

AT32 ERTC Application Note

前言

实时时钟 (ERTC) 是记录当前时间的计算机时钟。ERTC 不仅应用于个人计算机、服务器和嵌入式系统，几乎所有需要准确计时的电子设备也都会使用。支持 ERTC 的微控制器可用于精密计时器、闹钟、手表、小型电子记事簿以及其它多种设备。

本应用笔记介绍 AT32 微控制器中嵌入式实时时钟 (ERTC) 控制器的特性，以及将 ERTC 用于日历、闹钟、定时唤醒单元、入侵检测、时间戳和校准应用时所需的配置步骤。本应用笔记提供了含有配置信息的示例，有助于您快速准确地针对日历、闹钟、定时唤醒单元、入侵检测、时间戳和校准应用配置 ERTC。

支持型号列表：

支持型号	AT32F415xx
------	------------

目录

1	AT32 高级 ERTC 概述	6
1.1	ERTC 日历	6
1.1.1	初始化日历表	7
1.1.2	ERTC 时钟配置	7
1.2	ERTC 闹钟	9
1.2.1	ERTC 闹钟配置	9
1.2.2	闹钟亚秒配置	11
1.3	ERTC 定时唤醒单元	12
1.3.1	编码自动唤醒单元	13
1.3.2	最大和最小的 ERTC 唤醒周期	13
1.4	ERTC 数字校准功能	15
1.4.1	ERTC 粗略校准	15
1.4.2	ERTC 精密校准	16
1.5	ERTC 同步操作	17
1.6	ERTC 参考时钟检测	18
1.7	时间戳功能	18
1.8	ERTC 入侵检测功能	20
1.8.1	对入侵输入的边沿检测	20
1.8.2	对入侵输入的电平检测	20
1.8.3	激活入侵检测事件时间戳	22
1.9	备份寄存器	22
1.10	ERTC 和低功耗模式	22
1.11	备用功能 ERTC 输出	23
1.11.1	ERTC_CAL 输出	23
1.11.2	ERTC_ALARM 输出	24
1.12	ERTC 安全特性	25
1.12.1	ERTC 寄存器写保护	25
1.12.2	进入/退出初始化模式	25

1.12.3	ERTC 时钟同步	26
2	ERTC 固件驱动程序 API.....	27
2.1	开始使用 ERTC 驱动程序	27
2.2	时间和日期配置	27
2.3	闹钟配置	27
2.4	ERTC 唤醒配置	28
2.5	输出配置	28
2.6	数字校准配置	28
2.7	时间戳配置	28
2.8	入侵配置	28
2.9	备份数据寄存器配置	29
3	应用程序示例	30
4	版本历史	31

表目录

表 1. 初始化日历的步骤.....	7
表 2. 使用不同时钟源获得 1Hz 的日历时钟.....	8
表 3. 配置闹钟的步骤	10
表 4. 闹钟组合	10
表 5. 闹钟亚秒掩码组合	12
表 6. 配置自动唤醒单元的步骤.....	13
表 7. 使用时钟配置 1 时的时基/唤醒单元周期分辨率	14
表 8. 使用时钟配置 2 时的时基/唤醒单元周期分辨率.....	14
表 9. 最小和最大时基/唤醒周期 (ERTCCLK = 32768 时)	15
表 10. 时间戳功能.....	19
表 11. 入侵检测功能 (边沿检测)	20
表 12. 入侵检测功能 (电平检测)	21
表 13. ERTC_CAL 输出频率与时钟源.....	24
表 14. 示例说明	30
表 15. 文档版本历史	31

图目录

图 1. ERTC 日历阶段	6
图 2. AT32 ERTC 的时钟源	8
图 3. 从 ERTC 时钟源到日历单元的预分频器	8
图 4. 闹钟 A 字段	9
图 5. 闹钟亚秒字段	11
图 6. 配置 1 适用的预分频器与时基/唤醒单元的连接	14
图 7. 配置 2 和 3 适用的预分频器与唤醒单元的连接	14
图 8. 粗略校准模块	15
图 9. 精密校准模块	16
图 10. ERTC 移位寄存器	17
图 11. ERTC 参考时钟检测	18
图 12. 时间戳事件过程	19
图 13. 通过边沿检测入侵	20
图 14. 通过电平检测入侵	21
图 15. 使用预充电脉冲时的入侵采样	21
图 16. ERTC_CAL 时钟源	23
图 17. 闹钟标志连接到 ERTC_ALARM 输出	24
图 18. 定时唤醒连接到 ERTC_ALARM 引脚	25

1 AT32 高级 ERTC 概述

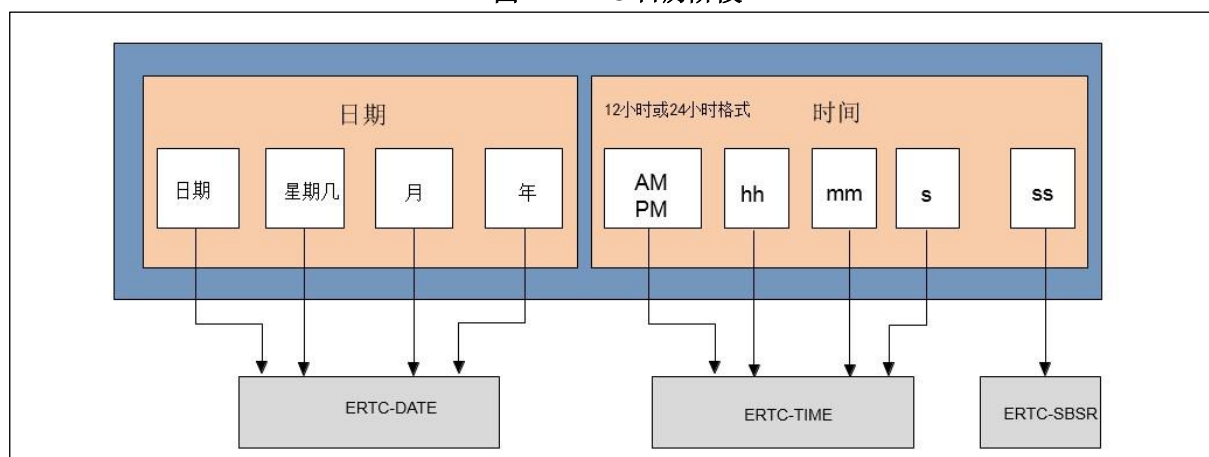
AT32 微控制器中的嵌入式实时时钟（ERTC）是一个独立的 BCD 定时器 / 计数器。ERTC 可用于实现全功能日历、闹钟、定时唤醒单元、数字校准、同步、时间戳和高级入侵检测。

1.1 ERTC 日历

日历用于记录时间（时、分和秒）和日期（日、周、月和年）。AT32 ERTC 日历具有多项功能，可轻松配置和显示下列日历数据字段：

- 含有下列字段的日历：
 - 亚秒（不可编程）
 - 秒
 - 分
 - 时（12 小时或 24 小时格式）
 - 星期
 - 日
 - 月
 - 年
- 二进制十进数（BCD）格式的日历
- 自动管理天数为 28、29（闰年）、30 和 31 的月份
- 夏令时调整可用软件编程

图 1. ERTC 日历阶段



1. ERTC_DATE，ERTC_TIME 是 ERTC 的日期和时间寄存器。
2. 亚秒字段是同步预分频器计数器的值。此字段不可写。

软件日历可以是表示秒数的软件计数器（通常为 32 位长）。软件程序将计数器值转换为小时、分钟、日期、星期、月份和年份。这些数据可以转换成 BCD 格式在标准 LCD 上显示，很适合采用 12 小时格式与 AM/PM 指示符的国家 / 地区。转换程序会占用大量程序存储器空间和 CPU 运行时间，这可能对某些实时应用很不利。

使用 AT32 ERTC 日历时，该功能通过硬件实现，因此不再需要软件转换程序。

AT32 ERTC 日历以 BCD 格式提供。这可以避免二进制转 BCD 的软件转换程序占用大量程序存

存储器空间和加重 CPU 负载而对某些实时应用产生不利影响。

1.1.1 初始化日历表

下表列出了正确配置日历的时间和日期所需的步骤。

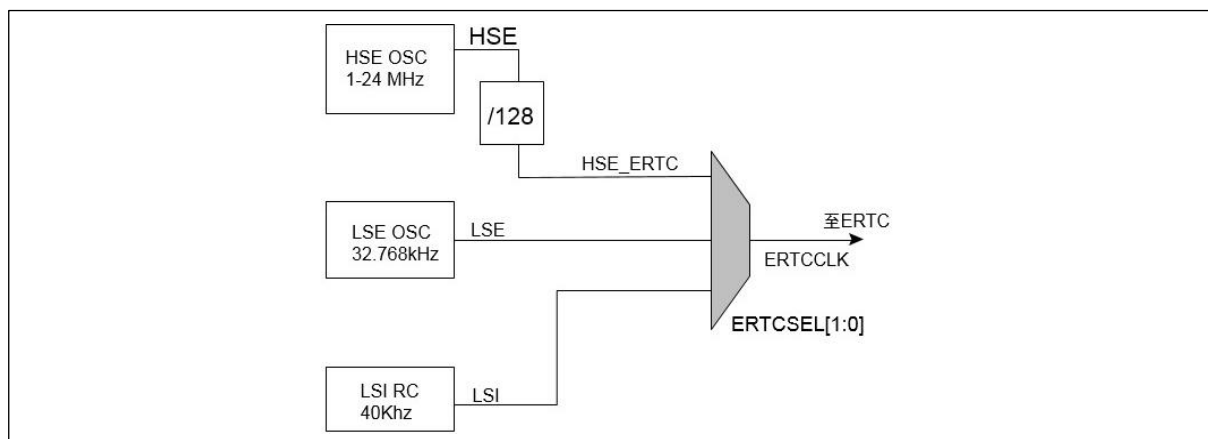
表 1. 初始化日历的步骤

步骤	操作	方法	注释
1	禁止 ERTC 寄存器写保护	将“0xCA”和“0x53”依次写入 ERTC_WPR 寄存器	可修改 ERTC 寄存器
2	进入初始化模式	将 ERTC_STS 寄存器中的 INITM 位置“1”	日历计数器将停止，以允许更新
3	等待确认初始化模式（时钟同步）	轮询 ERTC_STS 中的 INITF 位，直至该位置 1	中等容量器件大约需要 2 个 ERTCCLK 时钟周期
4	根据需要编程预分频器寄存器	ERTC_PSC 寄存器：首先写入同步值，然后写入异步值	默认情况下，ERTC_PSC 预分频器寄存器初始化为在 ERTCCLK = 32768Hz 时为日历单元提供 1Hz 时钟频率
5	在影子寄存器中加载时间和日期值	设置 ERTC_TIME 和 ERTC_DATE 寄存器	
6	配置时间格式（12h 或 24h）	将 ERTC_CTRL 寄存器中的 HFM 位置 1	FMT = 0: 24 小时 / 天格式 FMT = 1: AM/PM 小时格式
7	退出初始化模式	将 ERTC_STS 寄存器中的 INITM 位清零	自动加载当前日历计数器 在 4 个 ERTCCLK 时钟周期后重新开始计数
8	使能 ERTC 寄存器写保护	将“0xFF”写入 ERTC_WPR 寄存器	无法再修改 ERTC 寄存器

