

7 段 LED 控制器数据手册 LED7SEG V 1.20

Copyright © 2005-2014 Cypress Semiconductor Corporation. All Rights Reserved.

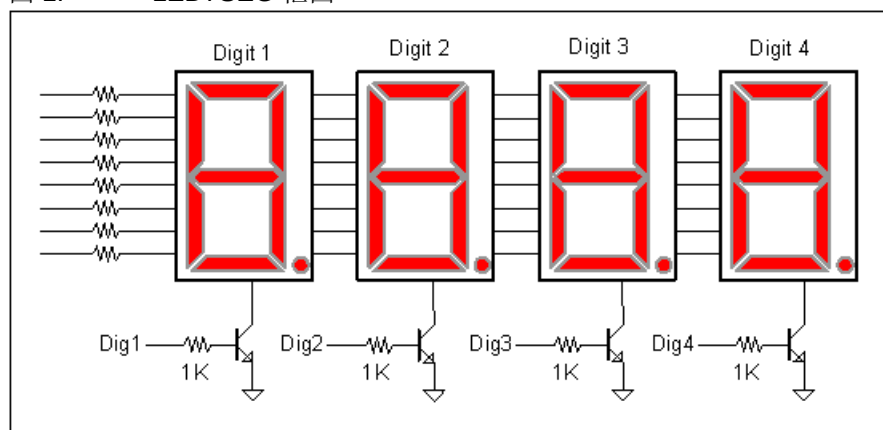
资源	PSoC® 模块			API 存储器（字节）		引脚数量 (针对每个外部 I/O)
	数字	模拟 CT	模拟 SC	闪存	RAM	
CY8C29/27/24/22/21xxx、 CY8C23x33、CY8C28xxx、 CY8CLED02/04/08/16、 CY7C64215、CY8CPLC20、 CY8CLED16P01、 CYWUSB6953	1 ^a	0	0	311	9	1

a. 如果选择了定时器项，那么只需要一个数字模块。

特性与概述

- 支持 1 到 8 个数字
- 单个显示屏最多可以组合成 8 个数字
- 可以显示十六进制数值和整数值
- 支持 7 段显示屏中构建的小数点
- 支持共阴极和共阳极显示
- 可配置高电平有效与低电平有效段，以及数字驱动

图 1. LED7SEG 框图

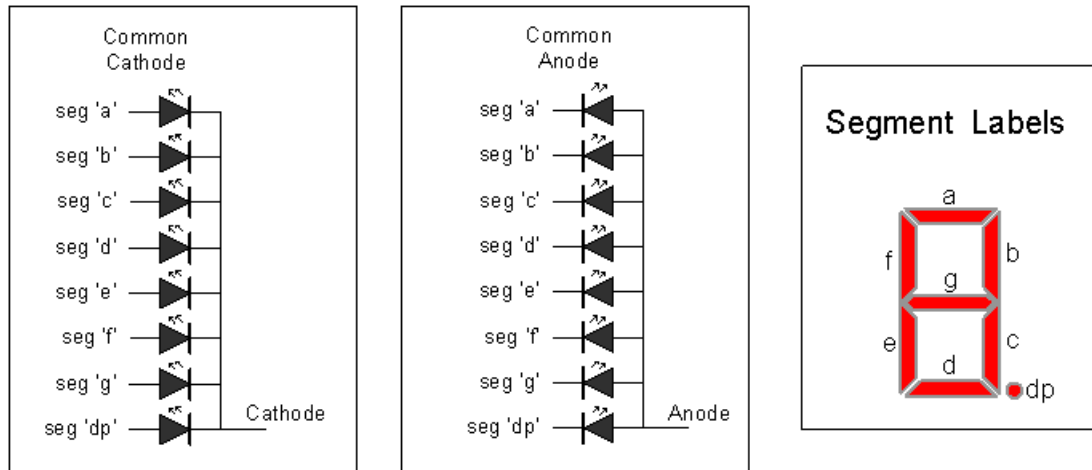


功能说明

LED7SEG 用户模块最多能够复用 8 个 7 段显示屏。该用户模块与共阴极、共阳极或任何驱动极性相兼容。这样对各种显示屏进行复用显得非常灵活。只要 PSoC 引脚的灌电流和拉电流不超过相应的范围，那么 PSoC 引脚就算没有使用晶体管或驱动器，但仍可以直接驱动数字和段。

