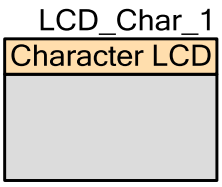
字符 LCD

1.90



特性

- 实现行业标准的 Hitachi HD44780 LCD 显示驱动器芯片协议
- 一个 I/O 端口上仅需要 7 个 I/O 插针
- 包含内置字符编辑器以创建用户定义的自定义字符
- 支持水平和垂直条形图

概述

字符 LCD 组件包含一组库例程，通过这些库例程易于使用遵循 Hitachi 44780 标准 4 位接口的一行、两行或四行 LCD 模块。该组件提供 API 用于实现水平和垂直条形图，您也可以创建和显示自己的自定义字符。

何时使用字符 LCD

使用字符 LCD 组件可向产品用户，或在设计和调试过程中的开发人员显示文本数据。

输入/输出连接

本节介绍可用于字符 LCD 的各种输入和输出连接。

LCD_Port – 引脚编辑器

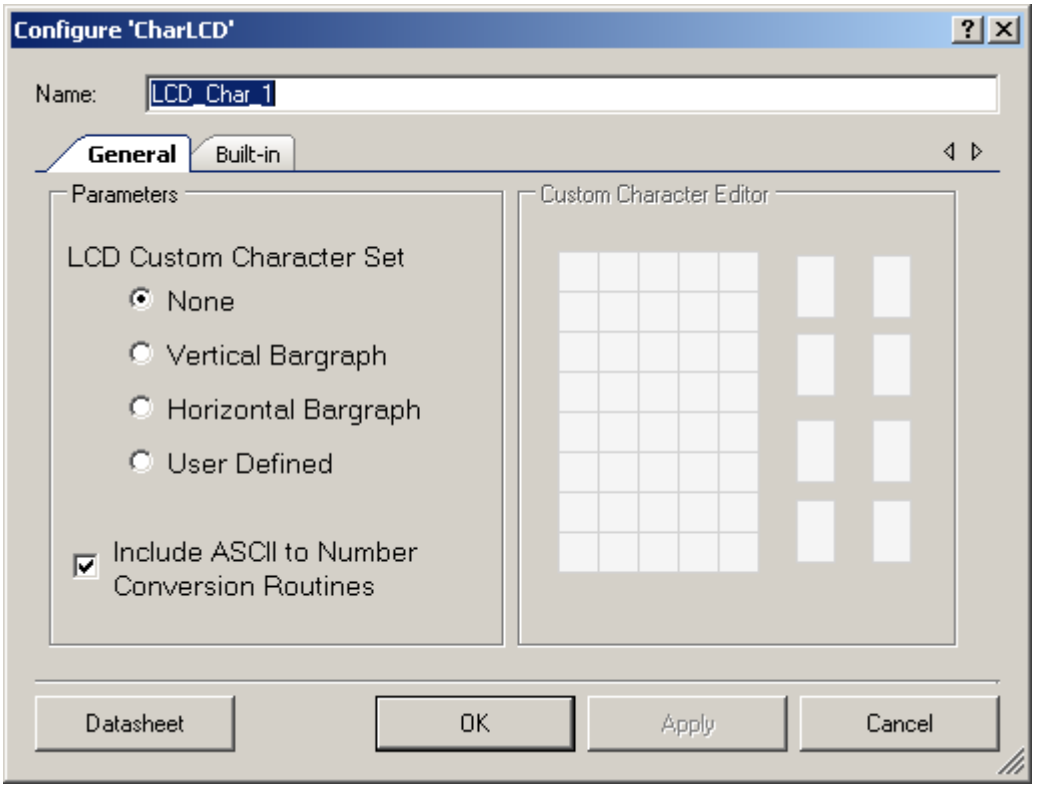
LCD 使用物理端口的七个连续引脚。若要将字符 LCD 放置到所需端口上，请使用设计范围资源引脚编辑器 （Design-Wide Resources Pin Editor）。通过引脚编辑器可以将此组件的数字端口放置在任何空闲输出端口上。

