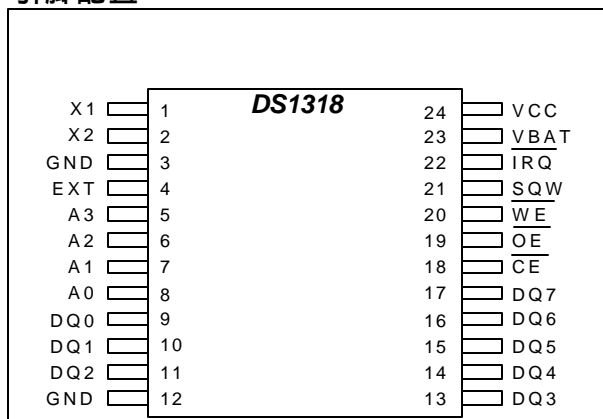


引脚配置



概述

DS1318 并行接口历时计数器(ETC)是一款 44 位计数器，提供分辨率 244 μ s 的历时。ETC 具有 6 个寄存器，其中 4 个寄存器表示为 32 位秒计数，其余两个寄存器使用 12 位记录亚秒级计数(表 1)。DS1318 能够用于时间和日期记录。ETC 寄存器里的零值必须定义为某个确定的时间起始点。比如，在 Unix 系统中，零计时起点为 1970 年 1 月 1 日午夜零时，则 32 位秒数寄存器可表示 1970 年 1 月 1 日午夜零时至 2038 年 1 月 19 日 03:14:07 之间的任何时间。转换程序通常完成将 32 秒位计数值转换成标准的时间和日期格式。请参考网站 www.maxim-ic.com/app517 上的应用笔记 517: *DS1371/DS1374 32-Bit Binary Counter Time Conversion*。DS1318 还可用于两个事件之间的计时。

表 1. 地址分配表

地址	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	功能	范围
00H	SS3	SS2	SS1	SS0	0	0	0	SQWS	Subseconds0	00–F0h
01H	SS11	SS10	SS9	SS8	SS7	SS6	SS5	SS4	Subseconds1	00–FFh
02H	S7	S6	S5	S4	S3	S2	S1	S0	Seconds0	00–FFh
03H	S15	S14	S13	S12	S11	S10	S9	S8	Seconds1	00–FFh
04H	S23	S22	S21	S20	S19	S18	S17	S16	Seconds2	00–FFh
05H	S31	S30	S29	S28	S27	S26	S25	S24	Seconds3	00–FFh
06H	ALM 7	ALM 6	ALM 5	ALM 4	ALM 3	ALM 2	ALM1	ALM0	Alarm0	00–FFh
07H	ALM 15	ALM 14	ALM 13	ALM 12	ALM 11	ALM 10	ALM 9	ALM 8	Alarm1	00–FFh
08H	ALM 23	ALM 22	ALM 21	ALM 20	ALM 19	ALM 18	ALM 17	ALM 16	Alarm2	00–FFh
09H	ALM 31	ALM 30	ALM 29	ALM 28	ALM 27	ALM 26	ALM 25	ALM 24	Alarm3	00–FFh
0AH	TE	ENOSC	CCFG1	CCFG0	EPOL	SQWE	PIE	AIE	ControlA	00–FFh
0BH	PRS3	PRS2	PRS1	PRS0	SRS3	SRS2	SRS1	SRS0	ControlB	00–FFh
0CH	OSF	UIP	0	0	0	0	PF	ALMF	Status	—

在任何应用中，ETC 运行时通常需要读取 ETC 寄存器值。DS1318 提供一组“用户寄存器”，允许在内部计数器运行的同时访问寄存器数据。每 244 μ s 内部寄存器更新一次用户寄存器。尽管计数器带有缓冲，但由于读操作和计数更新是异步工作的，仍有可能读取或写入错误的的数据。

