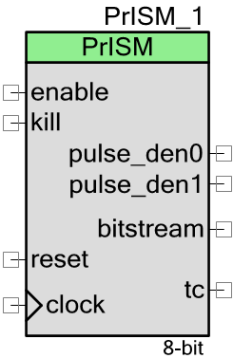
精确照明信号调制 (PrISM)

2.20



特性

- 2 到 32 位分辨率的可编程无闪烁调光
- 两路脉冲密度输出
- 可编程的输出信号密度
- 串行输出位流
- 连续运行模式
- 用户可配置序列启动值
- 为所有序列长度提供标准或自定义多项式
- 非同步停止输入端将禁用密度输出并强制它们处于低电平
- 启用输入端提供与其他组件的同步操作
- 复位输入端允许重新启动序列开始值以实现与其他组件的同步操作
- 适用于 8 位、16 位、24 位和 32 位序列长度的终端计数输出。

概述

精确照明信号调制 (PrISM) 组件使用线性反馈移位寄存器 (LFSR) 生成伪随机序列。此序列输出伪随机位流以及最多两个用户可调伪随机脉冲密度。这些脉冲密度的范围在 0 到 100% 之间。

LFSR 采用 Galois 形式（有时称为模形式），使用提供的最大长度代码。PrISM 组件启动后，只要使能输入处于高电平，此组件将持续运行。PrISM 伪随机数发生器可使用任意有效非零种子值启动。

何时使用 PrISM

PrISM 组件提供了调制技术，可大大降低低频闪烁和电磁辐射干扰 (EMI)，这些是高亮度 LED 设计的常见问题。PrISM 也可用于其他需要这种优势的应用，例如电机控制和供电电源。

输入/输出连接

本节介绍 PrISM 的各种输入和输出连接。I/O 列表中的星号 (*) 表示，在 I/O 说明中列出的情况下，该 I/O 可能不可见。

时钟 — 输入

时钟输入定义用于计算伪随机序列的信号。

复位 — 输入

复位输入用于将伪随机序列复位为高态的开始值。此输入仅对于已启动的组件有效，提供与其他组件的同步操作。

非同步停止 – 输入

高电平有效非同步停止输入用于禁用 PrISM 脉冲密度输出并将它们设置为 0 直至非同步停止输入释放为低电平。

启用 — 输入

PrISM 组件启动后，只要使能输入处于高电平，复位输入为低电平，此组件将继续运行。此输入提供与其他组件的同步操作。

pulse_den0/pulse_den1 – 输出

有两个脉冲密度输出可用；这两个输出都是派生自同一个伪随机序列。每个输出都是通过将需要的脉冲密度值与当前的伪随机序列数字进行比较而生成的。如果脉冲密度类型配置为 **Less Than or Equal**（小于或等于），则输出将处于高电平，而伪随机序列数字将小于或等于脉冲密度值。另一个选项是将脉冲密度类型设置为 **Greater Than or Equal**（大于或等于），则输出将处于高电平，而伪随机序列数字将大于或等于脉冲密度值。