注：七个引脚的起始位置可以为所选端口的引脚 1 或引脚 0，但不能跨多个端口。这些引脚专用于 LCD 端口，不能用于任何其他用途。

无需直接访问字符 LCD 的端口，因为软件 API 可为您管理所有读取和写入。功能描述中详细介绍了 LCD 模块与 PSoC 逻辑端口之间的引脚连接。

组件参数

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



参数

LCD 自定义字符集

通过此参数可以选择以下选项：

- **None**（无）（默认） - 不对自定义字符执行任何操作。
- **Vertical Bar Graph**（垂直条形图） - 生成自定义字符和 API 以操控垂直条形图。
- **Horizontal Bar Graph**（水平条形图） - 生成自定义字符和 API 以操控水平条形图。
- **User Defined**（用户定义的） - 创建自定义字符和 API 以操控它们。

在组件载入了字符之后，可以使用 LCD_Char_PutChar() 函数和自定义字符常量（来自头文件）显示它们。

转换例程

选择 **Include ASCII to Number Conversion Routines**（在数字转换例程中包括 ASCII）选项可将几个 API 函数添加到生成的代码中。（有关这些例程的详细信息，请参阅 API 表或函数说明。）

自定义字符编辑器

通过**自定义字符编辑器**，可以使用 GUI 轻松地创建用户定义的字符集。8 个字符中的每个字符最多可以为 5x8 个像素，不过某些硬件可能最多只能显示 5x7。

若要使用**自定义字符编辑器**，请选择 **User Defined**（用户定义的）作为 **LCD Custom Character Set**（LCD 自定义字符集）的选项。然后，单击要编辑字符的缩略图。

若要切换字符中的某个像素，请在放大的字符视图中单击所选像素。也可以单击并拖动以切换多个像素。

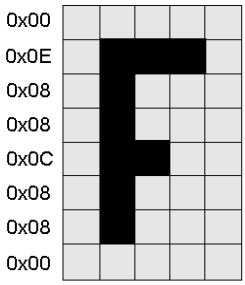
在创建自定义字符集之后，GUI 会生成八个自定义字符的查找数组。然后可以将该查找数组加载到 LCD 模块中。默认情况下，如果选择或创建了任何自定义字符，则 LCD_Char_Start() 例程会加载这些字符。

组件的功能使您可以在代码中创建自定义字符集并在运行时加载它们。在这种情况下，最后加载的字符集会覆盖上一个并呈活动状态。要恢复原始自定义字符集，请将 LCD_Char_customFonts[] 作为 LCD_Char_LoadCustomFonts() 的参数使用。无需添加 LCD_Char_customFonts[] 作为外部参考，因为其已经包含在 LCD_Char.h 中。

运行时，LCD_Char_LoadCustomFonts() 可以使用该代码作为参数来将原始字符集加载到 LCD 模块中。

图 1 显示了一个编码为 8 字节自定义字符查找数组行的自定义字符。

图 1. 自定义字符编码




```
        Custom character «F»:
{0x00, 0x0E, 0x08, 0x08, 0x0C, 0x08, 0x08, 0x00}
```

如图所示，字符的每行都编码为单个字节，该字节中仅使用五个最低有效位。第一个字符的顶部行采用自定义字体数组的第一个字节进行编码。第一个字符的下一行是数组中的第二个字节。第二个字符的第一行是数组中的第九个字节，依此类推。整个自定义字节数组包括八个自定义字符，从而形成总共 64 个字节的数组大小。

应用程序编程接口(API)

应用程序编程接口 (API) 库例程允许您使用软件配置组件。下表列出并描述了每个函数的接口以及“include”文件提供的相关常量。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_Char_1”分配给指定项目中组件的第一个实例。您可以将其重命名为遵循标识符语法规则的任何唯一值。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_Char”。

