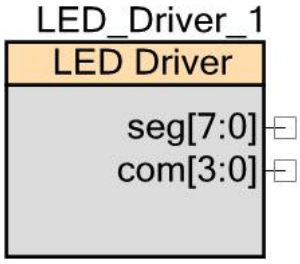
LED 段和矩阵驱动器

1.0



特性

- 多达 8 个 RGB 7 段数字，或 24 个单色 7 段数字
- 多达 8 个 14 段或 16 段数字
- 8x8 三色矩阵中具有多达 192 个 LED
- 高电平有效或低电平有效的共模信号
- 高电平有效或低电平有效的段信号
- 驱动器的复用无需 CPU 开销或中断
- 提供各函数用于使用 7 段、14 段和 16 段的数字和字符串显示
- 每个共模信号都具有独立的亮度水平

概述

LED 段和矩阵驱动器组件是一个可处理多达 24 个段信号和 8 个共模信号的复用 LED 驱动器。该组件可用于驱动 24 个 7 段 LED、8 个 14 或 16 段 LED、8 个 RGB 7 段 LED 或多达 192 个 8x8 三色矩阵 LED。通过所提供的 API，可以将字母数字值转换为段式代码，并且可以独立控制每个共模信号的亮度。该组件为 PSoC 3 和 PSoC 5LP 提供支持。

复用 LED 是节省 GPIO 引脚的有效方法，不过复用共模信号时，需要确保速率稳定。为了解决这个问题，组件在无 CPU 开销的情况下使用 PSoC 的 DMA 和 UDB 来复用 LED。因为仅使用硬件实现复用操作，这样可以排除非定期更新情况。仅在更新显示信息和修改亮度设置时才使用 CPU。

显示 7 段、14 段或 16 段数字时，不必将这些数字组合为单一的数字来显示。例如，想要显示 8 段数字时，可分为一个 2 段数字和两个 3 段数字来显示。在 LED 矩阵模式下运行时，不需要将单独显示信号组合为一个矩阵，可以将这些信号组合为许多单一的 LED 或分组 LED。该组件还支持使用信号器显示组合的数字。

何时使用 LED 段和矩阵驱动器

虽然 LCD 图形和真空荧光显示屏正在逐渐取代 LED，但是从面板式仪表到训练设备产品中仍然使用 LED。该组件提供了即快速又简便的方法来实现基于 PSoC 应用中的显示，另外设计者也需要使用最少的代码。因为复用操作由 UDB 和 DMA 处理，所以不需要 CPU 开销。如果没有定期更新，复用显示屏会使用户非常烦恼。通过使用 PSoC 3 和 5LP 的 UDB 和 DMA，可以避免非定期更新情况的发生。

输入/输出连接

本节介绍了 LED 段和矩阵驱动器组件的输入和输出连接。在描述内容所列出的条件中可能隐藏了某些 I/O 符号。

输入	是否被隐藏	说明
时钟	是	用于控制共模信号的刷新率的外部时钟终端。

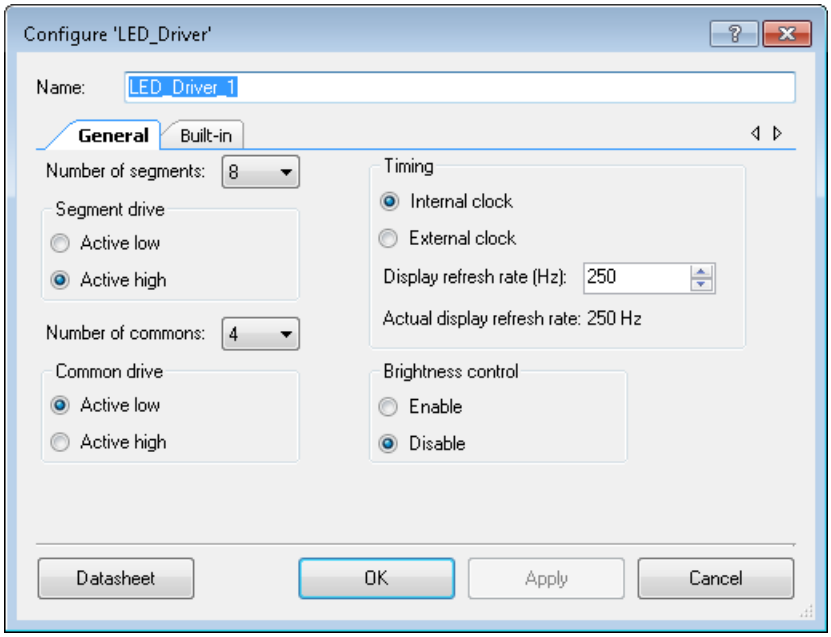
输出	是否隐藏	说明
seg[23:0]	否	驱动段式GPIO引脚的段式显示终端。根据段数设置的不同，段行数量也有所不同。
com[7:0]	否	驱动共模GPIO引脚的共模显示终端。根据共模数设置的不同，共模行数量也有所不同。

组件参数

将 LED 段和矩阵驱动器组件拖入设计窗口中，双击该组件，打开 **Configure** 对话框。

General（常规）选项卡

通用选项卡用于设置该组件的通用操作参数。它包含以下参数：所有这些设置为编译时选择项。



Number of segments（段数）

该参数用于配置段数量。它的取值范围为 1 至 24。

Segment drive（段驱动）

使用该参数可以将段设置为 **Active high（高电平有效）**或 **Active low（低电平有效）**。

Number of commons（共模数）

该参数用于配置共模数量。它的取值范围为 1 至 8。

Common drive（共模驱动）

使用该参数可以将共模设置为 **Active high** 或 **Active low**。

Timing（时序） — Clock source（时钟源）

可以将时钟源配置为内部时钟以使用最接近所需刷新率的时钟频率，或配置为外部时钟。

Timing（时序） — Display refresh rate（显示屏的刷新率）

该参数用于配置每个共模的刷新率。在无亮度控制的设计中，刷新率的取值范围为 1 Hz 至最大频率（总线时钟频率/（100 x 共模数量））。在控制亮度的情况下，推荐的最大刷新率等于（总线时钟频率/（100 x 共模数量 x 256）。下面显示的是 **Actual display refresh rate（显示屏的实际刷新率）**。根据系统的时钟分频器设置情况，实际刷新率与所需要的刷新率可能有差异。刷新率配置超过这些限制时，会导致不正确显示的结果。

如果使用外部时钟，禁用 **Brightness control（亮度控制）**时，刷新率等于时钟频率除以共模数量。**使能 Brightness control** 时，刷新率等于（时钟频率/（共模数量 x 256））。外部时钟配置中的最大刷新率限制与内部时钟配置中的相同，即不能超过上面的限制。

Brightness control（亮度控制）

每个共模的亮度可在运行时间内修改。通过该参数可以使能亮度控制功能。请注意，使能亮度控制时的组件时钟频率要比禁用亮度控制时的频率高 256 倍。

时钟选择

LED 段和矩阵驱动器组件中包含的嵌入式内部时钟由定义在定制器中的**显示屏的刷新率**配置。如果**时钟源**为外部时钟，要在设计中移除内部时钟，并且通过时钟终端提供一个时钟。

应用编程接口

通过应用编程接口（API），您可以使用软件对组件进行配置。下表列出并说明了每个函数的接口。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“LED_Driver_1”分配给指定设计中组件的第一个实例。您可以将该实例重新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“LED_Driver”。

函数	说明
LED_Driver_Init()	清除显示屏，并初始化显示阵列与寄存器。
LED_Driver_Enable()	初始化DMA并使能组件。
LED_Driver_Start()	使能并启动组件。
LED_Driver_Stop()	清除显示屏，禁用DMA并停止组件。
LED_Driver_SetDisplayRAM()	将某一值直接写入到显示屏RAM中的所指定的地址。
LED_Driver_SetRC()	在显示RAM中所指定的行和列置位该位。
LED_Driver_ClearRC()	在显示RAM中所指定的行和列清除该位。
LED_Driver_ToggleRC()	在显示RAM中所指定的行和列翻转该位。
LED_Driver_GetRC()	在显示RAM中所指定的行和列返回该位。
LED_Driver_ClearDisplay()	清零所指定共模的显示。
LED_Driver_ClearDisplayAll()	清零全部显示。
LED_Driver_Write7SegNumberDec()	显示长度可达8个字符的7段有符号整数，从指定的位置开始，扩展到所指定的数字数量。
LED_Driver_Write7SegNumberHex()	显示长度可达8个字符的7段十六进制数，从指定的位置开始，扩展到所指定的数字数量。
LED_Driver_WriteString7Seg()	显示7段、空结尾的字符串，从指定的位置开始，在字符串的结尾或显示屏的结尾结束。
LED_Driver_PutChar7Seg()	在指定的位置上显示7段ASCII编码的字符。
LED_Driver_Write7SegDigitDec()	在指定的显示屏上显示单7段的数字（0…9）。
LED_Driver_Write7SegDigitHex()	在指定的显示屏上显示单7段的数字（0…F）。
LED_Driver_Write14SegNumberDec()	显示长度可达8个字符的14段有符号的整数，从指定的位置开始，扩展到所指定的数字数量。
LED_Driver_Write14SegNumberHex()	显示长度可达8个字符的14段十六进制数，从指定的位置开始，扩展到所指定的数字数量。
LED_Driver_WriteString14Seg()	显示14段、空结尾的字符串，从指定的位置开始，在字符串的结尾或显示屏的结尾结束。
LED_Driver_PutChar14Seg()	在指定的位置上显示14段ASCII编码的字符。
LED_Driver_Write14SegDigitDec()	在指定的显示屏上显示单14段的数字（0…9）。
LED_Driver_Write14SegDigitHex()	在指定的显示屏上显示单14段的数字（0…F）。
LED_Driver_Write16SegNumberDec()	显示长度可达8个字符的16段、有符号的整数，从指定的位置开始，扩展到所指定的数字数量。
LED_Driver_Write16SegNumberHex()	显示长度可达8个字符的16段十六进制数，从指定的位置开始，扩展到所