1.1.2 ERTC 时钟配置

ERTC 日历可通过三个时钟源 LSE、LSI 或 HSE 驱动（见图 2）。

图 2. AT32 ERTC 的时钟源



注：ERTCSEL[1:0]位是 RCC 控制/状态寄存器 (RCC_BDC) [9:8]位
 如何调整 ERTC 日历时钟

ERTC 具有多个预分频器，无论使用哪个时钟源都可以为日历单元提供 1 Hz 时钟。

图 3. 从 ERTC 时钟源到日历单元的预分频器



注：同步预分频器的长度取决于产品，本文以 13 位表示。

CK_SPRE 的计算公式为：

$$CK_SPRE = \frac{ERTCCLK}{(PRDIV_A+1)*(PRDIV_S+1)}$$

其中：

- ERTCCLK 可以是任意一个可选的时钟源：HSE_ERTC、LSE 或 LSI
- PRDIV_A 可以是 1、2、3... 或 127
- PRDIV_S 可以是 0、1、2... 或 8191

表 2 列出了几种获得 1 Hz 日历时钟 (ck_spre) 的方法。

表 2. 使用不同时钟源获得 1Hz 的日历时钟

ERTCCLK 时钟源	预分频器		CK_SPRE
	PRDIV_A[6:0]	PRDIV_S[12:0]	
HSE_ERTC = 62.5KHz	124 (div 125)	499 (div 500)	1 Hz
LSE = 32.768 kHz	127 (div 128)	255 (div 256)	1 Hz
LSI = 40 kHz ⁽¹⁾	127 (div 128)	319 (div 320)	1 Hz

(1) 对于 AT32，LSI = 40 KHz，但 LSI 精度不适合日历应用。

1.2 ERTC 闹钟

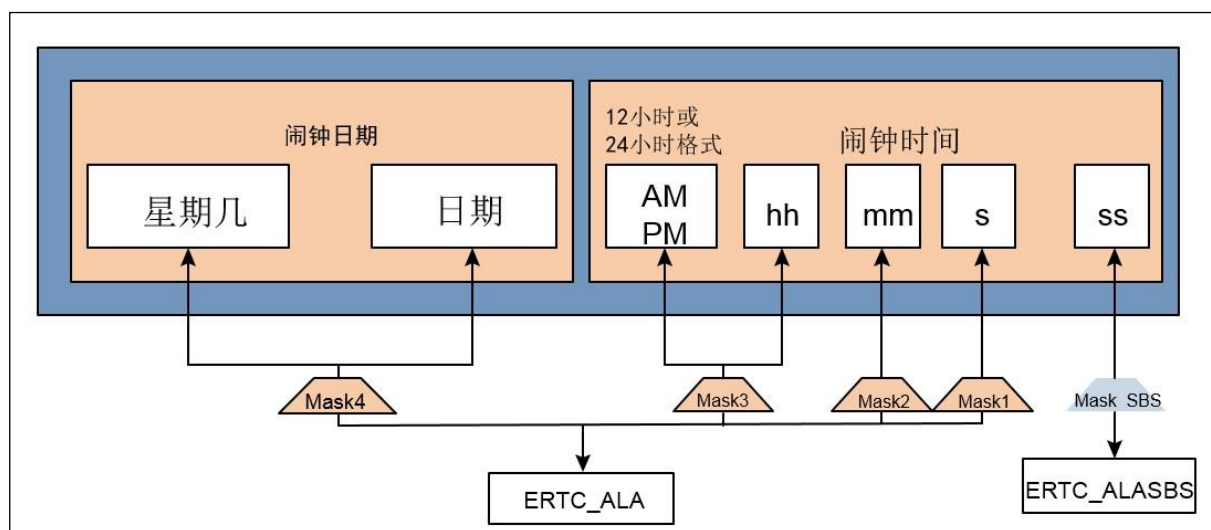
1.2.1 ERTC 闹钟配置

AT32 ERTC 内嵌两个相似的闹钟 —— 闹钟 A 和闹钟 B。闹钟可以在用户编程的给定时间和 / 或日期触发。

AT32 ERTC 提供了多种闹钟设置组合，并且具有多种功能，以方便用户轻松配置和显示这些闹钟设置。每个闹钟单元均具有以下功能：

- 支持对闹钟自由编程：亚秒（稍后讨论）、秒、分钟、小时和日期字段可以单独选择或屏蔽，以实现多种闹钟组合。
- 能够在发生闹钟事件时使器件退出低功耗模式。
- 可将闹钟事件发送到极性可配置的特定输出引脚。
- 具有专用的闹钟标志和中断。

图 4. 闹钟 A 字段



1. ERTC_ALA 是一个 ERTC 寄存器。其中的字段也可用于 ERTC_ALB 寄存器。
2. RT_ARMASSR 是一个 ERTC 寄存器。其中的字段也可用于 ERTC_ALB 寄存器。
3. Maskx 是 ERTC_ALA 寄存器中的位，可使能 / 禁止用于闹钟 A 和日历比较的 ERTC_ALARM 字段。有关详细信息，请参见表 4。
4. Mask SBS 是 ERTC_ALASBS 寄存器中的位。

闹钟包含一个与 ERTC 时间计数器等长的寄存器。当 ERTC 时间计数器达到闹钟寄存器中编程的值时，将设置一个标记，以指示发生了闹钟事件。

可以通过硬件配置 AT32 ERTC 闹钟来生成不同类型的闹钟。有关详细信息，请参见表 4。

编程闹钟

表 3 介绍了配置闹钟 A 所需的步骤。

表 3. 配置闹钟的步骤

步骤	操作	方法	注释
1	禁止 ERTC 寄存器写保护	将“0xCA”和“0x53”依次写入 ERTC_WPR 寄存器	可修改 ERTC 寄存器
2	禁止闹钟 A	将 ERTC_CTRL 寄存器中 ALAE ⁽¹⁾ 位清零。	
3	检查是否可以访问 ERTC_ALA 寄存器	轮询 ERTC_STS 中的 ALAWF ⁽²⁾ 位，直至该位置 1。	大约需要两个 ERTCCLK 时钟周期（时钟同步）。
4	配置闹钟	配置 ERTC_ALA ⁽³⁾ 寄存器。	闹钟小时格式必须与 ERTC_ALA 中的 ERTC 日历相同 ⁽⁴⁾ 。
5	重新使能闹钟 A	将 ERTC_CTRL 寄存器中的 ALAE ⁽⁵⁾ 位置 1。	
6	使能 ERTC 寄存器写保护	将“0xFF”写入 ERTC_WPR 寄存器	无法再修改 ERTC 寄存器

(1) 对于闹钟 B，为 ALBE 位。

(2) 对于闹钟 B，为 ALBWF 位。

(3) 对于闹钟 B，为 ERTC_ALB 寄存器。

(4) 例如，如果闹钟配置为在 3:00:00 PM 触发，即使日历时间为 15:00:00，也不会发生闹钟事件，这是因为 ERTC 日历为 24 小时格式，而闹钟为 12 小时格式。

(5) 对于闹钟 B，为 ALRBE 位。

(6) 仅当禁止相应的 ERTC 闹钟时，或在 ERTC 初始化模式下，才能对 ERTC 闹钟寄存器进行写操作。

使用 MSKx 位配置闹钟行为

对于闹钟 A，可以使用 ERTC_ALA 寄存器的 MSKx 位（x = 1、2、3、4）配置闹钟行为（对于闹钟 B，则使用 ERTC_ALB 寄存器）。

表 4 显示了所有可能的闹钟设置。例如，要将闹钟时间配置为星期一的 23:15:07（假定 WKSEL = 1），MASKx 位必须设置为 0000b。若 WDSEL = 0，MASKx 位也必须设置为 0000b，只是“闹钟掩码”字段与日期比较而不是与星期几比较。

表 4. 闹钟组合

MASK4	MASK3	MASK2	MASK1	闹钟行为
0	0	0	0	闹钟比较使用所有字段 闹钟事件发生在每星期一的 23:15:07。
0	0	0	1	闹钟比较忽略秒 闹钟事件发生在每星期一 23:15 的每一秒。
0	0	1	0	闹钟比较忽略分钟 闹钟事件发生在每星期一 23:XX 的每分钟第 7 秒。
0	0	1	1	闹钟比较忽略分钟和秒

0	1	0	0	闹钟比较忽略小时
0	1	0	1	闹钟比较忽略小时和秒
0	1	1	0	闹钟比较忽略小时和分钟
0	1	1	1	闹钟比较忽略小时、分钟和秒 每星期一全天的每一秒均设置闹钟。
1	0	0	0	闹钟比较忽略星期几（或日期，若已选择） 闹钟事件发生在每天的 23:15:07。
1	0	0	1	闹钟比较忽略星期几和秒
1	0	1	0	闹钟比较忽略星期几和分钟
1	0	1	1	闹钟比较忽略星期几、分钟和秒
1	1	0	0	闹钟比较忽略星期几和小时
1	1	0	1	闹钟比较忽略星期几、小时和秒
1	1	1	0	闹钟比较忽略星期几、小时和分钟
1	1	1	1	每秒都会发生闹钟事件

注：如果选择秒字段（*ERTC_ALA* 或 *ERTC_ALB* 中的 *MASK1* 位复位），则 *ERTC_PSC* 寄存器中设置的同步预分频器分频系数 *PRDIV_S* 必须至少为 3，才能确保正确的闹钟行为。

