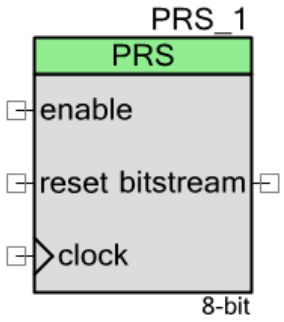
伪随机序列 (PRS)

2.30

特性



- 2-64 位 PRS 序列长度
- 时分复用模式
- 串行输出位流
- 连续或单步运行模式
- 标准或自定义多项式
- 标准或自定义种子值
- 启用输入提供与其他组件的同步操作
- 可以从线性反馈移位寄存器 (LFSR) 直接读取计算后得出的伪随机数字

概述

伪随机序列 (PRS) 组件使用 LFSR 生成伪随机序列，由此输出一个伪随机位流。LFSR 是 Galois 格式（有时也称为模块格式），并使用所提供的最大代码长度或周期。使能输入保持在高电平时，PRS 组件便可以在启动后连续运行。使用除 0 以后的任意有效种子值，可以启动 PRS 数字发生器。

何时使用 PRS

LFSR 可以在硬件中实现。这使得 LFSR 在要求非常快速地生成伪随机序列的应用程序中非常有用，例如，直接序列扩频无线电。

全球定位系统使用 LFSR 快速传输相对时间偏移的高精度序列。此外，某些视频游戏控制器也将 LFSR 用作音响系统的一部分。

用作计数器

可以接受非二进制序列时，通过 LFSR 状态的重复序列，将其用作分频器或计数器。与自然二进制计数器或格雷码计数器相比，LFSR 计数器具有简单的反馈逻辑，因此可以在较高的时间频率下

