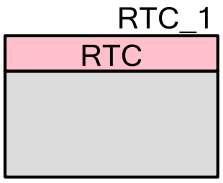
实时时钟 (RTC)

1.60



特性

- 多个警报选项
- 多个溢出选项
- 夏令时 (DST) 选项

概述

实时时钟 (RTC) 组件为系统提供精确的时间和日期信息。根据从 32.768-kHz 晶振生成的周期为 1 秒的脉冲中断，来实现每秒的时间和日期的更新。时钟精度取决于使用的晶振，通常为 20 ppm。

RTC 跟踪秒、分钟、小时、某周某日、某月某日、某年某日、月份和年份。某周某日是根据天、月份、年份自动计算的。可选择启用夏令时，它支持任何开始和结束日期，以及可编程夏令时间。开始和结束日期可以是绝对时间（如 3 月 24 日），或是相对时间（如五月的第二个周日）。

警报提供针对秒、分钟、小时、某周某日、某月某日、某年某日、月份和年份的匹配检测。掩码选择使用哪种时间和日期信息的组合来生成警报。警报灵活性支持定期警报（例如每小时后的每二十三分钟）或单个警报（例如 2043 年 9 月 28 日早上 4:52）。

用户代码存根用于基于各个主要时间间隔来进行定期代码执行。定时器间隔为 1 秒、1 分钟、1 小时、1 天、1 周、1 月、1 年。

何时使用 RTC 组件

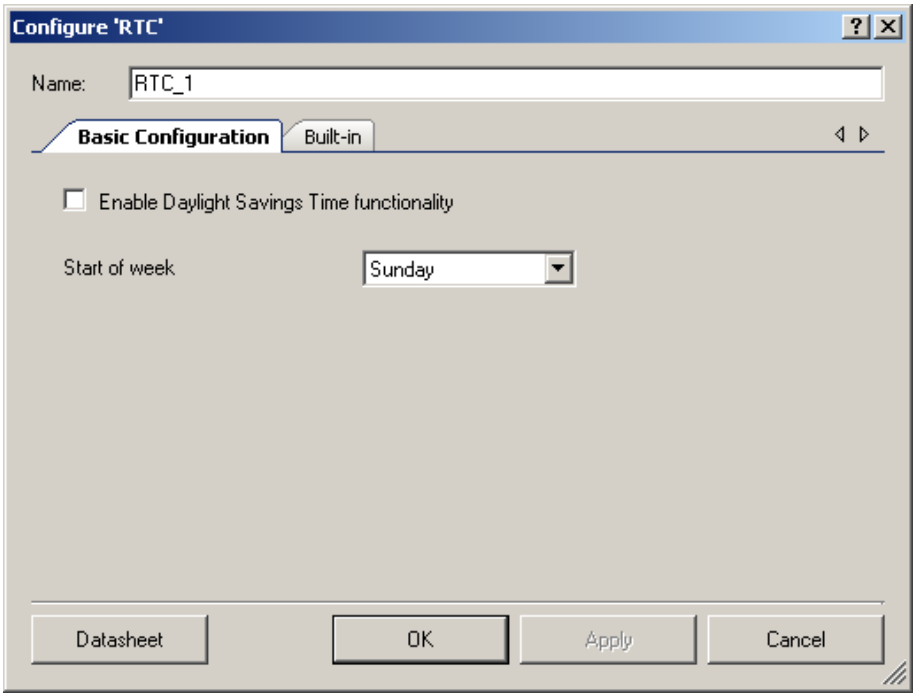
当系统需要当前时间或日期时，使用 RTC 组件。在不需要当前时间和日期，但需要事件的精确时序（精确到秒）时，也可使用 RTC。

输入/输出连接

RTC 组件没有输入或输出连接。

元件参数

将 RTC 组件拖入设计中，双击它以打开 **Configure**（配置）对话框。



RTC 组件包含以下选项：

Enable Daylight Savings Time functionality（启用夏令时功能）

此参数允许您选择是否在 RTC 中启用夏令时功能。默认值为已清除（假）。

一周开始

Start of week（一周开始）参数允许您选择周的开始日期。选项包括：**Sunday**（周日）（默认）、**Monday**（周一）、**Tuesday**（周二）、**Wednesday**（周三）、**Thursday**（周四）、**Friday**（周五）和 **Saturday**（周六）。

时钟选择

您应从外部晶振提供 32.768 kHz 时钟。此组件的精度由连接的外部时钟源的精度定义。有关如何连接和配置设计中的内置 XTAL_32KHZ 时钟的信息，请参见 PSoC Creator 帮助的时钟编辑器部分。

资源

分辨率	数字模块					API Memory（API 存储器）（字节）		Pins（引脚）（每个外部 I/O）
	Datapaths（数据路径）	Macro cells（宏单元）	Status Registers（状态寄存器）	Control Registers（控制寄存器）	Counter7（计数器 7）	Flash（闪存）	RAM	
RTC 固定 HW *	不可用	不可用	不可用	不可用	不可用	2738	25	不可用