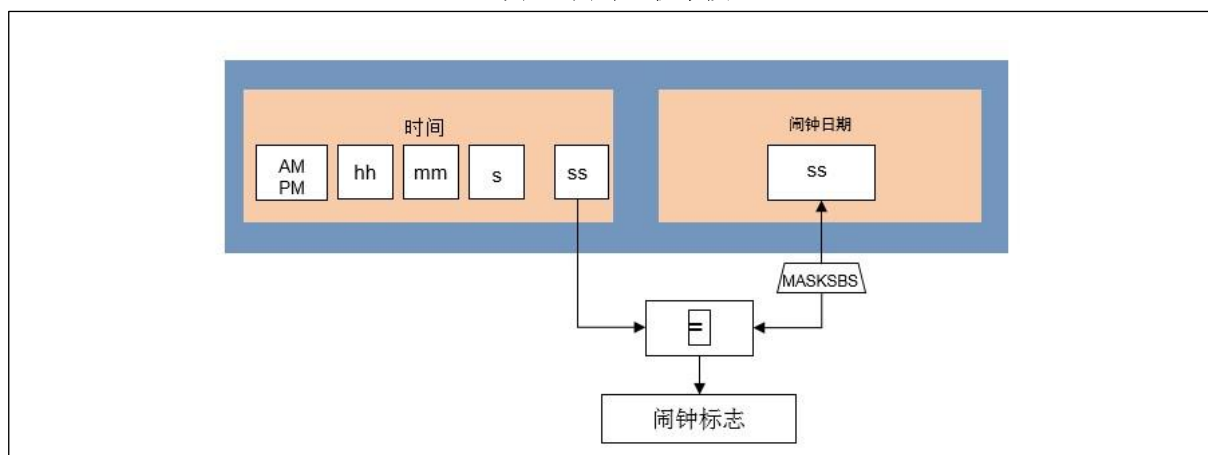
1.2.2 闹钟亚秒配置

AT32 ERTC 单元提供两个相似的可编程闹钟 - 亚秒 A 和 B。它们可生成高分辨率的闹钟（用于秒分频）。

闹钟亚秒寄存器中编程的值与日历单元中亚秒字段的内容进行比较。

亚秒字段计数器从同步预分频器中配置的值开始递减计数至零，然后重载 *ERTC_SPRE* 寄存器中的值。

图 5. 闹钟亚秒字段



注：*MASKSBS* 是亚秒闹钟的最高有效位。它们与同步预分频器寄存器进行比较。

可以使用闹钟亚秒寄存器中的 *MASKSBS* 位配置闹钟亚秒。表 5：闹钟亚秒掩码组合 给出了掩码寄存器的可能配置并提供了采用以下设置时的示例结果：

- 选择 LSE 作为 ERTC 时钟源（例如 LSE = 32768 Hz）。
- 将异步预分频器设置为 127。

- 将同步预分频器设置为 255（日历时钟等于 1Hz）。
- 将闹钟 A 的亚秒设置为 255（在 SS[14:0] 字段中写入 255）。

表 5. 闹钟亚秒掩码组合

MASKSS	闹钟 A 亚秒行为	结果示例
0	不对闹钟的亚秒进行比较。当秒单元递增 1 时，激活闹钟。	每 1 秒激活一次闹钟
1	仅 AlarmA_SBS[0] 位与 ERTC 亚秒寄存器 ERTC_SBSR 进行比较	每 (1/128) s 激活一次闹钟
2	仅 AlarmA_SBS[1:0] 位与 ERTC 亚秒寄存器 ERTC_SBSR 进行比较	每 (1/64) s 激活一次闹钟
3	仅 AlarmA_SBS[2:0] 位与 ERTC 亚秒寄存器 ERTC_SBSR 进行比较	每 (1/32) s 激活一次闹钟
4	仅 AlarmA_SBS[3:0] 位与 ERTC 亚秒寄存器 ERTC_SBSR 进行比较	每 (1/16) s 激活一次闹钟
5	仅 AlarmA_SBS[4:0] 位与 ERTC 亚秒寄存器 ERTC_SBSR 进行比较	每 125ms 激活一次闹钟
6	仅 AlarmA_SBS[5:0] 位与 ERTC 亚秒寄存器 ERTC_SBSR 进行比较	每 250ms 激活一次闹钟
7	仅 AlarmA_SBS[6:0] 位与 ERTC 亚秒寄存器 ERTC_SBSR 进行比较	每 500ms 激活一次闹钟
8	仅 AlarmA_SBS[7:0] 位与 ERTC 亚秒寄存器 ERTC_SBSR 进行比较	每 1 s 激活一次闹钟
9	仅 AlarmA_SBS[8:0] 位与 ERTC 亚秒寄存器 ERTC_SBSR 进行比较	每 1 s 激活一次闹钟
10	仅 AlarmA_SBS[9:0] 位与 ERTC 亚秒寄存器 ERTC_SBSR 进行比较	每 1 s 激活一次闹钟
11	仅 AlarmA_SBS[10:0] 位与 ERTC 亚秒寄存器 ERTC_SBSR 进行比较	每 1 s 激活一次闹钟
12	仅 AlarmA_SBS[11:0] 位与 ERTC 亚秒寄存器 ERTC_SBSR 进行比较	每 1 s 激活一次闹钟
13	仅 AlarmA_SBS[12:0] 位与 ERTC 亚秒寄存器 ERTC_SBSR 进行比较	每 1 s 激活一次闹钟
14	仅 AlarmA_SBS[13:0] 位与 ERTC 亚秒寄存器 ERTC_SBSR 进行比较	每 1 s 激活一次闹钟
15	仅 AlarmA_SBS[14:0] 位与 ERTC 亚秒寄存器 ERTC_SBSR 进行比较	每 1 s 激活一次闹钟

注：不比较亚秒寄存器位中的溢出位（15、16 和 17）。

1.3 ERTC 定时唤醒单元

AT32 提供了多种低功耗模式以降低功耗。

AT32 具有一个定时时基和唤醒单元，可在 AT32 以低功耗模式工作时唤醒系统。该单元是一个可编程的递减计数自动重载定时器。此计数器计到零后，将生成一个标记和一个中断（如果使能）。

唤醒单元具有以下特性：

- 可编程的递减计数自动重载定时器。
- 特定的标记和中断，能够将器件从低功耗模式中唤醒。
- 唤醒备用功能输出，可连接到极性可配置的 ERTC_ALARM 输出（用于闹钟 A、闹钟 B 或唤醒事件的独特焊盘）。

- 一整套用于选择所需等待周期的预分频器。

1.3.1 编码自动唤醒单元

表 6 介绍了配置自动唤醒单元所需的步骤。

表 6. 配置自动唤醒单元的步骤

步骤	操作	方法	注释
1	禁止 ERTC 寄存器写保护	将“0xCA”和“0x53”依次写入 ERTC_WPR 寄存器	可修改 ERTC 寄存器
2	禁止唤醒定时器	将 ERTC_CTRL 寄存器中的 WATE 位清零	
3	确保允许访问唤醒自动重载计数器和 WACKSEL[2:0] 位	轮询 ERTC_STS 中的 WATWF 位，直至该位置 1	大约需要 2 个 ERTCCLK 时钟周期
4	将值编程到唤醒定时器中	将 ERTC_WATR 寄存器中的 WAREV[15:0] 置 1	请参见第 1.3.2 节：最大和最小 ERTC 唤醒周期
5	选择所需时钟源	编程 ERTC_CTRL 寄存器中的 WACKSEL[2:0] 位	
6	重新使能唤醒定时器	将 ERTC_CTRL 寄存器中的 WATE 位置 1	唤醒定时器重新开始递减计数
7	使能 ERTC 寄存器写保护	将“0xFF”写入 ERTC_WPR 寄存器	无法再修改 ERTC 寄存器

1.3.2 最大和最小的 ERTC 唤醒周期

唤醒单元时钟通过 ERTC_CTRL 寄存器的 WACKSEL[2:0] 位进行配置。有如下三种可能的配置：

- 配置 1: WACKSEL[2:0] = 0xb, 表示短唤醒周期
(参见时钟配置 1 的定时时基/ 唤醒配置)
- 配置 2: WACKSEL[2:0] = 10xb, 表示中等时长的唤醒周期
(参见时钟配置 2 的定时时基/ 唤醒配置)
- 配置 3: WACKSEL[2:0] = 11xb, 表示长唤醒周期
(参见时钟配置 3 的定时时基/ 唤醒配置)

时钟配置 1 的定时时基 / 唤醒配置

图 6 显示了预分频器与时基 / 唤醒单元的连接，表 7 给出了与配置 1 对应的时基/唤醒时钟分辨率。

预分频比取决于唤醒时钟选择：

- WACKSEL[2:0] = 000: 选择 ERTCCLK/16 时钟
- WACKSEL[2:0] = 001: 选择 ERTCCLK/8 时钟
- WACKSEL[2:0] = 010: 选择 ERTCCLK/4 时钟

- WACKSEL[2:0] = 011: 选择 ERTCCLK/2 时钟

图 6. 配置 1 适用的预分频器与时基/唤醒单元的连接

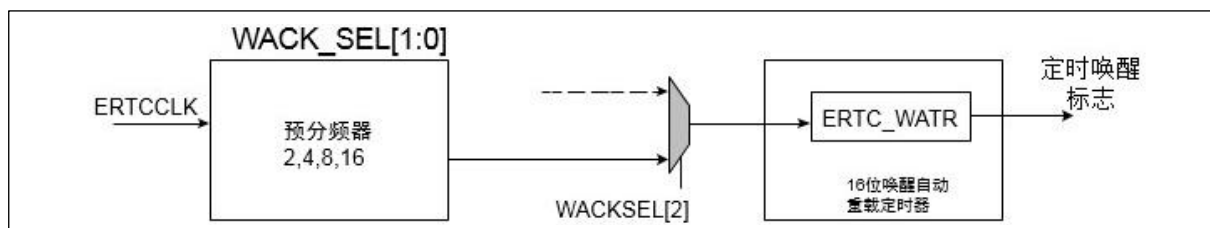


表 7. 使用时钟配置 1 时的时基/唤醒单元周期分辨率

时钟源	唤醒周期分辨率	
	WACKSEL[2:0] = 000b (div16)	WACKSEL[2:0] = 011b (div2)
LSE = 32.768 Hz	488.28 μ s	61.035 μ s

ERTCCLK = 32768 Hz 时，最小时基 / 唤醒分辨率为 61.035 μ s，最大时基 / 唤醒分辨率为 488.28 μ s。因此：

- 最小时基 / 唤醒周期为 $(0x0001 + 1) \times 61.035 \mu s = 122.07 \mu s$
WACKSEL[2:0]=011b(fERTCCLK/2) 时，时基 / 唤醒定时计数器 WATR[15:0] 不能设置为 0x0000，因为此配置已被禁用。有关详细信息，请参见 AT32 参考手册。
- 最大时基 / 唤醒周期为 $(0xFFFF + 1) \times 488.28 \mu s = 2 s$ 。

时钟配置 2 的定时时基 / 唤醒配置

图 7 显示了预分频器与时基 / 唤醒单元的连接，表 8 给出了与配置 2 对应的时基 / 唤醒时钟分辨率。

图 7. 配置 2 和 3 适用的预分频器与唤醒单元的连接

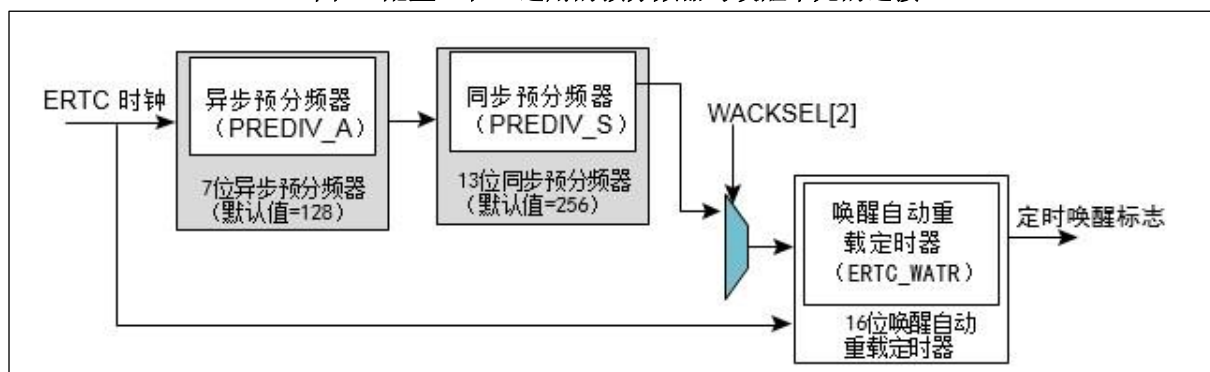


表 8. 使用时钟配置 2 时的时基/唤醒单元周期分辨率

时钟源	唤醒周期分辨率	
	PRDIV_A[6:0] = div128 PRDIV_S[12:0] = div8192	PRDIV_A[6:0] = div2 PRDIV_S[12:0] = div1
LSE = 32 768 Hz	32 s	61.035 μ s