例子 1

在读取 ETC 寄存器时，数据可能在更新。比如，如果寄存器中的数据是 0x55555555.FFF，在读亚秒和秒寄存器之间时，数据可能更新，读取的值可能为 0x55555556.FFF，而不是 0x55555556.000。

例子 2

当访问某个寄存器时，数据正从内部计数器传输到用户寄存器。在这种情况下，数据更新和读操作可能仅相差几个纳秒。虽然出现的概率很低，但当它真的发生时，则读取的数据是无效的。

例子 3

执行写操作，同时内部计数器和用户寄存器之间正在传输数据。由于内部数据总线在传输过程中被占用，对 DS1318 内部任何寄存器的写操作都破坏数据(数据冲突)。在这种情况下，如果发生的时间差仅几个纳秒时，则用户寄存器中的数据无效，直至 244 μ s 之后的再次更新为止。

例子 4

当写计数器时，如果对所有计数器的写操作时间超过 244 μ s，则更新操作将发生，必然增加先前写入寄存器中的数据。

有几种办法可避免上述错误。DS1318 包含两个控制位，用于避免在更新过程中进行读或者写操作。以下段落讨论这些方法。

更新操作可能发生在读计数寄存器时。一种验证所读数据是否正确的方法是：以 LSB 到 MSB 的顺序读取对应的寄存器，并存储结果，然后再次读取这些寄存器。如果发生了更新操作，则第一次读取数据和第二次读取数据不同。如果数据不同，需要再次读取这些寄存器，直到前后数据匹配为止。如果使用了亚秒级寄存器，则每 244 μ s 发生一次更新操作。如果仅使用了秒寄存器，则更新操作每秒一次(图 1)。