函数	说明
LCD_Char_Start()	启动模块并将自定义字符集加载到 LCD（如果定义了自定义字符集）。
LCD_Char_Stop()	关闭 LCD
LCD_Char_DisplayOn()	打开 LCD 模块的显示
LCD_Char_DisplayOff()	关闭 LCD 模块的显示
LCD_Char_PrintString()	将以空字符结尾的字符串逐个字符地打印到屏幕
LCD_Char_PutChar()	将单个字符发送到当前位置的 LCD 模块数据寄存器。
LCD_Char_Position()	设置与提供的行和列匹配的光标位置
LCD_Char_WriteData()	将单个字节的数据写入 LCD 模块数据寄存器
LCD_Char_WriteControl()	将单字节指令写入 LCD 模块控制寄存器
LCD_Char_ClearDisplay()	从 LCD 模块的屏幕清除数据
LCD_Char_IsReady()	轮询 LCD，直到准备（Ready）绪位
LCD_Char_Sleep()	准备让组件进入睡眠模式
LCD_Char_Wakeup()	恢复组件配置并打开 LCD
LCD_Char_Init()	执行组件正常工作所需的初始化
LCD_Char_Enable()	打开显示
LCD_Char_SaveConfig()	在进入睡眠模式之前提供的空的 API 存储任何所需数据。
LCD_Char_RestoreConfig()	在退出睡眠模式之后提供的空的 API 以恢复保存的数据。

如果选择了用户选择的自定义字体，则在需要时包括以下可选函数。LCD_Char_LoadCustomFonts() 函数附带每个自定义字符集（无论是用户定义还是 PSoC Creator 生成的）。LCD_Char_LoadCustomFonts() 函数可以用于将用户定义的字符或条形图字符加载到 LCD 硬件中。如果加载由工具创建的自定义字体，需要先将指向自定义字体的指针导入项目，然后再使用此函数（请参阅 LCD_Char_LoadCustomFonts() 的说明）。默认情况下，LCD_Char_Init() 例程加载用户选择的自定义字体。在选择了条形图并启用了条形图的方便动态调整时，会生成绘制条形图命令。

可选自定义字体函数	说明
LCD_Char_LoadCustomFonts()	将自定义字符加载到 LCD 模块中
LCD_Char_DrawHorizontalBG()	绘制水平条形图。仅当选择了条形图字符集时可用。
LCD_Char_DrawVerticalBG()	绘制垂直条形图。仅当选择了条形图字符集时可用。

根据您的选择需要，会包括以下可选函数：

可选的数字到 ASCII 转换例程	说明
LCD_Char_PrintInt8()	将 8 位值的双 ASCII 字符十六进制表示形式打印到字符 LCD 模块。
LCD_Char_PrintInt16()	将 16 位值的四 ASCII 字符十六进制表示形式打印到字符 LCD 模块。
LCD_Char_PrintNumber()	以左对齐 ASCII 字符的形式打印 16 位值的十进制值

void LCD_Char_Start(void)

说明： 此函数初始化 LCD 硬件模块，如下所示：

- 启用 4 位接口
- 清除显示
- 启用自动光标增加
- 将光标复位到起始位置

它还将自定义字符集加载到 LCD（如果在自定义程序的 GUI 中进行了定义）。

参数： None

返回值： None

副作用： None

void LCD_Char_Stop(void)

说明： 关闭 LCD 屏幕的显示。

参数： None

返回值： None

副作用： None

void LCD_Char_PrintString(char8 const string[])

说明： 从当前光标位置开始，将以空字符结尾的字符串写入屏幕。

参数： char8 const string[]: 要在 LCD 模块屏幕上显示的以空字符结尾的 ASCII 字符数组。

返回值： None

副作用： None

void LCD_Char_PutChar(char8 character)

说明： 将单个字符写入屏幕上的当前光标位置。用于通过字符的命名值显示自定义字符。（LCD_Char_CUSTOM_0 到 LCD_Char_CUSTOM_7）。

参数： char8 字符：将要在 LCD 模块屏幕上显示的 ASCII 字符。

返回值： None

副作用： None

void LCD_Char_Position(uint8 row, uint8 column)