下图显示的是配置共阳极和共阴极 7 段显示屏的方式：

图 2. 7 段显示屏的配置



如果项目使用了端口上控制引脚的 LED7SEG 用户模块（其他 LED7SEG 用户模块也共同使用该端口），那么必须避免直接对 PRTxDR 进行写入操作。使用映像寄存器来阻止 LED7SEG 执行的错误操作。

复用的工作原理

复用多个显示屏时，尽管我们的眼睛看到全部数字都连续显示，但一次只能打开一个数字。为了实现这种视觉上的假象，显示屏打开和关闭的速度必须高于人眼的反应速度。对于四位数字的显示屏，重复频率应接近 1 kHz；而对于八位数字的显示屏，重复频率应加倍为 2 kHz。对 N 位数字的显示屏进行复用时，每个数字的显示时间为总时间的 1/N。例如，对于具有 8 个数字和刷新率为 2 kHz（500 uS 周期）的系统，在每 4 mSec 的时间内，每个显示屏开启的时间为 500 uSec（8 个数字 * 500 uSec/ 数字）。

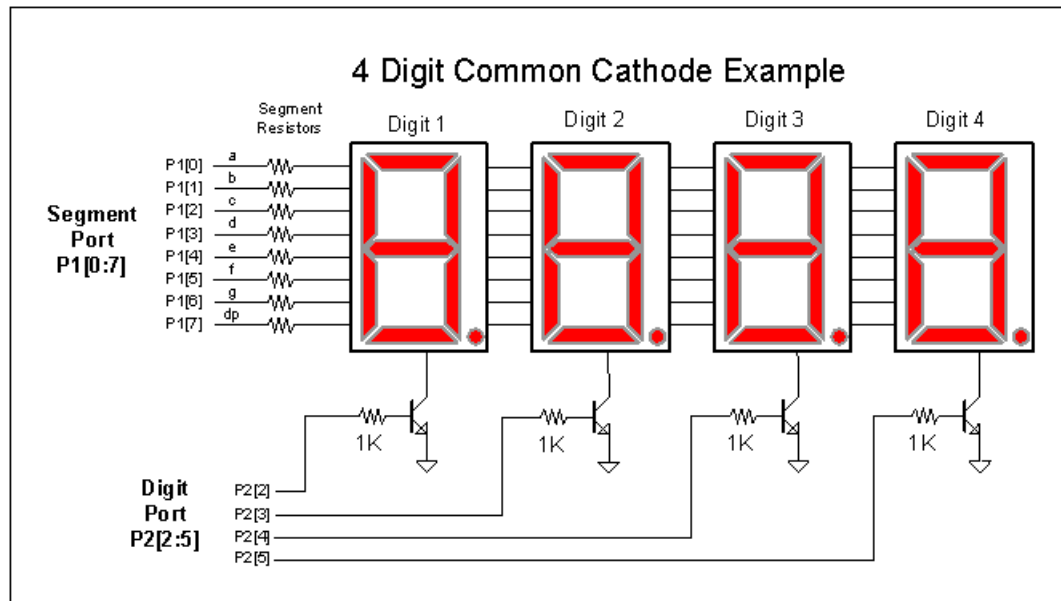
为何要进行复用？

复用一个显示屏的主要原因是为了节省 I/O 引脚。这样进行复用八位数字的显示屏时，仅需要使用 16 个引脚。如果直接驱动同一个显示屏，那么需要使用 64 个 I/O 引脚。另外，复用操作会占用一定量的 CPU 开销。在大部分系统中，该开销非常小，从而不被关注的。而对于该用户模块，如果 CPU 的速度被设为 12 MHz，那么该开销小于 2%（针对重复频率为 1 kHz 的情况）。

下图显示的是一个使用了该用户模块的四位数字显示屏的示例。数字号码总是从左边（MSB）到右边（LSB）开始。必须在段信号上使用电阻，用以限制 LED 电流和每个引脚上的 PSoC 灌电流或拉电流。要在 PSoC 端口引脚和所有数字的相同段信号之间连接段电阻。例如，并行连接所有显示屏的“a”段信号，并行连接所有显示屏的“b”段信号，其他段信号依此类推。驱动显示屏的段信号（段端口）时需要使用全部 8 位端口。在该示例中，使用了端口 1。用于驱动共阳极或共阴极的端口（数字端口）需要引脚数量等于数字数量，但引脚必须是连续的。