运行。然而，必须确保 LFSR 从未进入全零状态，例如，在启动时，通过将其预设为序列中其他任何状态。

输入/输出连接

本节介绍 PRS 组件的各种输入和输出连接。I/O 列表中的星号 (*) 表示该 I/O 是可隐藏 I/O，其隐藏条件在该 I/O 的说明中。

clock – Input *（时钟 – 输入）*

时钟输入定义要计算 PRS 的信号。当选择 API 单步运行模式时，此输入不可用。

复位 – 输入*

复位输入用来定义要同步复位 PRS 的信号。此输入在选择时钟模式时不可用。只有使能输入保持在高电平时，才能复位 PRS。

启用 — 输入

使能输入保持在高电平时，PRS 组件便可以在启动后运行。此输入提供与其他组件的同步操作。

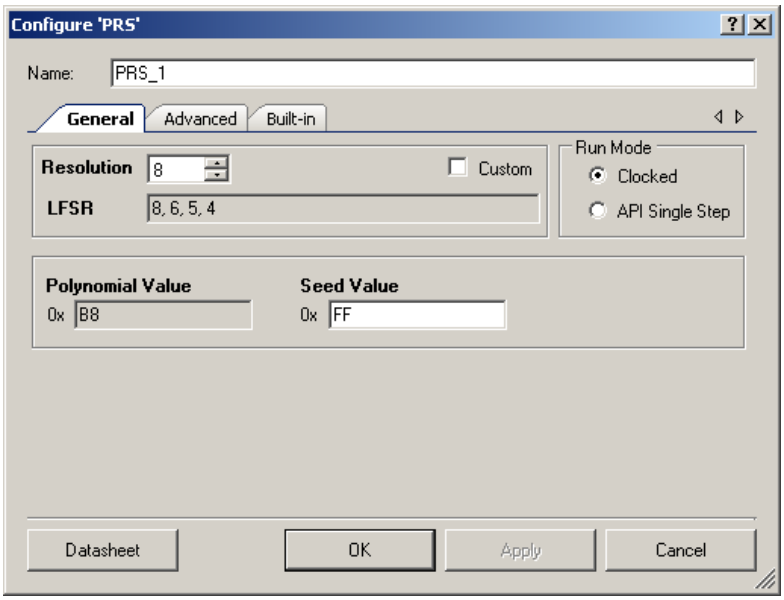
位流 - 输出

LFSR 输出。

组件参数

将 PRS 组件拖入设计中，双击该组件，打开 **Configure**（配置）对话框。该对话框有若干选项卡，可引导您完成 PRS 组件的设置过程。

一般选项卡



分辨率

这用来定义 PRS 序列的长度。该值设置介于 2-64 之间。默认值为 **8**。

默认情况下，**分辨率**用来定义 **LFSR** 系数**和多项式值**。这些系数取自下表。此外，此参数还用来定义最大代码长度或周期，如下表所示。

分辨率	LFSR	周期（$2^{分辨率} - 1$）		分辨率	LFSR	周期（$2^{分辨率} - 1$）
2	2, 1	3		34	34, 31, 30, 26	17179869183
3	3, 2	7		35	35, 34, 28, 27	34359738367
4	4, 3	15		36	36, 35, 29, 28	68719476735
5	5, 4, 3, 2	31		37	37, 36, 33, 31	137438953471
6	6, 5, 3, 2	63		38	38, 37, 33, 32	274877906943
7	7, 6, 5, 4	127		39	39, 38, 35, 32	549755813887
8	8, 6, 5, 4	255		40	40, 37, 36, 35	1099511627775
9	9, 8, 6, 5	511		41	41, 40, 39, 38	2199023255551
10	10, 9, 7, 6	1023		42	42, 40, 37, 35	4398046511103

分辨率	LFSR	周期（$2^{分辨率} - 1$）		分辨率	LFSR	周期（$2^{分辨率} - 1$）
11	11, 10, 9, 7	2047		43	43, 42, 38, 37	8796093022207
12	12, 11, 8, 6	4095		44	44, 42, 39, 38	17592186044415
13	13, 12, 10, 9	8191		45	45, 44, 42, 41	35184372088831
14	14, 13, 11, 9	16383		46	46, 40, 39, 38	70368744177663
15	15, 14, 13, 11	32767		47	47, 46, 43, 42	140737488355327
16	16, 14, 13, 11	65535		48	48, 44, 41, 39	281474976710655
17	17, 16, 15, 14	131071		49	49, 45, 44, 43	562949953421311
18	18, 17, 16, 13	262143		50	50, 48, 47, 46	1125899906842623
19	19, 18, 17, 14	524187		51	51, 50, 48, 45	2251799813685247
20	20, 19, 16, 14	1048575		52	52, 51, 49, 46	4503599627370495
21	21, 20, 19, 16	2097151		53	53, 52, 51, 47	9007199254740991
22	22, 19, 18, 17	4194303		54	54, 51, 48, 46	18014398509481983
23	23, 22, 20, 18	8388607		55	55, 54, 53, 49	36028797018963967
24	24, 23, 21, 20	16777215		56	56, 54, 52, 49	72057594037927935
25	25, 24, 23, 22	33554431		57	57, 55, 54, 52	144115188075855871
26	26, 25, 24, 20	67108863		58	58, 57, 53, 52	288230376151711743
27	27, 26, 25, 22	134217727		59	59, 57, 55, 52	576460752303423487
28	28, 27, 24, 22	268435455		60	60, 58, 56, 55	1152921504606846975
29	29, 28, 27, 25	536870911		61	61, 60, 59, 56	2305843009213693951
30	30, 29, 26, 24	1073741823		62	62, 59, 57, 56	4611686018427387903
31	31, 30, 29, 28	2147483647		63	63, 62, 59, 58	9223372036854775807
32	32, 30, 26, 25	4294967295		64	64, 63, 61, 60	18446744073709551615
33	33, 32, 29, 27	8589934591				