位流 - 输出

位流输出用于连续输出 LFSR 的 LSb。

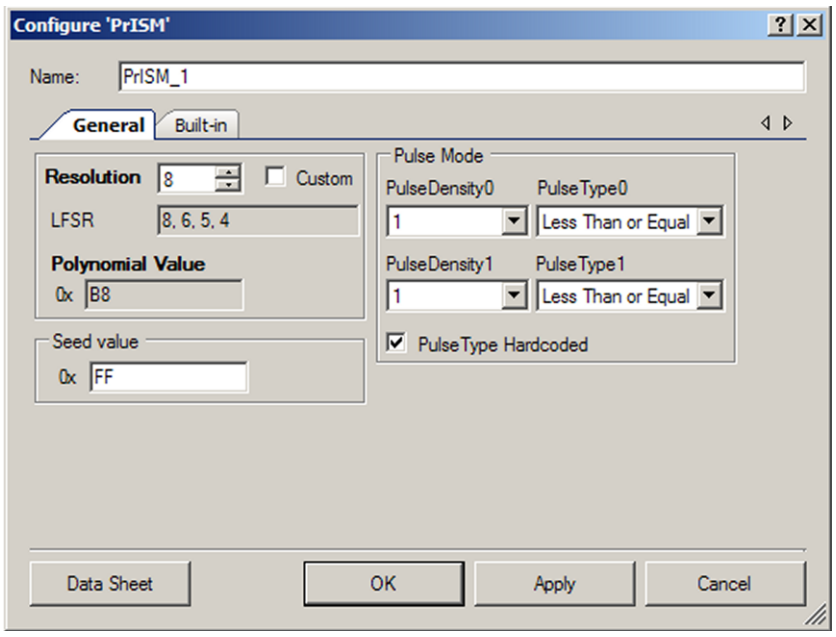
tc – 输出 *

终端计数输出适用于 8 位、16 位、24 位 和 32 位长度 PrISM 组件。在每个时钟周期内，每次伪随机序列数字等于 0xFF（8 位）、0xFFFF（16 位）、0xFFFFFF（24 位）或 0xFFFFFFFF（32 位）时，终端计数输出都会处于高电平，每个时钟伪随机序列数字发生器的每个周期都会出现一次这种情况。

组件参数

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

图 1. 配置对话框



PrISM 组件包含下列参数：

分辨率

此参数用于定义 PrISM 最大代码长度（周期）。最大代码长度为（$2^{分辨率} - 1$）。可能值包括 2 - 32 位。最大长度代码用于设置伪随机序列数字发生器的长度，因此，即要生成的序列长度。序列越长，脉冲密度分辨率将越高，同时辐射的 EMI 就越低。下表中列出的最大长度代码以 Galois 形式提供，在 PSoC 3 UDB ALU 中使用这些代码之前不需要进行转换。

表 1. 最大代码长度

分辨率	LFSR
2	2, 1
3	3, 2
4	4, 3
5	5, 4, 3, 2
6	6, 5, 3, 2
7	7, 6, 5, 4

分辨率	LFSR
13	13, 12, 10, 9
14	14, 13, 11, 9
15	15, 14, 13, 11
16	16, 14, 13, 11
17	17, 16, 15, 14
18	18, 17, 16, 13

分辨率	LFSR
24	24, 23, 21, 20
25	25, 24, 23, 22
26	26, 25, 24, 20
27	27, 26, 25, 22
28	28, 27, 24, 22
29	29, 28, 27, 25

分辨率	LFSR
8	8, 6, 5, 4
9	9, 8, 6, 5
10	10, 9, 7, 6
11	11, 10, 9, 7
12	12, 11, 8, 6

分辨率	LFSR
19	19, 18, 17, 14
20	20, 19, 16, 14
21	21, 20, 19, 16
22	22, 19, 18, 17
23	23, 22, 20, 18

分辨率	LFSR
30	30, 29, 26, 24
31	31, 30, 29, 28
32	32, 30, 26, 25

要手动设置 LFSR 系数：

1. 定义**分辨率**。
2. 选择 **Custom**（自定义）复选框。
3. 在 LFSR 文本框中输入用逗号分隔的系数并按 [**Enter**（输入）]。系统将自动重新计算多项式值。

Polynomial Value（多项式值）按十六进制格式显示。

注意 LFSR 系数值不能大于 **Resolution**（分辨率）值。

Polynomial Value（多项式值）

此参数使用十六进制格式。根据选择的 **Resolution**（分辨率）选择正确的多项式。可以指定自定义多项式。

Seed Value（种子值）

默认情况下，此参数设置为最大的可能值（$2^{\text{分辨率}} - 1$）。此值可更改为任意值（0 除外）。**Seed value**（种子值）按十六进制格式显示。

注意更改 **Resolution**（分辨率）会将 **Seed value**（种子值）设置为默认值。

脉冲模式

这些参数值是从组合框中选定的。可用值范围为 1 到 $2^{\text{分辨率}} - 1$，阶为 $2^{\text{分辨率}}$。脉冲比较类型可设置为 **Less Than or Equal**（小于或等于）或 **Greater Than or Equal**（大于或等于）。

PulseType Hardcoded（硬编码 PulseType）

PulseType Hardcoded（硬编码 PulseType）参数在启用后可节省硬件资源（控制寄存器），但是会造成无法使用 PrISM_SetPulse0Mode() 或 PrISM_SetPulse1Mode() API 更改脉冲类型。