ERTCCLK= 32768 Hz 时，配置 2 的最小分辨率为 61.035 μ s，最大分辨率为 32s。

因此：

- 最小时基 / 唤醒周期为 $(0x0000 + 1) \times 61.035 \mu s = 122.07 \mu s$ 。
- 最大时基 / 唤醒周期为 $(0xFFFF + 1) \times 32 s = 131072 s$ （超过 36 小时）。

时钟配置 3 的定时时基 / 唤醒配置

此配置的分辨率与配置 2 相同。但是，时基 / 唤醒计数器会从 0x1FFFF 开始递减计数至 0x00000，而不是像配置 2 那样从 0xFFFF 递减计数至 0x0000。

ERTCCLK= 32768 时，

- 最小时基 / 唤醒周期为：
 $(0 \times 10000 + 1) \times 61.035 \mu s = 250.06 ms$
- 最大时基 / 唤醒周期为：
 $(0x1FFFF + 1) \times 32 s = 4194304 s$ （超过 48 天）。

时基 / 唤醒周期极值汇总

表 9 中按配置列出了 ERTCCLK= 32768 Hz 时的最小周期值和最大周期值

表 9. 最小和最大时基/唤醒周期（ERTCCLK = 32768 时）

配置	最小周期	最大周期
1	122.07 μs	2s
2	122.07 μs	超过 36 小时
3	250.06 ms	超过 48 天

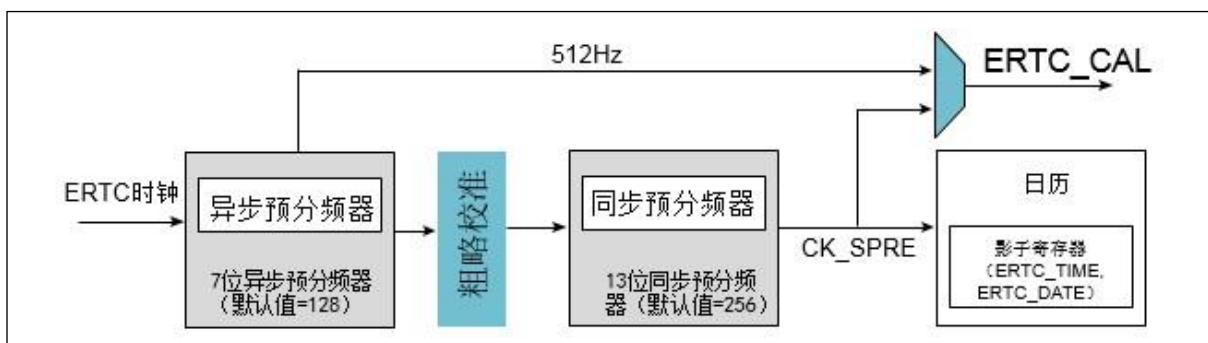
1.4 ERTC 数字校准功能

1.4.1 ERTC 粗略校准

数字粗略校准功能通过在异步预分频器 (ck_apre) 的输出端增加（正校准）或减少（负校准）时钟周期来实现晶振误差补偿。

可采用约 2 ppm 的分辨率执行负校准，采用约 4 ppm 的分辨率执行正校准。最大校准范围为 -63 ppm 到 126 ppm。

图 8. 粗略校准模块



可以使用 ERTC_CAL 计算时钟偏差，然后更新校准模块。由于 512Hz 输出在粗略校准模块之前，因此无法检查校准结果。但由于在粗略校准模块之后有 1Hz CK_Spre 输出，因此可以检查校准结果。

注：只能在初始化期间更改校准设置。

整个校准周期将持续 64 分钟。

校准过程在校准周期的前数分钟内完成（0 到 62 分钟，具体取决于配置）。

建议仅对静态修正使用粗略校准。根据注 1 中列出的几点，更改校准设置会产生误差：

- 进入初始化模式会停止日历并重新初始化预分频器；
- 校准变化率必须远小于校准窗口大小，以尽可能减小因最终精度改变而导致的误差影响；因此，粗略校准不适合动态校准（例如对由于外部温度变化产生的石英晶振变化进行补偿）。

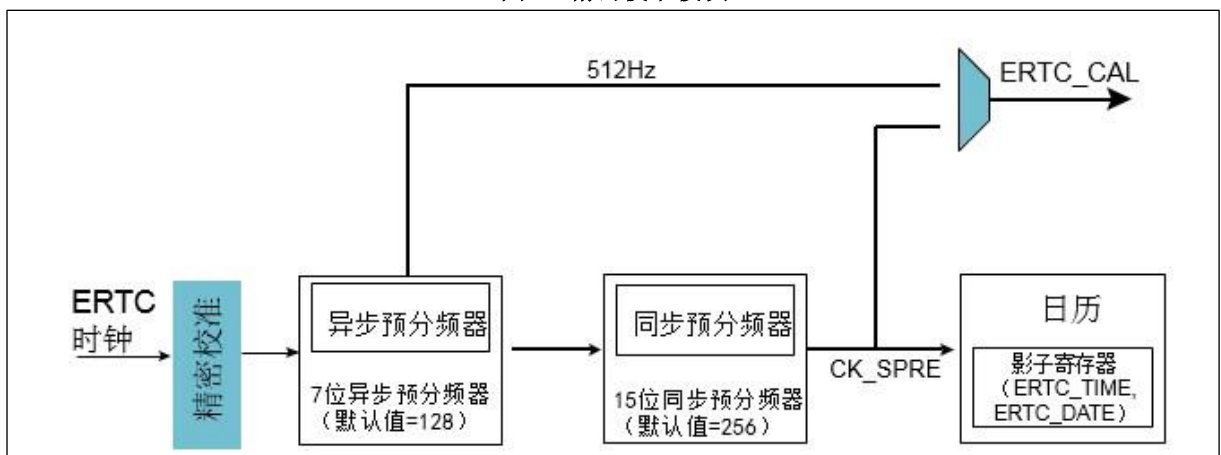
参考时钟校准和粗略校准不可同时使用。

注：如果 $PRDIV_A < 6$ ，数字粗略校准可能无法正常工作。

1.4.2 ERTC 精密校准

通过增加或减少 ERTCCLK 脉冲等一系列细微调整可修正 ERTC 时钟频率。可采用约 0.954 ppm 的分辨率以及 -487.1 ppm 到 +488.5 ppm 的校准范围来校准 ERTC 时钟。此数字精密校准功能用于补偿因温度、晶振老化引起的晶振误差。

图 9. 精密校准模块



可以使用 ERTC_CAL 计算时钟偏差，然后更新校准模块。可以使用 ERTC_CAL 信号的校准输出 512 Hz 或 1 Hz（具体取决于产品）检查校准结果。

精密校准通过在可配置窗口（8 s、16 s 或 32 s）中减少或增加适当分布的 N（可配置）个 32 kHz 脉冲来实现。

使用 ERTC_CCR 寄存器中的 CALAD 和 CALM 定义减少或增加的脉冲数。

默认情况下，校准窗口为 32 秒。将 ERTC_CCR 寄存器中的 CAL8 位或 CAL16 位置 1 可使校准窗口减至 8 秒或 16 秒。

例1：当 CALAD = 0 且校准窗口为 32 秒时，CALM[0] 置 1 将导致每 32 秒去除 1 个脉冲。

例2：当 CALAD = 0 且校准窗口为 32 秒时，CALM[2] 置 1 将导致每 32 秒去除 4 个脉冲。

注：可以同时使用 CALM 和 CALAD，在这种情况下，32 秒（校准窗口）内可增加的偏移量范围为-511 到 +512 个脉冲。

异步预分频器小于 3 时, CALAD 不能设置为 1。

若输入频率 (FERTCCLK) 已知, 可通过以下公式计算有效校准频率 (FCAL):

$$FCAL = FERTCCLK \times [1 + (CALAD \times 512 - CALM) / (220 + CALM - CALAD \times 512)]$$
。精密校准可以实时执行, 从而在温度发生变化或检测到其它情况时可相应进行调整。

检查精密校准

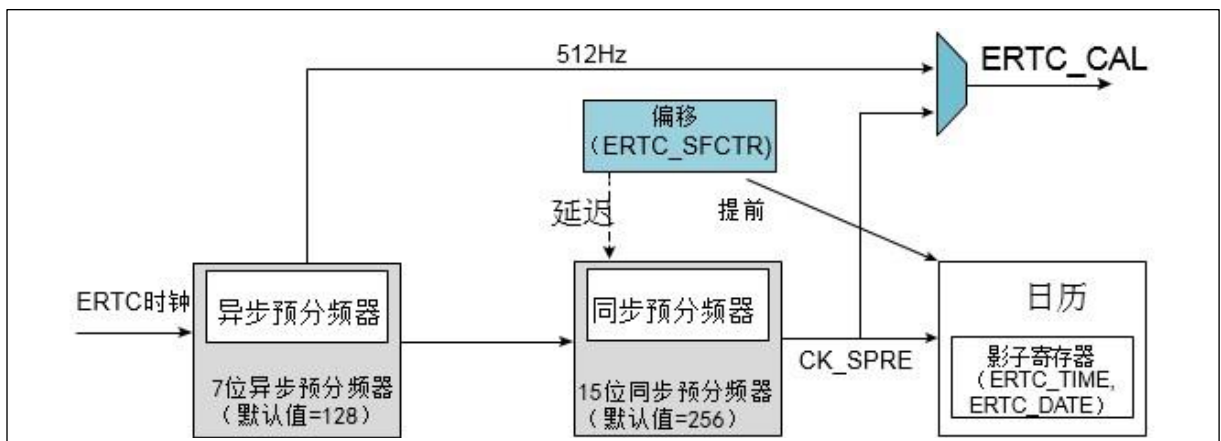
可通过以下方式检查精密校准对日历时钟 (ERTC 时钟) 的影响:

- 采用 ERTC_CAL (512 Hz 或 1 Hz) 的校准。
- 采用亚秒闹钟的校准。
- 采用唤醒定时器的校准。

1.5 ERTC 同步操作

通过 ERTC 移位特性, 可以将 ERTC 日历与更精确的时钟 (“远程时钟”) 同步。在读取 ERTC 亚秒字段后, 即可计算远程时钟的时间与 ERTC 之间的精准偏差。可以使用移位寄存器控制进行微调来消除此偏差, 从而调整 ERTC。