函数	说明
	指定的数字数量。
LED_Driver_WriteString16Seg()	显示16段、空结尾的字符串，从指定的位置开始，在字符串的结尾或显示屏的结尾结束。
LED_Driver_PutChar16Seg()	在指定的位置上显示16段ASCII编码的字符。
LED_Driver_Write16SegDigitDec()	在指定的显示屏上显示单16段的数字（0…9）。
LED_Driver_Write16SegDigitHex()	在指定的显示屏上显示单16段的数字（0…F）。
LED_Driver_PutDecimalPoint()	在指定的位置上置位或清除小数点。
LED_Driver_GetDecimalPoint()	禁用小数点时，将返回0；使能小数点时，则返回1。
LED_Driver_EncodeNumber7Seg()	将输入的低4位转换为7段数据，在显示屏上显示为十六进制的数字。
LED_Driver_EncodeChar7Seg()	将ASCII编码的字母字符输入转换为7段数据，在显示屏上显示为字母字符。
LED_Driver_EncodeNumber14Seg()	将输入的低4位转换为14段数据，在显示屏上显示为十六进制的数字。
LED_Driver_EncodeChar14Seg()	将ASCII编码的字母字符输入转换为14段数据，在显示屏上显示为字母字符。
LED_Driver_EncodeNumber16Seg()	将输入的低4位转换为16段数据，在显示屏上显示为十六进制数字。
LED_Driver_EncodeChar16Seg()	将ASCII编码的字母字符输入转换为16段数据，在显示屏上显示为字母字符。
LED_Driver_SetBrightness()	为所选的共模设置所需的亮度值 （0：关闭显示；255：以最大亮度显示）
LED_Driver_GetBrightness()	返回所指定共模的亮度值。
LED_Driver_Sleep()	停止该组件并保存用户配置
LED_Driver_Wakeup()	恢复用户配置并使能组件

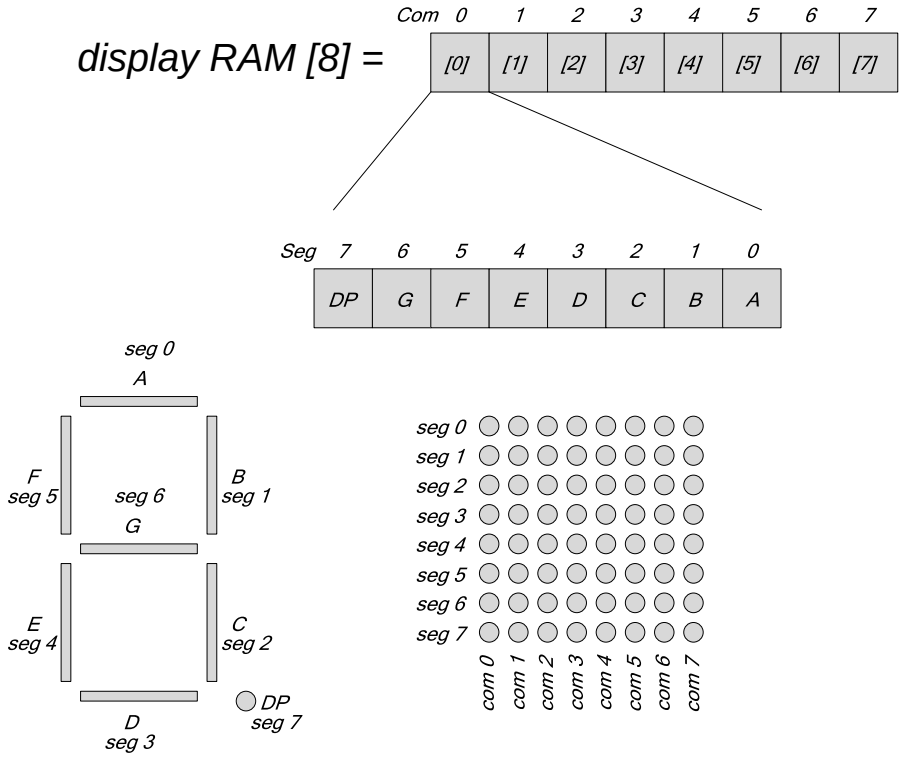
下面的多个 API 引用了“display”和“value”参数。“display”指的是显示屏 RAM 阵列中的共模位置，而“value”指的是被写入显示屏 RAM 中某个位置上的段值。图 1 显示的是一个 7 段显示和一个 8x8 LED 矩阵。显示 RAM 具有 8 个元素，每个元素对应于一个共模。这就是“display”。每个共模中保存被写入段行的 8 位“value”。

API 还引用“行”和“列”。“行”和“段”的引用相同，“列”和“共模”的引用相同。

下面显示了 7 段显示和 8x8 LED 阵列的显示 RAM 结构以及该结构与每个共模和段的映射情况。

段驱动配置为低电平有效时，LED 段的“off”值相当于显示 RAM 中的“1”。对显示 RAM 进行读写的所有 API 将自动处理显示 RAM 值的反转事项。不同的段驱动配置具有相同的应用代码，“1”和“0”值分别被翻译为“ON”和“OFF”。

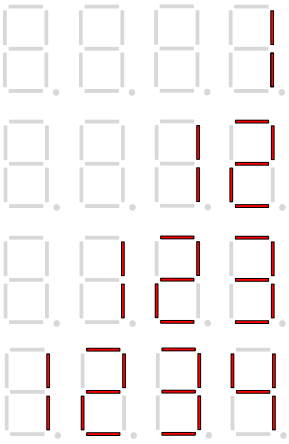
图 1. 显示 RAM



寻址多位数显示时，左边的第一个数字或最高有效数字为最低的列号。例如，如果显示的数字为“1234”，“1”由共模 0 驱动、“2”由共模 1 驱动，依此类推。共模 0、共模 1 等的引用指的是显示的“位置”，对于 7 段显示，它的范围为 0 至 23。对于 14 段和 16 段显示，允许的“位置”数量为 0 至 7。

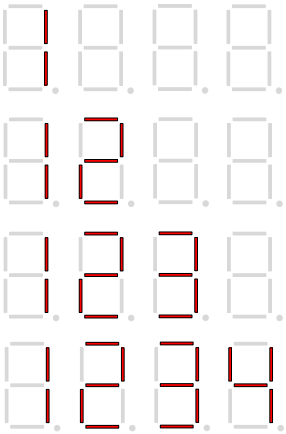
在右对齐显示模式下，最低有效数字始终被放置在最右边的显示位置，如图 2 所示。

图 2. “1”、“12”、“123”和“1234”值的右对齐显示



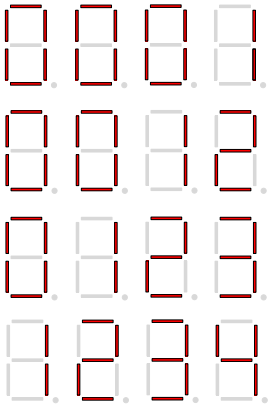
在左对齐显示模式下，最高有效数字始终被放置在最左边的显示位置，如图 3 所示。

图 3. “1”、“12”、“123”和“1234”值的左对齐显示



如果选定零填充功能，数据将以右对齐模式显示，并且有效值左边的所有位置以 0 填充。如图 4 所示。如果值为负数，“-”符号被放置在最左边。

图 4. “1”、“12”、“123”和“1234”值的零填充显示



void LED_Driver_Init(void)