如果启用了此函数，也无法使用 PrISM_Stop() 函数。在这种情况下，要停止 PrISM，使用“使能”输入。

本地参数（供 API 使用）

这些参数用于 API 中，不在 **Configure**（配置）对话框中显示。

- **PolyValue(uint32) –** 包含使用十六进制格式的多项式值。默认值为 0xB8h (LFSR= [8,6,5,4])。
- **Density0(uint32) –** 包含使用十六进制格式的 density0 值。
- **Density1(uint32) –** 包含使用十六进制格式的 density1 值。
- **CompareType0(CompareType) –** 包含 Density0 的 **Pulse Type**（脉冲类型），其可以为 **Less Than or Equal**（小于或等于）或 **Greater Than or Equal**（大于或等于）。
- **CompareType1(CompareType) –** 包含 Density1 的 **Pulse Type**（脉冲类型），其可以为 **Less Than or Equal**（小于或等于）或 **Greater Than or Equal**（大于或等于）。

时钟选择

此组件中没有内部时钟。您必须附加时钟源。此组件根据连接到组件的单时钟进行操作。

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“PrISM_1”分配给设计中的第一个组件实例。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。为增加可读性，下表中使用了实例名称“PrISM”。

表 2. 函数接口

函数	说明
PrISM_Start()	此启动函数用于设置定制器提供的多项式、种子和脉冲密度寄存器。
PrISM_Stop()	停止 PrISM 计算。
PrISM_SetPulse0Mode()	设置 Density0 的脉冲密度类型。
PrISM_SetPulse1Mode()	设置 Density1 的脉冲密度类型。
PrISM_ReadSeed()	读取 PrISM 种子寄存器。
PrISM_WriteSeed()	使用开始值写入 PrISM 种子寄存器。

PrISM_ReadPolynomial()	读取 PrISM 多项式寄存器。
PrISM_WritePolynomial()	使用开始值写入 PrISM 多项式寄存器。
PrISM_ReadPulse0()	PrISM 脉冲 Density0 值寄存器。
PrISM_WritePulse0()	使用新的脉冲密度值写入 PrISM 脉冲 Density0 值。
PrISM_ReadPulse1()	读取 PrISM 脉冲 Density1 值寄存器。
PrISM_WritePulse1()	使用新的脉冲密度值写入 PrISM 脉冲 Density1 值。
PrISM_Sleep()	停止并保存用户配置。
PrISM_Wakeup()	恢复并使能用户配置
PrISM_Init()	初始化随自定义程序提供的默认配置。
PrISM_Enable()	使能 PrISM 模块操作。
PrISM_SaveConfig()	保存当前用户配置。
PrISM_RestoreConfig()	恢复当前用户配置。

表 3. 全局变量

变量	说明
PrISM_initVar	说明 PrISM 是否已初始化。该变量初始化为 0，并在第一次调用 PrISM_Start() 时设置为 1。这样，第一次调用 PrISM_Start() 子程序后，组件不用重新初始化即可重启。 如果需要重新初始化此组件，则在 PrISM_Start() 或 PrISM_Enable() 函数之前可调用 PrISM_Init() 函数。

void PrISM_Start(void)

说明： 这是开始执行组件操作的首选方法。PrISM_Start() 用于设置 initVar 变量，调用 PrISM_Init() 函数并调用 PrISM_Enable() 函数。此启动函数用于设置定制器提供的多项式、种子和脉冲密度寄存器。PrISM 计算在输入时钟的上升沿上开始执行。

参数： 无

返回值： 无

副作用： 无

void PrISM_Stop(void)

说明： 停止 PrISM 计算。输出保持固定。

参数： 无

返回值： 无

副作用： 只有在禁用 **PulseType Hardcoded**（硬编码 PulseType）参数时才有效。

void PrISM_SetPulse0Mode(uint8 pulse0Type)

说明： 设置 Density0 的脉冲密度类型。小于或等于 (<=) 或大于或等于 (>=)。

参数： uint8 pulse0Type：选择脉冲密度类型

参数值	说明
PrISM_LESSTHAN_OR_EQUAL	当伪随机数字小于或等于 PulseDensity0 寄存器值时，pulse_den0 处于高电平。
PrISM_GREATERTHAN_OR_EQUAL	当伪随机数字大于或等于 PulseDensity0 寄存器值时，pulse_den0 处于高电平。

