一：

- 7 段、14 段 或 16 段显示屏
- 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）

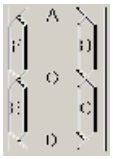
可通过在 **Helpers**（助手）列表中选择所需的助手类型并单击右箭头按钮，向 **Selected Helpers**（选定助手）添加一个或多个助手。如果没有足够引脚支持新助手，则将不添加助手。要删除助手，在 **Selected Helpers**（选定助手）列表中选择该助手，并单击左箭头按钮。

注意： 一旦将显示助手添加到组件，您将不可更改通用线路或段线路数。请务必注意，在定义任何显示助手之前，为组件设置通用线路和段线路的数量。更改通用线路和段线路的数量之前，必须移除任何已定义的显示助手。

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选定像素和指定像素左侧或右侧的所有像素。



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。

从 **Selected Helpers**（选定助手）列表中选择助手。

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。

要重新名称作为组件函数一部分的像素，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。

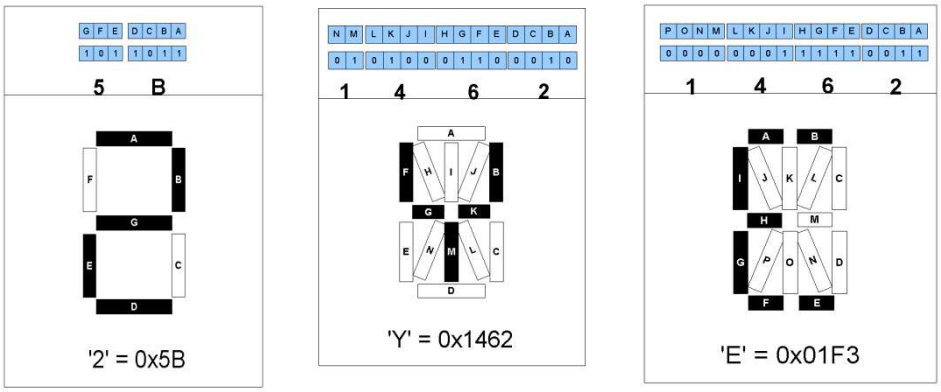
注意: 所有像素名称必须是唯一的。

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。

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，它构建了特定字符反射。以下范例显示特定字符的编码方式（段名称可能与定制器中显示的不同）。

7 段编码 **14 段编码** **16 段编码**



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 LCD 显示屏的数据表。

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。

可通过双击表显示屏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。可使用此方法对与某可用助手类型不关联的像素进行命名。

Print（打印）按钮用于打印像素映射表。

时钟选择

静态段 LCD 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。时钟生成帧率频率

放置

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, 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	状态单元	Control/Count7 单元	同步单元	中断	Flash（闪存）	RAM	
基本	0	0	0	0	0	1	1372	40	2 到 62
基本，7 段助手	0	0	0	0	0	1	2373	42	2 到 62
基本，14 段助手	0	0	0	0	0	1	2496	42	2 到 62
基本，16 段助手	0	0	0	0	0	1	2480	42	2 到 62
基本，条形图助手	0	0	0	0	0	1	3268	42	2 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_SegStat_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“LCD_SegStat”。

函数	说明
LCD_SegStat_Start()	启动 LCD 组件和 DMA 通道。初始化帧缓冲区。如果之前预定义了帧缓冲区 RAM，切勿清除它。
LCD_SegStat_Stop()	禁用 LCD 组件、相关联的中断和 DMA 通道。切勿清除帧缓冲区。
LCD_SegStat_EnableInt()	启用 LCD 中断。如果调用了 LCD_SegStat_Start()，则无需此函数。
LCD_SegStat_DisableInt()	禁用 LCD 中断。如果调用了 LCD_SegStat_Stop()，则无需此函数。
LCD_SegStat_ClearDisplay()	清除帧缓冲区的显示屏 RAM。
LCD_SegStat_WritePixel()	基于 PixelState（像素状态）设置或清除像素。通过封装数对此像素进行寻址。
LCD_SegStat_ReadPixel()	读取帧缓冲区中像素的状态。通过封装数对此像素进行寻址。
LCD_SegStat_Sleep()	停止 LCD 并保存用户配置。
LCD_SegStat_Wakeup()	恢复并启用用户配置。
LCD_SegStat_Init()	配置各个帧中断，并初始化帧缓冲区。
LCD_SegStat_Enable()	启用组件的时钟生成。
LCD_SegStat_SaveConfig()	保存 LCD 配置。
LCD_SegStat_RestoreConfig()	恢复 LCD 配置。