说明： 将光标移动到参数 **row**（行）和 **column**（列）指定的位置。

参数： uint8 行：放置光标的行号。最小值为零。

uint8 列：放置光标的列号。最小值为零。

返回值： None

副作用： None

void LCD_Char_WriteData(uint8 dByte)

说明： 将数据写入 LCD RAM 的当前位置。在写入完成时，位置会根据指定的输入模式递增或递减。

参数： dByte：要写入 LCD 模块的字节值。

返回值： None

副作用： None

void LCD_Char_WriteControl(uint8 cByte)

说明： 将命令字节写入 LCD 模块。不同 LCD 模式可能具有自己的命令。查看特定 LCD 数据手册以了解对该模式有效的命令。

参数： cByte：8 位值，表示要加载到 LCD 模块命令寄存器中的命令。下表中指定了有效命令参数：

值	说明
LCD_Char_CLEAR_DISPLAY	清除显示
LCD_Char_RESET_CURSOR_POSITION LCD_Char_CURSOR_HOME	将光标和 LCD 返回到主位置
LCD_Char_CURSOR_LEFT	设置左光标移动方向
LCD_Char_CURSOR_RIGHT	设置右光标移动方向
LCD_Char_DISPLAY_CURSOR_ON	启用显示和光标
LCD_Char_DISPLAY_ON_CURSOR_OFF	启用显示，光标关闭
LCD_Char_CURSOR_WINK	启用显示，光标关闭，设置光标闪烁
LCD_Char_CURSOR_BLINK	启用显示和光标，设置光标闪烁
LCD_Char_CURSOR_SH_LEFT	左移光标/移位显示
LCD_Char_CURSOR_SH_RIGHT	右移光标/移位显示
LCD_Char_DISPLAY_2_LINES_5x10	将显示设置为 2 行 10 个字符

返回值： None

副作用： None

void LCD_Char_ClearDisplay(void)

说明： 清除屏幕内容并将光标位置复位为零行和零列。它调用带有合适参数的函数 LCD_Char_WriteControl() 以激活显示。

参数： None

返回值： None

副作用： 光标位置复位为 行0 （row 0) 列0（column0)。

void LCD_Char_IsReady(void)

说明： 轮询 LCD，直到准备（Ready）绪位

参数： None

返回值： None

副作用： 将引脚更改为 HI-Z。

void LCD_Char_DisplayOff(void)

说明： 关闭显示，但是不以任何方式复位 LCD 模块。它调用带有合适参数的函数 LCD_Char_WriteControl() 以停用显示。

参数： None

返回值： None

副作用： None

void LCD_Char_DisplayOn(void)

说明： 打开显示，而*不*初始化它。它调用带有合适参数的函数 LCD_Char_WriteControl() 以激活显示。

参数： None

返回值： None

副作用： None

void LCD_Char_Sleep(void)

说明： 这是准备组件睡眠的首选例程。LCD_Char_Sleep() 例程保存当前组件的状态。然后调用 LCD_Char_Stop() 函数并调用 LCD_Char_SaveConfig() 以保存硬件配置。

在调用 CyPmSleep() 或 CyPmHibernate() 函数之前调用 LCD_Char_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

返回值： None

副作用： 不更改组件引脚的驱动模式。将端口组件 API 用于该用途。由于字符LCD是具有特定协议的接口组件，所以您需要在保存或恢复了组件引脚状态之后重新初始化该组件。

void LCD_Char_Wakeup(void)

说明： 恢复组件配置并打开 LCD。

参数： None

返回值： None

副作用： None

void LCD_Char_Init(void)

说明： 执行组件正常工作所需的初始化。LCD_Char_Init() 还加载自定义字符集（如果在 Configure（配置）对话框中进行了定义）。

参数： None

返回值： None

副作用： None

void LCD_Char_Enable(void)

说明： 打开显示。

参数： None

返回值： None

副作用： None

void LCD_Char_SaveConfig(void)

说明： 清空的API 以在进入睡眠模式之前存储任何所需数据。

参数： None

返回值： None

副作用： None

void LCD_Char_RestoreConfig(void)