要手动设置 LFSR 系数：

定义**分辨率**。

选中 **Custom**（自定义）复选框。

在 **LFSR** 文本框中输入系数，用逗号分隔，然后按 [**Enter**]。系统将自动重新计算多项式值。多项式值显示为十六进制格式。

注意：任何 LFSR 系数的值均不大于**分辨率**的值。

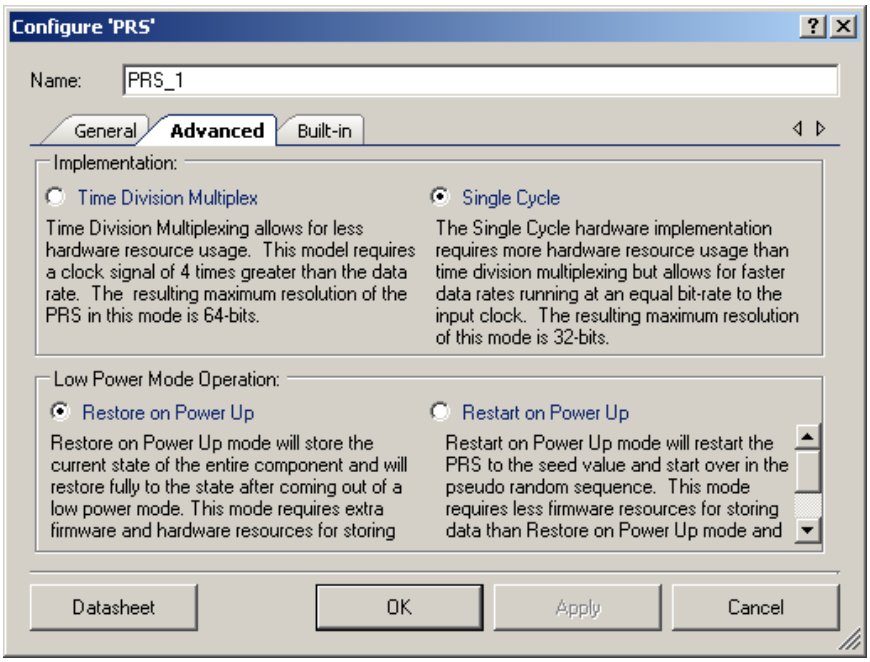
默认情况下，种子值设置为最大可能值（2 分辨率 – 1）。其值可以更改为除 0 以外的任何值。种子值显示为十六进制格式。

注意：更改**分辨率**时，可以将**种子值**复位到默认值。

运行模式

此参数用于将组件运行模式定义为连续运行或单步运行模式。您可以选择 **Clocked**（时钟）（默认值）或 **API Single Step**（API 单步执行）。如果连续读取 PRS 的值，或需要读取一个值，则必须停止时钟或在时钟模式下将使能设置为低电平。

“高级”选项卡



PRS **Advanced**（高级）选项卡包含以下设置：

实现

这用来定义 PRS 组件的实现：包含或不包含时间复用（**单周期**）。默认值为 **Single Cycle**（单周期）。

低功耗模式运行

这用来定义低功耗模式下的 PRS 行为。默认值为 **Restore on Power Up**（加电时存储）。

本地参数（供 API 使用）

以下参数用于 API，不在 GUI 中使用：

- **PolyValueLower(uint32)** – 包含下半部分多项式值（十六进制格式）。默认值为 0xB8h (**LFSR**= [8,6,5,4])，因为默认分辨率值为 8。
- **PolyValueUpper(uint32)** – 包含下半部分多项式值（十六进制格式）。默认值为 0x00h，因为默认分辨率为 8。
- **SeedValueLower (uint32) —** 包含十六进制格式的种子值的下半部分。默认值为 0xFFh，因为默认分辨率为 8。
- **SeedValueUpper (uint32) —** 包含十六进制格式的种子值的上半部分。默认值为 0，因为默认分辨率为 8。

时钟选择

如果选择 **Run Mode**（运行模式）参数的 **Clocked**（时钟）选项，则必须连接时钟源。

注意：如果为 **Implementation**（实现）参数选择 **Time Division Multiplex**（时分复用），想要生成分辨率大于 8 的正确 PRS 序列，则要求时钟信号比数据速率大 4 倍。

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“PRS_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 "PRS"。

函数	说明
PRS_Start()	初始化由自定义程序提供的种子和多项式寄存器。在输入时钟上升沿启动 PRS 计算功能。
PRS_Stop()	停止 PRS 计算功能。
PRS_Sleep()	停止 PRS 计算功能并保存 PRS 配置。
PRS_Wakeup()	存储 PRS 配置，并在输入时钟上升沿启动 PRS 计算功能。
PRS_Init()	使用初始值初始化种子和多项式寄存器。
PRS_Enable()	在输入时钟上升沿启动 PRS 计算功能。