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”索引值。

全局变量

变量	说明
LCD_SegStat_initVar	指示是否已初始化 LCD_SegStat。变量将初始化为 0，并在第一次调用 LCD_SegStat_Start() 时设置为 1。这样，第一次调用 LCD_SegStat_Start() 子程序后，组件不用重新初始化即可重启。 如果需要重新对组件进行初始化，可在调用 LCD_SegStat_Start() 或 LCD_SegStat_Enable() 函数之前先调用 LCD_SegStat_Init() 函数。

uint8 LCD_SegStat_Start(void)

说明： 启动 LCD 组件、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_BAD_PARAM	一个或多个函数参数无效
CYRET_SUCCESS	函数成功完成

Side Effects（副作用）： None（无）

void LCD_SegStat_Stop(void)

说明： 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）：

void LCD_SegStat_EnableInt(void)

说明： 启用 LCD 中断。如果调用了 LCD_SegStat_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_SegStat_DisableInt(void)

说明： 禁用 LCD 中断。如果调用了 LCD_SegStat_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_SegStat_ClearDisplay(void)

说明： 此函数清除页面缓冲区的显示屏 RAM。

参数： None（无）

Return Value（返回值）： None（无）

Side Effects（副作用）： None（无）

uint8 LCD_SegStat_WritePixel(uint16 PixelNumber, uint8 PixelState)

说明： 此函数基于“PixelState”参数设置或清除帧缓冲区中的像素。通过封装数对此像素进行寻址。

参数： uint16 PixelNumber：指向帧缓冲区中像素位置的封装数。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（四位）是复用器行中的字节地址，MSB 低位半字节（四位）是复用器行数。

uint8 PixelState：指定的 pixelNumber 设为此像素状态。提供了像素状态的符号名称，其相关值如下所示：

值	说明
LCD_SegStat_PIXEL_STATE_OFF	将像素设为关闭。
LCD_SegStat_PIXEL_STATE_ON	将像素设为开启。
LCD_SegStat_PIXEL_STATE_INVERT	反转像素的当前状态。

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。

返回值	说明
CYRET_BAD_PARAM	封装字节地址或行值无效
CYRET_SUCCESS	函数成功完成

Side Effects（副作用）： None（无）

uint8 LCD_SegStat_ReadPixel(uint16 PixelNumber)

说明： 此函数在帧缓冲区中读取像素状态。通过封装数对此像素进行寻址。

参数： uint16：PixelNumber：指向帧缓冲区中像素位置的封装数。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（四位）是复用器行中的字节地址，MSB 低位半字节（四位）是复用器行数。

Return Value（返回值）： uint8 PixelState：返回指定 PixelNumber 的当前状态。

值	说明
0	像素为关闭状态。
1	像素为打开状态。

Side Effects（副作用）： None（无）

void LCD_SegStat_Init(void)

说明： 根据自定义程序“配置”对话框设置来初始化或恢复组件。无需调用 LCD_SegStat_Init()，因为 LCD_SegStat_Start() 子程序会调用该函数并是开始组件操作的首选方法。配置各个帧中断，并初始化帧缓冲区。

参数： None（无）

Return Value（返回值）： None（无）

Side Effects（副作用）： None（无）

void LCD_SegStat_Enable(void)