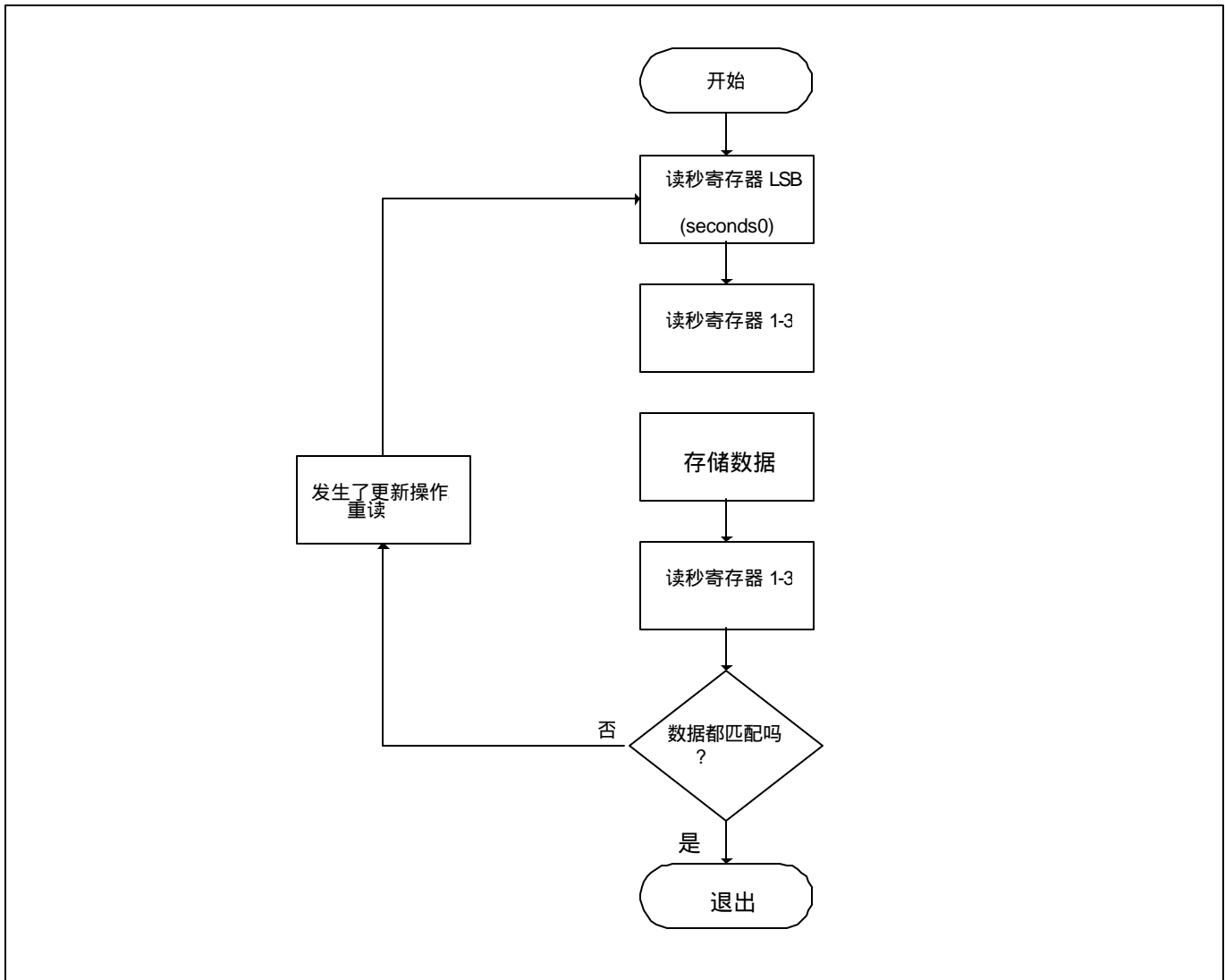


图 1. 在两次更新操作之间读取时钟寄存器

当读计数器时，通过判断更新(UIP)标志位，以避免出错。UIP 每 244 μ s 置位一次，如果更新操作在 61 μ s 之内发生，则其值为 1。当读计数寄存器时，应该判断 UIP 位。如果其值为 1，应该延时读操作，直到 UIP 为 0。如果 UIP 为 0，则可以读取寄存器。应该在 60 μ s 之内读寄存器，以避免下一次更新操作（图 2）。如果读寄存器程序的执行时间大于 60 μ s(例如，程序被另一个程序中断，且执行时间大于 60 μ s)，则必须注意，应当确保数据是否有效。