*使用的电源管理中每秒一脉冲中断

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“RTC_1”分配给指定设计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“RTC”。

在调用读取或修改全局变量的函数时，您应禁用组件的中断。

有关更多信息，请参见此数据表的寄存器部分（如果需要）。

函数	说明
RTC_Start()	启用 RTC 组件
RTC_Stop()	停止 RTC 组件操作
RTC_EnableInt()	启用 RTC 组件的中断
RTC_DisableInt()	禁用 RTC 组件的中断；时间和日期停止运行
RTC_WriteTime()	写入时间和日期值作为当前时间和日期
RTC_ReadTime()	读取当前时间和日期
RTC_WriteSecond()	写入“秒”软件寄存器值
RTC_WriteMinute()	写入“分钟”软件寄存器值
RTC_WriteHour()	写入“小时”软件寄存器值
RTC_WriteDayOfMonth()	写入“某月某日”软件寄存器值
RTC_WriteMonth()	写入“月份”软件寄存器值

函数	说明
RTC_WriteYear()	写入“年份”软件寄存器值
RTC_WriteAlarmSecond()	写入“警报秒”软件寄存器值
RTC_WriteAlarmMinute()	写入“警报分钟”软件寄存器值
RTC_WriteAlarmHour()	写入“警报小时”软件寄存器值
RTC_WriteAlarmDayOfMonth()	写入“警报某月某日”软件寄存器值
RTC_WriteAlarmMonth()	写入“警报月份”软件寄存器值
RTC_WriteAlarmYear()	写入“警报年份”软件寄存器值
RTC_WriteAlarmDayOfWeek()	写入“警报某周某日”软件寄存器值
RTC_WriteAlarmDayOfYear()	写入“警报某年某日”软件寄存器值
RTC_ReadSecond()	读取“秒”软件寄存器值
RTC_ReadMinute()	读取“分钟”软件寄存器值
RTC_ReadHour()	读取“分钟”软件寄存器值
RTC_ReadDayOfMonth()	读取“某月某日”软件寄存器值
RTC_ReadMonth()	读取“月份”软件寄存器值
RTC_ReadYear()	读取“年份”软件寄存器值
RTC_ReadAlarmSecond()	读取“警报秒”软件寄存器值
RTC_ReadAlarmMinute()	读取“警报分钟”软件寄存器值
RTC_ReadAlarmHour()	读取“警报小时”软件寄存器值
RTC_ReadAlarmDayOfMonth()	读取“警报某月某日”软件寄存器值
RTC_ReadAlarmMonth()	读取“警报月份”软件寄存器值
RTC_ReadAlarmYear()	读取“警报年份”软件寄存器值
RTC_ReadAlarmDayOfWeek()	读取“警报某周某日”软件寄存器值
RTC_ReadAlarmDayOfYear()	读取“警报某年某日”软件寄存器值
RTC_WriteAlarmMask()	写入“警报掩码”软件寄存器值，每个时间/日期输入一位
RTC_WriteIntervalMask()	配置从 RTC ISR 中调用的间隔处理程序
RTC_ReadStatus()	读取状态软件寄存器，它具有 DST (DST)、闰年 (LY)、AM/PM (AM_PM) 和警报活动 (AA) 的标志。
RTC_WriteDSTMode()	写入“DST 模式”软件寄存器
RTC_WriteDSTStartHour()	写入“DST 开始小时”软件寄存器

函数	说明
RTC_WriteDSTStartDayOfMonth()	写入“DST 开始某月某日”软件寄存器
RTC_WriteDSTStartMonth()	写入“DST 开始月份”软件寄存器
RTC_WriteDSTStartDayOfWeek()	写入“DST 开始某周某日”软件寄存器
RTC_WriteDSTStartWeek()	写入“DST 开始周”软件寄存器
RTC_WriteDSTStopHour()	写入“DST 停止小时”软件寄存器
RTC_WriteDSTStopDayOfMonth()	写入“DST 停止某月某日”软件寄存器
RTC_WriteDSTStopMonth()	写入“DST 停止月份”软件寄存器
RTC_WriteDSTStopDayOfWeek()	写入“DST 停止某周某日”软件寄存器
RTC_WriteDSTStopWeek()	写入“DST 停止周”软件寄存器
RTC_WriteDSTOffset()	写入“DST 偏移”软件寄存器
RTC_Init()	初始化和恢复随定制器提供的默认配置
RTC_Enable()	启用每秒一脉冲的中断，并启用基于 OPPS 事件的中断生成