说明： 启用组件的时钟生成。

参数： None（无）

Return Value（返回值）： None（无）

Side Effects（副作用）： None（无）

注意 PSoC 5 不支持以下四个 API。

void LCD_SegStat_Sleep(void)

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

在调用 CyPmSleep() 或 CyPmHibernate() 函数之前调用 LCD_SegStat_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_SegStat_Wakeup(void)

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

参数： None（无）

Return Value（返回值）： None（无）

Side Effects（副作用）： 调用 LCD_SegStat_Wakeup() 函数前未调用 LCD_SegStat_Sleep() 或 LCD_SegStat_SaveConfig() 函数可能会产生意外行为。

void LCD_SegStat_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_SegStat_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_SegStat_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_SegStat_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 LCD_SegStat_RestoreConfig() 函数前未调用 LCD_SegStat_Sleep() 或 LCD_SegStat_SaveConfig() 函数可能会产生意外行为。

可选助手 APIs

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_SegStat_Write7SegDigit_n	显示在 7 段显示元素阵列上的十六进制数字。
LCD_SegStat_Write7SegNumber_n	显示 7 段显示元素的 1 位到 5 位阵列上的整数。
LCD_SegStat_WriteBargraph_n	显示线性或圆形条形图上的整数位置
LCD_SegStat_PutChar14Seg_n	显示 14 段字母数字字符显示元素阵列上的字符。
LCD_SegStat_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_SegStat_PutChar16Seg_n	显示 16 段字母数字字符显示元素阵列上的字符。
LCD_SegStat_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。

void LCD_SegStat_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_SegStat Write7SegNumber_n(uint8 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_SegStat_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。
2-10	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_SegStat_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_SegStat_WriteString14Seg_n(*uint8 Character, uint8 Position)

说明： 此函数显示 14 段字母数字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 段字母数字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_SegStat_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_SegStat_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）

引脚 APIs

这些 API 函数用于更改静态段 LCD 组件所用引脚的驱动模式。这些 API 大多用于将静态段 LCD 组件引脚置于 HI-Z 模式，以便在器件处于低功耗模式时将漏电流降到最低。

函数	说明
LCD_SegStat_ComPort_SetDriveMode	针对静态段 LCD 组件的通用线路使用的引脚设置驱动模式。
LCD_SegStat_SegPort_SetDriveMode	针对静态段 LCD 组件的段线路使用的所有引脚设置驱动模式。

void LCD_SegStat_ComPort_SetDriveMode(uint8 mode)

说明： 针对静态段 LCD 组件的通用线路使用的引脚设置驱动模式。

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_SegStat_SegPort_SetDriveMode(uint8 mode)

说明：	针对静态段 LCD 组件的段线路使用的所有引脚设置驱动模式。
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）

定义

LCD_SegStat_SEG_NUM

此宏针对组件当前的用户定义显示屏配置定义段线路数。

LCD_SegStat_FRAME_RATE

此宏针对组件当前的用户定义显示屏配置定义刷新率。

LCD_SegStat_WRITE_PIXEL

这是返回 void 的 LCD_SegStat_WritePixel() 函数的宏定义。

LCD_SegStat_READ_PIXEL

这是 LCD_SegStat_ReadPixel() 函数的宏定义。

LCD_SegStat_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**（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

静态段 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。

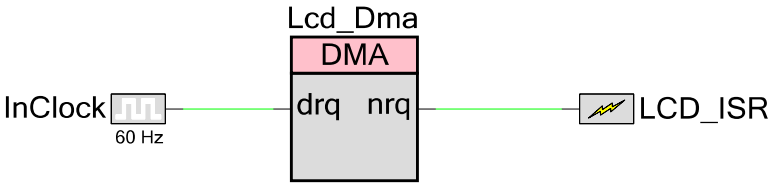
默认配置

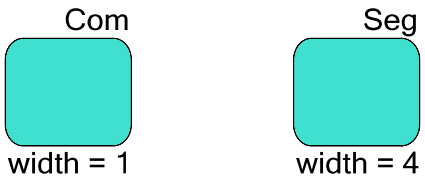
LCD_SegStat 组件的默认配置提供普通的 LCD 直接段驱动控制器。默认 LCD_SegStat 配置为：