图 10. ERTC 移位寄存器



不能使用 ERTC_CAL 输出检测“同步”移位功能, 因为移位操作对 ERTC 时钟没有影响, 只是在日历计数器上增加或减去一部分值。

修正 ERTC 日历时间

如果 ERTC 时钟比远程时钟快 n 分之一秒, 则必须向 SUBSBS 中写入偏差值, 该值将增加到同步预分频器的计数器中。由于该计数器递减计数, 此操作可有效地从时钟减去 (延迟) 以下时间:

延迟 (秒) = $SUBSBS / (PRDIV_S + 1)$

如果 ERTC 时钟比远程时钟慢 n 分之一秒, 则可将 ADD1S 函数与 SUBSBS 结合使用, 有效地将偏差值增加到时钟, 使时钟提前以下时间:

提前 (秒) = $(1 - (SUBSBS / (PRDIV_S + 1)))$ 。

1.6 ERTC 参考时钟检测

参考时钟（50 Hz 或 60 Hz 时）的精度应高于 32.768 kHz LSE 时钟。因此 ERTC 提供了可用来补偿不精确日历频率（1 Hz）的参考时钟输入（ERTC_50Hz 引脚）。

应在输入悬空模式下对 ERTC_50Hz 引脚进行配置。

该机制可使日历像参考时钟一样精确。

将 ERTC_CTRL 寄存器的 RFCKON 位置 1 即可使能参考时钟检测。

使能参考时钟检测后，必须将 PRDIV_A 和 PRDIV_S 设置为各自的默认值：PRDIV_A = 0x007F，PREVID_S = 0x00FF。

使能参考时钟检测后，每个 1 Hz 时钟边沿都与最近的参考时钟边沿进行比较（如果在给定的时间窗口内发现一个边沿）。在大多数情况下，两个时钟边沿恰好对齐。当 1 Hz 时钟由于 LSE 时钟不精确而发生偏离时，ERTC 会稍微偏移 1 Hz 时钟，以便后续的 1 Hz 时钟边沿能够对齐。更新窗口为 3 个 ck_calib 周期（ck_calib 是粗略校准模块的输出）。

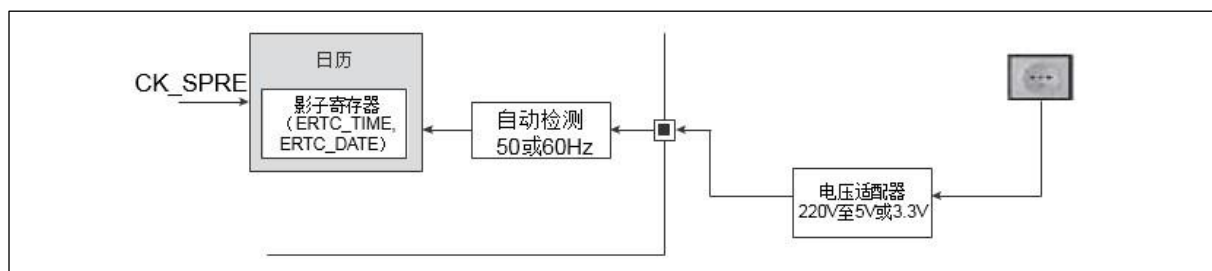
如果参考时钟停止，日历将仅根据 LSE 时钟进行连续更新。ERTC 随后使用同步预分频器输出时钟（ck_spre）边沿上居中的检测窗口等待参考时钟。检测窗口为 7 个 ck_calib 周期。

参考时钟可能有较大的局部偏差（例如在 500 ppm 范围内），但从长期来看，它的精度远高于 32 kHz 石英时钟。

仅当参考时钟丢失而需要重新检测时，才会使用检测系统。当检测窗口比参考时钟周期略大时，由于检测窗口中可能存在 2 个 ck_ref 边沿，该检测系统可能产生 1 个 ck_ref 周期（对于 50 Hz 参考时钟为 20 ms）的不确定度。随后将使用更新窗口，由于其小于参考时钟周期，故不会产生任何误差。

假设 ck_ref 一天丢失不超过一次。这样每个月的总不确定度为 $20\text{ ms} * 1 * 30 = 0.6\text{ s}$ ，这比典型石英时钟的不确定度（35 ppm 石英时钟每月为 1.53 分钟）要小得多。

图 11. ERTC 参考时钟检测



注：参考时钟校准和粗略校准不可同时使用。如果 50 Hz 始终可用，则参考时钟校准是最佳方法（确保精确校准的时间）。如果 50 Hz 输入丢失，可使用 LSE。

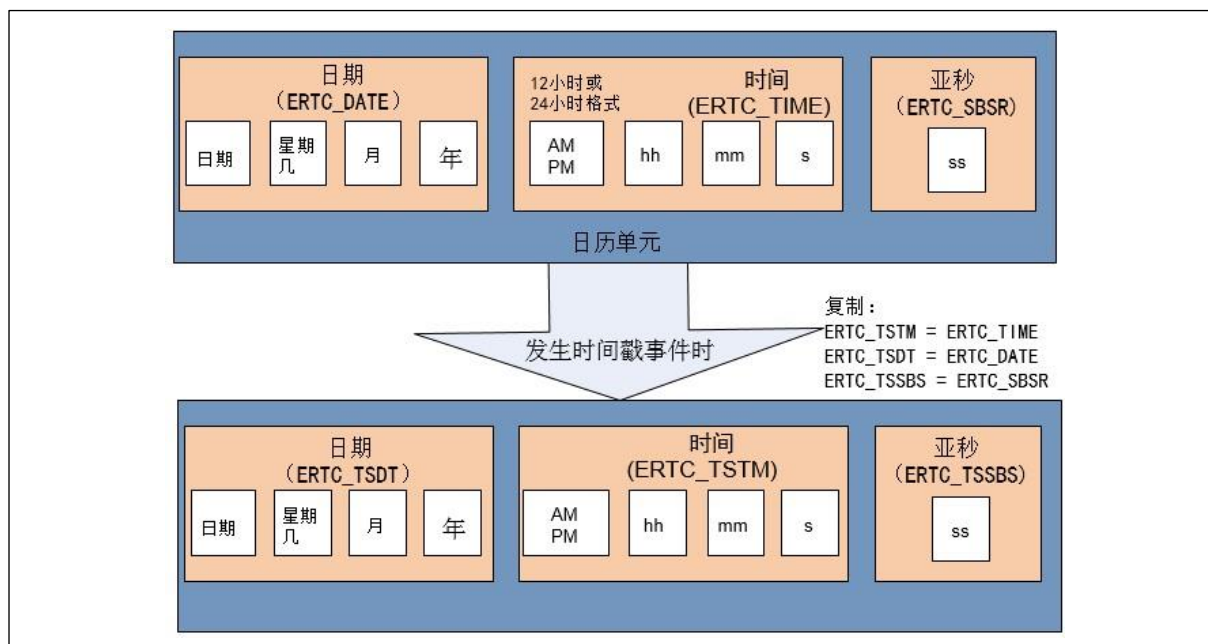
不能在 Vbat 模式下使用参考时钟检测。

仅当提供精确的 50 或 60 Hz 输入时，才能使用参考时钟校准。

1.7 时间戳功能

时间戳功能可用来自动保存当前日历。

图 12. 时间戳事件过程



若时间戳功能已使能，则在 **TIMESTAMP** 备用功能映射到的引脚上检测到时间戳事件时，日历会保存到时间戳寄存器（**ERTC_TSTM**、**ERTC_TSDT**、**ERTC_TSSBS**）中。发生时间戳事件时，**ERTC_STS** 寄存器中的时间戳标志位 (**TSF**) 将置 1。

注：时间戳亚秒寄存器并不适用于所有产品。请参见表 15: **ERTC** 高级功能。

表 10. 时间戳功能

操作	方法	注释
使能时间戳	将 ERTC_CTRL 寄存器的 TSE 位置 1	
通过中断检测时间戳事件	将 ERTC_CTRL 寄存器中的 TSIE 位置 1	发生时间戳事件时生成中断。
通过轮询检测时间戳事件	轮询 ERTC_STS 寄存器中的时间戳标志 (TSF) ⁽¹⁾	要将标志清零，请在 TSF 位中写入零。 ⁽²⁾
检测时间戳溢出事件 ⁽³⁾	轮询 ERTC_STS 寄存器中的时间戳溢出标志 (TSOF) ⁽⁴⁾	要将标志清零，请在 TSOF 位中写入零。 时间戳寄存器（ ERTC_TSTM 、 ERTC_TSDT 和 ERTC_TSSBS) ⁽¹⁾ 保存前一事件的结果。如果在 TSF 位清零后紧接着发生时间戳事件，则 TSF 和 TSOF 位都将置 1。

(1) 在发生由同步过程引发的时间戳事件后，**TSF** 在 2 个 **ck_apre** 周期置为 1。

(2) 为防止在时间戳事件发生的同时屏蔽该事件，除非已将 **TSF** 位读取为“1”，否则应用程序不得将“0”写入 **TSF** 位。

(3) 时间戳溢出事件不产生中断。

(4) **TSOF** 的产生中不存在延迟。也就是说如果两个时间戳事件接连发生，**TSOF** 可能为“1”而 **TSF** 为“0”。因

此，建议只在检测到 TSF 为“1”后再轮询 TSOF。

1.8 ERTC 入侵检测功能

ERTC 包括 n 个入侵检测输入。可配置入侵输入为电平有效或者边沿有效，并且每一个输入都有单独的标志（ERTC_STS 寄存器中的 TPF 位）。

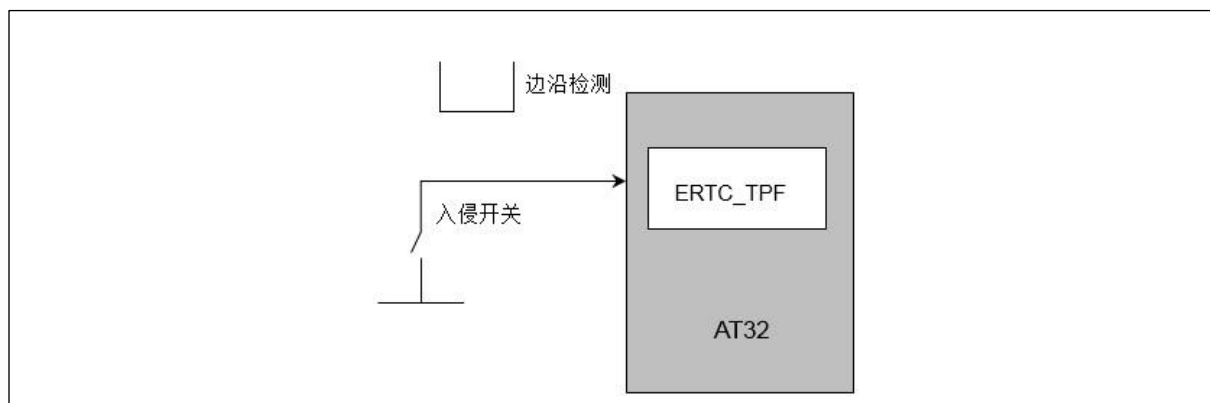
ERTC_TPAF 寄存器中的 TMIE 位置 1 时，入侵检测事件将产生中断。