说明： 清除显示，并初始化DMA。如果使能亮度控制，并初始化亮度阵列。

参数： 无

返回值： 无

其他影响： 无

void LED_Driver_Enable(void)

说明： 使能DMA，如果使能亮度控制，并使能PWM。完成这些操作后，将使能该组件。

参数： 无

返回值： 无

其他影响： 无

void LED_Driver_Start(void)

说明： 调用LED_Driver_Init()和LED_Driver_Enable()后，可以配置硬件（DMA和PWM（根据设置情况）），并使能LED显示。如果先前已经调用了LED_Driver_Init()，LED会显示RAM中的所有当前值。如果是第一次调用该函数，显示RAM将被清除。

参数： 无

返回值： 无

其他影响： 无

void LED_Driver_Stop(void)

说明： 清除显示RAM，禁用所有DMA通道并停止PWM（如果使能了亮度控制）

参数： 无

返回值： 无

其他影响： 无

void LED_Driver_SetDisplayRAM(uint8 value, uint8 position)

说明： 将‘值’直接写入显示RAM中。该函数将单一字节写入到与‘position’参数所指定的8段组对应的显示RAM内。

参数： uint8 value：写入显示RAM内的所需值。写入“1”值会使能该段，写入“0”值会禁用该段。

uint8 position：从0至23范围内的值表示显示RAM中的写入位置。对于共模0和段0到段7，写入位置从0开始，并根据共模和段按顺序给各个位置编号。例如，对于具有4数字（4共模）的16段LED，如果要写入数字1上的段8至段15，需要使用位置5。更多有关如何编号各个位置的详细信息，请参考图9和图10。

返回值： 无

其他影响： 无

void LED_Driver_SetRC(uint8 row, uint8 column)

说明： 置位显示RAM中的位。显示RAM与LED的行和列相应。请注意：行指的是段，列指的是共模。

参数： uint8 row：行值范围为0至23

uint8 column：列值范围为0至7

返回值： 无

其他影响： 无

void LED_Driver_ClearRC(uint8 row, uint8 column)

说明： 清除显示RAM中的位。显示RAM与LED的行和列相应。

参数： uint8 row：行值范围为0至23

uint8 column：列值范围为0至7

返回值： 无

其他影响： 无

void LED_Driver_ToggleRC(uint8 row, uint8 column)

说明： 翻转显示RAM中的位。显示RAM与LED的行和列相应。

参数： uint8 row：行值范围为0至23
uint8 column：列值范围为0至7

返回值： 无

其他影响： 无

uint8 LED_Driver_GetRC(uint8 row, uint8 column)

说明： 返回显示RAM中的位值。显示RAM与LED的行和列相应。

参数： uint8 row：行值范围为0至23
uint8 column：列值范围为0至7

返回值： uint8：如果位值为高电平，将返回“1”；如果位值为低电平，则返回“0”。

其他影响： 无

void LED_Driver_ClearDisplay(uint8 position)

说明： 清除‘position’参数所指定的8段组的显示（禁用所有LED）。

参数： uint8 position：从0至23范围内的值表示的是显示RAM中的写入位置。对于共模0和段0到段7，写入位置从0开始，并根据共模和段按顺序给各个位置编号。例如，对于具有4数字（4共模）的16段LED，如果要写入数字1上的段8至段15，需要使用位置5。更多有关如何编号各个位置的详细信息，请参考图9和图10。

返回值： 无

其他影响： 无

void LED_Driver_ClearDisplayAll(void)

说明： 将“0”写入显示RAM中的所有位置以清除全部显示。

参数： 无

返回值： 无

其他影响： 无。

void LED_Driver_Write7SegNumberDec(int32 number, uint8 position, uint8 digits, uint8 alignment)

说明： 显示长度可达8个字符的7段有符号的整数，从指定的位置开始，扩展到所指定的数字数量。负号（若有）将占用一个位数。如果数值超过了所指定的数字数量，将显示各最低有效数字。例如，如果数值为-1234，位置为0而且有4个数字，那么显示的结果将为：-234。注意，各个数字的位置是连续的，因此用户可以根据应用的要求选择正确的位置。另外，超过所配置的共模数量的数字将被清除。更多信息，请参考图9和图10。

参数： int32 number：需要显示的带符号的整数。用户需要保证显示有足够的位数以准确显示输入该函数的数值。否则，只会显示各最低有效位。也要注意，与等值的正数相比，负数需要多一个数字，用于显示负号。

uint8 position：显示/共模RAM的起始数字位置。

uint8 digits：显示数值时所需的位数数量。负号（若有）将占用一个位数。如果数字超过了所配置的共模数量，这些数字将被清除。

uint8 alignment：如何在所分配的数字上对齐所提供的数值。

值	说明
LED_Driver_RIGHT_ALIGN	最低有效数字占用最右边的位数（位置 + 数字）。未使用的数字被关闭。
LED_Driver_LEFT_ALIGN	最高有效数字（或负号）占用位置参数所指定的位数。未使用的数字被关闭。
LED_Driver_ZERO_PAD	左边的未使用数字用“0”填充。

返回值： 无

其他影响： 无

void LED_Driver_Write7SegNumberHex(uint32 number, uint8 position, uint8 digits, uint8 alignment)

说明： 显示长度可达8个字符的7段有符号的十六进制数，从指定的位置开始，扩展到所指定的数字字符。如果数值超过所指定的数字数量，将显示各最低有效数字。例如，如果数值为0xDEADBEEF，位置为0而且有4个数字，那么显示的结果为：BEEF。注意，各个数字的位置是连续的，因此用户可以根据应用的要求选择正确的位置。另外，超过所配置的共模数量的数字将被清除。更多信息，请参考图9和图10。

参数： uint32 number：要显示的十六进制数。用户需要保证显示有足够的位数以准确显示输入该函数的数值。否则，只会显示各最低有效位。

uint8 position：显示/共模RAM的起始数字位置。

uint8 digits：显示数值时所需的数字数。如果数字超过所配置的共模数量，这些数字将被清除。

uint8 alignment：如何在所分配的数字上对齐所提供的数值。

值	说明
LED_Driver_RIGHT_ALIGN	最低有效数字占用最右边的位数（位置 + 数字）。未使用的数字被关闭。
LED_Driver_LEFT_ALIGN	最高有效位占用位置参数所指定的数字。未使用的数字被关闭。
LED_Driver_ZERO_PAD	左边的未使用数字用“0”填充。

返回值： 无

其他影响： 无

void LED_Driver_WriteString7Seg(char8 const character[], uint8 position)

说明： 显示7段、空结尾的字符串，从指定的位置开始，在字符串的结尾或配置的共模数结尾结束。有关可显示字符的信息，请参考功能描述部分中介绍的内容。不可显示的字符将产生一个空格。注意，各个数字的位置是连续的，因此用户可以根据应用的要求选择正确的位置。更多信息，请参考图9和图10。

参数： char8 const character[]：要显示的空结尾字符串。

字符串的起始位置。

返回值： 无

其他影响： 无

void LED_Driver_PutChar7Seg(char8 character, uint8 position)

说明： 在指定的位置上显示7段且ASCII编码的字符。该函数可以显示所有字母数字字符，以及‘-’、‘.’、‘_’、‘,’和‘=’。 所有未知字符以空格显示。注意，各个数字的位置是连续的，因此用户可以根据应用的要求选择正确的位置。更多信息，请参考图9和图10。

参数： char8 character：ASCII字符

uint8 position：字符的位置

返回值： 无

其他影响： 无

void LED_Driver_Write7SegDigDec(uint8 digit, uint8 position)

说明： 在指定的显示屏上显示单7段的数字。指定的‘数字’（0 – 9）被放置在指定的位置上。注意，各个数字的位置是连续的，因此用户可以根据应用的要求选择正确的位置。更多信息，请参考图9和图10。

参数： uint8 digit：0至9范围内的显示数字。

uint8 position：数字的位置。

返回值： 无

其他影响： 无

void LED_Driver_Write7SegDigHex(uint8 digit, uint8 position)

说明： 在指定的显示屏上显示单7段的数字。指定的‘数字’（0 – F）被放置在指定的位置上。注意，各个数字的位置是连续的，因此用户可以根据应用的要求选择正确的位置。更多信息，请参考图9和图10。

参数： uint8 digit：0x0至0xF（0至15）范围内的数字

uint8 position：数字的位置。

返回值： 无

其他影响： 无

void LED_Driver_Write14SegNumberDec(int32 number, uint8 position, uint8 digits, uint8 alignment)

说明： 显示长度可达8个字符的14段有符号的整数，从指定的位置开始，扩展到所指定数量的字符。负号（若有）将占用一个位数。如果数值超过了所指定的数字数量，将显示各最低有效数字。例如，如果数值为-1234，位置为0并且有4个数字，那么显示的结果为：-234。