在该示例中，使用了端口引脚 P2[2]、P2[3]、P2[4] 以及 P2[5]。使用端口引脚 P2[0]、P2[1]、P2[2]、P[3] 或者其他组合也一样。

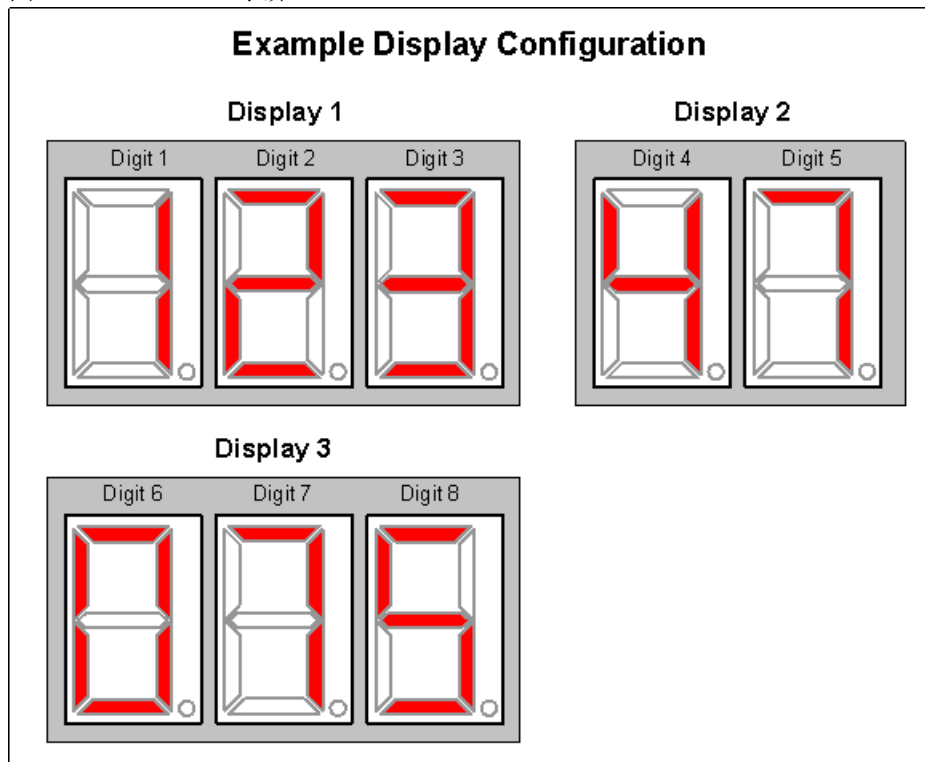
图 3. 使用 LED7SEG 用户模块的 4 数字显示屏的示例



尽管一到八位数字的显示屏可能像一个连续显示屏一样，但它可以被分配为一个八位数字的显示屏或八个 1 位数字的显示屏等待。例如，您可以将 8 个数字分配为两个 3 位数字的显示屏和一个 2 位数字的显示屏或四个 2 位数字的显示屏。只要数字的组是连续的，API 可以处理任意组合。

在下图中，将八个数字放置在三个显示屏组中。显示屏 1 包括数字 1、2 以及 3。显示屏 2 包括数字 4 和 5，另外显示屏 3 包括数字 6、7 和 8。请注意，虽然 8 个数字被分配为三组，但数字号码仍是连续的。

图 4. LED 显示屏组



下面是一个使用了上述定义的配置结合的代码片段的示例：

```
iNum1 = 123;
iNum2 = 47
iNum3 = 75;
LED7SEG_Displnt( iNum1, 1, 3); // Start at Digit 1, three digits long
LED7SEG_Displnt( iNum2, 4, 2); // Start at Digit 4, two digits long
LED7SEG_Displnt( iNum3, 6, 3); // Start at Digit 6, three digits long.
```

直流和交流电气特性

请参见器件数据手册中介绍的内容。

放置

放置 LED7SEG 用户模块前，用户需要选择内置的复用定时器或用户生成的复用定时器。如果选择内置的复用定时器，那么一个数字块将被添加到项目中，用以执行唯一的目的，既复用显示屏。如果存在一个可用的额外数字模块，那么这是最简单的选项。如果选择了该项，那么存在一个额外用户参数 “Multiplex Rate”（重复频率）。

