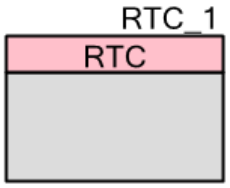
实时时钟（RTC）

2.0



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。灵活的警报功能支持定期报警（例如每小时的第 23 分钟）或单次报警（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

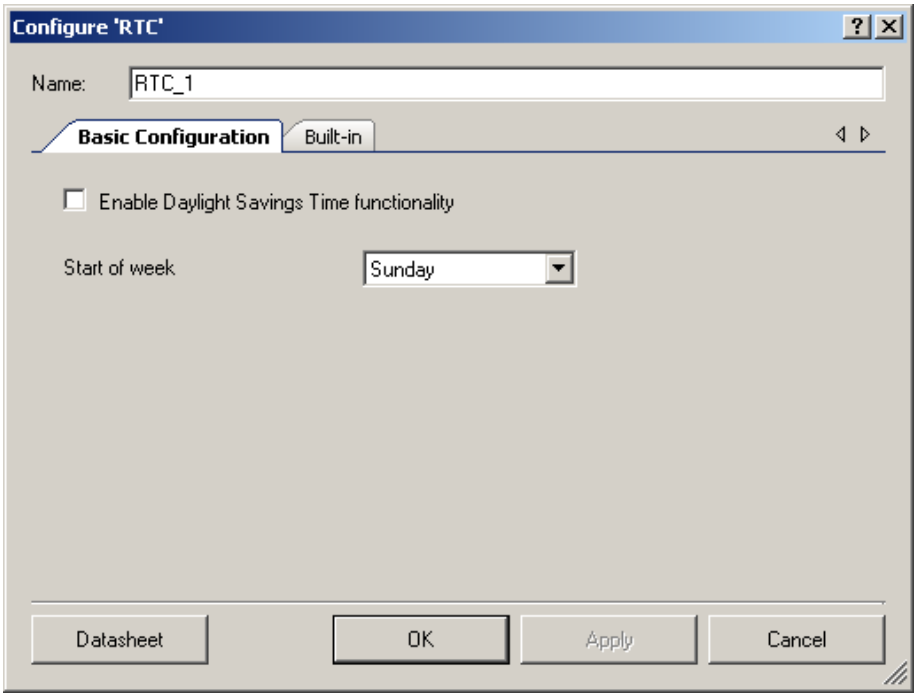
当系统需要当前时间或日期时，使用 RTC 组件。在不需要当前时间和日期，但需要事件的精确时序（精确到秒）时，也可使用 RTC。

输入/输出连接

RTC 组件没有输入或输出连接。

组件参数

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



RTC 组件包含以下选项：

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。默认情况下未选中该项（禁用）。

一周开始

一周开始参数允许您选择周的开始日期。选项包括：**周日**（默认），**周一**，**周二**，**周三**，**周四**，**周五**和**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。

应用编程接口

通过应用编程接口（API）子程序，您可以使用软件对软件进行配置。下面的表格列出并说明了每个函数的接口。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。您可以将其重新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、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
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

函数	说明
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
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

函数	说明
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	此变量包括以下标志：夏时制（DST）、闰年、AM/PM以及活动警报状态。
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模式类型值：‘0’ – 固定和‘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。
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： 无

返回值： 无

其他影响： 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，并保持这两种信号为启用状态。

void RTC_Stop(void)

说明： 停止RTC组件操作。

参数： 无

返回值： 无

其他影响： 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： 无

返回值： 无

其他影响： 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： 无

返回值： 无

其他影响： 无

RTC_TIME_DATE* RTC_ReadTime(void)

说明： 读取当前时间和日期。

参数： 无

返回值： 指向RTC_TIME_DATE。

其他影响： 无

void RTC_WriteTime(const 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

返回值： 无

其他影响： 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”值

返回值： 无

其他影响： 无

void RTC_WriteMinute(uint8 minute)

说明： 写入“分钟”软件寄存器值。

参数： minute：“分钟”值

返回值： 无

其他影响： 无

void RTC_WriteHour(uint8 hour)