全局变量

变量	说明
RTC_initVar	指示 RTC 是否已初始化。变量将初始化为 0，并在第一次调用 RTC_Start() 时设置为 1。这允许第一次调用 RTC_Start() 子程序后组件无需重新初始化便可重新启动。 如需重新初始化组件，可在 RTC_Start() 或 RTC_Enable() 函数前调用 RTC_Init() 函数。
RTC_currentTimeDate	当前的时间和日期值存储在此变量中。
RTC_statusDateTime	此变量具有以下标志：DST、Leap Year、AM/PM 和 Active Alarm 状态。
RTC_intervalCfgMask	此变量用于定义应执行的中断存根。
RTC_alarmCfgTimeDate	警报时间和日期值存储在此变量中..
RTC_alarmCurStatus	此变量用于指示当前活动警报：秒警报处于活动状态，分钟警报处于活动状态，等等。
RTC_alarmCfgMask	此变量用于掩码警报事件：掩码秒警报，掩码分钟警报，等等。
RTC_dstModeType	此变量存储 DST 模式类型的值：“0”- 固定，“1”- 相对。
RTC_dstTimeDateStart	DST 开始的时间和日期值
RTC_dstTimeDateStop	DST 停止的时间和日期值。

变量	说明
RTC_dstStartStatus	DST 开始状态。
RTC_dstStopStatus	DST 停止状态。
RTC_dstOffsetMin	DST 偏移值（单位为分钟）。

void RTC_Start(void)

说明： 启用 RTC 组件。此函数配置计数器，设置中断，执行所有需要的计算，并启动计数器。

参数： None（无）

Return Value（返回值）： None（无）

Side Effects（副作用）： 启用每秒一脉冲信号和中央时轮信号以将器件从低功耗模式（睡眠和备用活动）中唤醒，并保持这两种信号被启用。

void RTC_Stop(void)

说明： 停止 RTC 组件操作。

参数： None（无）

Return Value（返回值）： None（无）

Side Effects（副作用）： 保持启用每秒一脉冲信号和中央时轮，这将把器件从低功耗模式（睡眠和备用活动）中唤醒。

void RTC_EnableInt(void)

说明： 从 RTC 组件中启用中断。

参数： None（无）

Return Value（返回值）： None（无）

Side Effects（副作用）： None（无）

void RTC_DisableInt(void)

说明： 从 RTC 组件中禁用中断。时间和日期停止运行。

参数： None（无）

Return Value（返回值）： None（无）

Side Effects（副作用）： None（无）

RTC_TIME_DATE* timeDate RTC_ReadTime(void)

说明： 读取当前时间和日期。

参数： None（无）

Return Value（返回值）： 指向 RTC_TIME_DATE..

Side Effects（副作用）： None（无）

void RTC_WriteTime(RTC_TIME_DATE * timeDate)

说明： 写入时间和日期值，作为当前时间和日期。仅通过“秒”、“分钟”、“小时”、“月份”、“某月某日”和“年份”。

参数： timeDate：指向 RTC_TIME_DATE 全局结构，其中存储了新的时间和日期值

Return Value（返回值）： None（无）

Side Effects（副作用）： 调用此函数前，应禁用 RTC 组件的中断，并在之后启用，以避免在写入时间和日期时发生 RTC 计数器递增。

void RTC_WriteSecond(uint8 second)

说明： 写入“秒”软件寄存器值。

参数： second（秒）：“秒”值

Return Value（返回值）： None（无）

Side Effects（副作用）： None（无）

void RTC_WriteMinute(uint8 minute)

说明： 写入“分钟”软件寄存器值。

参数： minute（分钟）：“分钟”值

Return Value（返回值）： None（无）

Side Effects（副作用）： None（无）

void RTC_WriteHour(uint8 hour)

说明： 写入“小时”软件寄存器值。

参数： hour（小时）：“小时”值

Return Value（返回值）： None（无）

Side Effects（副作用）： None（无）

void RTC_WriteDayOfMonth(uint8 dayOfMonth)

说明： 写入“某月某日”软件寄存器值。

参数： dayOfMonth：“某月某日”值

Return Value（返回值）： None（无）

Side Effects（副作用）： None（无）

void RTC_WriteMonth(uint8 month)

说明： 写入“月份”软件寄存器值。

参数： month（月份）：“月份”值

Return Value（返回值）： None（无）

Side Effects（副作用）： None（无）

void RTC_WriteYear(uint16 year)

说明： 写入“年份”软件寄存器值。

参数： year（年份）：“年份”值

Return Value（返回值）： None（无）

Side Effects（副作用）： None（无）

void RTC_WriteAlarmSecond(uint8 second)

说明： 写入“警报秒”软件寄存器值。

参数： second（秒）：“警报秒”值

Return Value（返回值）： None（无）

Side Effects（副作用）： None（无）

void RTC_WriteAlarmMinute(uint8 minute)

说明： 写入“警报分钟”软件寄存器值。

参数： minute（分钟）：“警报分钟”值

Return Value（返回值）： None（无）

Side Effects（副作用）： None（无）

void RTC_WriteAlarmHour(uint8 hour)

说明： 写入“警报小时”软件寄存器值。

参数： hour（小时）：“警报小时”值

Return Value（返回值）： None（无）

Side Effects（副作用）： None（无）

void RTC_WriteAlarmDayOfMonth(uint8 dayOfMonth)