如果额外数字模块不可用，那么用户可以使用任何用户模块的中断服务子程序，该子程序是周期性的，并且频率接近于 1 kHz 或更高。具有周期性的中断服务程序的用户模块包括计数器、定时器、PWM 以及某些 ADC。应将 LED7SEG_Update() 函数调用放在任意用户模块的中断服务子程序中。不同于大部分函数，LED7SEG_Update() 函数保存了 A 和 X 寄存器的值。

参数与资源

段端口

通过该参数可以选择用于驱动显示屏的段信号的 GPIO 端口（请参见图 3）。该用户模块采用了所选定的 GPIO 端口上所有八个引脚。具有八个引脚可用的任意 GPIO 端口被显示为该参数的选项。

数字端口

通过该参数可以选择用于驱动 7 段显示屏的“公用”信号的 GPIO 端口。显示屏中每一个数字都拥有一个公用信号，该信号控制着相应数字被打开还是关闭。要了解详情将公用信号连接到 7 段显示屏硬件的方式，请参见图 3。一旦有一个公用信号处于活动状态，将创建相应数字的 LED 阴极被接地。因此，如果有任意段信号被驱动，则相应的 LED 会发光。然而，如果数字的公用信号当前为非活动状态，那么相应的 LED 数字的阴极没有接地，因此不能使数字的 LED 发光。因此，数字公用信号可用作全部数字的局部 on/off（开关）信号。

输出公用信号的引脚必须作为同一个 GPIO 端口的一部分，以便使用该用户模块的代码是简单而有效的。

该参数仅用于选择被用于公用信号的 GPIO 端口。数字驱动引脚参数会选择哪个 GPIO 端口中的特定引脚将被使用。

数字驱动引脚

该参数用于选择创建 7 段显示屏的数字总数以及驱动与每个数字相对应的“公用”信号的特定引脚。要了解将公用信号连接到 7 段显示屏硬件的方式，请参见图 3。

例如，假设正在使用带有三个数字的 7 段显示屏，并假设公用信号所需的引脚分别为 P2[2]、P2[3] 以及 P2[4]。在这种情况下，数字端口参数必须选择“Port_2”，并且数字驱动引脚参数必须选择“3 位数字 Px[2:4]”。

该参数的选项文本中的字符‘x’表示数字端口参数已选的端口编号。每个参数选项中的“[x:y]”指定了端口中所使用的引脚。第一个数字表示第一个引脚，最后的数字表示最后引脚。因此，“[2:4]”表示 GPIO 端口的 GPIO 引脚 2、3 和 4 被使用。公用信号引脚必须在 GPIO 端口上是连续的。必须将第一个公用引脚（示例中的 P2[2]）连接至第一个数字，将第二个公用引脚（示例中的 P2[3]）连接至第二个数字，并依此类推。

数字驱动器极性

通过该参数可以选择打开单一数字所需的极性。

选项	说明
低电平有效	表示引脚信号被驱动为低，用以打开数字。
高电平有效	表示引脚信号被驱动为高，用以打开数字。

段驱动极性

通过该参数选择打开显示屏上各段所需的极性。

选项	说明
低电平有效	表示引脚信号被驱动为低，用以打开段。
高电平有效	表示引脚信号被驱动为高，用以打开段。

重复频率（如果选择了复用定时器项，重复频率才会有效）

该函数用于选择显示屏的刷新率。下表显示的是该参数的有效选项。内置定时器使用频率为 **32 kHz** 的时钟，这样它可以独立于 **VC1**、**VC2** 以及 **VC3** 时钟。较大的显示屏使用的重复频率更高。较小的显示屏（具有 2 至 4 个数字）使用较低的重复频率。如果显示屏发生闪烁现象，请增大重复频率。

选项	说明
500 Hz	重复频率为 500 Hz（对于 2 ~ 3 个数字）
1 kHz	重复频率为 1 kHz（对于 3 ~ 5 个数字）
2 kHz	重复频率为 2 kHz（对于 4 ~ 6 个数字）
4 kHz	重复频率为 4 kHz（对于 6 ~ 8 个数字）

应用编程接口（API）

所提供的应用编程接口（API）子程序作为用户模块的一部分，设计人员通过它可以采用更高级别的方式进行处理模块。

虽然某些用户模块 API 函数可以保持 A 和 X 不变，但无法保证它们将来也会如此。

对于大型储存器模型器件，调用程序需要保留 CUR_PP、IDX_PP、MVR_PP 以及 MVW_PP 等寄存器中的所有值。尽管目前尚未修改某些寄存器，但无法保证在将来的版本中也会如此。