返回值： 无

副作用： 只有在禁用 **PulseType Hardcoded**（硬编码 PulseType）参数时才有效。

void PrISM_SetPulse1Mode(uint8 pulse1Type)

说明： 设置 Density1 的脉冲密度类型。小于或等于 (<=) 或大于或等于 (>=)。

参数： uint8 pulse1Type：选择脉冲密度类型

参数值	说明
PrISM_LESSTHAN_OR_EQUAL	当伪随机数字小于或等于 PulseDensity1 寄存器值时，pulse_den1 处于高电平。
PrISM_GREATERTHAN_OR_EQUAL	当伪随机数字大于或等于 PulseDensity1 寄存器值时，pulse_den1 处于高电平。

返回值： 无

副作用： 只有在禁用 **PulseType Hardcoded**（硬编码 PulseType）参数时才有效。

uint8/16/32 PrISM_ReadSeed(void)

说明： 读取 PrISM 种子寄存器。

参数： 无

返回值： uint8/16/32：设置寄存器值

副作用： 无

void PrISM_WriteSeed(uint8/16/32 seed)

说明： 使用开始值写入 PrISM 种子寄存器。

参数： uint8/16/32) 种子：设置寄存器值

返回值： 无

副作用： 无

uint8/16/32 PrISM_ReadPolynomial(void)

说明： 读取 PrISM 多项式。

参数： 无

返回值： uint8/16/32：多项式值

副作用： 无

void PrISM_WritePolynomial(uint8/16/32 polynomial)

说明： 写入 PrISM 多项式。

参数： uint8/16/32 polynomial：多项式寄存器值

返回值： 无

副作用： 无

uint8/16/32 PrISM_ReadPulse0(void)

说明： 读取 PrISM 脉冲 Density0 值寄存器。

参数： 无

返回值： uint8/16/32：PulseDensity0 寄存器值

副作用： 无

void PrISM_WritePulse0(uint8/16/32 pulseDensity0)

说明： 使用新的脉冲密度值写入 PrISM 脉冲 Density0 值。

参数： (unit8/16/32) pulseDensity0：脉冲密度值。

返回值： 无

副作用： 无

uint8/16/32 PrISM_ReadPulse1(void)

说明： 读取 PrISM 脉冲 Density1 值寄存器。

参数： 无

返回值： uint8/16/32：PulseDensity1 寄存器值

副作用： 无

void PrISM_WritePulse1(uint8/16/32 pulseDensity1)

说明： 使用新的脉冲密度值写入 PrISM 脉冲 Density1 值。

参数： uint8/16/32 pulseDensity1：脉冲密度值

返回值： 无

副作用： 无

void PrISM_Sleep(void)

说明： 这是让组件进入睡眠的首选 API。PrISM_Sleep() API 保存当前组件的状态。然后，它将调用 PrISM_Stop() 函数，并调用 PrISM_SaveConfig() 以保存硬件配置。

在调用 CyPmSleep() 或 CyPmHibernate() 函数之前调用 PrISM_Sleep() 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： 无

返回值： 无

副作用： 无

void PrISM_Wakeup(void)

说明： 此函数是将组件恢复到调用 PrISM_Sleep() 时状态的首选 API。PrISM_Wakeup() 函数调用 PrISM_RestoreConfig() 函数以恢复配置。如果组件在系统调用 PrISM_Sleep() 函数前已启用，则 PrISM_Wakeup() 函数也会重新启用组件。

参数： 无

返回值： 无

副作用： 调用 PrISM_Wakeup() 函数前未调用 PrISM_Sleep() 或 PrISM_SaveConfig() 函数可能会产生意外行为。

void PrISM_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 PrISM_Init()，因为 PrISM_Start() API 会调用该函数，这是开始组件操作的首选方法。

参数： 无

返回值： 无

副作用： 所有寄存器均根据定制器“配置”对话框设置为相应的值。

void PrISM_Enable(void)