说明： 写入“警报某月某日”软件寄存器值。

参数： dayOfMonth：“警报某月某日”值

Return Value（返回值）： None（无）

Side Effects（副作用）： None（无）

void RTC_WriteAlarmMonth(uint8 month)

说明： 写入“警报月份”软件寄存器值。

参数： month（月份）：“警报月份”值

Return Value（返回值）： None（无）

Side Effects（副作用）： None（无）

void RTC_WriteAlarmYear(uint16 year)

说明： 写入“警报年份”软件寄存器值。

参数： year（年份）：“警报年份”值

Return Value（返回值）： None（无）

Side Effects（副作用）： None（无）

void RTC_WriteAlarmDayOfWeek(uint8 dayOfWeek)

说明： 写入“警报某周某日”软件寄存器值。

参数： dayOfWeek（某周某日）：“警报某周某日”值

Return Value（返回值）： None（无）

Side Effects（副作用）： None（无）

void RTC_WriteAlarmDayOfYear(uint16 dayOfYear)

说明： 写入“警报某年某日”软件寄存器值。

参数： dayOfYear：“警报某年某日”值

Return Value（返回值）： None（无）

Side Effects（副作用）： None（无）

uint8 RTC_ReadSecond(void)

说明： 读取“秒”软件寄存器值。

参数： None（无）

Return Value（返回值）： None（无）

Side Effects（副作用）： None（无）

uint8 RTC_ReadMinute(void)

说明： 读取“分钟”软件寄存器值。

参数： None（无）

Return Value（返回值）： 返回当前分钟值。

Side Effects（副作用）： None（无）

uint8 RTC_ReadHour(void)

说明： 读取“分钟”软件寄存器值。

参数： None（无）

Return Value（返回值）： 返回当前小时值。

Side Effects（副作用）： None（无）

uint8 RTC_ReadDayOfMonth(void)

说明： 读取“某月某日”软件寄存器值。

参数： None（无）

Return Value（返回值）： 返回当前某月某日值。

Side Effects（副作用）： None（无）

uint8 RTC_ReadMonth(void)

说明： 此函数读取“月份”软件寄存器值。

参数： None（无）

Return Value（返回值）： 返回当前月份值。

Side Effects（副作用）： None（无）

uint16 RTC_ReadYear(void)

说明： 读取“年份”软件寄存器值。

参数： None（无）

Return Value（返回值）： 返回当前年份值。

Side Effects（副作用）： None（无）

uint8 RTC_ReadAlarmSecond(void)

说明： 读取“警报秒”软件寄存器值。

参数： None（无）

Return Value（返回值）： 返回当前秒的警报值。

Side Effects（副作用）： None（无）

uint8 RTC_ReadAlarmMinute(void)

说明： 读取“警报分钟”软件寄存器值。

参数： None（无）

Return Value（返回值）： 返回当前分钟的警报值。

Side Effects（副作用）： None（无）

uint8 RTC_ReadAlarmHour(void)

说明： 读取“警报小时”软件寄存器值。

参数： None（无）

Return Value（返回值）： 返回当前小时的警报值。

Side Effects（副作用）： None（无）

uint8 RTC_ReadAlarmDayOfMonth(void)

说明： 读取“警报某月某日”软件寄存器值。

参数： None（无）

Return Value（返回值）： 返回当前某月某日的警报值。

Side Effects（副作用）： None（无）

uint8 RTC_ReadAlarmMonth(void)

说明： 读取“警报月份”软件寄存器值。

参数： None（无）

Return Value（返回值）： 返回当前月份的警报值。

Side Effects（副作用）： None（无）

uint16 RTC_ReadAlarmYear(void)

说明： 读取“警报年份”软件寄存器值。

参数： None（无）

Return Value（返回值）： 返回当前年份的警报值。

Side Effects（副作用）： None（无）

uint8 RTC_ReadAlarmDayOfWeek(void)

说明： 读取“警报某周某日”软件寄存器值。

参数： None（无）

Return Value（返回值）： 返回当前某周某日的警报值。

Side Effects（副作用）： None（无）

uint16 RTC_ReadAlarmDayOfYear(void)

说明： 返回“警报某年某日”软件寄存器值。

参数： None（无）

Return Value（返回值）： 返回当前某年某日的警报值。

Side Effects（副作用）： None（无）

void RTC_WriteAlarmMask(uint8 mask)

说明： 写入“警报掩码”软件寄存器，每个时间/日期输入一位。当所有掩码时间/日期值与“警报”值匹配时，警报为真。

参数： mask（掩码）：“警报掩码”软件寄存器值。有关此参数的更多信息，请参见警报掩码寄存器一节。

Return Value（返回值）： None（无）。

Side Effects（副作用）： None（无）。

void RTC_WriteIntervalMask(uint8 mask)

说明： 配置从 RTC ISR 中调用的间隔处理程序。有关如何使用此功能的更多信息，请参见中断服务子程序一节。

参数： mask（掩码）：“间隔掩码”软件寄存器值。有关此参数的更多信息，请参见间隔掩码寄存器一节。

Return Value（返回值）： None（无）。

Side Effects（副作用）： None（无）。

uint8 RTC_ReadStatus(void)