参数： int32 number：需要显示的带符号的整数。用户需要保证显示有足够的位数以准确显示输入该函数的数值。否则，只会显示各最低有效位。也要注意，与等值的正数相比，负数需要多一个数字，用于显示负号。

uint8 position：显示/共模RAM的起始数字位置。

uint8 digits：显示数值时所需的位数数量。负号（若有）将占用一个位数。

uint8 alignment：如何在所分配的数字上对齐提供的数值。

值	说明
LED_Driver_RIGHT_ALIGN	最低有效数字占用最右边的位数（位置 + 数字）。未使用的数字被关闭。
LED_Driver_LEFT_ALIGN	最高有效数字（或负号）占用位置参数所指定的位数。未使用的数字被关闭。
LED_Driver_ZERO_PAD	左边的未使用数字用“0”填充。

返回值： 无

其他影响： 无

void LED_Driver_Write14SegNumberHex(uint32 number, uint8 position, uint8 digits, uint8 alignment)

说明： 显示长度可达8个字符的14段十六进制数，从指定的位置开始，扩展到所指定的数字字符。如果数值超过所指定的位数数量，将显示各最低有效数字。例如，如果数值为0xDEADBEEF，位置为0而且有4个数字，那么显示的结果为：BEEF。

参数： uint32 number：要显示的十六进制数。用户需要保证显示有足够的位数以准确显示输入该函数的数值。否则，只会显示各最低有效位。

uint8 position：显示/共模RAM的起始数字位置。

uint8 digits：显示数值时所需的数字数量。

uint8 alignment：如何在所分配的数字上对齐提供的数值。

值	说明
LED_Driver_RIGHT_ALIGN	最低有效数字占用最右边的位数（位置 + 数字）。未使用的数字被关闭。
LED_Driver_LEFT_ALIGN	最高有效位占用位置参数所指定的数字。未使用的数字被关闭。
LED_Driver_ZERO_PAD	左边的未使用数字用“0”填充。

返回值： 无

其他影响： 无

void LED_Driver_WriteString14Seg(char8 const character[], uint8 position)

说明： 显示14段的空结尾的字符串，从指定的位置开始，在字符串的结尾或显示屏的结尾结束。有关可显示字符的信息，请参考《功能说明》部分中介绍的内容。不可显示的字符将产生一个空格。

参数： char8 const character[]：要显示的空结尾字符串。

字符串的起始位置。

返回值： 无

其他影响： 无

void LED_Driver_PutChar14Seg(char8 character, uint8 position)

说明： 在指定的位置上显示14段且ASCII编码的字符。该函数可以显示所有字母数字字符，以及‘-’、‘.’、‘_’、‘,’和‘=’。 所有未知字符以空格显示。

参数： uint8 character：ASCII字符
uint8 position：字符的位置

返回值： 无

其他影响： 无

void LED_Driver_Write14SegDigDec(uint8 digit, uint8 position)

说明： 在指定的显示屏上显示单14段数字。指定的‘数字’（0 – 9）被放置在指定的位置上。

参数： uint8 digit：0至9范围内的显示数字。
uint8 position：数字的位置。

返回值： 无

其他影响： 无

void LED_Driver_Write14SegDigHex(uint8 digit, uint8 position)

说明： 在指定的显示屏上显示单14段数字。指定的‘数字’（0 – F）被放置在指定的位置上。

参数： uint8 digit：0x0至0xF（0至15）范围内的数字
uint8 position：数字的位置。

返回值： 无

其他影响： 无

void LED_Driver_Write16SegNumberDec(int32 number, uint8 position, uint8 digits, uint8 alignment)

说明： 显示长度可达8个字符的16段有符号的整数，从指定的位置开始，扩展到所指定数量的字符。负号（若有）将占用一个位数。如果数值超过了所指定的数字数量，将显示各最低有效数字。例如，如果数值为-1234，位置为0并且有4个数字，那么显示的结果将为：-234。

参数： int32 number：需要显示的带符号的整数。用户需要保证显示有足够的位数以准确显示输入该函数的数值。否则，只会显示各最低有效位。也要注意，与等值的正数相比，负数需要多一个数字，用于显示负号。

uint8 position：显示/共模RAM的起始数字位置。

uint8 digits：显示数值时所需的数字数。

uint8 alignment：如何在所分配的数字上对齐提供的数值。

值	说明
LED_Driver_RIGHT_ALIGN	最低有效数字占用最右边的位数（位置 + 数字）。未使用的数字被关闭。
LED_Driver_LEFT_ALIGN	最高有效数字（或负号）占用位置参数所指定的位数。未使用的数字被关闭。
LED_Driver_ZERO_PAD	左边的未使用数字用“0”填充。

返回值： 无

其他影响： 无

void LED_Driver_Write16SegNumberHex(uint32 number, uint8 position, uint8 digits, uint8 alignment)

说明： 显示长度可达8个字符的16段十六进制数，从指定的位置开始，扩展到所指定的数字字符。如果数值超过所指定的数字数量，将显示各最低有效数字。例如，如果数值为0xDEADBEEF，位置为0而且有4个数字，那么显示的结果为：BEEF。

参数： uint32 number：要显示的十六进制数。用户需要保证显示有足够的位数以准确显示输入该函数的数值。否则，只会显示各最低有效位。

uint8 position：显示/共模RAM的起始数字位置。

uint8 digits：显示数值时所需的数字数量。

uint8 alignment：如何在所分配的数字上对齐提供的数值。

值	说明
LED_Driver_RIGHT_ALIGN	最低有效数字占用最右边的位数（位置 + 数字）。未使用的数字被关闭。
LED_Driver_LEFT_ALIGN	最高有效位占用位置参数所指定的数字。未使用的数字被关闭。
LED_Driver_ZERO_PAD	左边的未使用数字用“0”填充。

返回值： 无

其他影响： 无

void LED_Driver_WriteString16Seg(char8 const character[], uint8 position)

说明： 显示16段的空结尾的字符串，从指定的位置开始，在字符串的结尾或显示屏的结尾结束。有关可显示字符的信息，请参考《功能说明》部分中介绍的内容。不可显示的字符将产生一个空格。

参数： char8 const character[]：要显示的空结尾字符串。

字符串的起始位置。

返回值： 无

其他影响： 无

void LED_Driver_PutChar16Seg(char8 character, uint8 position)

说明： 在指定的位置上显示16段且ASCII编码的字符。该函数可以显示所有字母数字字符，以及‘-’、‘.’、‘_’、‘ ,’和‘=’。 所有未知字符以空格显示。

参数： char8 character：ASCII字符
uint8 position：字符的位置

返回值： 无

其他影响： 无

void LED_Driver_Write16SegDigDec(uint8 digit, uint8 position)

说明： 在指定的显示屏上显示单16段数字。指定的‘数字’（0 – 9）被放置在指定的位置上。

参数： uint8 digit：0至9范围内的显示数字。
uint8 position：数字的位置。

返回值： 无

其他影响： 无

void LED_Driver_Write16SegDigHex(uint8 digit, uint8 position)

说明： 在指定的显示屏上显示单16段数字。指定的‘数字’（0 – F）被放置在指定的位置上。

参数： uint8 digit：0x0至0xF（0至15）范围内的数字
uint8 position：数字的位置。

返回值： 无

其他影响： 无

void LED_Driver_PutDecimalPoint(uint8 dp, uint8 position)

说明： 在指定的位置上置位或清除小数点。

参数： uint8 dp：如果数值大于0，置位小数点，如果数值等于0，则清除小数点。
uint8 position：调整小数点的位置。

返回值： 无

其他影响： 无

uint8 LED_Driver_GetDecimalPoint(uint8 position)

说明： 禁用小数点时，将返回0；使能小数点时，则返回1。

参数： uint8 position：检查小数点的值的位置。

返回值： 返回小数点的当前状态（7段显示上的段7）。‘1’表示小数点被使能。‘0’表示小数点被禁用。

其他影响： 无

uint8 LED_Driver_EncodeNumber7Seg(uint8 number)

说明： 将输入的低4位转换为7段数据，在显示屏上显示为十六进制数字。返回的数据可以被直接写入到显示RAM内，以显示所需的数字。使用更高级别的API，既可以对数值进行译码又可以将该值写入显示RAM内，所以不需要使用该函数。

参数： uint8 number：被转换为段数据的数值，其取值范围为0x0至0xF。

返回值： 写入到显示RAM内的值，用于显示特定的数字。

其他影响： 无

uint8 LED_Driver_EncodeChar7Seg(char8 input)

说明： 将ASCII编码的字母字符输入转换为7段数据，在显示屏上显示为字母字符。返回的数据可以被直接写入显示RAM内，以显示所需的数字。使用更高级别的API，既可以对数值进行译码又可以将该值写入显示RAM内，所以不需要使用该函数。