说明： 激活硬件并开始执行组件操作。无需调用 PrISM_Enable()，因为 PrISM_Start() 会调用此函数，该函数是开始执行组件操作的首选方法。

参数： 无

返回值： 无

副作用： 无

void PrISM_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 PrISM_Sleep() 函数调用。

参数： 无

返回值： 无

副作用： 无

void PrISM_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 PrISM_Sleep() 函数之前的值。

参数： 无

返回值： 无

副作用： 调用该函数前未调用 PrISM_Sleep() 或 PrISM_SaveConfig() 函数可能会产生意外行为。

MISRA 合规性

本节介绍了本组件与 MISRA-C:2004 的合规和偏差情况。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 合规性章节中介绍项目偏差以及有关 MISRA 合规性验证环境的信息。

此 PrISM 组件没有任何特定偏差。

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的例子项目。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，请参见 PSoC Creator 帮助中的“Find Example Project（查找示例项目）”主题。

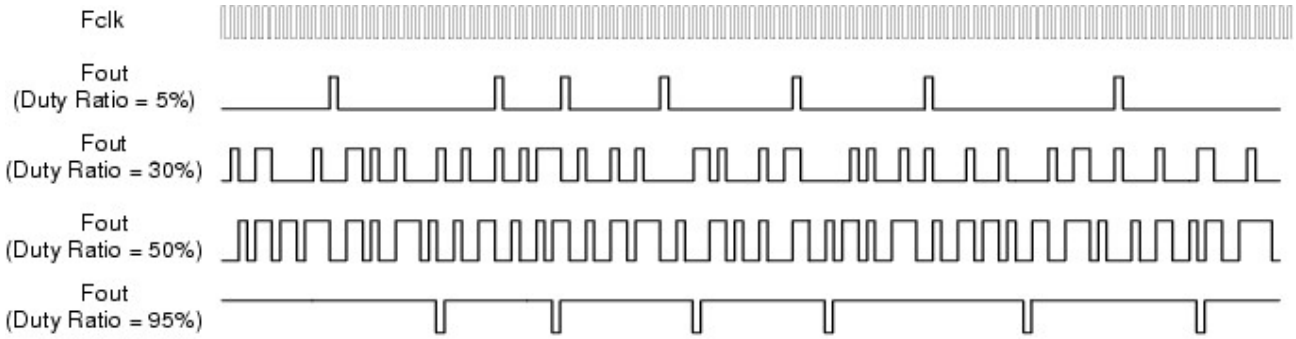
功能描述

PrISM 组件启动后，只要“使能”输入处于高电平，此组件将持续运行。PrISM 伪随机数字发生器可以任意有效值（0 除外）进行启动。从而允许多个 PrISM 组件相互之间不同步运行，从而进一步减少 EMI。“复位”输入用于将伪随机数字复位为开始值。高电平有效非同步停止输入用于禁用 PrISM 脉冲密度输出并将它们设置为 0 直至非同步停止输入释放为低电平。“位流”输出用于连续输出 LFSR 的 LSb。