函数	说明
PRS_SaveConfig()	保存种子和多项式寄存器。
PRS_RestoreConfig()	存储种子和多项式寄存器。
PRS_Step()	使用 API 单步执行模式时，PRS 增量为 1。
PRS_WriteSeed()	写入种子值。
PRS_WriteSeedUpper()	写入上半部分种子值。仅为 33-64 位 PRS 生成。
PRS_WriteSeedLower()	写入下半部分种子值。仅为 33-64 位 PRS 生成。
PRS_Read()	读取 PRS 值。
PRS_ReadUpper()	读取上半部分 PRS 值。仅为 33-64 位 PRS 生成。
PRS_ReadLower()	读取下半部分 PRS 值。仅为 33-64 位 PRS 生成。
PRS_WritePolynomial()	写入 PRS 多项式值。
PRS_WritePolynomialUpper()	写入上半部分 PRS 多项式值。仅为 33-64 位 PRS 生成。
PRS_WritePolynomialLower()	写入下半部分 PRS 多项式值。仅为 33-64 位 PRS 生成。
PRS_ReadPolynomial()	读取 PRS 多项式值。
PRS_ReadPolynomialUpper()	读取上半部分 PRS 多项式值。仅为 33-64 位 PRS 生成。
PRS_ReadPolynomialLower()	读取下半部分 PRS 多项式值。仅为 33-64 位 PRS 生成。

全局变量

变量	说明
PRS_initVar	表示是否完成初始化 PRS。变量将初始化为 0，并在第一次调用 PRS_Start() 时设置为 1。这样，第一次调用 PRS_Start() 子程序后，组件不用重新初始化即可重启。 如需重新初始化组件，可在 PRS_Start() 或 PRS_Enable() 函数前调用 PRS_Init() 函数。

void PRS_Start(void)

说明： 初始化种子和多项式寄存器。在输入时钟上升沿启动 PRS 计算功能。

参数： 无

返回值： 无

副作用： 无

void PRS_Stop(void)

说明： 停止 PRS 计算功能。

参数： 无

返回值： 无

副作用： 无

void PRS_Sleep(void)

说明： 停止 PRS 计算功能，然后保存 PRS 配置。

参数： 无

返回值： 无

副作用： 无

void PRS_Wakeup(void)

说明： 存储 PRS 配置，并在输入时钟上升沿启动 PRS 计算功能。

参数： 无

返回值： 无

副作用： 无

void PRS_Init(void)

说明： 用初始值初始化种子和多项式寄存器。

参数： 无

返回值： 无

副作用： 无

void PRS_Enable(void)

说明： 在输入时钟上升沿启动 PRS 计算功能。

参数： 无

返回值： 无

副作用： 无

void PRS_SaveConfig(void)

说明： 保存种子和多项式寄存器。

参数： 无

返回值： 无

副作用： 无

void PRS_RestoreConfig(void)

说明： 恢复种子和多项式寄存器。

参数： 无

返回值： 无

副作用： 无

void PRS_Step(void)

说明： 使用 API 单步执行模式时，PRS 增量为 1。

参数： 无

返回值： 无

副作用： 无

void PRS_WriteSeed(uint8/16/32 seed)

说明： 写入种子值。

参数： uint8/16/32 seed：种子值

返回值： 无

副作用： 根据掩码 = $2^{\text{分辨率}}$ — 1 剪切种子值。例如，如果 PRS 分辨率为 14 位，则掩码值为：掩码 = 2^{14} – 1 = 0x3FFFu。截取种子值 = 0xFFFFu：种子与掩码 = 0xFFFFu AND 0x3FFFu = 0x3FFFu。

void PRS_WriteSeedUpper(uint32 seed)

说明： 写入种子值的上半部分。仅为 33-64 位 PRS 生成。

参数： uint32 seed：种子值的上半部分

返回值： 无

副作用： 根据掩码 = 2 (分辨率 – 32) – 1 截取上半部分种子值。
例如，如果 PRS 分辨率为 35 位，则掩码值为：
2(35 – 32) – 1 = 2^3 – 1 = 0x0000 0007u.
种子值的上半部分 = 0x0000 00FFu 被剪切。
上半部分种子与掩码 = 0x0000 00FFu AND 0x0000 0007u = 0x0000 0007u。

void PRS_WriteSeedLower(uint32 seed)

说明： 写入种子值的下半部分。仅为 33-64 位 PRS 生成。

参数： uint32 seed：种子值的下半部分

返回值： 无

副作用： 无

uint8/16/32 PRS_Read(void)