说明： 写入“小时”软件寄存器值。

参数： hour：“小时”值

返回值： 无

其他影响： 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

返回值： 无

其他影响： 无

void RTC_WriteMonth(uint8 month)

说明： 写入“月份”软件寄存器值。

参数： month：“月份”值

返回值： 无

其他影响： 无

void RTC_WriteYear(uint16 year)

说明： 写入“年份”软件寄存器值。

参数： year：“年份”值

返回值： 无

其他影响： 无

void RTC_WriteAlarmSecond(uint8 second)

说明： 写入“警报秒”软件寄存器值。

参数： second：“警报秒”值

返回值： 无

其他影响： 无

void RTC_WriteAlarmMinute(uint8 minute)

说明： 写入“警报分钟”软件寄存器值。

参数： minute：“警报分钟”值

返回值： 无

其他影响： 无

void RTC_WriteAlarmHour(uint8 hour)

说明： 写入“警报小时”软件寄存器值。

参数： hour：“警报小时”值

返回值： 无

其他影响： 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

返回值： 无

其他影响： 无

void RTC_WriteAlarmMonth(uint8 month)

说明： 写入“警报月份”软件寄存器值。

参数： month：“警报月份”值

返回值： 无

其他影响： 无

void RTC_WriteAlarmYear(uint16 year)

说明： 写入“警报年份”软件寄存器值。

参数： year：“警报年份”值

返回值： 无

其他影响： 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：“每周警报日”值

返回值： 无

其他影响： 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：“每年警报日”值

返回值： 无

其他影响： 无

uint8 RTC_ReadSecond(void)

说明： 读取“秒”软件寄存器值。

参数： 无

返回值： 无

其他影响： 无

uint8 RTC_ReadMinute(void)

说明： 读取“分钟”软件寄存器值。

参数： 无

返回值： 返回当前分钟值。

其他影响： 无

uint8 RTC_ReadHour(void)

说明： 读取“分钟”软件寄存器值。

参数： 无

返回值： 返回当前小时值。

其他影响： 无

uint8 RTC_ReadDayOfMonth(void)

说明： 读取“某月某日”软件寄存器值。

参数： 无

返回值： 返回当前某月某日值。

其他影响： 无

uint8 RTC_ReadMonth(void)

说明： 此函数读取“月份”软件寄存器值。

参数： 无

返回值： 返回当前月份值。

其他影响： 无

uint16 RTC_ReadYear(void)

说明： 读取“年份”软件寄存器值。

参数： 无

返回值： 返回当前年份值。

其他影响： 无

uint8 RTC_ReadAlarmSecond(void)

说明： 读取“警报秒”软件寄存器值。

参数： 无

返回值： 返回当前秒的警报值。

其他影响： 无

uint8 RTC_ReadAlarmMinute(void)

说明： 读取“警报分钟”软件寄存器值。

参数： 无

返回值： 返回当前分钟的警报值。

其他影响： 无

uint8 RTC_ReadAlarmHour(void)

说明： 读取“警报小时”软件寄存器值。

参数： 无

返回值： 返回当前小时的警报值。

其他影响： 无

uint8 RTC_ReadAlarmDayOfMonth(void)

说明： 读取“警报某月某日”软件寄存器值。

参数： 无

返回值： 返回当前某月某日的警报值。

其他影响： 无

uint8 RTC_ReadAlarmMonth(void)

说明： 读取“警报月份”软件寄存器值。

参数： 无

返回值： 返回当前月份的警报值。

其他影响： 无

uint16 RTC_ReadAlarmYear(void)

说明： 读取“警报年份”软件寄存器值。

参数： 无

返回值： 返回当前年份的警报值。

其他影响： 无

uint8 RTC_ReadAlarmDayOfWeek(void)

说明： 读取“警报某周某日”软件寄存器值。

参数： 无

返回值： 返回当前某周某日的警报值。

其他影响： 无

uint16 RTC_ReadAlarmDayOfYear(void)