有两个脉冲密度输出可用；这两个输出都是派生自同一个伪随机序列。每个输出都是通过将需要的脉冲密度值与当前的伪随机序列数字进行比较而生成的。

下表显示了基于多个脉冲密度比率的 PrISM 输出。

图 2. PrISM 输出的时序

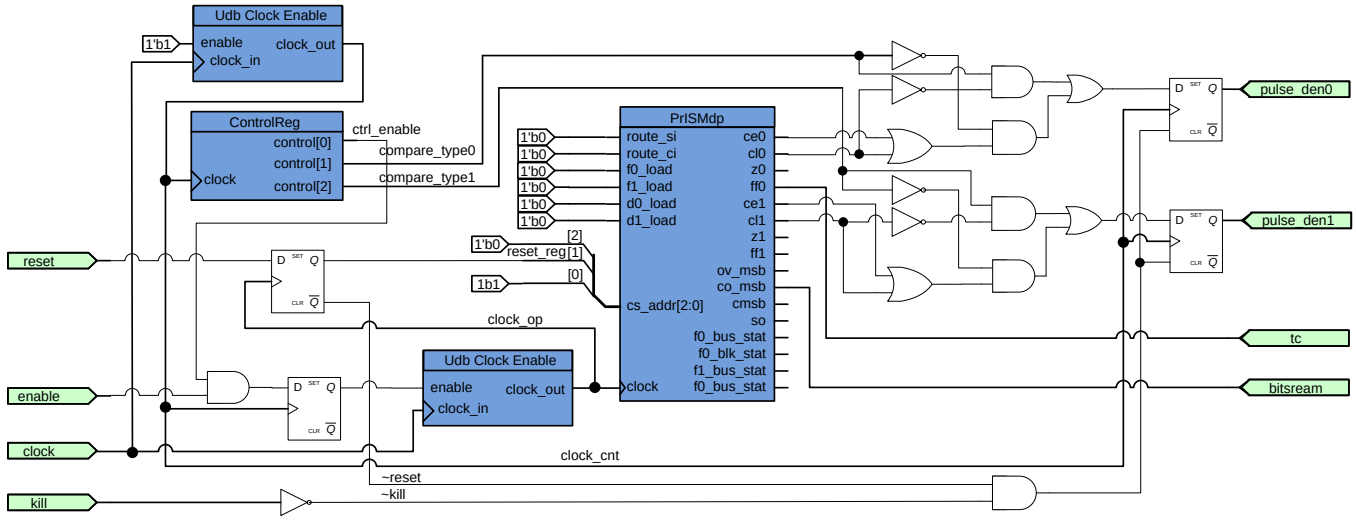


框图和配置

PrISM 仅作为 UDB 配置提供。上面所描述的 API 和此处所描述的寄存器用于定义 PrISM 的整体实现。

下面的框图中描述了实现。

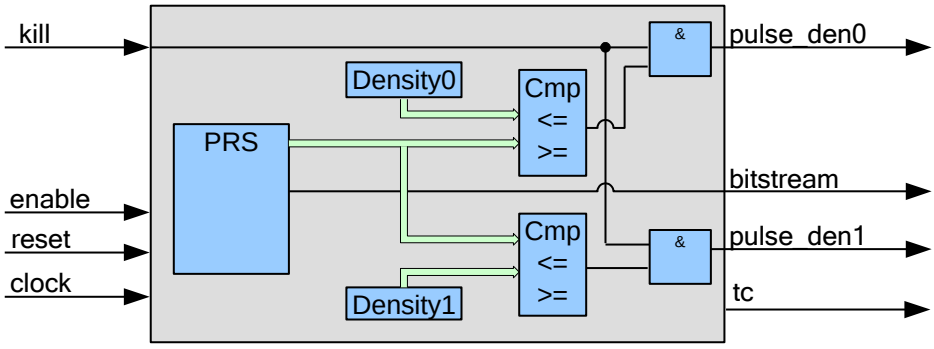
图 3. PrISM 实现



顶层架构

2 位到 32 位的硬件 PrISM 组件将伪随机计数器的输出与单个密度值进行比较。当计数值小于（或大于）或等于密度值寄存器中的值时，比较器输出置位。

图 4. PrISM 顶层架构



寄存器

PrISM_CONTROL

位	7	6	5	4	3	2	1	0
值	保留					比较 type1	比较 type0	ctrl 启用

- ctrl 启用：此位启用前面章节中说明的所有信号的生成功能。此值可 PrISM_Start() 和 PrISM_Stop() 函数进行更改。
- 比较 type0：此位执行 pulse_den0 输出的比较类型。此位的值由组件“Configure”（配置）对话框中做出的脉冲比较类型选项确定。同时，此值可由 PrISM_SetPulse0Mode() 函数进行更改。
- 比较 type1：此位执行 pulse_den1 输出的比较类型。此位的值由组件“Configure”（配置）对话框中做出的脉冲比较类型选项确定。同时，此值可由 PrISM_SetPulse1Mode() 函数进行更改。

如果选择了 **PulseType Hardcoded**（硬编码 PulseType）选项，则不使用控制寄存器。

PrISM_SEED

位	7	6	5	4	3	2	1	0
值	种子							

- 种子：包含计算计数时的初始种子值和 PRS 余值。此寄存器的值由组件“Configure”（配置）对话框中的 **Seed value**（种子值）确定。同时，此值可由 PrISM_WriteSeed() 函数进行更改，且可由 PrISM_ReadSeed() 函数进行读取。