说明： 清空的 API 以在退出睡眠模式之后恢复保存的数据。

参数： None

返回值： None

副作用： None

void LCD_Char_LoadCustomFonts(uint8 const customData[])

说明： 将八个自定义字符（条形图或用户定义的字体）加载到 LCD 模块中以在运行时使用自定义字体。仅当在自定义程序中选择了自定义字符集时才可用。

参数： uint8 const customData[]: 指向字节数组头的指针。数组长度应为 64 个字节，因为 5x8 字符需要每个字符 8 字节。

返回值： None

副作用： 会覆盖可能已存储在 LCD 模块中的任何之前自定义的字符。

void LCD_Char_DrawHorizontalBG(uint8 row, uint8 column, uint8 maxCharacters, uint8 value)

说明： 绘制水平条形图。仅当选择了水平或垂直条形图时才可用。

参数： uint8 行：条形图中第一个字符的行。

uint8 列：条形图中第一个字符的列。

uint8 maxCharacters：条形图使用的所有字符数量。根据条形图选择表示高度或宽度。每个字符为 5 像素宽和 8 像素高。

uint8 值：要绘制的阴影像素数。不能超过条形图的总像素长度（高度）。

返回值： None

副作用： None

void LCD_Char_DrawVerticalBG(uint8 row, uint8 column, uint8 maxCharacters, uint8 value)

说明： 绘制垂直条形图。仅当选择了水平或垂直条形图时才可用。

参数： uint8 行：条形图中第一个字符的行。

uint8 列：条形图中第一个字符的列。

uint8 maxCharacters：条形图使用的所有字符数量。根据条形图选择表示高度或宽度。每个字符为 5 像素宽和 8 像素高。

uint8 值：要绘制的阴影像素数。不能超过条形图的总像素长度（高度）。

返回值： None

副作用： None

void LCD_Char_PrintInt8(uint8 value)

说明： 将 8 位值的双 ASCII 字符表示形式打印到字符 LCD 模块。

参数： uint8 值：要以十六进制 ASCII 字符打印的 8 位值。

返回值： None

副作用： None

void LCD_Char_PrintInt16(uint16 value)

说明： 将 16 位值的四 ASCII 字符表示形式打印到字符 LCD 模块。

参数： uint16 值：要以十六进制 ASCII 字符打印的 16 位值。

返回值： None

副作用： None

void LCD_Char_PrintNumber(uint16 value)

说明： 以左对齐 ASCII 字符的形式打印 16 位值的十进制值。

参数： uint16 值：要以十进制数字的 ASCII 字符打印的 16 位值。

返回值： None

副作用： None

MISRA 兼容性

本节介绍了本组件与 MISRA-C:2004 的兼容和偏差情况。定义了两种类型的偏差：项目偏差 - 适用于所有 PSoC Creator 组件的偏差；特定偏差 - 仅适用于此组件的偏差。本节提供了有关组件特定偏差的信息。*系统参考指南*的 MISRA 兼容性章节中介绍项目偏差以及有关 MISRA 兼容性验证环境的信息。

字符 LCD 组件具有以下特定偏差：

MISRA-C:2004 Rule	规则类（要求/咨询）	规则说明	偏差说明
19.7	R	函数的使用应优先于类似于函数的宏。	以下的宏标记为过期： LCD_PrintDecUint16(x)、 LCD_PrintHexUint8(x)、 LCD_PrintHexUint16(x) 它们是以下函数的重新定义功能： LCD_PrintNumber()、LCD_PrintInt8()、 LCD_PrintInt16() 它们将从新版本的组件中移除。