说明： 读取“状态”软件寄存器，它具有 DST (DST)、闰年 (LY)、AM/PM (AM_PM) 和警报活动 (AA) 的标志。

参数： None（无）

Return Value（返回值）： 当前组件的状态，已清除活动警报位。有关返回值的更多信息，请参见Status Register（状态寄存器）一节。

Side Effects（副作用）： 读取警报活动 (AA) 标记后将其清除。

void RTC_WriteDSTMode(uint8 mode)

说明： 写入“DST 模式”软件寄存器，以启用或禁用 DST 更改，并将日期模式设置为固定日期或相对日期。仅在启用了 DST 时才生成此函数。

参数： mode（模式）：“DST 模式”软件寄存器值

Return Value（返回值）： None（无）

Side Effects（副作用）： None（无）

void RTC_WriteDSTStartHour(uint8 hour)

说明： 写入“DST 开始小时”软件寄存器。用于绝对日期输入。仅在启用了 DST 时才生成此函数。

参数： hour（小时）：“DST 开始小时”软件寄存器。

Return Value（返回值）： None（无）

Side Effects（副作用）： None（无）

void RTC_WriteDSTStartDayOfMonth(uint8 dayOfMonth)

说明： 写入“DST 开始某月某日”软件寄存器。用于绝对日期输入。仅在启用了 DST 时才生成此函数。

参数： dayOfMonth：“DST 开始某月某日”软件寄存器值

Return Value（返回值）： None（无）

Side Effects（副作用）： None（无）

void RTC_WriteDSTStartMonth(uint8 month)

说明： 写入“DST 开始月份”软件寄存器。用于绝对日期输入。仅在启用了 DST 时才生成此函数。

参数： month（月份）：“DST 开始月份”软件寄存器值

Return Value（返回值）： None（无）

Side Effects（副作用）： None（无）

void RTC_WriteDSTStartDayOfWeek(uint8 dayOfWeek)

说明： 写入“DST 开始某周某日”软件寄存器。用于相对日期输入。仅在启用了 DST 时才生成此函数。

参数： dayOfWeek（某周某日）：“DST 开始某周某日”软件寄存器值

Return Value（返回值）： None（无）

Side Effects（副作用）： None（无）

void RTC_WriteDSTStartWeek(uint8 week)

说明： 写入“DST 开始周”软件寄存器。用于相对日期输入。仅在启用了 DST 时才生成此函数。

参数： Week（周）：“DST 开始周”软件寄存器值。

Return Value（返回值）： None（无）

Side Effects（副作用）： None（无）

void RTC_WriteDSTStopHour(uint8 hour)

说明： 写入“DST 停止小时”软件寄存器。用于绝对日期输入。仅在启用了 DST 时才生成此函数。

参数： hour（小时）：“DST 停止小时”软件寄存器值

Return Value（返回值）： None（无）

Side Effects（副作用）： None（无）

void RTC_WriteDSTStopDayOfMonth(uint8 dayOfMonth)

说明： 写入“DST 停止某月某日”软件寄存器。用于绝对日期输入。仅在启用了 DST 时才生成此函数。

参数： dayOfMonth：“DST 停止某月某日”软件寄存器值

Return Value（返回值）： None（无）

Side Effects（副作用）： None（无）

void RTC_WriteDSTStopMonth(uint8 month)

说明：	写入“DST 停止月份”软件寄存器。用于绝对日期输入。仅在启用了 DST 时才生成此函数。
参数：	month（月份）：“DST 停止月份”软件寄存器值
Return Value（返回值）：	None（无）
Side Effects（副作用）：	None（无）

void RTC_WriteDSTStopDayOfWeek(uint8 dayOfWeek)

说明：	写入“DST 停止某周某日”软件寄存器。用于相对日期输入。仅在启用了 DST 时才生成此函数。
参数：	dayOfWeek（某周某日）：“DST 停止某周某日”软件寄存器值
Return Value（返回值）：	None（无）
Side Effects（副作用）：	None（无）

void RTC_WriteDSTStopWeek(uint8 week)

说明：	写入“DST 停止周”软件寄存器。用于相对日期输入。仅在启用了 DST 时才生成此函数。
参数：	week：“DST 停止周”软件寄存器值
Return Value”（返回值）：	None（无）
Side Effects"（副作用）：	None（无）

void RTC_WriteDSTOffset(uint8 offset)

说明：	写入“DST 偏移”寄存器。允许 0 到 255 分钟范围内的可配置时间递增或递减。递增发生在 DST 开始时，递减发生在 DST 停止时。仅在启用了 DST 时才生成此函数。
参数：	offset（偏移）：“DST 偏移”软件寄存器值
Return Value（返回值）：	None（无）
Side Effects（副作用）：	None（无）

void RTC_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 RTC_Init()，因为 RTC_Start() API 会调用该函数，这是开始组件操作的首选方法。
参数：	None（无）
Return Value（返回值）：	None（无）
Side Effects（副作用）：	所有寄存器将设置为自定义程序“配置”对话框中的值。

void RTC_Enable(void)