参数： char8 input：被转换为段数据的ASCII字母字符。

返回值： 写入到显示RAM内的值，用于显示所指定的字符。

其他影响： 无

uint16 LED_Driver_EncodeNumber14Seg(uint8 number)

说明： 将输入的低4位转换为14段数据，在显示屏上显示为十六进制的数字。返回的数据可以被直接写入显示RAM内，以显示所需的数字。使用更高级别的API，既可以对数值进行译码又可以将该值写入显示RAM内，所以不需要使用该函数。

参数： uint8 number：被转换为段数据的数值，其取值范围为0x0至0xF。

返回值： 写入到显示RAM内的值，用于显示特定的数字。

其他影响： 无

uint16 LED_Driver_EncodeChar14Seg(char8 input)

说明： 将ASCII编码的字母字符输入转换为14段数据，在显示屏上显示为字母字符。返回的数据可以被直接写入显示RAM内，以显示所需的数字。使用更高级别的API，既可以对数值进行译码又可以将该值写入显示RAM内，所以不需要使用该函数。

参数： char8 input：被转换为段数据的ASCII字母字符。

返回值： 写入到显示RAM内的值，用于显示所指定的字符。

其他影响： 无

uint16 LED_Driver_EncodeNumber16Seg(uint8 number)

说明： 将输入的低4位转换为16段数据，在显示屏上显示为十六进制数字。返回的数据可以被直接写入显示RAM内，以显示所需的数字。使用更高级别的API，既可以对数值进行译码又可以将该值写入显示RAM内，所以不需要使用该函数。

参数： uint8 number：被转换为段数据的数值，其取值范围为0x0至0xF。

返回值： 写入到显示RAM内的值，用于显示特定的数字。

其他影响： 无

uint16 LED_Driver_EncodeChar16Seg(char8 input)

说明： 将ASCII编码的字母字符输入转换为16段数据，在显示屏上显示为字母字符。返回的数据可以被直接写入显示RAM内，以显示所需的数字。使用更高级别的API，既可以对数值进行译码又可以将该值写入显示RAM内，所以不需要使用该函数。

参数： char8 input：被转换为段数据的ASCII字母字符。

返回值： 写入到显示RAM内的值，用于显示所指定的字符。

其他影响： 无

void LED_Driver_SetBrightness(uint8 bright, uint8 position)

说明： 显示有效时为共模提供一个PWM占空比，可以为所选的显示设置需要的亮度值（0：关闭显示；255：以最大亮度显示）。

参数： uint8 bright：亮度的占空比的取值范围为0到255。

uint8 position:设置亮度的位置。

返回值： 无

其他影响： 无

uint8 LED_Driver_GetBrightness(uint8 position)

说明： 返回所指定显示位置的亮度值。

参数： uint8 position：返回亮度值的位置。

返回值： 无

其他影响： 无

void LED_Driver_Sleep(void)

说明： 为组件睡眠做准备。如果组件当前启用，它将被禁用，并将由LED_Driver_Wakeup()重新启用。

参数： 无

返回值： 无

其他影响： 无

void LED_Driver_Wakeup(void)

说明： 将组件返回到调用LED_Driver_Sleep()之前的状态。

参数： 无

返回值： 无

其他影响： 无

示例固件源代码

PSoC Creator 在“Find Example Project”（查找示例项目）对话框中提供了多种包括原理图和代码示例的示例项目。要查看特定组件实例，请打开“Component Catalog”中的对话框或原理图中的组件示例。要查看通用示例，请打开‘Start Page’或 **File** 菜单中的对话框。根据要求，可以通过使用对话框中的 **Filter Options** 选项来限定可选的项目列表。

更多有关信息，请参考《PSoC Creator 帮助》部分中主题为“查找示例项目”的内容。

功能描述

LED 段和矩阵驱动器组件通过使用 PSoC 3 或 PSoC 5LP 驱动 7 段、14 段或 16 段的显示或 LED 矩阵显示。通过数字引脚，将组件的段和共模引脚连接到 GPIO 引脚，这样可以使用 PSoC 控制外部 LED 的显示。下面内容说明了显示的引脚分配情况，以及该组件可用的各种配置模式。

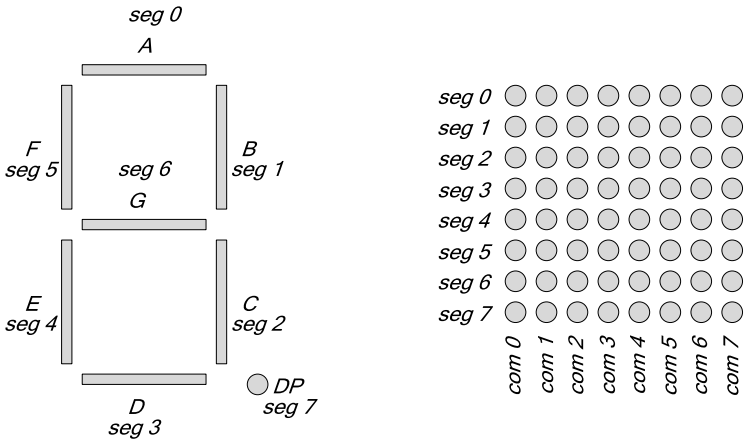
引脚分配

假设下面的引脚分配适用于显示 LED 上的数据。这些显示的支持符号如下所示。

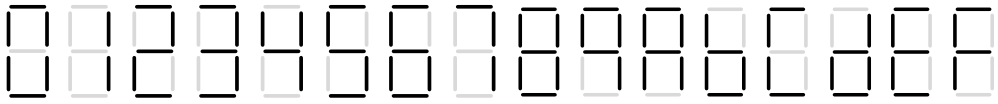
7 段与矩阵阵列

对于 7 段 LED，各段的安排如图 5 的左边所示。右边显示的是样例 8x8 LED 阵列的引脚分配。

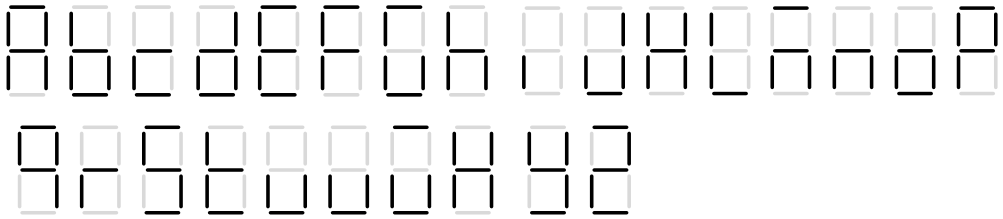
图 5. 7 段与矩阵阵列



译码数字时，7 段显示会使用下面模型：



译码大写或小写的“A”至“Z”字母时，均使用下面的模型：



特别的字符如下所示：

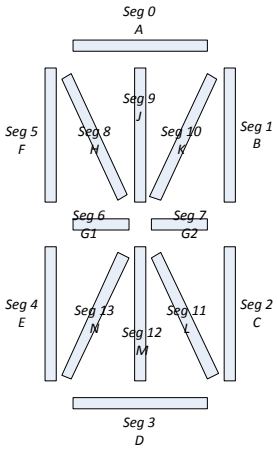
- 下划线 “_”
- 等号 “=”
- 负号 “-”
- 空格 “ ”
- 小数点 “.”



14 段显示

图 6 显示的是 14 段显示的引脚分配情况。

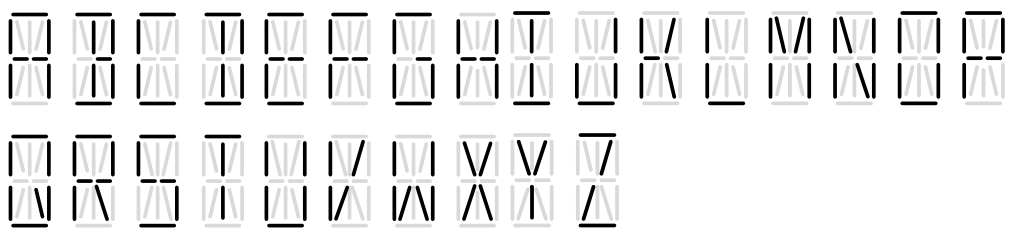
图 6. 14 段显示



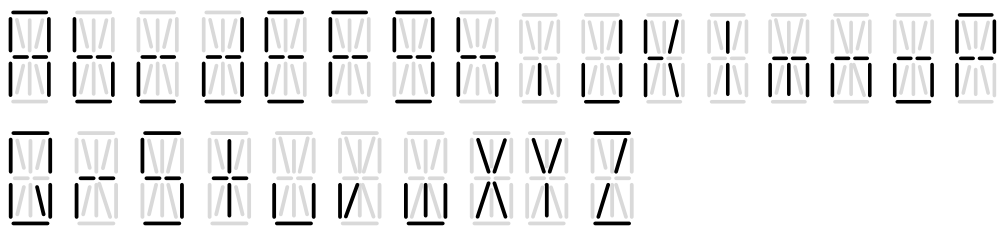
译码数字时，14 段显示会使用下面模型：



译码大写的“A”至“Z”字母时，使用下面模型：



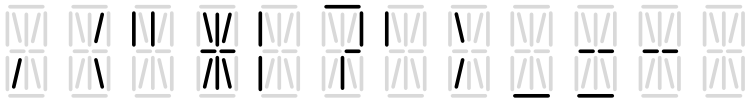
译码小写的“a”至“z”字母时，使用下面的模型：



特别的字符如下所示：

- 逗号“,”
- 左括号“（”
- 双引号“ “”
- 星号“*”
- 感叹号“!”
- 问号“?”
- 单引号“ ‘”
- 右括号“)”
- 下划线“_”
- 等号“=”
- 负号“-”