说明： 返回“警报某年某日”软件寄存器值。

参数： 无

返回值： 返回当前某年某日的警报值。

其他影响： 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：“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。

返回值： 无。

其他影响： 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：“时间间隔警报掩码”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。

返回值： 无。

其他影响： 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： 无

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，请参见状态寄存器一节。

其他影响： 警报活动（AA）标记被读取后会被清除。

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：“DST模式”软件寄存器值

返回值： 无

其他影响： 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：“DST开始小时”软件寄存器。

返回值： 无

其他影响： 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

返回值： 无

其他影响： 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：“DST开始月份”软件寄存器值

返回值： 无

其他影响： 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：“DST开始星期值”软件寄存器值

返回值： 无

其他影响： 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：“DST开始周”软件寄存器值。

返回值： 无

其他影响： 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：“DST停止小时”软件寄存器。

返回值： 无

其他影响： 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

返回值： 无

其他影响： 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：“DST停止月份”软件寄存器值

返回值： 无

其他影响： 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：“DST停止星期值”软件寄存器值

返回值： 无

其他影响： 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。

返回值： 无

其他影响： 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：“DST偏移”软件寄存器值

返回值： 无

其他影响： 无

void RTC_Init (void)

说明： 根据自定义程序“Configure”对话框的设置，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： 无

返回值： 无

其他影响： 根据自定义程序“Configure”对话框中的内容设置所有寄存器。

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： 无

返回值： 无

其他影响： 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，并保持这两种信号为启用状态。

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
{
    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
{
    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。编写代码时，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。要获取器件特定的示例，请打开器件目录中的对话框或原理图中的器件实例。要查看通用示例，请打开“Start Page”或 **File** 菜单中的对话框。根据要求，可以通过使用对话框中的 **Filter Options** 选项来限定可选的项目列表。