此组件具有以下嵌入式组件 - 引脚。请参阅相应的组件数据手册，以获得有关其 MISRA 合规性和特定偏差的信息。

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**（筛选选项）可缩小可选项目的列表。

有关更多信息，请参考 PSoC Creator 帮助中的“查找示例项目”主题。

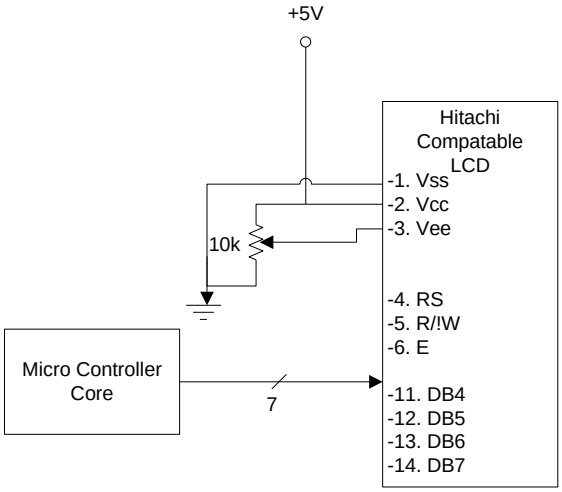
功能描述

LCD 模块为字母数字字符以及有限自定义字体提供可视显示。API 根据需要配置 PSoC 组件，以便轻松地在标准 Hitachi LCD 显示驱动器与 PSoC 设备之间实现连接。

下表介绍 LCD 逻辑端口引脚到物理 LCD 模块引脚的映射。LCD 的逻辑端口可以映射为在端口的第一个或第二个物理引脚上开始；不能跨端口。即，LogicalPort_0 理论上可以是端口 2 引脚 0 或端口 2 引脚 1。使用引脚编辑器强制 LCD 逻辑端口在引脚 0 上开始可减少为写入对齐数据所需的位移数，从而提高效率。

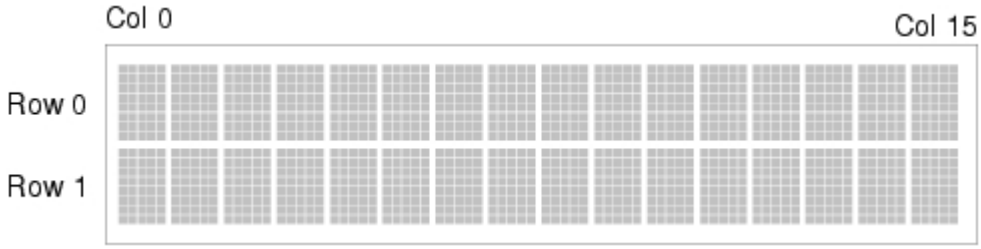
逻辑端口引脚	LCD 模块引脚	说明
LCDPort_0	DB4	数据位 0
LCDPort_1	DB5	数据位 1
LCDPort_2	DB6	数据位 2
LCDPort_3	DB7	数据位 3
LCDPort_4	E	LCD 启用（用于确认新数据可用的探针）
LCDPort_5	RS	寄存器选择（选择数据或控制输入数据）
LCDPort_6	R/!W	读取/不写入（切换以便轮询 LCD 的就绪位）

图 2. 引脚编辑器图



LCD_Char_Position() 函数管理显示寻址按如下所示。行零、列零在左上角，列号向右递增。在四行显示中，写入超过行 0 的列 19 可能会导致行 2 损坏，因为寻址将行 0、列 20 映射到行 2、列 0。在标准 2x16 Hitachi 模块中没有此问题。

图 3. 2x16 Hitachi LCD 模块



资源

字符 LCD 组件使用 7 个 I/O 引脚。

API 存储器使用

根据编译器、组件、所用 API 数量和组件配置的不同，组件内存使用会出现较大变化。下表提供指定组件配置中可用的 API 的存储器使用。

已利用释放模式中配置的相关编译器进行了测量，大小采用了优化设定。对于特定设计，可以分析编译器生成的映射文件以确定存储器使用。

配置	PSoC 3 (Keil_PK51)		PSoC 4 (GCC)		PSoC 5LP (GCC)	
	闪存 字节	SRAM 字节	闪存 字节	SRAM 字节	闪存 字节	SRAM 字节
None	652	3	610	6	642	6
垂直	1049	3	874	6	894	6
水平	979	3	830	6	852	6
用户定义的	790	3	724	6	760	6
无 + 转换例程	918	3	772	6	826	6