入侵筛选器“TMFLT 位”的配置定义了入侵检测是边沿激活（将 TMFLT 设置为“00”）还是电平激活（TMFLT 设置为非“00”值）。

1.8.1 对入侵输入的边沿检测

如果 TMFLT 位设置为零，当相应的 TM1TRG 位上出现上升沿或下降沿时，将触发入侵输入检测。

图 13. 通过边沿检测入侵



注：边沿检测时不能使用采样和预充电功能。

表 11. 入侵检测功能（边沿检测）

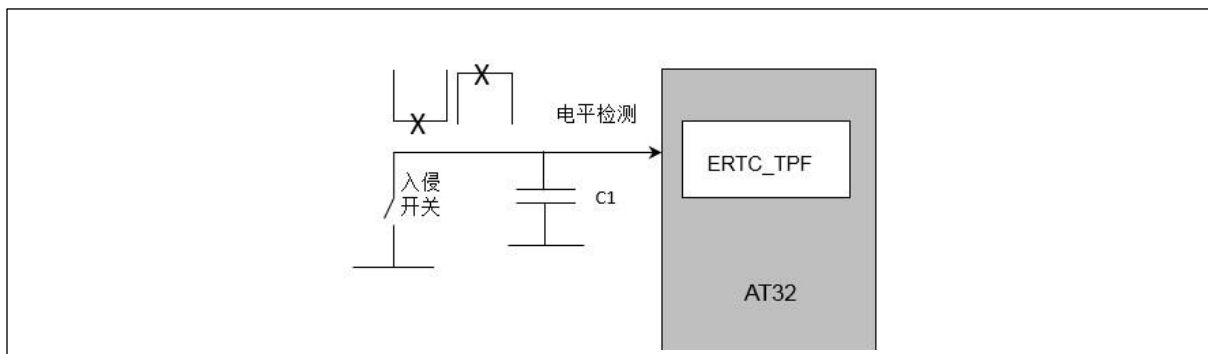
操作	方法	注释
使能入侵检测	将 ERTC_TPAF 寄存器的 TM1E 位置 1	
选择 Tamper1 有效边沿检测	使用 ERTC_TPAF 寄存器中的 TM1TRG 位选择	默认边沿为上升沿
通过中断检测 Tamper1 事件	将 ERTC_TPAF 寄存器中的 TMIE 位置 1	发生入侵检测事件时生成中断
通过轮询检测 Tamper1 事件	轮询 ERTC_STS 寄存器中的时入侵标志 (TPF)	要将标志清零，请在 TPF 位中写入零

1.8.2 对入侵输入的电平检测

将入侵筛选器“TMFLT”设置为非零值表示当相应的 TM1TRG 位上出现所选电平（高或低）时将触发入侵输入检测。

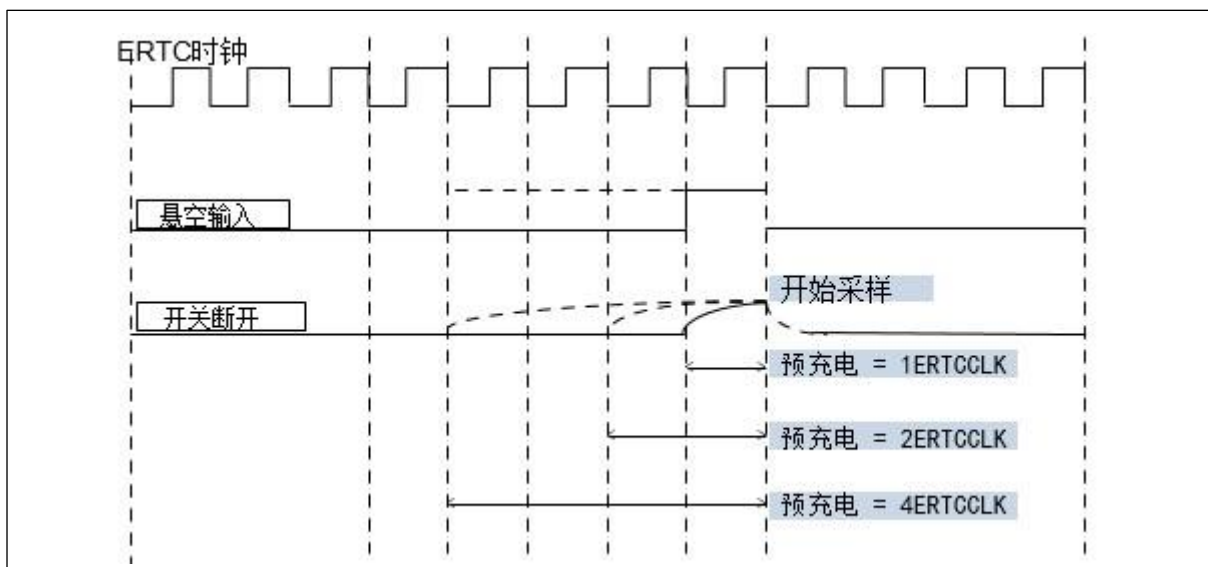
在所选电平连续出现 2、4 或 8 个（取决于 TMFLT 值）采样时生成入侵检测事件。

图 14. 通过电平检测入侵



使用电平检测（入侵筛选器设置为非零值）时，可对入侵输入引脚进行预充电，方法为对输入的状态进行采样前通过内部电阻将 **TMPUDIS** 复位。为支持不同的电容值，期间应用内部上拉的脉冲宽度可以是 1、2、4 或 8 个 **ERTCCLK** 周期。

图 15. 使用预充电脉冲时的入侵采样



注：如果未应用内部上拉，将禁止 I/O 施密特触发器，以避免入侵开关断开出现额外消耗。通过设置入侵采样的频率，可有效应对入侵检测延时（使用预充电功能）与弱上拉 / 下拉的功耗之间的冲突。通过配置 **ERTC_TAMP** 寄存器中的 **TAMPFREQ** 位来决定入侵采样频率。使用 **LSE(32768 Hz)** 作为 **ERTC** 时钟源时，采样频率可以是 1、2、4、8、16、32、64 或 128 Hz。

表 12. 入侵检测功能（电平检测）

操作	方法	注释
使能入侵检测	将 ERTC_TPAF 寄存器的 TM1E 位置 1	
配置 Tamper1 筛选器计数	配置 ERTC_TPAF 寄存器中的 TMFLT 位	默认值为 0
配置 Tamper1 采样频率	配置 ERTC_TPAF 寄存器中的 TMFREQ 位	默认值为 1Hz

配置入侵预充电 / 放电持续时间	将 ERTC_TAMPCR 寄存器中的 TMPUDIS 位置 1/复位	
选择 Tamper1 有效边沿 / 电平检测	使用 ERTC_TPAF 寄存器中的 TM1TRG 位选择	边沿或电平取决于入侵筛选器配置
通过中断检测 Tamper1 事件	将 ERTC_TPAF 寄存器中的 TMIE 位置 1	发生入侵检测事件时生成中断
通过轮询检测 Tamper1 事件	轮询 ERTC_STS 寄存器中的入侵标志 (TPF)	要将标志清零, 请在 TPF 位中写入零

1.8.3 激活入侵检测事件时间戳

将 TMTS 位置 1 后, 任何入侵事件 (使用边沿或电平检测) 都会导致出现时间戳。因此, 当入侵标志置 1 时, 时间戳标志和时间戳溢出标志也将置 1, 并且起作用的方式与发生常规时间戳事件时相同。

注: 使用此功能时不必使能或禁止时间戳功能。

1.9 备份寄存器

ERTC_BKPxD_T (其中 $x = 0$ 到 n 个备份寄存器 (80 字节)) 在发生入侵检测事件时复位。当 VDD 关闭时, 这些寄存器由 VBAT 供电, 因而系统复位时, 这些寄存器不会复位, 并且当器件在低功耗模式下工作时, 寄存器的内容仍然有效。

注: 备份寄存器的数量“ n ”取决于产品。请参见表 15: ERTC 高级功能。

1.10 ERTC 和低功耗模式

ERTC 设计为最大程度地降低功耗。用于日历的预分频器分为同步和异步两种。

增加异步预分频器的值可降低功耗。

如果 VDD 和 VBAT 先前均已关闭或者在 AT32F2xx 器件上复位了备份域, 则 ERTC 将保持在复位模式下工作, 并且仅在 VDD 或 VBAT 上电时才会复位。

仅当上电复位时, 寄存器才会复位。ERTC 寄存器的值在复位后不会丢失, 从而日历可保持正确的时间和日期。

系统复位或上电复位后, AT32 在运行模式下工作。此外, 该器件还支持五种低功耗模式, 以在低功耗、短启动时间和可用唤醒源之间寻求最佳平衡。

ERTC 外设可在以下三种低功耗模式下保持活动状态:

- 睡眠模式
- 待机模式
- 停止模式

有关低功耗模式的更多详细信息, 请参见 AT32 参考手册的低功耗模式部分。

1.11 备用功能 ERTC 输出

ERTC 外设具有以下两个输出：

- ERTC_CAL，用于生成外部时钟。
- ERTC_ALARM，复用 ERTC 闹钟和唤醒事件而产生的唯一输出。

1.11.1 ERTC_CAL 输出

ERTC_CAL 输出用于生成频率可变的信号。根据用户应用，此信号可以用作校准外部器件的参考时钟，或者连接到蜂鸣器以生成声音。

信号频率使用异步预分频器 PRDIV_A[7:0] 的 7 个 LSB 位 (PRDIV_A [6:0]) 进行配置。

ERTC_CAL 是 7 位异步预分频器 PRDIV_A 的第 4 位的输出。如果 PRDIV_A[5]=0，则在 ERTC_CAL 上无信号输出。

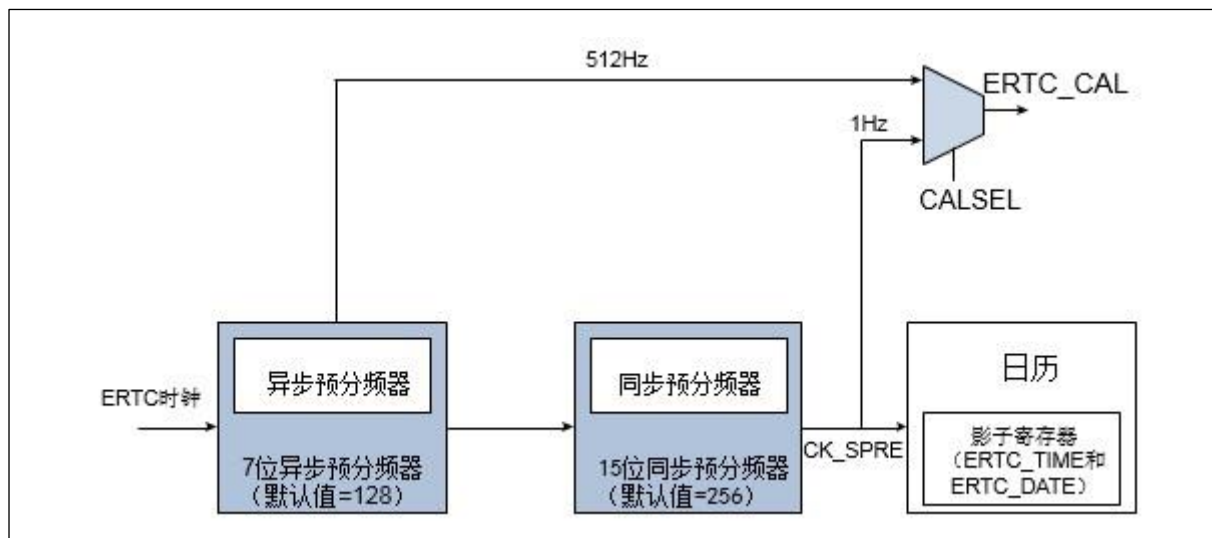
将 512 Hz 设置为输出信号

1. 选择 LSE“32768 Hz”作为 ERTC 时钟源。
2. 将异步预分频器设置为默认值“128”。
3. 将“CALOE”设置为“1”，使能输出校准。
4. 将 CALSEL 设置为“0”，选择 512 Hz 作为校准输出。