说明：	启用每秒一脉冲的中断，并启用基于 OPPS 事件的中断生成。
参数：	None（无）
Return Value（返回值）：	None（无）
Side Effects（副作用）：	启用每秒一脉冲信号和中央时轮信号以将器件从低功耗模式（睡眠和备用活动）中唤醒，并保持这两种信号被启用。

数据结构

RTC_TIME_DATE

这是一种用于保存当前时间和日期 (RTC_currentTimeDate) 以及警报时间和日期 (RTC_alarmCfgTimeDate) 的数据结构。

```
typedef struct _RTC_TIME_DATE
{
    uint8 Sec;
    uint8 Min;
    uint8 Hour;
    uint8 DayOfWeek;
    uint8 DayOfMonth;
    uint16 DayOfYear;
    uint8 Month;
    uint16 Year;
} volatile RTC_TIME_DATE;
```

RTC_DSTIME

这是一种用于保存夏令时开始和停止（RTC_dstTimeDateStart 和 RTC_dstTimeDateStop）的时间和日期值的数据结构。

```
typedef struct _RTC_DSTIME
{
    uint8 Hour;
    uint8 DayOfWeek;
    uint8 Week;
    uint8 DayOfMonth;
    uint8 Month;
} volatile RTC_DSTIME;
```

常量

有数个定义某周某日、某月某日和月份的常量。写入代码时，使用包头 (.h) 文件中定义的常量。

固件源代码示例

固件源代码示例 PSoC Creator 在“Find Example Projec（查找示例项目）”对话框中提供了大量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**（滤波器选项）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。

中断服务子程序

RTC 组件使用每秒触发一次的单个中断。中断处理程序更新内部日期和时间结构，然后基于用 RTC_WriteIntervalMask() 函数配置的设置，以适当的时间间隔调用特定函数。如果在间隔掩码寄存器中设置了对应的位，将调用以下函数：

- 每秒处理程序 - RTC_EverySecondHandler()
- 每分钟处理程序 - RTC_EveryMinuteHandler()
- 每小时处理程序 - RTC_EveryHourHandler()
- 每天处理程序 - RTC_EveryDayHandler()
- 每周处理程序 - RTC_EveryWeekHandler()
- 每月处理程序 - RTC_EveryMonthHandler()

- 每年处理程序 - RTC_EveryYearHandler()

提供了这些函数的存根子程序，可在其中添加您自己的代码。首次构建项目时，在 *RTC_INT.c* 文件中生成子程序存根。您的代码必须添加在所提供的注释标签之间，如下所示：

```
void RTC_EverySecondHandler( void )
{
    /*  Place your every second handler code here. */
    /* `#START EVERY_SECOND_HANDLER_CODE` */

    /* `#END` */
}
```

在中断处理程序中清除电源管理器中断状态寄存器的所有中断状态位。如果在清除时生成了一个中断，则该位保持不变（这将导致另一个中断）。

功能描述

时间和日期

所有时间和日期寄存器都和软件变量一样可访问。时间和日期是基于计数器组件中的中断事件而更改的。提供了以下变量：

- Sec - 秒：0 到 59
- Min - 分钟：0 到 59
- Hour - 小时（仅适用于 24 小时格式）：0 到 23
- DayOfMonth - 某月某日：1 到 31
- DayOfWeek - 某周某日：1 到 7。数字取决于“StartOfWeek”参数设置。如果 **Start of week**（周开始）设为 **Sunday**（周日），则：1 - 周日，2 - 周一…，7 - 周六
- DayOfYear - 某年某日：1 到 366
- Month - 月份：1 到 12
- Year - 年份：1900 到 2200（实际范围为 1 到 65536）

使用蔡勒公式计算 DayOfWeek。蔡勒公式是针对基于年份、月份和某月某日计算某周某日的整数算法优化的简单算法。它计算闰年和闰世纪。

当您调用 RTC_Start() 函数时，将调用 RTC_SetInitValues() 函数，并执行所有需要的标志和日期计算。这包括所有需要计算的变量：

- DayOfWeek
- DayOfYear
- LY
- AM_PM
- DST

警报函数

警报函数用于秒、分钟、小时、某月某日、某周某日、月份、年份和某年某日。警报设置使用相同的变量名称。您可设置这些警报设置的全部或部分，并配置其中那些设置用于触发警报。

定期中断

中断存根（单独函数中用户代码的位置）可每秒、每分钟、每小时、每天、每周、每月和每年运行一次。如果存根中有代码，将以适当的间隔运行。

夏令时

要启用夏令时功能，选择“配置”对话框上的复选框（请参见此数据表的元件参数一节）。夏令时作为 API 更新时间、日期和持续时间的集合进行实施。如果当前时间和日期与 DST 开始时间和日期匹配，则设置 DST 标志，且时间按设置的持续时间递增。