直流和交流电气特性

无

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
1.90	添加了 PSoC4 支持。	修正了组件源代码中多种类型的转换问题。
	更新了数据手册和 PSoC 4 内存使用情况	
1.80	添加了 MISRA 兼容性章节。	此组件具有特定的描述偏差。
	函数 LCD_PrintDecUint16(x)、LCD_PrintHexUint8(x) 和 LCD_PrintHexUint16(x) 已过时。	
	更新了 API 存储器使用表。	
	MISRA 相关内容发生了更改。 以下 API 函数的声明发生了更改： 由 LCD_Char_LoadCustomFonts(uint8* customData) 更改为 LCD_Char_LoadCustomFonts(uint8 const customData[]) ； 由 LCD_Char_PrintString(const char8* string) 更改为 LCD_Char_PrintString(char8 const string[])。	将变量声明为指针时，MISRA 不允许将变量作为阵列使用。
	在“自定义字符编辑器”部分，删除了有关添加和 LCD_Char_customFonts[] 外部参考的注释。	LCD_Char.h 文件现在包括 LCD_Char_customFonts[]。
1.70	写入 LCD 显示屏时信号转换的延迟增加了。	在配有编译器设置的 CY8CKIT-050 开发板上，至 LCD 的信号不进行完全转换。
1.60	向数据手册中添加了特性部分	
	向.cyre file.APIs 文件中包括的所有组件 API 添加了 CYREENTRANT 关键词。	并非所有 API 都是真正可重入的。组件 API 源文件中的注释指出了适用的函数。 对于以安全方式使用的不是可重入的函数，需要此更改来消除编译器警告：通过标志或关键节防止并发调用。添加了此功能，使客户能够将单个生成的函数指定为可重入。
	LCD_Char_DrawHorizontalBG() 和 LCD_Char_DrawVerticalBG() 条形图已添加到条件编译中。	要防止使用具有不当自定义字符集的条形图 API 函数。
1.50.c	向数据手册添加了指令以恢复默认客户字体集	
1.50.b	删除了 void LCD_Char_WriteControl(uint8 command) API。	void LCD_Char_WriteControl(uint8 command) API 在 API 节中介绍了两次，因此删除了额外说明。

版本	更改说明	更改/影响原因
	从 LCD_Char_DrawHorizontalBG() 和 LCD_Char_DrawVerticalBG() API 的说明中删除了有关调用 LCD_Char_LoadCustomFonts() 的必要性的说明。	现在在使用 LCD_Char_DrawHorizontalBG() 或 LCD_Char_DrawVerticalBG() 之前无需调用 LCD_Char_LoadCustomFonts()，因为它在组件的 LCD_Char_Start() 例程中进行。
1.50.a	在数据手册中更新了 LCD_Char_WriteData()、LCD_Char_WriteControl() 和 LCD_Char_Sleep() API 的说明	不存在对函数 LCD_Char_WriteData() 中参数的说明，因此添加了合适说明。LCD_Char_Sleep() 的说明非常糟糕，因此使用更多详细信息对其进行了更新。使用可能输入参数值的详细信息，对 LCD_Char_WriteControl() 的说明进行了扩展。
	对数据手册进行了少量编辑和更新	
1.50	添加了 LCD_Char_Sleep()、LCD_Char_Wakeup()、LCD_Char_Enable()、LCD_Char_Init()、LCD_Char_SaveConfig()、LCD_Char_RestoreConfig() API。	用于支持低功耗模式并为大多数组件提供常用接口。
	添加了新 API 文件 – *CharLCD_PM.c*，该文件包含睡眠模式 API 的声明。	用于支持低功耗模式。
	在 LCD_Char_Init() API 中添加了 LCD_Char_LoadCustomFonts() 调用。	在项目中首次调用 LCD_Char_Start() 时，会自动加载 Configure（配置）对话框中的所选自定义字体。
1.40	向 LCD_Char_Start() 函数的初始化序列添加了额外延迟，以便满足显示的时序要求。	防止高频运行系统时初始化出现故障。
1.30	在字符 LCD 原理图中使用引脚组件替换了数字端口组件。	旧数字端口已过期，替换为新引脚组件。
	添加了 IsReady 函数的说明。	旧版本的数据手册中没有提到 LCD_Char_IsReady()。
	删除了 LCD_Char_DelayUS()，将其替换为 LCD_Char_CyDelay() 或 LCD_Char_IsReady()。	这是为了解决与组件有关的时序问题，该问题导致了一些故障。
1.20.a	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组件的修订版。
1.20	更新的符号	为符合公司标准。
1.10.a	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组件的修订版。
1.10	自版本 0.2 以来进行的各种更新	0.2 版本随 alpha 构建附带，但不是完整功能的组件。