更多有关信息，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：

```
static 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（仅使用 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**周开始** 设为**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：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_Init()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（请参见此数据表的组件参数一节）。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。例如，固定日期为“03 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 — 某周某日：1 到 7。数字取决于“StartOfWeek”参数设置。如果**周开始** 设为**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：1 到 12（固定和相对）

寄存器

状态寄存器

状态寄存器是只读寄存器，它包含各种 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。#定义位于已生成的头文件（.h）中，具体如下：

- **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。通过调用 RTC_WriteAlarmMask()函数写入此寄存器。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（从配置对话框中）。它在整个 API 中用于编译数据夏令时功能。

MISRA 兼容性

本节介绍了 MISRA-C:2004 合规性和本组件的偏差情况。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 — 仅适用于该组件的偏差

本节介绍了有关组件特定偏差的信息。*《系统参考指南》*的“MISRA 符合性”章节中介绍了项目偏差以及有关 MISRA 符合性验证环境的信息。

此 RTC 组件具有如下特定偏差：

MISRA-C:2004规则	规则类别（必须/建议）	规则说明	偏差说明
1.1	必须	此规则规定，代码必须符合C ISO/IEC 9899:1990 标准。	控制结构嵌套（说明）超过15 — 程序未严格达到ISO:C90。 实际上，大多数编译器支持更自由的嵌套限制，所以只有当要求严格操作时才涉及此限制。通过比较，ISO:C99规定模块的限制为127嵌套级。 支持的编译器支持更多的控制结构嵌套。

19.7	建议	使用时，函数应优先于类函数宏。	如果使用的宏具有二进制掩码的语句，将会导致偏差。如果采用以下函数代替宏语句，便可以避免该偏差，不过将降低性能：
			`RTC_LEAP_YEAR(),`
			`RTC_IS_BIT_SET()`

资源

RTC 组件使用电源管理中的每秒一脉冲（one-pulse-per-second）中断。

API 存储器大小

根据编译器、器件、所使用的 API 数量以及组件的配置情况不同，组件的存储器大小也不一样。下表提供了组件配置中所有 API 占用的存储器大小。

通过使用“释放”模式下的相应编译器，可以进行测量操作。在该模式下，存储器的大小得到优化。对于特定的设计，分析编译器生成映射文件后可以确定存储器的使用大小。

配置	PSoC 3 （Keil_PK51）		PSoC 5LP（GCC）	
	闪存 字节	SRAM 字节	闪存 字节	SRAM 字节
默认值	2719	25	2232	29

过时的定义

从器件中删除的定义	适用的定义
RTC_IsLeapYear	RTC_LEAP_YEAR
RTC_Dst	RTC_DSTIME
RTC_TimeDate	RTC_TIME_DATE
RTC_CurTimeDate	RTC_currentTimeDate
RTC_AlarmTimeDate	RTC_alarmCfgTimeDate
RTC_DstMode	RTC_dstModeType
RTC_DstStartTimeDate	RTC_dstTimeDateStart
RTC_DstStopTimeDate	RTC_dstTimeDateStop
RTC_DstOffset	RTC_dstOffsetMin
RTC_DstStatusStart	RTC_dstStartStatus
RTC_DstStatusStop	RTC_dstStopStatus
RTC_AlarmMask	RTC_alarmCfgMask
RTC_AlarmStatus	RTC_alarmCurStatus
RTC_IntervalMask	RTC_intervalCfgMask
RTC_Status	RTC_statusDateTime
RTC_Dim	RTC_daysInMonths
RTC_Seq	RTC_monthTemplate

组件更改

本节列出了各版本的主要组件更改内容。

版本	更改内容	更改原因/影响
2.0	更新了“MISRA合规性”章节。	此组件具有特定的描述偏差。

版本	更改内容	更改原因/影响

	将下面各函数设为静态
	`RTC_EverySecondHandler(),`
	`RTC_EveryMinuteHandler(),`
	`RTC_EveryHourHandler(),`
	`RTC_EveryDayHandler(),`
	`RTC_EveryWeekHandler(),`
	`RTC_EveryMonthHandler(),`
	`RTC_EveryYearHandler().`
		这些函数不是API函数，而只是从RTC ISR调用的子程序。

	更新了RTC_SET_ALARM()宏的执行代码。根据新的执行代码，宏被调用后必须带有分号。	之前，通过if() {}结构来执行RTC_SET_ALARM()宏，所以调用宏后的分号不是必须的。根据MIRSA要求，更新了这些宏的执行代码，即通过do {} while ()结构来执行，并且必须使用宏调用后的分号。
	向RTC ISR处理器中的DST子程序添加了对闰年的2月29日进行处理的事项。	在触发DST的情况下，切换到闰年的2月29日时发生了问题。
	增加了过时的定义表。	对大量定义进行更换，如表中所示。
1.80	更新了内部中断组件。	
	已添加了MISRA符合性章节。	此器件未进行MISRA合规性验证。
1.70	添加了PSoC 5LP器件支持。	
	对所有RTC API添加了CYREENTRANT关键词（当它们包含在.cyre文件中时）。	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。 对于采用了安全方式并且是不可重入的函数，则需要需要该项变更，以消除编译器警告：通过标志或关键节防止并发调用。
	对数据手册进行了少量编辑。	提高可读性。
1.60.a	更新了RTC_WriteIntervalMask()函数的说明。更新了“中断服务子程序”部分，以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。
	更新了代码注释，使其更加明确。修正了拼写错误。	提供了有关组件操作的正确信息。
	对数据手册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。

版本	更改内容	更改原因/影响
	更新了“示例固件源代码”一节，以说明已经添加了数据手册的样例项目。	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。
	为以下各函数的参数说明添加了参考内容：RTC_WriteAlarmMask()、RTC_WriteIntervalMask()以及RTC_ReadStatus()。	对函数使用加以说明。
	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。
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。
	重命名了某些全局变量，使其符合编码标准。更新了函数标题，以说明函数全局参数的使用和返回。	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。已将空白的“其他影响”一节添加到“应用编程接口”一节中。
	RTCIsLeapYear(uint16 year)函数已替换为RTC_LEAP_YEAR宏。已创建了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。单行函数应替换为宏。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，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，WICED，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。