PrISM_SEED_COPY

位	7	6	5	4	3	2	1	0
值	Seed_Copy							

- Seed_Copy：包含开始的种子值，当“复位”输入有效时，此值将自动加载 PrISM_SEED 寄存器。此寄存器的值由组件“Configure”（配置）对话框中的**种子值**确定，而且如果调用了 PrISM_WriteSeed() 函数将自动进行更新。

PrISM_POLYNOM

位	7	6	5	4	3	2	1	0
值	多项式							

- 多项式：根据选择的分辨率选择正确的多项式。此值可由 PrISM_WritePolynomial() 函数进行更改，且可由 PrISM_ReadPolynomial() 函数进行读取。

PrISM_DENSITY0

位	7	6	5	4	3	2	1	0
值	脉冲 density0							

- 脉冲 density0 确定 PrISM pulse_den0 输出的值。此寄存器的值由“Configure”（配置）对话框中的 **PulseDensity0** 参数确定。此值可由 PrISM_WritePulse0() 函数进行更改。

PrISM_DENSITY1

位	7	6	5	4	3	2	1	0
值	脉冲 density1							

- 脉冲 density1 确定 PrISM pulse_den1 输出的值。此寄存器的值由“Configure”（配置）对话框中的 **PulseDensity1** 参数确定。此值可由 PrISM_WritePulse1() 函数进行更改。

参考

另请参见 PRS 组件数据表。

资源

PrISM 组件放置在整个 UDB 阵列中。该组件利用以下资源。

表 4. 资源类型

配置	资源类型					
	数据路径单元	宏单元	状态单元	控制单元[1]	DMA 通道	中断
8 位	1	4	–	1	–	–
16 位	2	4	–	1	–	–
24 位	3	4	–	1	–	–
32 位	4	4	–	1	–	–

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。有关特定的设计，可分析编译器生成的映射文件以确定内存使用情况。

表 5. API 内存资源使用

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
8 位	281	6	440	9	428	9
16 位	428	9	456	9	444	9
24 位	419	15	520	17	512	17
32 位	421	15	456	17	432	17

直流和交流电气特性

除非另有说明，否则这些规范的适用条件是 -40 °C ≤ T_A ≤ 85 °C 且 T_J ≤ 100 °C。除非另有说明，否则这些规范的适用范围为 1.71 V 到 5.5 V。

[1] 如果选中 **PulseType Hardcoded**（硬编码 PulseType）参数，则不会使用控制单元。

表 6. 直流电特性

参数	说明	最小值	典型值 [2]	最大值	单位
I_{DD}	组件电流消耗				
	8 位	–	15	–	µA/MHz
	16 位	–	22	–	µA/MHz
	24 位	–	28	–	µA/MHz
	32 位	–	35	–	µA/MHz

表 7. 交流特性

参数	说明	最小值	典型值	最大值[3]	单位
f_{CLOCK}	组件时钟频率				
	8 位			66	MHz
	16 位			55	MHz
	24 位			48	MHz
	32 位			40	MHz

2. 未包括设备 IO 和时钟分配的电流。这些值是在 25 °C 时的值。

3. 这些值提供了此组件的最大安全工作频率。可以在更高的时钟频率运行此组件，在该频率将需要使用 STA 结果验证时序要求。

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
2.20.a	更新了数据表，添加了 PSoC 4 的内存使用情况。	
2.20	已添加 MISRA 合规性章节。	此组件没有任何特定偏差。
2.10	添加了所有包括在 .cyre 文件中的带 CYREENTRANT 关键词的 API。	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。 需要此更改为采用安全方式使用并且不是可重入函数消除编译器警告：通过标志或关键节防止并发调用。
	添加了支持 PSoC 5LP 芯片。	
2.0.a	对数据表进行了少量编辑和更新	
2.0	已寄存的脉冲密度输出，用于消除短时脉冲。	任何组合输出都可能出现短时脉冲，具体取决于放置和信号之间的延迟。要消除短时脉冲，应将输出寄存。
	已寄存的使能输入和复位输入，以提高最大运行。	这些输入已组合使用，因此不会被 Creator 自动寄存，且有违规行为。寄存用于提高最大速度，并避免可能的短时脉冲。
	向数据表中添加了特性数据	
	对数据表进行了少量编辑和更新	