- 四条段线路
- 30 Hz 刷新率
- 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。

框图和配置

下图显示静态段 LCD 组件的原理图展示。组件不需要专用的 LCD 驱动硬件。组件利用 DMA 和标准数字端口 I/O。





此图显示静态段 LCD 组件的内部原理图。它包含 DMA 组件、ISR 组件、时钟组件和两个 LCD 端口。

- 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 端口（Com 和 Seg）用于将逻辑信号映射到物理引脚上。 有两个 LCD 端口实例：一个是通用线路实例，一个是段线路实例。
- ISR 组件可用于用户终端。

顶层架构

组件的架构非常简单。它仅使用几个模块。组件以包含 LCD 数据的帧缓冲区为基础。可使用 LCD_SegStat_WritePixel() 函数或引用此参数的高级函数修改帧缓冲区。然后，DMA 使用伪存储器将 LCD 数据传输到端口寄存器。DMA 数据操作由内部时钟触发，此时钟设置预计算的值，以为 LCD 提供正确的刷新率。

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
2.0	已从 PSoC 5 芯片中移除低功耗 API 支持。	以下 API 已添加到条件编译中： LCD_SegStat_SaveConfig(); LCD_SegStat_RestoreConfig(); LCD_SegStat_Sleep(); LCD_SegStat_Wakeup(). 如果项目在 PSoC 5 器件上运行，则此项目不包括这些 API。
	已解决 LCD_SegStat_Stop() API 问题。	LCD_SegStat_Stop() 函数未释放 DMA 配置。这导致组件在数个 LCD_SegStat_Start() -> LCD_SegStat_Stop() 序列之后停止运行。出现此情况是因为组件用完了在 LCD_SegStat_Start() function 函数中分配的 DMA 资源，且未在 LCD_SegStat_Stop() API 中释放这些资源。在 LCD_SegStat_Stop() API 中实施和使用了正确的 DMA 释放程序。
	已将 WRITE_PIXEL() 宏定义从 #define LCD_SegStat_WRITE_PIXEL(pixelNumber, pixelState) LCD_SegStat_WritePixel(pixelNumber, pixelState) 更改为 #define LCD_SegStat_WRITE_PIXEL(pixelNumber, pixelState) (void) LCD_SegStat_WritePixel(pixelNumber, pixelState)	无需在组件实现过程中分析 LCD_SegStat_WritePixel() API 的返回值，因此更新宏以处理此情况。
	已从头文件中删除以下废弃名称： LCD_SegStat_Buffer LCD_SegStat_Channel LCD_SegStat_TermOut	已在之前的组件版本中将这些名称标记为废弃。