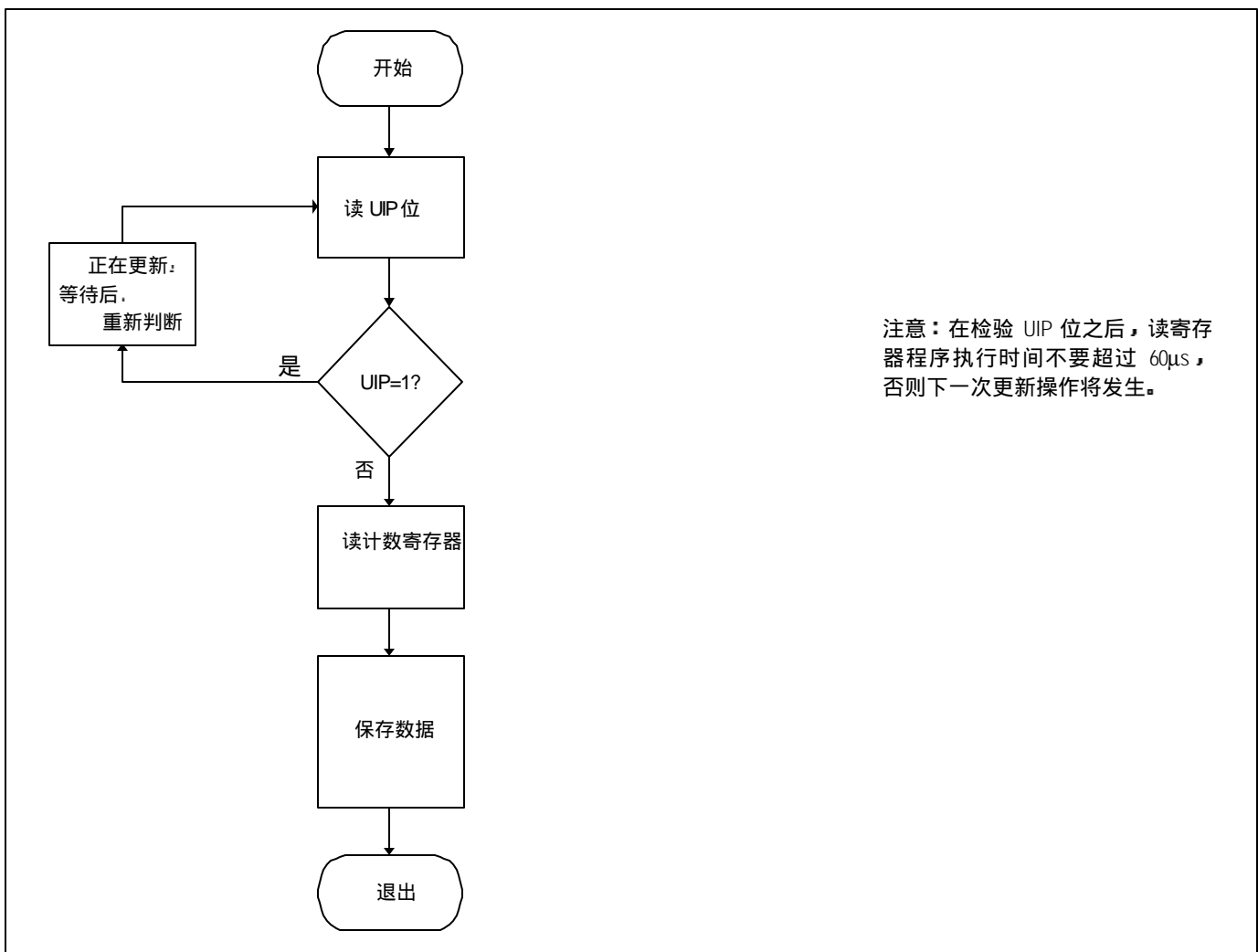


图 2. 判断 UIP 位读取时钟寄存器

另一种读寄存器的方法是使用 UIP 位和传输使能位(TE)。当 TE 位被设置为 0 时，将停止内部计数器和用户寄存器之间的数据传输。由于上次数据更新可能会被写 TE 位操作破坏，因此在改写 TE 位时，应当判断 UIP 位，避免发生更新操作。一旦 TE 被设置为 0，数据还可以从寄存器中读出。由于更新被禁止，因而不必在意 UIP 位是否被设置，或者要花多长时间读取寄存器数据。这个例程可代替前面的例程，在那些应用中可能需要花费 60μs 以上的时间读取所有寄存器。

一旦数据被读出，应当设置 TE 位为 1，允许数据更新。注意：为发生一次计时更新，传输使能应至少保持 244μs。由于这个要求，因此采用这个方法不可能读到后续变化的 subsecond0 寄存器数据(图 3)。