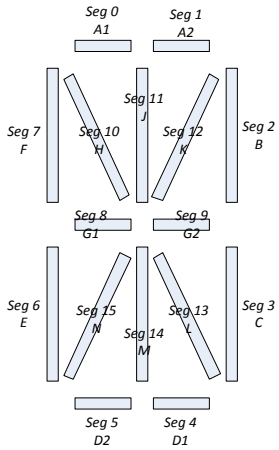
- 空格“ ”



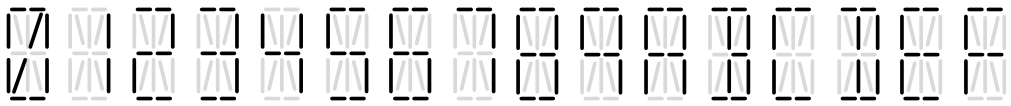
16 段显示

图 7 显示的是 16 段显示的引脚分配情况。

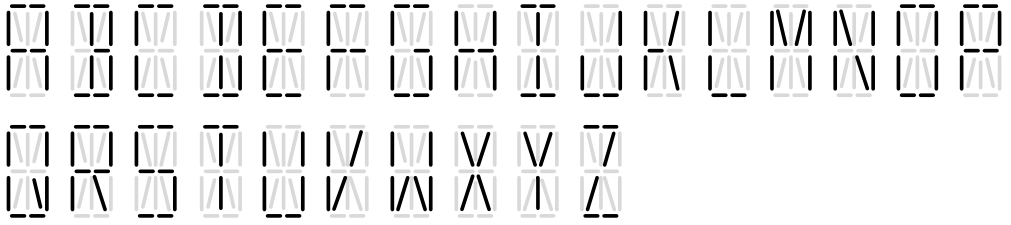
图 7. 16 段显示



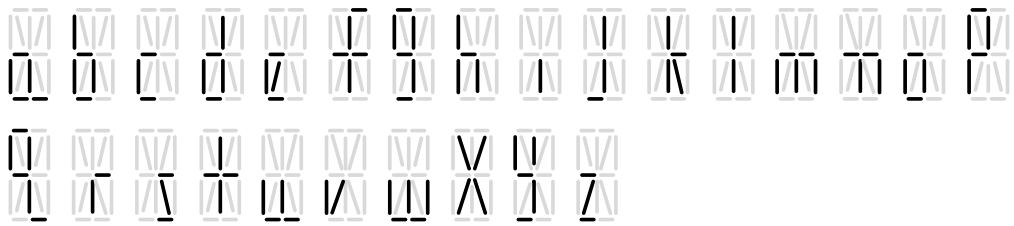
译码数字时，16 段显示会使用下面的模型：



译码大写的“A”至“Z”字母时，使用下面的模型：

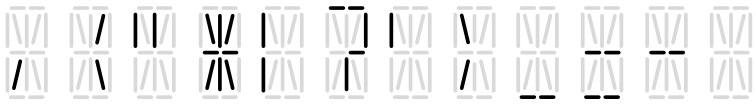


译码小写的“a”至“z”字母时，使用下面的模型：



特别的字符如下所示：

- 逗号“,”
- 左括号“(”
- 双引号“ “”
- 星号“*”
- 感叹号“!”
- 问号“?”
- 单引号“ ‘”
- 右括号“)”
- 下划线“_”
- 等号“=”
- 负号“-”
- 空格“ ”



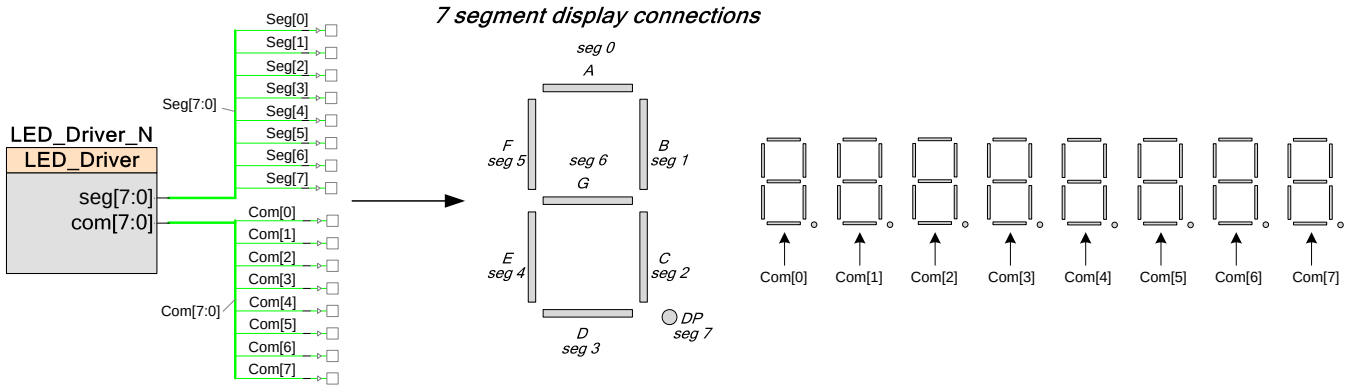
受支持的配置

LED 段和矩阵驱动器组件支持 7 段、14 段、16 段的显示和矩阵 LED。本章节介绍的是这些模式的配置情况。

7 段显示

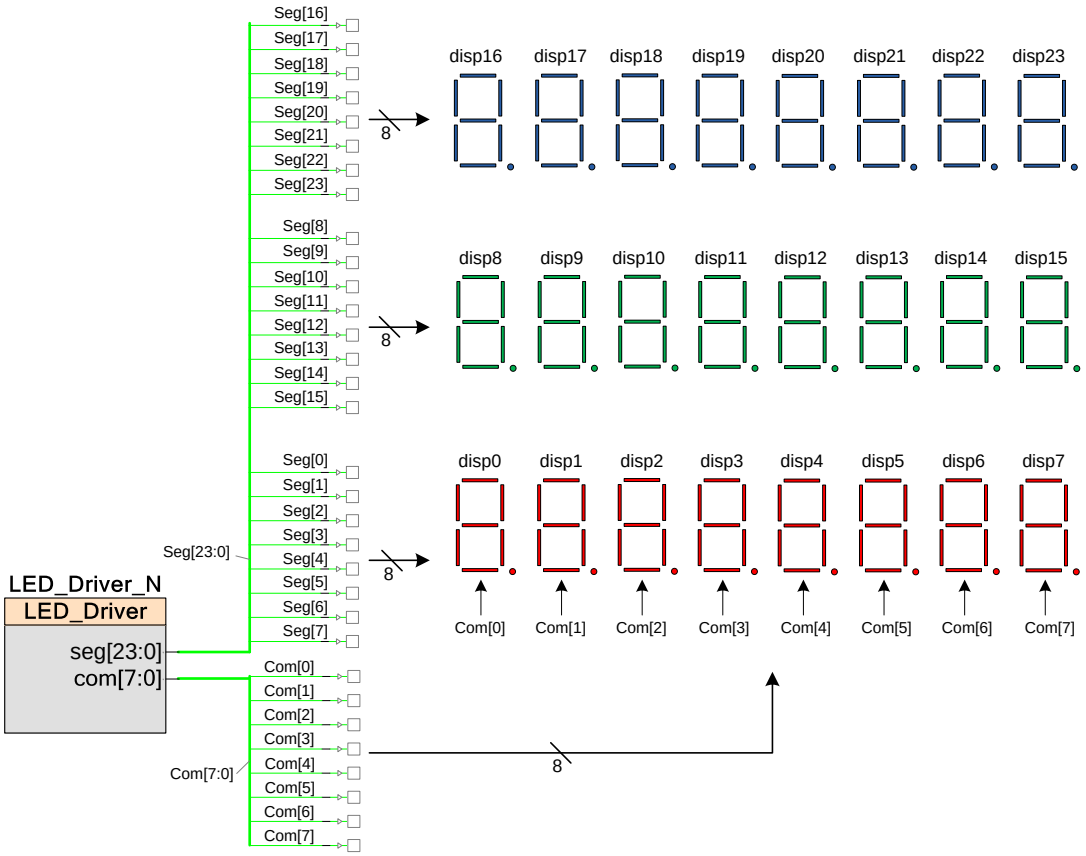
通过使用 8 个共模和 8 个段，该组件可以驱动多达八个 7 段显示。该标准配置显示在图 8 内。