每次放置用户模块时，都要为它分配一个实例名称。默认情况下，PSoC Designer 向给定项目中该用户模块的第一个实例分配了 LED7SEG_1。可将该值更改为任意一个符合标识符语法规则的值。分配的实例名称成为每个全局函数名称、变量和常量符号的前缀。为简便起见，在以下说明中将实例名称缩写为 LED7SEG。

在用户模块头文件中定义了以下常量：

常量	数值
LED7SEG_Digit1	0x01
LED7SEG_Digit2	0x02
LED7SEG_Digit3	0x04
LED7SEG_Digit4	0x08
LED7SEG_Digit5	0x10
LED7SEG_Digit6	0x20
LED7SEG_Digit7	0x40
LED7SEG_Digit8	0x80
LED7SEG_DimOn	0x01
LED7SEG_DimOff	0x00
LED7SEG_DpOn	0x01
LED7SEG_DpOff	0x00

LED7SEG_Start

说明：

清除所有数字存储器中的内容，并使能扫描。如果选择复用定时器，则该参数会使能定时器及其中断。进行复用时需要使能全局中断。

C 语言原型：

```
void LED7SEG_Start(void)
```

汇编程序：

```
lcall LED7SEG_Start
```

参数:

无

返回值:

无

其他影响:

无

LED7SEG_Stop

说明:

停止显示屏扫描，并关闭所有数字。如果选择复用定时器，则该函数会禁用定时器及其中断。

C 语言原型:

```
void LED7SEG_Stop(void)
```

汇编程序:

```
lcall LED7SEG_Stop
```

参数:

无。

返回值:

无

其他影响:

更新后的 ISR 不再扫描显示屏。

LED7SEG_PutHex

说明:

将一个十六进制数字（0 至 F）放在给定的数字中。

C 语言原型:

```
void LED7SEG_PutHex(BYTE bValue, BYTE bDigit)
```

汇编程序:

```
mov a,0x5      ; Write a '5' to the display  
mov x,0x01     ; Write the '5' at digit 1
```

```
lcall LED7SEG_PutHex
```

参数:

bValue: 显示 0 至 F 之间的十六进制数字。

bDigit: 表示用于放置 bValue 的数字位置。

返回值:

无

其他影响：

该函数会更改 A 和 X 寄存器。

LED7SEG_PutPattern**说明：**

将字节格式直接写入到显示屏存储器中。无论显示屏是共阴极还是共阳极，一个高位都会对应一个有效段。LSB 是 ‘a’ 段，MSB 是 “dp”（即小数点）。

C 语言原型：

```
void LED7SEG_PutPattern(BYTE bPattern, BYTE bDigit)
```

汇编程序：

```
mov  a,0xFF    ; Turn all segments on
mov  x,0x01
```

```
lcall LED7SEG_PutPattern
```

参数：

bPattern: 用于显示的位模式。例如，0x80 仅会激活小数点，并且 0x7F 会激活所有段，而非激活小数点。

bDigit: 表示用于放置 bValue 的数字位置。

返回值：

无

LED7SEG_DP**说明：**

在给定的位置，打开或关闭小数点。

C 语言原型：

```
void LED7SEG_DP(BYTE bOnOff, BYTE bDigit)
```

汇编程序：

```
mov  a,0x1
mov  x,0x01
lcall LED7SEG_DP
```

参数：

bOnOff: 0 表示关闭小数点，1 表示打开小数点。

bDigit: 表示小数点的数字位置。

返回值：

无

其他影响：

该函数会更改 A 和 X 寄存器。

LED7SEG_Dim

说明:

调暗或不调暗显示屏。使能调暗时，更新操作会跳过所有其他刷新周期。

C 语言原型:

```
void LED7SEG_Dim(BYTE bDim)
```

汇编程序:

```
mov    x, 0x01          ; Dim the display
lcall  LED7SEG_Dim
```

参数:

bDim: 0 表示不调暗显示屏，1 表示调暗显示屏

返回值:

无

其他影响:

该函数会更改 A 和 X 寄存器。

LED7SEG_Displnt

说明:

显示显示屏上的一个整数值。它的长度介于 1 至 5 个数字，并在任意位置开始。数值的范围为 0 ~ 65535。

C 语言原型:

```
void LED7SEG_Displnt(int iValue, BYTE bPos, BYTE bLSD)
```

汇编程序:

```
mov    X, SP
add    SP, 4
mov    [X], 0x34          ; Load MSB of value to display
mov    [X+1], 0x12        ; Load LSB of value to display
mov    [X+2], 0           ; Set starting location for value in display
mov    [X+3], 4           ; Set number of digits to display
lcall  LED7SEG_Displnt
```

参数:

iValue: 显示整数值。

bPos: 表示显示屏上开始显示数字的位置。

bLSD: 数字的数量，例如：仅显示 4 个数字。

返回值:

无

其他影响:

该函数会更改 A 和 X 寄存器。

LED7SEG_Update

说明:

Update（更新）函数控制显示屏的重复频率。每次调用该函数，都会关闭先前被显示的数字，然后打开下一个数字。如果在“**User Module Selection Options...**”（用户模块选择的选项）菜单中选择了“**LED7SEG with MPX Timer**”（使用 MPX 定时器的 LED7SEG），那么会定期自动调用该函数。如果选择了“不使用 MPX 定时器的 LED7SEG”项，用户需要每 2 到 0.5 mSec 定期调用该函数，用以实现在复用时不发生闪烁。可以在用户主程序循环中或从一个现有的中断服务子程序中调用该函数。

C 语言原型:

```
void LED7SEG_Update(void)
```

汇编程序:

```
lcall LED7SEG_Update
```

参数:

无

返回值:

无

其他影响:

保留所有寄存器的值，因此调用该函数前不用保存 A 和 X 寄存器值。

示例固件源代码

示例汇编代码

```
-----  
; // Example ASM code using LED7SEG User Module  
-----  
  
include "m8c.inc"      ; part specific constants and macros  
include "memory.inc"   ; Constants & macros for SMM/LMM and Compiler  
include "PSOCAPI.inc"  ; PSoC API definitions for all User Modules  
  
export _main  
  
_main:  
    lcall LED7SEG_Start  
  
;; Enable multiplex timer here if using  
;; non-built-in-timer version.  
  
M8C_EnableGInt      ; Enable global interrupts  
    push X  
    mov A,4          ; Push display size on stack "4"  
    push A  
    mov A,1          ; Push display start on stack "1"  
    push A  
    mov A,>1234       ; Push MSB of value 1234 on stack  
    push A  
    mov A,<1234       ; Push LSB of value 1234 on stack
```

```
push  A
lcall LED7SEG_DisInt  ; Display "1234"
mov   X,3             ; Turn on decimal point at digit 3
mov   A,1             ; Enable DP
lcall LED7SEG_DP
```

```
.terminate:
    jmp .terminate
```

示例 C 语言代码

```
//-----
// Example C code using LED7SEG User Module
//-----

#include <m8c.h>
#include "PSoCAPI.h"

void main(void)
{
    LED7SEG_Start();           // Enable display
    M8C_EnableGInt;           // Enable IRQ

    // Enable multiplex timer here if using
    // non-built-in-timer version.

    M8C_EnableGInt;
    LED7SEG_DisInt(1234, 1, 4); // Display "1234"
    LED7SEG_DP(1, 3);           // Turn on DP ( 123.4 )
}
```

配置寄存器

无

版本历史记录

版本	修订人	说明
1.1	DHA	添加了版本历史记录。
1.20	DHA	添加了对 CY8C21x12 器件的支持。

注意： PSoC Designer 5.1 介绍了所有用户模块数据手册中的 “版本历史记录”。本数据手册详细介绍了当前和先前的用户模块版本之间存在的区别。

文档编号: 001-84323 Rev. *A

修订日期 November 20, 2014

页 13/13

Copyright © 2005-2014 赛普拉斯半导体公司。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。也不会根据专利权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，对于合理预计会发生运行异常和故障并对用户造成严重伤害的生命支持系统，赛普拉斯将不批准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 Designer™ 和 Programmable System-on-Chip™ 是赛普拉斯半导体公司的商标，PSoC® 是赛普拉斯半导体公司的注册商标。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。除上述指定用途外，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：赛普拉斯不针对该材料提供任何类型的明示或暗示保证，包括（但不限于）针对特定用途的适销性和适用性的暗示保证。赛普拉斯保留在不另行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，则表示制造商将承担因此类使用而导致的所有风险，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。