将 1 Hz 设置为输出信号

1. 选择 LSE“32768 Hz”作为 ERTC 时钟源。
2. 将异步预分频器设置为默认值“128”。
3. 将同步预分频器设置为默认值“256”。
4. 将“CALOE”设置为“1”，使能输出校准。
5. 将 CALSEL 设置为“1”，选择 1 Hz 作为校准输出。

图 16. ERTC_CAL 时钟源



最大和最小 ERTC_CAL 输出频率

ERTC 可输出由 7 位异步预分频器分频的 ERTCCLK 时钟。分频系数使用 ERTC_PSC 寄存器的

PRDIV_A[6:0] 位进行配置。

ERTC_CAL 最大频率和最小频率分别为 31.250 kHz 和 512 Hz。

表 13. ERTC_CAL 输出频率与时钟源

ERTC 时钟源	ERTC_CAL 输出频率	
	最小值 (PRDIV_A[6:0] = 111 111b) (div 64)	最大化 (PRDIV_A[6:0] = 100 000b ⁽¹⁾) (div 32)
HSE_ERTC = 1 MHz	15.625 kHz	31.250 KHz
LSE = 32768 Hz	512 Hz (默认输出频率)	1.024 KHz
LSI = 40 kHz	625 Hz	1250 Hz

(1) PRDIV_A[5] 必须设置为“1”才能使用 ERTC_CAL 输出信号生成。如果 PRDIV_A[5] 位为零，则 ERTC_CAL 上不会输出任何信号。

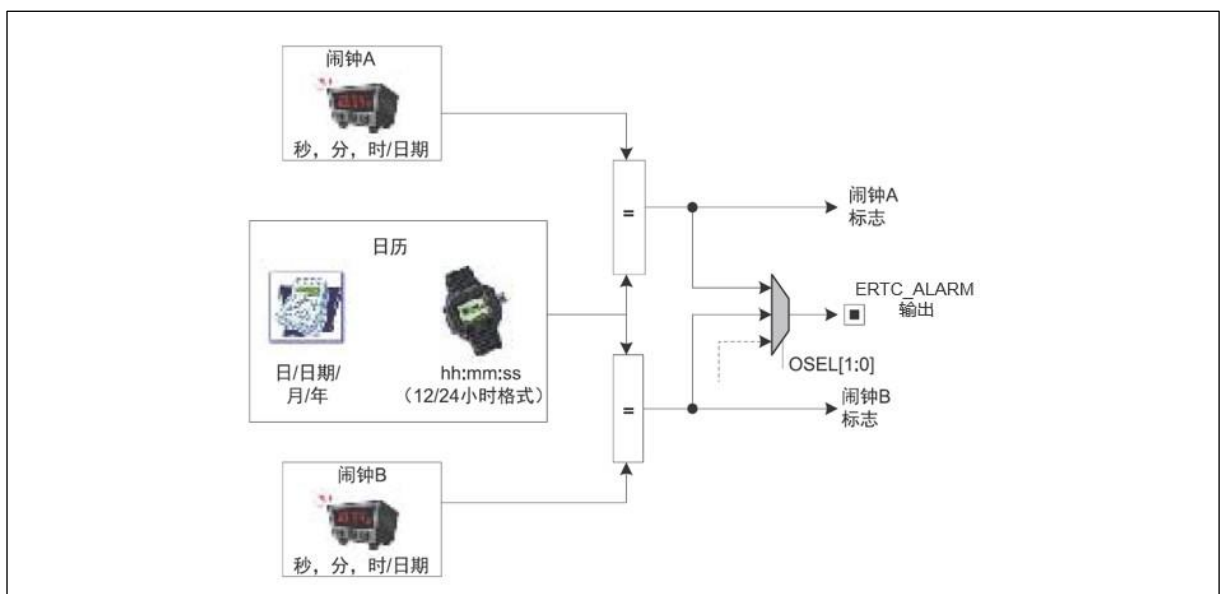
1.11.2 ERTC_ALARM 输出

ERTC_ALARM 输出可以连接到 ERTC 闹钟单元 A 或 B 以触发外部动作，或连接到 ERTC 唤醒单元以唤醒外部器件。

ERTC_ALARM 输出连接到 ERTC 闹钟单元

当日历达到 ERTC_ALA 寄存器中的闹钟 A 预编程值时（ERTC_ALA 寄存器对应闹钟 B），ERTC_STS 寄存器中的闹钟标志 ALAF 位（ALRBF 位）置“1”。如果闹钟 A 或闹钟 B 标志连接到 ERTC_ALARM 输出（对于闹钟 A，ERTC_CTRL_OSEL[1:0] = 01；对于闹钟 B，ERTC_CTRL_OSEL[1:0] = 10），该引脚将设置为 VDD 或 GND，具体取决于所选极性。将所选闹钟标志清零时，输出将翻转。

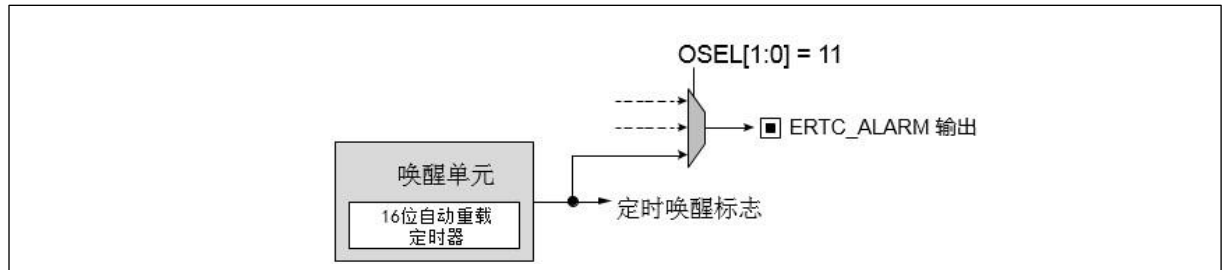
图 17. 闹钟标志连接到 ERTC_ALARM 输出



ERTC_ALARM 输出连接到唤醒单元

唤醒递减计数定时器达到 0 时，唤醒标志置“1”。如果选择此标志作为 ERTC_ALARM 输出的源（ERTC_CTRL 寄存器中的 OSEL[1:0] 位设置为“11”），输出将根据所选的极性进行设置，并且只要该标志未清零，输出就会保持置 1 状态。

图 18. 定时唤醒连接到 ERTC_ALARM 引脚



1.12 ERTC 安全特性

1.12.1 ERTC 寄存器写保护

为防止 ERTC 寄存器在复位后受到可能的寄生写访问，ERTC 寄存器会自动锁定。要更新当前日历时间和日期，必须将这些寄存器解锁。

通过在写保护寄存器 (ERTC_WPR) 中编程一个密钥来允许对 ERTC 寄存器进行写操作。要解锁 ERTC 寄存器的写保护，需要执行以下步骤：

1. 将 0xCA 写入 ERTC_WPR 寄存器。
2. 将 0x53 写入 ERTC_WPR 寄存器。

写入不正确的密钥会自动重新激活 ERTC 寄存器写访问保护。

1.12.2 进入/退出初始化模式

ERTC 可在以下两种模式下工作：

- 初始化模式，在该模式下将停止计数器。
- 自由运行模式，在该模式下将运行计数器。

计数器正在运行时无法更新日历。因此必须将 ERTC 切换到初始化模式，才能更新时间和日期。在初始化模式下工作时，计数器将停止。当 ERTC 进入自由运行模式时，计数器从新值开始计数。

ERTC_STS 寄存器的 INITM 位可用于从一种模式切换到另一种模式，INITF 位可用于检查 ERTC 当前模式。

ERTC 必须处于初始化模式才能对时间和日期寄存器 (ERTC_TIME 和 ERTC_DATE) 以及预分频器寄存器 (ERTC_PSC) 进行编程。通过将 INITM 位置 1 并等到 ERTC_STS_INITF 标志置 1 来进入初始化模式。

要返回到自由运行模式并重新开始计数，ERTC 必须退出初始化模式。这通过复位 INITM 位来完成。

只有上电复位能复位日历。系统复位不会影响日历，但会复位由应用读取的影子寄存器。RSF 位置 1 时，影子寄存器会再次更新。系统复位后，应用可检查 ERTC_STS 寄存器中的 INITS 状态标志，以验证日历是否已初始化。当日历的年字段为 0x00（上电复位值）时，该标志被复位，表示日历必须初始化。

1.12.3 ERTC 时钟同步

当应用读取日历时，将访问影子寄存器，其中包含由 ERTC 时钟 (ERTCCLK) 驱动的实际日历时间和日期的副本。每次用实际日历值更新日历时间和日期影子寄存器时，ERTC_STS 寄存器中的 RSF 位都会置 1。该副本每两个 ERTCCLK 周期更新一次，与系统时钟 (SYSCLK) 同步。系统复位后或退出初始化模式后，应用必须等待 RSF 置 1 才能读取日历影子寄存器。

当系统从低功耗模式唤醒时 (SYSCLK 已关闭)，应用必须首先将 RSF 位清零，然后等到该位再次置 1，才能读取日历寄存器。这样可确保应用读取的值是当前日历值，而不是进入低功耗模式前的值。

将 ERTC_CTRL 寄存器中的 “BYP SHDW” 位置 “1” 后，日历值将直接从日历计数器获取，而不是通过读取影子寄存器获取。在这种情况下，不必等待同步时间，但必须用软件检查日历寄存器的一致性。用户必须读取所需日历字段的值。然后必须再次执行读操作。随后比较两个读序列的结果。如果结果匹配，则读结果正确。如果不匹配，必须再读一次字段，第三次读取的结果为有效结果。