图 8。7 段显示的标准配置



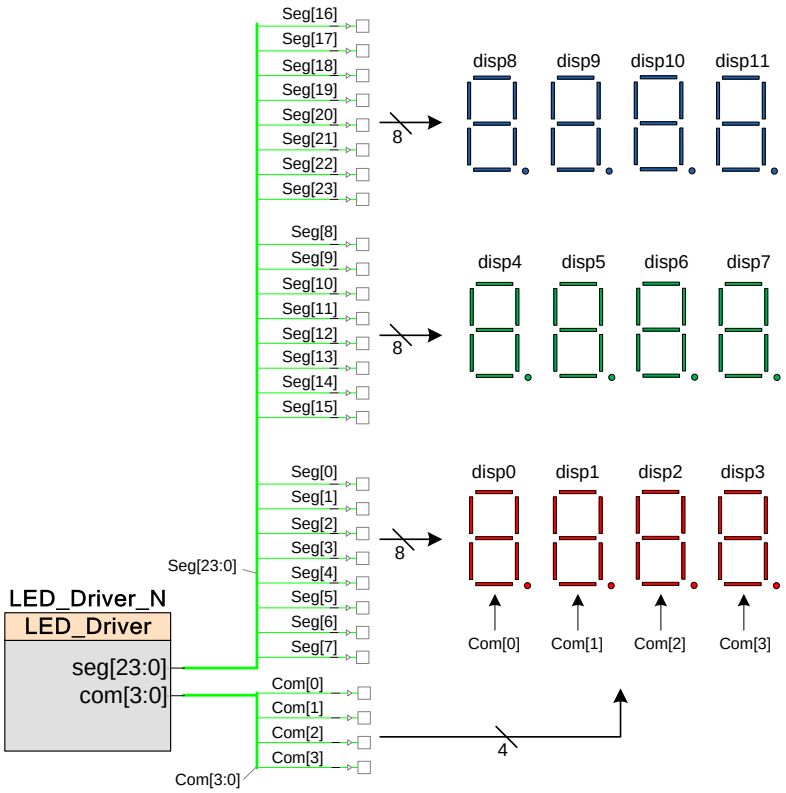
通过使用 24 个段和 8 个共模，该组件还能够显示多达 24 个 7 段显示。当用户需要使用 7 段显示来显示 24 数字或 8 数字 RGB（红色、绿色，蓝色）时，会发生这种情况。图 9 显示的是使用 RGB 7 段显示来驱动某个 8 数字显示时的组件引脚分配。

图 9. 大型的 24 数字显示或 8 数字 RGB 显示



如果定义共模的数量小于 8 而段数量大于 8，那么显示位置的引用将显示为连续。例如，图 10 显示的是某一个 7 段显示具有 4 个共模和 24 个段，用于驱动 12 个显示。该显示屏的编号将连续显示为 disp[0、1、2、3、4、5、…、11]。

图 10. 具有 4 个共模的显示阵列以及 12 个数字显示



14 段和 16 段显示

LED 段和矩阵驱动器组件可以驱动多达 8 个 14 段显示和 8 个 16 段显示。具体内容分别显示在图 11 和图 12 内。

图 11. 14 段显示的标准配置

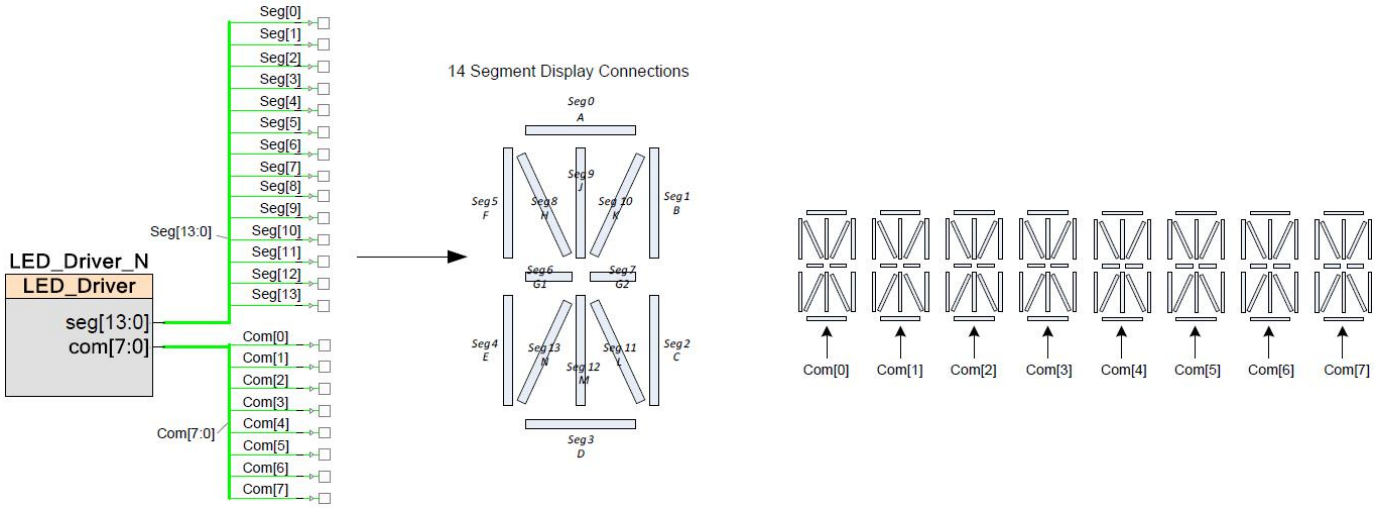
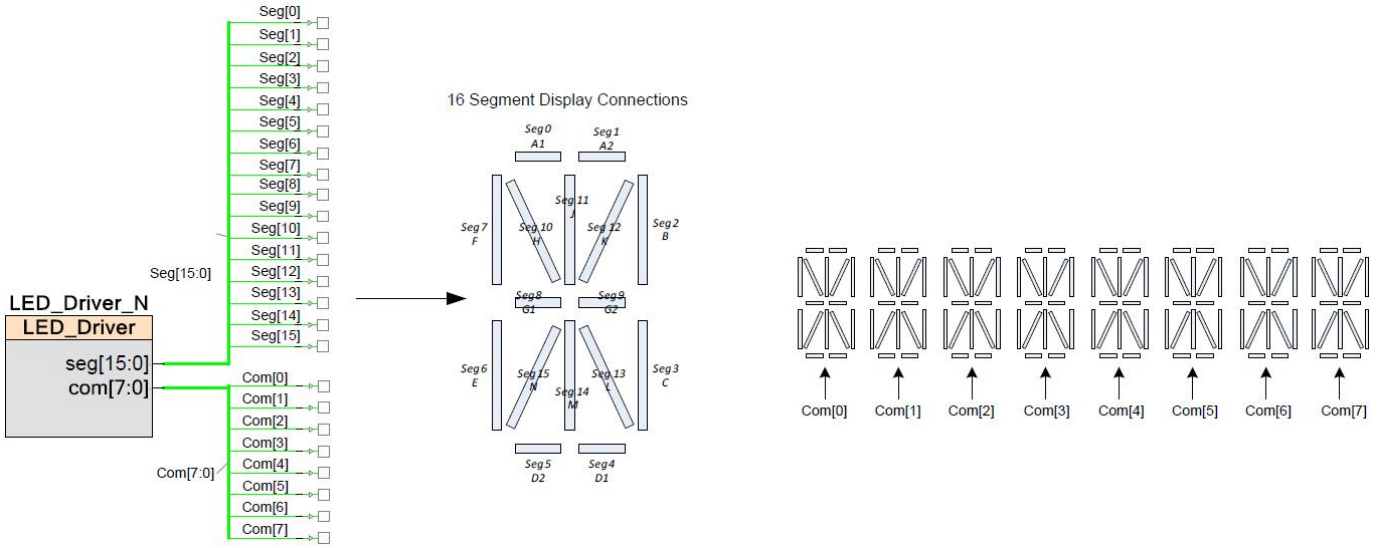