赛普拉斯半导体公司，2013-2016 年。本文件是赛普拉斯半导体公司及其子公司，包括 Spansion LLC（“赛普拉斯”）的财产。本文件，包括其包含或引用的任何软件或固件（“软件”），根据全球范围内的知识产权法律以及美国与其他国家签署条约由赛普拉斯所有。除非在本款中另有明确规定，赛普拉斯保留在该等法律和条约下的所有权利，且未就其专利、版权、商标或其他知识产权授予任何许可。如果软件并不附随有一份许可协议且贵方未以其他方式与赛普拉斯签署关于使用软件的书面协议，赛普拉斯特此授予贵方属人性质的、非独家且不可转让的如下许可（无再许可权）：（1）在赛普拉斯特软件著作权项下的下列许可权（一）对以源代码形式提供的软件，仅出于在赛普拉斯硬件产品上使用之目的且仅在贵方集团内部修改和复制软件，和（二）仅限于在有关赛普拉斯硬件产品上使用之目的将软件以二进制代码形式的向外部最终用户提供（无论直接提供或通过经销商和分销商间接提供），和（2）在被软件（由赛普拉斯公司提供，且未经修改）侵犯的赛普拉斯专利的权利主张项下，仅出于在赛普拉斯硬件产品上使用之目的制造、使用、提供和进口软件的许可。禁止对软件的任何其他使用、复制、修改、翻译或汇编。

在适用法律允许的限度内，赛普拉斯未对本文件或任何软件作出任何明示或暗示的担保，包括但不限于关于适销性和特定用途的默示保证。赛普拉斯保留更改本文件的权利，届时将不另行通知。在适用法律允许的限度内，赛普拉斯不对因应用或使用本文件所述任何产品或电路引起的任何后果负责。本文件，包括任何样本设计信息或程序代码信息，仅为供参考之目的提供。文件使用人应负责正确设计、计划和测试信息应用和由此生产的任何产品的功能和安全性。赛普拉斯产品不应被设计为、设定为或授权用作武器操作、武器系统、核设施、生命支持设备或系统、其他医疗设备或系统（包括急救设备和手术植入物）、污染控制或有害物质管理系统中的关键部件，或产品植入之设备或系统故障可能导致人身伤害、死亡或财产损失其他用途（“非预期用途”）。关键部件指，若该部件发生故障，经合理预期会导致设备或系统故障或会影响设备或系统安全性和有效性的部件。针对由赛普拉斯产品非预期用途产生或相关的任何主张、费用、损失和其他责任，赛普拉斯不承担全部或部分责任且贵方不应追究赛普拉斯之责任。贵方应赔偿赛普拉斯因赛普拉斯产品任何非预期用途产生或相关的所有索赔、费用、损失和其他责任，包括因人身伤害或死亡引起的主张，并使之免受损失。

赛普拉斯、赛普拉斯徽标、Spansion、Spansion 徽标，及上述项目的组合，及 PSoC、CapSense、EZ-USB、F-RAM 和 Traveo 应视为赛普拉斯在美国和其他国家的商标或注册商标。请访问 cypress.com 获取赛普拉斯商标的完整列表。其他名称和品牌可能由其各自所有者主张为该方财产。