版本	更改说明	更改/影响原因
	LCD_SegStat_TD LCD_SegStat_GCommons LCD_SegStat_Commons	
	已优化函数 LCD_SegStat_WriteBargraph()。	代码重构为此组件的 API 节省了一些闪存空间。
	优化了函数 LCD_SegStat_WriteString14Seg() 和 LCD_SegStat_WriteString16Seg()。	已将阵列索引更改为指针索引，从而使组件节省了 RAM 空间，并从这些函数中获得更多执行时间。
	修正了整个源代码上一行中的多个变量声明。	这种情况违反了内部代码标准规范。
	优化了 LCD_SegStat_Write7SegNumber() API 实现。	LCD_SegStat_Write7SegNumber() API 中有一些在此 API 的较长执行时间内出现的无效代码。
	修正了 LCD_SegStat_Wakeup() API。	LCD_SegStat_Wakeup() API 函数即使正确执行，也始终返回 CYRET_LOCKED。已实施正确返回状态处理的程序。
1.50.a	已在数据表中将手动调用解决方案添加到 LCD_SegStat_Stop() 说明	丢失 LCD_SegStat_Stop() 子程序中对 LCD_SegStat_DmaDispose() 的调用。
	已将引脚 API 部分添加到数据表。	
	数据手册的微小编辑和更新。	
1.50	已添加睡眠 API 和唤醒 API。	用于支持低功耗模式。
	已添加存储组件状态的备份结构。	在进入睡眠模式之前，备份结构存储组件状态（已启用/已禁用）。当调用了 LCD_SegStat_Wakeup() API 以将组件从睡眠状态中唤醒时，此结构使组件返回进入睡眠模式之前的状态。
	已在“Helpers”（助手）选项卡中添加了工具提示。	执行此操作的原因是“Configure”（配置）对话框的默认大小不足以显示像素映射表。
	已为设置为启用（之前为禁用）的“段线路数”添加数字上下控制和编辑字段，即使用户添加了助手函数。	之前，如要在添加助手之后更改段线路数，您必须删除助手并丢失所有配置信息。将一个助手添加到段 LCD 组件之后，可返回并更改段线路数。如果段数增加了，清除所有信息。如果段数减少了，您将看到一则警告，提醒您会丢失配置信息。 如果要移除的段线路上有已分配的像素，将取消这些像素的分配。
	已将错误提供商添加到选定的像素名称文本框。如果在文本框中输入错误的值，将显示一个错误图标，同时您无法离开此文本框，直至输入正确的值。错误图标提供带问题说	错误提供商可更好的处理错误的值。使用错误提供商，而非显示消息框。

版本	更改说明	更改/影响原因
	明的工具提示。	
	以下全局变量用骆驼式小写名称进行了调整： #define `$LCD_SegStat`_Buffer #define `$LCD_SegStat`_Channel #define `$LCD_SegStat`_TermOut #define `$LCD_SegStat`_TD #define `$$LCD_SegStat`_GCommons #define `$$LCD_SegStat`_Commons	编码标准要求变量符合骆驼式小写命名风格。这些变量被废弃，并将在之后移除。
1.30.b	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，该工具将报告错误/警告。如果发生此情况，请更新到支持您的目标器件的修订版。
	进行了 v1.30 数据表日志累积更改	
1.30.a	已将本地参数移动到形式参数列表。	为了解决 PSoC Creator v1.0 Beta 4.1 和更早版本中存在的缺陷，更新了组件，以使它可以继续在该工具的较新版本中使用。此组件使用了本地参数（这些参数不向用户公开），以在用户输入时执行后台计算。这些参数已更改为可见的形式参数，但是不可编辑。组件没有功能上的更改，但是现在可在自定义程序对话框的“expression view”（表达式视图）中看到受影响的参数。
1.30	已更新生成的源代码。	已将 *StaticSegLCD_INT.c* 文件添加到组件的库中。 已更改组件的源代码，以针对内部中断配置使用 *CyLib.h* 函数。 已添加 DMA 补丁，以在 32 位地址空间中正常工作。 已将一个等式添加到组件源文件中，以从 DMA 中选择正确的终止信号。
1.20.b	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，该工具将报告错误/警告。如果发生此情况，请更新到支持您的目标器件的修订版。
	进行了数据手册日志累积更改	
1.10.b	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，该工具将报告错误/警告。如果发生此情况，请更新到支持您的目标器件的修订版。

版本	更改说明	更改/影响原因
1.10.a	已将本地参数移动到形式参数列表。	为了解决 PSoC Creator v1.0 Beta 4.1 和更早版本中存在的缺陷，更新了组件，以使它可以继续在该工具的较新版本中使用。此组件使用了本地参数（这些参数不向用户公开），以在用户输入时执行后台计算。这些参数已更改为可见的形式参数，但是不可编辑。组件没有功能上的更改，但是现在可在自定义程序对话框的“expression view”（表达式视图）中看到受影响的参数。

© 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。