说明： 读取 PRS 值。

参数： 无

返回值： uint8/16/32：返回 PRS 值。

副作用： 无

uint32 PRS_ReadUpper(void)

说明： 读取上半部分 PRS 值。仅为 33-64 位 PRS 生成。

参数： 无

返回值： uint32：返回上半部分 PRS 值。

副作用： 无

uint32 PRS_ReadLower(void)

说明： 读取下半部分 PRS 值。仅为 33-64 位 PRS 生成

参数： 无

返回值： uint32：返回下半部分 PRS 值。

副作用： 无

void PRS_WritePolynomial(uint8/16/32 polynomial)

说明： 写入 PRS 多项式值。

参数： uint8/16/32 polynomial：PRS 多项式。

返回值： 无

副作用： 根据掩码 = $2^{分辨率}$ — 1 剪切多项式值。
例如，如果 PRS 分辨率为 14 位，则掩码值为：掩码 = 2^{14} – 1 = 0x3FFFu。
多项式值 = 0xFFFFu 被剪切：
多项式与掩码 = 0xFFFFu AND 0x3FFFu = 0x3FFFu。

void PRS_WritePolynomialUpper(uint32 polynomial)

说明： 写入上半部分 PRS 多项式值。仅为 33-64 位 PRS 生成。

参数： uint32 polynomial：上半部分 PRS 多项式值。

返回值： 无

副作用： 根据掩码 = $2^{(分辨率 - 32)}$ – 1 截取上半部分多项式值。
例如，如果 PRS 分辨率为 35 位，则掩码值为：
$2^{(35 - 32)}$ — 1 = 2^3 — 1 = 0x0000 0007u。
多项式值的上半部分 = 0x0000 00FFu 被剪切：
上半部分多项式与掩码 = 0x0000 00FFu AND 0x0000 0007u = 0x0000 0007u。

void PRS_WritePolynomialLower(uint32 polynomial)

说明： 写入下半部分 PRS 多项式值。仅为 33-64 位 PRS 生成。

参数： uint32 polynomial：下半部分 PRS 多项式值

返回值： 无

副作用： 无

uint8/16/32 PRS_ReadPolynomial(void)

说明： 读取 PRS 多项式值。

参数： 无

返回值： uint8/16/32：返回 PRS 多项式值。

副作用： 无

uint32 PRS_ReadPolynomialUpper(void)

说明： 读取上半部分 PRS 多项式值。仅为 33-64 位 PRS 生成。

参数： 无

返回值： uint32：返回上半部分 PRS 多项式值。

副作用： 无

uint32 PRS_ReadPolynomialLower(void)