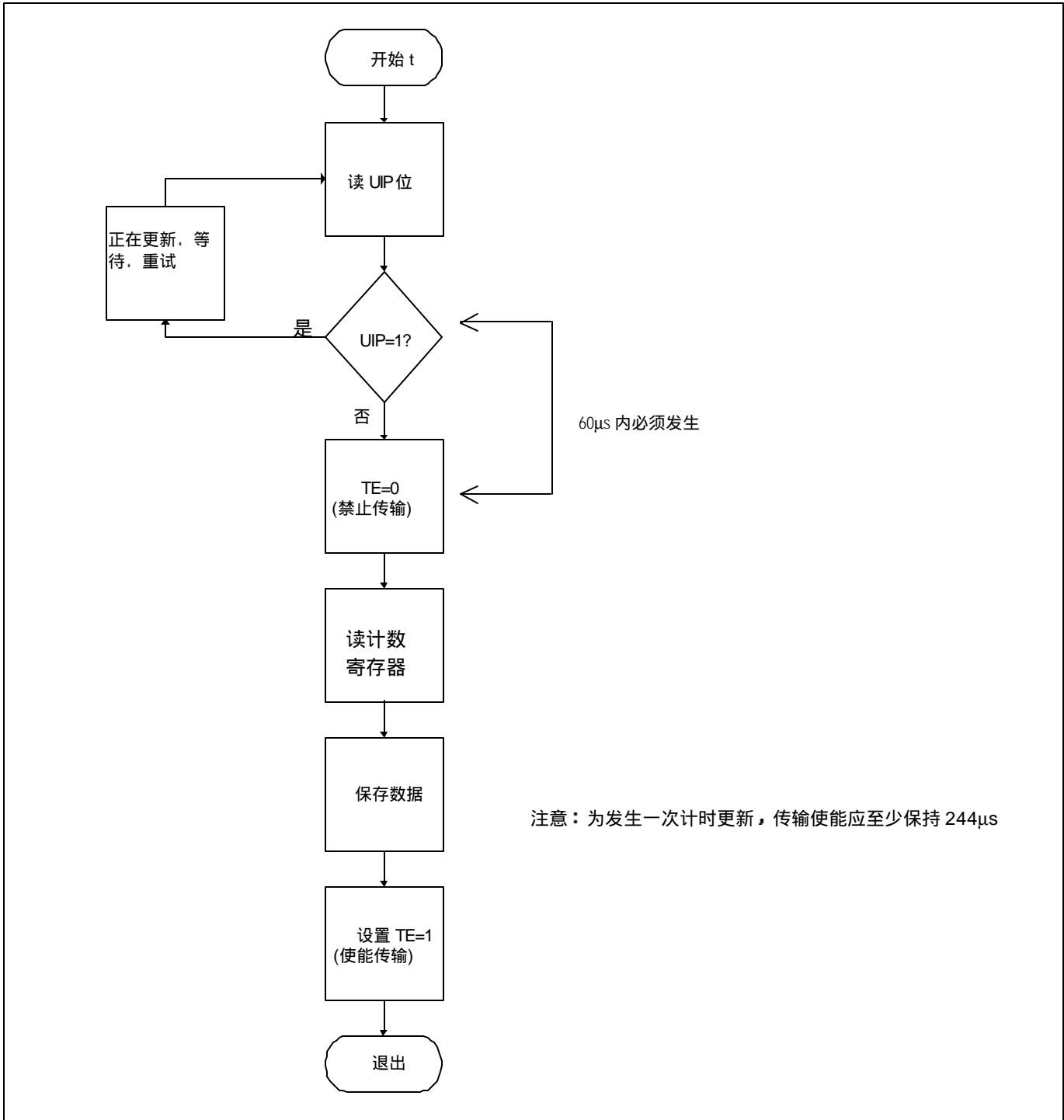


图 3. 使用 UIP 和 TE 位读时钟寄存器

周期中断与时钟寄存器的更新保持同步。如果 PF 标志用作中断输出，则在数据更新后马上读取时钟寄存器。如果准备读取亚秒寄存器，则在 244 μ s 内不会遇到计时更新。如果仅使用秒寄存器，则下次更新将在 1 秒后发生。该程序可使用在时间和日期周期性更新的应用中，例如要求显示时间和日期。使用 PF 标志位驱动控制器的中断输入，允许系统在时间未发生变化时，执行其他的功能。