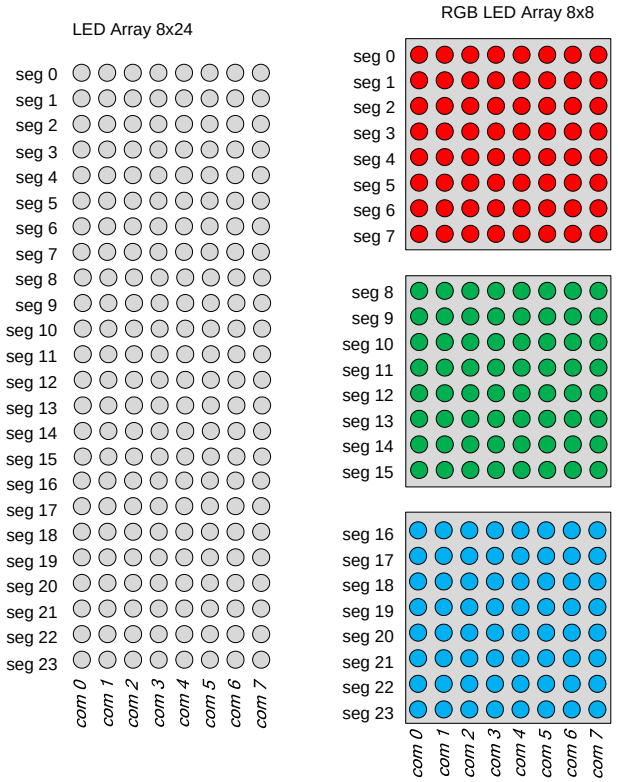
图 12. 16 段显示的标准配置



LED 矩阵显示

除了驱动段显示外，该组件还可以驱动多达 8x24LED 的矩阵。它可以是 LED 的 8x24 阵列，如图 13 的左侧所示，也可以是一个 8x8 RGB LED 阵列，如图 13 的左侧所示。

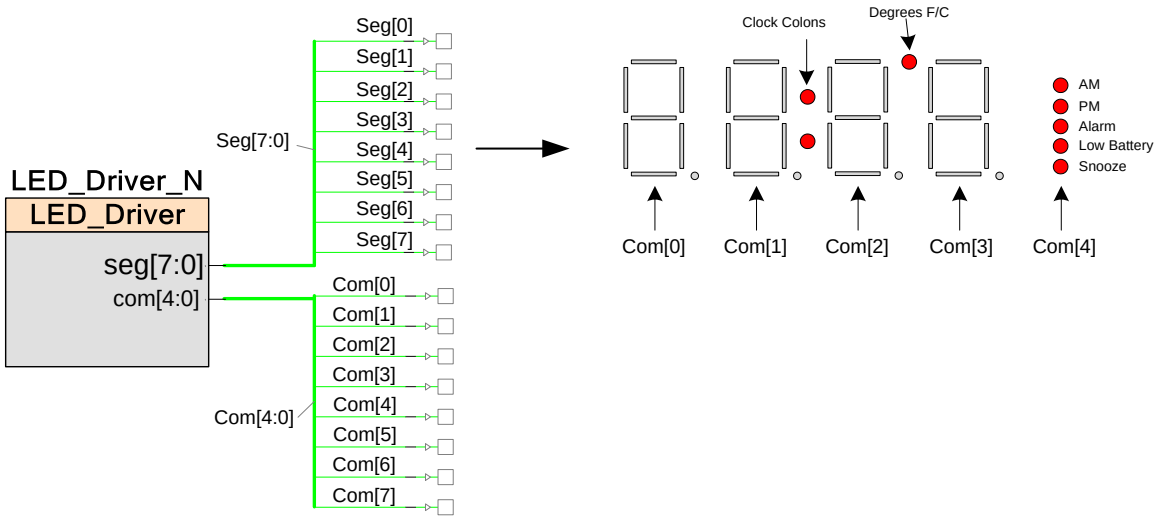
图 13. LED 的矩阵配置



信号器显示

该组件也为编码 7/14/16 段的组合显示支持信号器所需要的随机 LED。图 14 显示的是一个实例，其中 LED 段和矩阵驱动器组件驱动包含 8 个信号器的 4 个 7 段显示。AM、PM、Alarm 等信号器由 Com[4]驱动，而各数字由 Com[3:0]显示。

图 14. 7 段和随机 LED 的结合，用于各信号器



段寄存器

有三个用于保持段值的寄存器。每个寄存器包含某些元件数量（等于共模数量）。可以直接访问这些寄存器，以修改显示屏。下表显示的是各个寄存器的名称，其中“LED_Driver”表示该组件的实例名称，“*n*”即为使能的共模数量。如果段数量小于 8，则 LED_Driver_SegMBuffer 和 LED_Driver_SegHBuffer 不可用。如果段数量小于 16，则 LED_Driver_SegHBuffer 不可用。

寄存器名称	说明
LED_Driver_SegLBuffer[*n*]	用于保持段0到段7的段值。
LED_Driver_SegMBuffer[*n*]	用于保持段8到段15的段值。
LED_Driver_SegHBuffer[*n*]	用于保持段16到段23的段值。

资源

下表显示的是 LED 段和矩阵驱动器组件的资源要求。

配置		资源类型					
		数据路径单元	宏单元	状态单元	控制单元	DMA 通道	中断
禁用亮度控制	1-8段	0	1	_	2	2	_
	9-16段	0	1	_	3	2	_
	17-24段	0	1	_	4	2	_
亮度控制有效	1-8段	1	7	_	3	3	_
	9-16段	1	7	_	4	3	_
	17-24段	1	7	_	5	3	_

API 存储器大小

根据编译器、器件、所使用的 API 数量以及组件的配置情况的不同，组件所用的存储器使用情况也不一样。下表提供了给定组件配置中的所有 API 所使用存储空间。

下表中的存储器大小是在将相应编译器设置为 Release 模式并且优化选项为 Size 的情况下测得的。对于特定的设计，分析编译器生成的映射文件后可以确定存储器的使用情况。

配置		PSoC 3（Keil_PK51）		PSoC 5LP（GCC）	
		闪存字节	SRAM 字节	闪存字节	SRAM 字节
亮度控制无效	1-8段	5924	64	4560	73
	9-16段	7375	88	5164	101
	17-24段	8178	112	5572	125
亮度控制有效	1-8段	6284	75	4844	82
	9-16段	7655	99	5440	110
	17-24段	8458	123	5848	134

MISRA 合规性

本节介绍了 MISRA-C:2004 规范以及本组件的偏差情况。有两种偏差类型，如下定义：

- 项目偏差 — 适用于所有 PSoC Creator 组件的偏差
- 特定偏差 — 仅适用于该组件的偏差

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

段和矩阵驱动器组件没有任何特定偏差。

该组件具有以下嵌入式组件 — DMA。MISRA 合规性与特定偏差的相关信息请参见相应组件数据规格书。

直流和交流电气特性

除非另有说明，否则这些规范的适用范围是：-40 °C ≤ TA ≤ 85 °C 且 TJ ≤ 100 °C。电压范围为 1.71 V 到 5.5 V。

直流特性

参数	说明	最小值	典型值[1]	最大值	单位[2]
I_{DD}	组件的电流消耗（亮度控制无效）				
	8个共模，8段	_	30	_	µA/kHz
	8个共模，16段	_	40	_	µA/kHz
	8个共模，24段	_	50	_	µA/kHz
	组件的电流消耗（亮度控制有效）				
	8个共模，8段	_	120	_	µA/kHz
	8个共模，16段	_	145	_	µA/kHz
	8个共模，24段	_	150	_	µA/kHz

[1] 未包括设备 IO 和时钟分配的电流。这些值是在温度为 25 °C 条件下测得的。

[2] 电流消耗与刷新率相关。更多有关组件时钟和刷新率之间关系的信息，请参考 Timing（时序） — Display refresh rate（显示屏的刷新率部分的内容。

组件勘误表

本节列出了组件的已知问题。

赛普拉斯 ID	组件版本	问题	解决方案
191257	v1.0	在没有修正PSoC Creator 3.0 SP1中的版本编号时进行更改这组件。更多信息，请参见基础知识文章 KBA94159（网页地址：www.cypress.com/go/kba94159）。	解决方案是不必要的。设计不受任何影响。

组件更改

本节列出了该组件各版本中的主要更改内容。

版本	更改说明	更改/影响原因
1.0.a	编辑数据手册并将其添加到组件勘误章节。	文档的组件被更改，但设计不受任何影响。
1.0	第一版	

©赛普拉斯半导体公司，2014。此处所包含的信息可能会随时更改，恕不另行通知。除赛普拉斯产品内嵌的电路以外，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。也不会以明示或暗示的方式授予任何专利许可或其他权利。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，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。该处引用的所有其它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。对于合理预计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于赛普拉斯软件许可协议的限制。