说明： 读取下半部分 PRS 多项式值。仅为 33-64 位 PRS 生成。

参数： 无

返回值： uint32：返回下半部分 PRS 多项式值。

副作用： 无

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。

尚未根据 MISRA-C:2004 编码准则合规性，验证 PRS 组件源代码。

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

PRS 运行模式：时钟

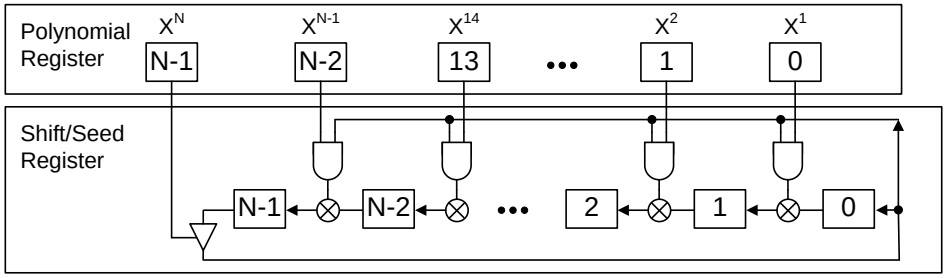
在此模式下，只要使能输入保持在高电平时，PRS 组件便可以在启动后连续运行。

PRS 运行模式：API 单步执行

在此模式下，通过 API 调用增加 PRS。

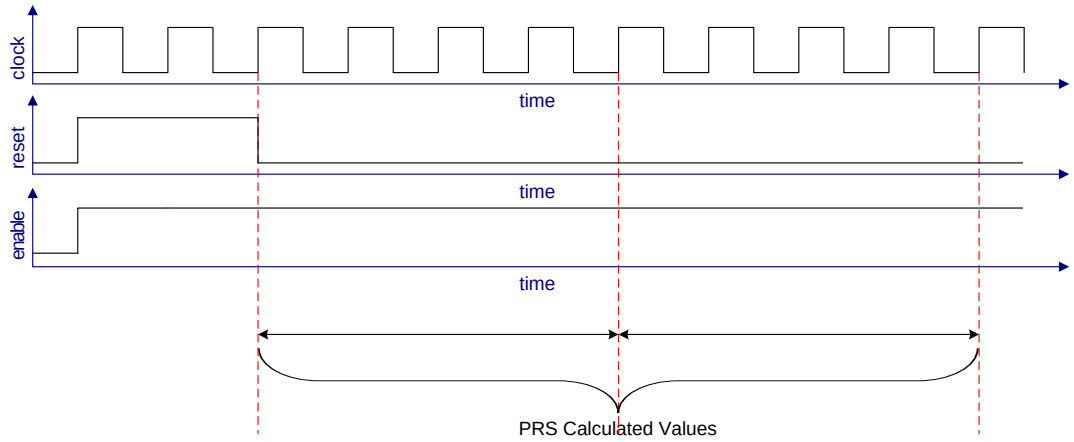
框图和配置

PRS 作为一组 UDB 配置得以实现。在以下框图中显示该实现。

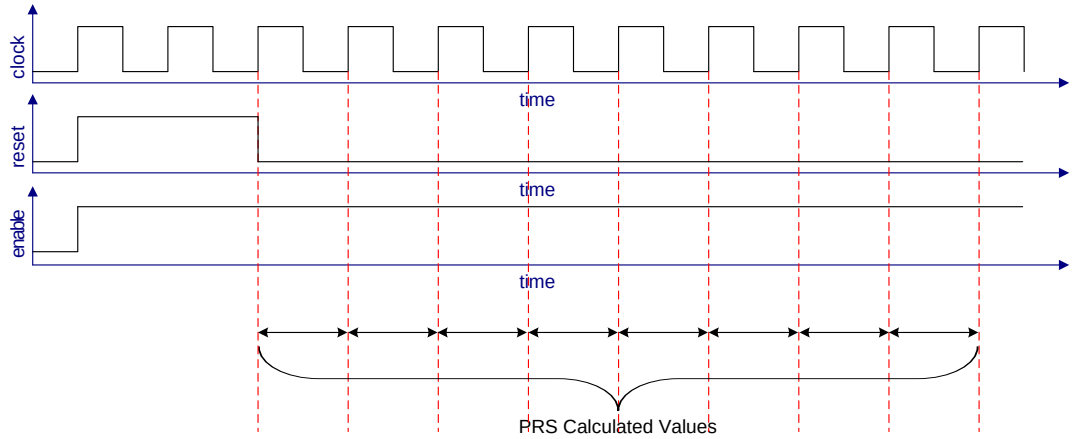


时序图

时分复用实现模式



单周期实现模式



寄存器

多项式寄存器（根据分辨率，从 2 位到 64 位）

多项式寄存器包含多项式值。通过 PRS_WritePolynomial()、PRS_WritePolynomialUpper() 或 PRS_WritePolynomialLower() 函数，可以对该值进行更改。此外，还可以使用 PRS_ReadPolynomial()、PRS_ReadPolynomialUpper() 或 PRS_ReadPolynomialLower() 读取当前多项式值。

移位/种子寄存器（2-64 位，根据寄存器）

移位/种子寄存器包含种子值。可以使用 PRS_WriteSeed()、PRS_WriteSeedUpper() 或 PRS_WriteSeedLower() 函数，可以对该值进行更改。此外，还可以使用 PRS_ReadSeed()、PRS_ReadSeedUpper() 或 PRS_ReadSeedLower() 读取当前种子值。

资源

PRS 组件放置在整个 UDB 阵列中。该组件利用以下资源。

配置	资源类型					
	数据路径单元	宏单元[1]	状态单元	控制单元	DMA 通道	中断
8 位单周期	1	1	–	1	–	–
16 位单周期	2	1	–	1	–	–
24 位单周期	3	1	–	1	–	–
32 位单周期	4	1	–	1	–	–
16 位时分	1	9	1	1	–	–
24 位时分	2	10	1	1	–	–
32 位时分	2	9	1	1	–	–
40 位时分	3	10	1	1	–	–
48 位时分	3	9	1	1	–	–
56 位时分	4	10	1	1	–	–
64 位时分	4	9	1	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。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。