DST 的开始和停止日期可给定为固定或相对日期。相对日期转换为固定日期，并根据当前时间将其作为警报函数进行检查。固定日期的示例为“3 月 24 日”。相对日期的示例为“5 月的第四个周日”。

相对日期到固定日期的转换是作为单独的函数进行实施的。此函数是在调用了 RTC_Start() 函数之后的第一个小时结束时调用的，它在 RTC_Start() 函数自身中设置了转换标志，指示转换已完成。下一次转换将在下一年进行。

开始和停止时间的 DST 变量如下：

- Hour - 小时：0 到 23（固定和相对）
- DayOfWeek - 周一到周日。数字取决于“StartOfWeek”参数设置。如果 **Start of week**（周开始）设为 **Sunday**（周日），则：1 - 周日，2 - 周一，…，7 - 周六（相对）
- Week - 某月某周：1 到 5（相对）
- DayOfMonth - 某月某日：1 到 31（固定）
- Month - 月份：0 到 12（固定和相对）

寄存器

Status Register（状态寄存器）

状态寄存器是只读寄存器，它包含为各种 RTC 状态位。可使用 RTC_ReadStatus() 函数读取此值。有一些为状态寄存器定义的位字段掩码。#定义在如下已生成的头文件 (.h) 中可用：

- **RTC_STATUS_DST** - 夏令时状态。当当前时间和日期与 DST 时间和日期匹配时，此位变为高电平状态，且时间递增。在 DST 间隔之后此位变为低电平状态，且时间递减。
- **RTC_STATUS_LY** - 闰年状态。当本年为闰年时，此位变为低电平状态。
- **RTC_STATUS_AM_PM** - 当前时间状态。此位在午夜到中午时处于低电平状态，从中午到午夜处于高电平状态。
- **RTC_STATUS_AA** - 警报活动状态（即警报位）。当当前时间和日期与警报时间和日期匹配时，此位处于高电平状态。读取此状态后，此位变为低电平状态。

警报掩码寄存器

警报掩码寄存器是只写寄存器，允许您控制状态寄存器中的警报位。警报位由此寄存器中的掩码位字段 ORing 生成。通过 RTC_WriteAlarmMask() 函数调用写入此寄存器。当写入警报掩码寄存器时，必须使用包头 (.h) 文件中定义的位字段定义。警报掩码寄存器的定义如下：

- **RTC_ALARM_SEC_MASK** - 秒警报掩码允许您将警报秒寄存器与当前秒寄存器进行匹配。通过 RTC_WriteAlarmSecond() 函数调用写入警报秒寄存器，并通过 RTC_ReadAlarmSecond() 函数读取警报秒寄存器。
- **RTC_ALARM_MIN_MASK** - 分钟警报掩码允许您将警报分钟寄存器与当前分钟寄存器进行匹配。通过 RTC_WriteAlarmMinute() 函数调用写入警报分钟寄存器，并通过 RTC_ReadAlarmMinute() 函数读取警报分钟寄存器。
- **RTC_ALARM_HOUR_MASK** - 小时警报掩码允许您将警报小时寄存器与当前小时寄存器进行匹配。通过 RTC_WriteAlarmHour() 函数调用写入警报小时寄存器，并通过 RTC_ReadAlarmHour() 函数读取警报小时寄存器。
- **RTC_ALARM_DAYOFWEEK_MASK** - 某周某日警报掩码允许您将警报某周某日寄存器与当前某周某日寄存器相匹配。通过 RTC_WriteAlarmDayOfWeek() 函数调用写入警报某周某日寄存器，并通过 RTC_ReadAlarmDayOfWeek() 函数读取警报某周某日寄存器。
- **RTC_ALARM_DAYOFMONTH_MASK** - 某月某日警报掩码允许您将警报某月某日寄存器与当前某月某日寄存器相匹配。通过 RTC_WriteAlarmDayOfMonth() 函数调用写入警报某月某日寄存器，并通过 RTC_ReadAlarmDayOfMonth() 函数读取警报某月某日寄存器。

- **RTC_ALARM_DAYOFYEAR_MASK** - 某年某日警报掩码允许您将警报某年某日寄存器与当前某年某日寄存器相匹配。通过 RTC_WriteAlarmDayOfYear() 函数调用写入警报某年某日寄存器，并通过 RTC_ReadAlarmDayOfYear() 函数读取警报某年某日寄存器。
- **RTC_ALARM_MONTH_MASK** - 月份警报掩码允许您将警报月份寄存器与当前月份寄存器相匹配。通过 RTC_WriteAlarmMonth() 函数调用写入警报月份寄存器，并通过 RTC_ReadAlarmMonth() 函数读取警报月份寄存器。
- **RTC_ALARM_YEAR_MASK** - 年份警报掩码允许您将警报年份寄存器与当前年份寄存器相匹配。通过 RTC_WriteAlarmYear() 函数调用写入警报年份寄存器，并通过 RTC_ReadAlarmYear() 函数读取警报年份寄存器。