注： *BYP SHDW 位复位后，影子寄存器可能直到下次同步才会正确。在这种情况下，软件应将“RSF”位清零，然后等待同步 (“RSF” 应置 1)，最后读取影子寄存器。*

2 ERTC 固件驱动程序 API

固件驱动程序提供一系列固件函数来管理 ERTC 外设的下列功能：

- 初始化
- 日历（时间和日期）配置
- 闹钟（闹钟 A 和闹钟 B）配置
- 唤醒定时器配置
- 夏令时配置
- 输出引脚配置
- 数字校准配置
- 同步配置
- 时间戳配置
- 入侵配置
- 备份数据寄存器配置
- ERTC 入侵和时间戳引脚选择以及输出类型配置
- 中断和标志管理

对于 AT32F2xx 系列，ERTC 驱动程序 `at32f4xx_ertc.c/h` 位于以下目录：

`AT32F4xx_StdPeriph_Lib_Vxxx\Libraries\AT32F4xx_StdPeriph_Driver\src`

2.1 开始使用 ERTC 驱动程序

在使用 ERTC 功能前：

- 使能 ERTC 域访问（见下面的注）
- 使用 `ERTC_Init()` 函数配置 ERTC 预分频器（异步和同步）和 ERTC 小时格式。

注：复位后，备份域（ERTC 寄存器、ERTC 备份数据寄存器和备份 SRAM）将受到保护，以防止任何不需要的写访问。要使能对 ERTC 域和 ERTC 寄存器的访问：

- 使用 `RCC_APB1PeriphClockCmd()` 函数使能电源控制器 (PWR) APB1 接口时钟。
- 使用 `RCC_ERTCCLKConfig()` 函数选择 ERTC 时钟源。
- 使用 `RCC_ERTCCLKCmd()` 函数使能 ERTC 时钟。

2.2 时间和日期配置

要配置 ERTC 日历（时间\日期），请使用 `ERTC_SetTimeValue()`和 `ERTC_SetDateValue()` 函数。

要读取 ERTC 日历，请使用 `ERTC_GetTimeValue()`、`ERTC_GetDateValue()` 和 `ERTC_GetSubSecondValue()` 函数。

要对 ERTC 日历增加或减少一小时，请使用 `ERTC_DayLightSavingConfig()` 函数。

2.3 闹钟配置

ERTC 闹钟

要配置 ERTC 闹钟，请使用 `ERTC_SetAlarmValue()` 函数。

要使能所选的 ERTC 闹钟，请使用 ERTC_AlarmCmd () 函数。

要读取 ERTC 闹钟，请使用 ERTC_GetAlarmValue () 函数。

ERTC 闹钟的亚秒

要配置 ERTC 闹钟的亚秒，请使用 ERTC_AlarmSubSecondConfig () 函数。

要读取 ERTC 闹钟的亚秒，请使用 ERTC_GetAlarmSubSecond () 函数。

2.4 ERTC 唤醒配置

要配置 ERTC 唤醒时钟源，请使用 ERTC_WakeUpClockConfig () 函数。

要配置 ERTC 唤醒计数器，请使用 ERTC_SetWakeUpCounter () 函数。

要使能 ERTC 唤醒，请使用 ERTC_WakeUpCmd () 函数。

要读取 ERTC 唤醒计数器寄存器，请使用 ERTC_GetWakeUpCounter () 函数。

2.5 输出配置

ERTC 有两个不同的输出：

- AFO_ALARM，用于管理 ERTC 闹钟 A、闹钟 B 和唤醒信号。要在 ERTC_AF1 引脚上输出所选的 ERTC 信号，请使用 ERTC_OutputConfig () 函数。
- AFO_CALIB，用于管理 64 分频 (512 Hz) 的 ERTC 时钟信号和日历时钟 (1 Hz)。要在 ERTC_AF1 引脚上输出 ERTC 时钟，请使用 ERTC_CalOutputCmd () 函数。

2.6 数字校准配置

要配置 ERTC 粗略校准值和相应的符号，请使用 ERTC_CoarseCalConfig () 函数。

要使能 ERTC 粗略校准，请使用 ERTC_CoarseCalCmd () 函数。

要配置 ERTC 精密校准值和校准周期，请使用 ERTC_SmoothCalConfig () 函数。

2.7 时间戳配置

要配置 ERTC_AF1 触发信号并使能 ERTC 时间戳，请使用 ERTC_TimeStampCmd () 函数。

要读取 ERTC 时间戳时间和日期寄存器，请使用 ERTC_GetTimeStamp () 函数。

要读取 ERTC 时间戳亚秒寄存器，请使用 ERTC_GetTimeStampSubSecond () 函数。

2.8 入侵配置

要配置 ERTC 入侵触发，请使用 ERTC_TamperTriggerConfig () 函数。

要配置 ERTC 入侵筛选器，请使用 ERTC_TamperFilterConfig () 函数。

要配置 ERTC 入侵采样频率，请使用 ERTC_TamperSamplingFreqConfig () 函数。

要配置 ERTC 入侵引脚输入预充电持续时间，请使用 ERTC_TamperPinsPrechargeDuration () 函数。

要使能入侵引脚的预充电，请使用 ERTC_TamperPullUpCmd () 函数。

要使能入侵检测事件的时间戳，请使用 ERTC_TimeStampOnTamperDetectionCmd () 函数。

要使能 ERTC 入侵，请使用 ERTC_TamperCmd () 函数。

2.9 备份数据寄存器配置

要写入 ERTC 备份数据寄存器，请使用 ERTC_WriteBackupRegister () 函数。

要读取 ERTC 备份数据寄存器，请使用 ERTC_ReadBackupRegister () 函数。

3 应用程序示例

ERTC 固件驱动程序提供了一组示例，以帮助您快速熟悉 ERTC 外设。

对于 AT32F415 系列，示例位于以下目录：

AT32F4xx_StdPeriph_Lib_Vxxx\Project\AT_START_F415\Examples\ERTC

表 14. 示例说明

示例	说明	涉及的功能
ERTC 硬件日历	此示例介绍如何使用 ERTC 外设日历功能：秒、分钟、小时（12 或 24 小时格式）、星期几、日期、月份和年份。示例使用预分频器与中断来保持时间和生成闹钟中断，从应用角度说明了设置 ERTC 外设的方法。	硬件日历闹钟（中断）预分频器 ERTC 备份寄存器
ERTC 备份域	此示例演示并说明如何使用备份域上可用的外设。这些外设为 RCC BDCR 寄存器，其中包含 LSE 振荡器配置和 ERTC 时钟使能 / 禁止位。示例使用了 ERTC 外设及其相关备份数据寄存器，以及备份 SRAM (4KB) 及其低功耗调压器（使备份 SRAM 在产品由 VBAT 引脚供电时保留其内容），从应用角度说明了如何设置 ERTC 硬件日历以及对 ERTC 备份数据寄存器和 BKPSRAM（备份 SRAM）的读 / 写操作。	ERTC 备份寄存器 备份 SRAM 备份 SRAM 的低功耗调压器 硬件日历唤醒（中断）
使用 LSI 进行自动校准	此示例演示并说明如何使用 LSI 时钟源自动校准来获取精确的 ERTC 时钟。示例采用低速内部 (LSI) 时钟作为 ERTC 时钟源。ERTC 唤醒配置为每 1 秒生成一次中断。唤醒计数器的时钟由 ERTC CK_SPRE 信号 (1Hz) 提供，其计数器设置为零。	预分频器 ERTC 备份寄存器 硬件日历唤醒（中断）
入侵检测	此示例介绍如何对 ERTC 备份数据寄存器进行写 / 读数据操作，并演示了入侵检测功能。示例将 ERTC_AF1 引脚配置为下降沿触发入侵，并使能入侵中断。在 ERTC_AF1 引脚上施加低电平时，ERTC 备份数据寄存器复位，并且生成入侵中断。	入侵（中断） ERTC 备份寄存器
时间戳	此示例介绍如何使用 ERTC 外设和时间戳功能。示例将 ERTC_AF1 引脚配置为下降沿触发时间戳，并使能时间戳检测。在 ERTC_AF1 引脚上施加低电平时，由于使能了时间戳事件检测，日历将保存在时间戳寄存器中。	时间戳（中断）预分频器唤醒（中断） 硬件日历 ERTC 备份寄存器
秒表	此示例介绍如何使用 AT32F4xx 新增的 ERTC 亚秒和入侵（筛选、采样）功能。示例模拟一个精密计时器，可在备份寄存器（10 个寄存器表示时间（秒、分钟和小时），10 个寄存器表示亚秒）中存储 10 条记录时间。	时间戳（中断）入侵（中断） 硬件日历 ERTC 备份寄存器
ERTC 定时器	此示例简要介绍了如何使用具有闹钟亚秒功能的 ERTC 外设来模拟刷新时间等于 250 ms $((1 \text{ 秒} / 8) * 2)$ 的定时器。ERTC 配置为每 125 ms 生成一次亚秒中断（每秒 8 个中断）。	硬件日历闹钟亚秒

4 版本历史

表 15. 文档版本历史

日期	版本	变更
2019.10.12	1.0.0	最初版本

重要通知 - 请仔细阅读

买方自行负责对本文所述雅特力产品和服务的选择和使用，雅特力概不承担与选择或使用本文所述雅特力产品和服务相关的任何责任。

无论之前是否有过任何形式的表示，本文档不以任何方式对任何知识产权进行任何明示或默示的授权或许可。如果本文档任何部分涉及任何第三方产品或服务，不应被视为雅特力授权使用此类第三方产品或服务，或许可其中的任何知识产权，或者被视为涉及以任何方式使用任何此类第三方产品或服务或其中任何知识产权的保证。

除非在雅特力的销售条款中另有说明，否则，雅特力对雅特力产品的使用和/或销售不做任何明示或默示的保证，包括但不限于有关适销性、适合特定用途(及其依据任何司法管辖区的法律的对应情况)，或侵犯任何专利、版权或其他知识产权的默示保证。

雅特力产品并非设计或专门用于下列用途的产品：(A) 对安全性有特别要求的应用，如：生命支持、主动植入设备或对产品功能安全有要求的系统；(B) 航空应用；(C) 汽车应用或汽车环境；(D) 航天应用或航天环境，且/或(E) 武器。因雅特力产品不是为前述应用设计的，而采购商擅自将其用于前述应用，即使采购商向雅特力发出了书面通知，风险由购买者单独承担，并且独力负责在此类相关使用中满足所有法律和法规要求。

经销的雅特力产品如有不同于本文档中提出的声明和/或技术特点的规定，将立即导致雅特力针对本文所述雅特力产品或服务授予的任何保证失效，并且不应以任何形式造成或扩大雅特力的任何责任。

© 2020 雅特力科技 (重庆) 有限公司 保留所有权利