1. 使用 API 单步运行模式为单周期实现使用其他宏。

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 位单周期	170	3	274	5	284	5
16 位单周期	232	4	296	5	306	5
24 位单周期	304	6	356	9	338	9
32 位单周期	302	6	312	9	326	9
16 位时分	306	6	424	9	434	9
24 位时分	595	8	508	13	510	13
32 位时分	671	8	536	13	546	13
40 位时分	842	12	696	17	730	17
48 位时分	977	12	748	17	804	17
56 位时分	1083	12	824	17	880	17
64 位时分	1175	12	850	17	864	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。

直流特性

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 位单周期	–	11	–	µA/MHz
	16 位单周期	–	17	–	µA/MHz
	24 位单周期	–	23	–	µA/MHz
	32 位单周期	–	31	–	µA/MHz
	16 位时分	–	22	–	µA/MHz
	24 位时分	–	30	–	µA/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。

参数	说明	最小值	典型值[2]	最大值	单位
	32 位时分	–	31	–	µA/MHz
	40 位时分	–	40	–	µA/MHz
	48 位时分	–	41	–	µA/MHz
	56 位时分	–	51	–	µA/MHz
	64 位时分	–	47	–	µA/MHz

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 位单周期	–	–	39	MHz
	16 位单周期	–	–	33	MHz
	24 位单周期	–	–	30	MHz
	32 位单周期	–	–	29	MHz
	16 位时分	–	–	29	MHz
	24 位时分	–	–	22	MHz
	32 位时分	–	–	28	MHz
	40 位时分	–	–	24	MHz
	48 位时分	–	–	28	MHz
	56 位时分	–	–	24	MHz
	64 位时分	–	–	26	MHz

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
2.30	更新了数据表，添加了 PSoC 4 的内存使用情况。 更新了对 PSoC 4 的 [25 – 32] 位单周期实现模式的定义。	
2.20	已添加 MISRA 合规性章节	尚未根据 MISRA 合规性验证此组件
2.10	添加了 PSoC 5LP 支持	
	已添加所有包括在 .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。
2.0.b	更新了数据表中的资源信息	
2.0.a	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	
2.0	添加了对 PSoC 3 Production 的芯片的支持。更改包括： ■ 添加了时分复用实现的 4x 时钟 ■ 1x 时钟上现可用 1 到 32 位的单周期实现。 ■ 4x 时钟上现可用 9 到 64 位的时分复用实现。 ■ 补充了同步输入信号复位。 ■ 补充了同步输入信号使能。 ■ 在 Configure（配置）对话框新增实现与低功耗模式参数的 'Advanced'（‘高级’）页面	新要求支持 PSoC 3 Production 组件，由此创建 PRS 组件新的 2.0 版本
	补充了 PRS_Sleep()/PRS_Wakeup() 和 PRS_Init()/PRS_Enable() APIs。	为支持低功耗模式并提供常用接口，以单独控制大多数组件的初始化和启用。
	更新了函数 PRS_WriteSeed() 和 PRS_WriteSeedUpper()。	掩码参数用于在写入时截取种子值以定义分辨率。
	将复位 DFF 触发器添加到多项式写入函数：PRS_WritePolynomial()、PRS_WritePolynomialUpper() 和 PRS_WritePolynomialLower()。	开始计算前，必须在正确状态下设置 DFF 触发（多项式最重要的位始终为 1）。要满足此条件，任何写入种子或多项式寄存器中的值均可以复位 DFF 触发。

版本	更改说明	更改/影响原因
	更新了 Configure（配置）对话框以用来支持某些参数的“表达式查看”。	“Expression View”（表达式视图）用于直接访问符号参数。此视图允许您用外部参数连接组件参数（如果需要）。
	已更新“Configure”（配置）对话框，以为各种参数添加错误图标。	如果在文本框中输入了错误的值，将显示错误图标以及问题说明的工具提示。这种方法比单独提供错误消息更方便。

© 赛普拉斯半导体公司，2013。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。