间隔掩码寄存器

间隔掩码寄存器是只写寄存器，允许您控制 RTC 组件中断存根的处理。中断存根是为每秒、每分钟、每小时、每天、每周、每月和每年提供的。要启用中断存根执行，在此寄存器中设置正确的位。通过 RTC_WriteIntervalMask() 函数调用写入此寄存器。当写入间隔掩码寄存器时，必须使用包头 (.h) 文件中定义的位字段定义。间隔掩码寄存器的定义如下：

- **RTC_INTERVAL_SEC_MASK** - 使用秒间隔掩码，可每秒处理一个中断存根。
- **RTC_INTERVAL_MIN_MASK** - 使用分钟间隔掩码，可每分钟处理一个中断存根。
- **RTC_INTERVAL_HOUR_MASK** - 使用小时间隔掩码，可每小时处理一个中断存根。
- **RTC_INTERVAL_DAY_MASK** - 使用天间隔掩码，可每天处理一个中断存根。
- **RTC_INTERVAL_WEEK_MASK** - 使用周间隔掩码，可每周处理一个中断存根。
- **RTC_INTERVAL_MONTH_MASK** - 使用月间隔掩码，可每月处理一个中断存根。
- **RTC_INTERVAL_YEAR_MASK** - 使用年间隔掩码，可每年处理一个中断存根。

DST 模式寄存器

DST 模式寄存器是只写寄存器，允许您设置夏令时模式和启用 DST 操作。通过 RTC_WriteDSTMode() 函数调用写入此寄存器。当写入 DST 模式寄存器时，必须使用包头 (.h) 文件中定义的位字段定义。DST 模式寄存器的定义如下：

- **RTC_DST_ENABLE** - 启用位控制启用夏令时功能。
- **RTC_DST_FIXDATE** - 定义夏令时功能的时间和日期（开始和停止）的固定格式。例如，固定日期为 3 月 24 日。

- **RTC_DST_RELDATE** - 定义夏令时功能的时间和日期（开始和停止）的相对格式。例如，相对日期为 5 月的第二个周日。

有条件编译信息

RTC API 需要一个条件编译定义以处理夏令时功能。仅当在“Configure”（配置）对话框中启用了此选项时，才可对 DST 功能进行条件编译。该软件决不可直接使用此参数。而应使用定义的符号名称。

- **RTC_DST_FUNC_ENABLE** - 在构建时，分配的夏令时功能启用定义值等于 DstEnable 值（从“Configure”（配置）对话框中）。它在整个 API 中用于编译数据夏令时功能。

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
1.60.a	已更新 RTC_WriteIntervalMask() 函数的说明。已更新“中断服务子程序”部分，以清楚的说明间隔处理程序的配置方式。	中断处理程序使用说明更新。
1.60	解决了当全局变量在代码和 ISR 中使用时可能被编译器优化的问题。	避免可能导致意外结果的优化问题。
	合并区域被添加到 ISR（RTC_INT.c 文件）。	向 RTC 组件提供更多的灵活性。
	已更新代码注释，以进行说明。修复几个拼写错误。	提供有关组件操作的正确信息。
	对数据表进行了少量编辑和更新	
1.50.a	RTC 1.50 版本需要使用 cy_boot 2.x 版本。	cy_boot 组件应更新到 2.x 版本，以便可使用 RTC 1.50 版本。
	已更新“样品固件源代码”一节，以说明已添加了数据表样品项目。	数据表样品项目已添加到 PSoC Creator 中。
	已添加 RTC_Enable() 函数说明。	应在数据表中说明用户可用的各个函数。
	已添加以下函数的参数说明的参考：RTC_WriteAlarmMask()、RTC_WriteIntervalMask() 和 RTC_ReadStatus()。	有关函数使用的说明注释。
	已在 1.50 版中添加了 Keil 重新进入支持。	带 Keil 编译器的 PSoC 3 支持此功能，以便能够从多个控制流调用函数。

版本	更改说明	更改/影响原因
1.50	已更改 API 流 - 已添加 RTC_Init() 函数。	为了符合公司标准，并提供 API 以便无需启动组件即可初始化或恢复组件。
	已重命名一些全局变量，以符合编码标准。已更新函数包头，以说明函数全局参数使用和返回。	符合编码规则。无影响 - 已创建宏以便向后兼容。
	已更新数据表。	已添加“组件更改”一节。已将空“副作用”一节添加到“应用程序编程接口”一节。
	RTCIsLeapYear(uint16 year) 函数已替换为 RTC_LEAP_YEAR 宏。已创建 RTC_SET_ALARM 和 RTC_IS_BIT_SET。RTC_IS_BIT_SET 宏用于更多条件表达式中。	已修复一个不良实践以在 ISR 中调用函数，它们被移出或替换为宏。在线函数应替换为宏。在少数几个地方使用的代码应替换为宏。

© 赛普拉斯半导体公司，2012。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。