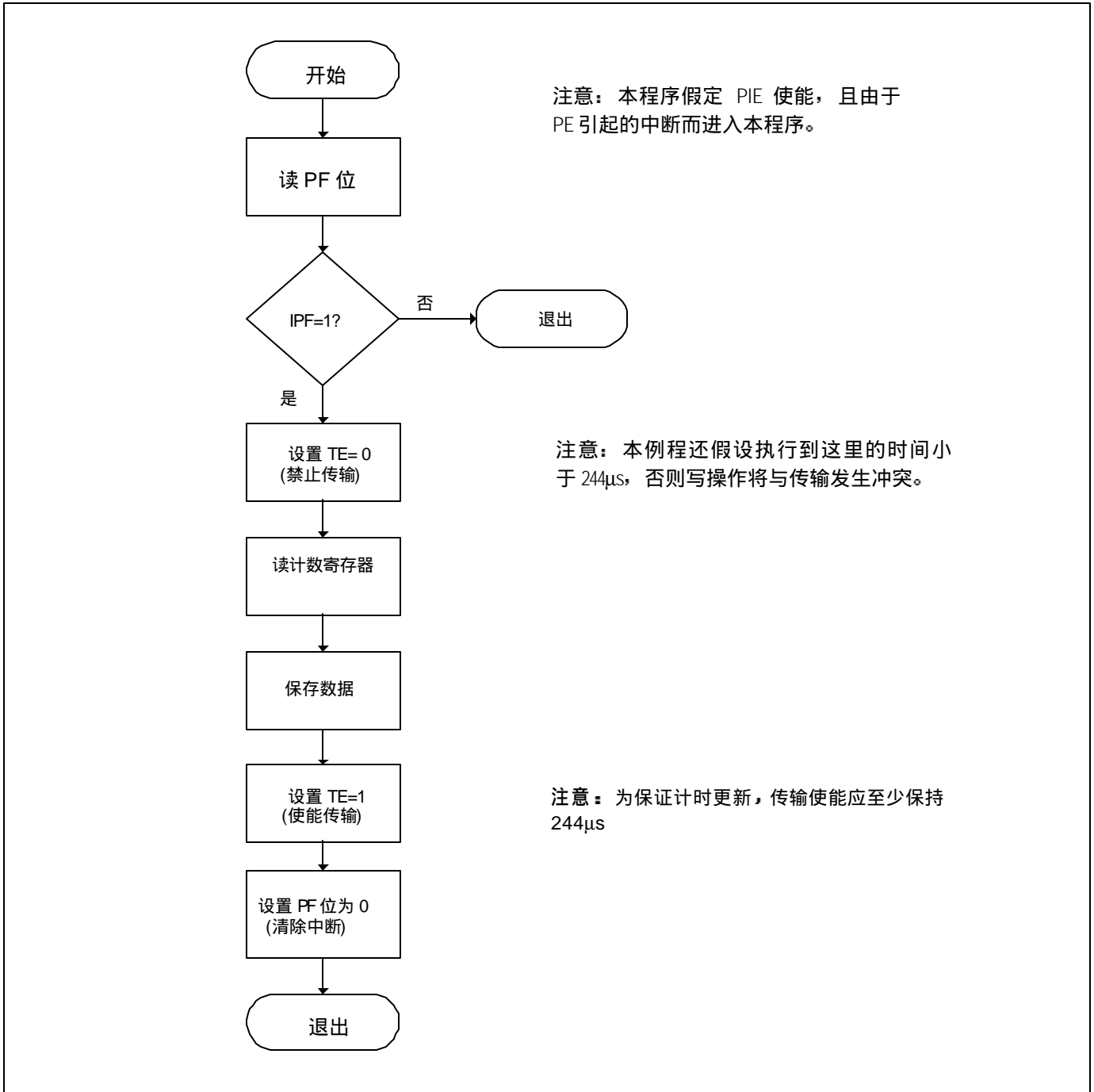


图 4. 使用周期中断标志位读时钟寄存器

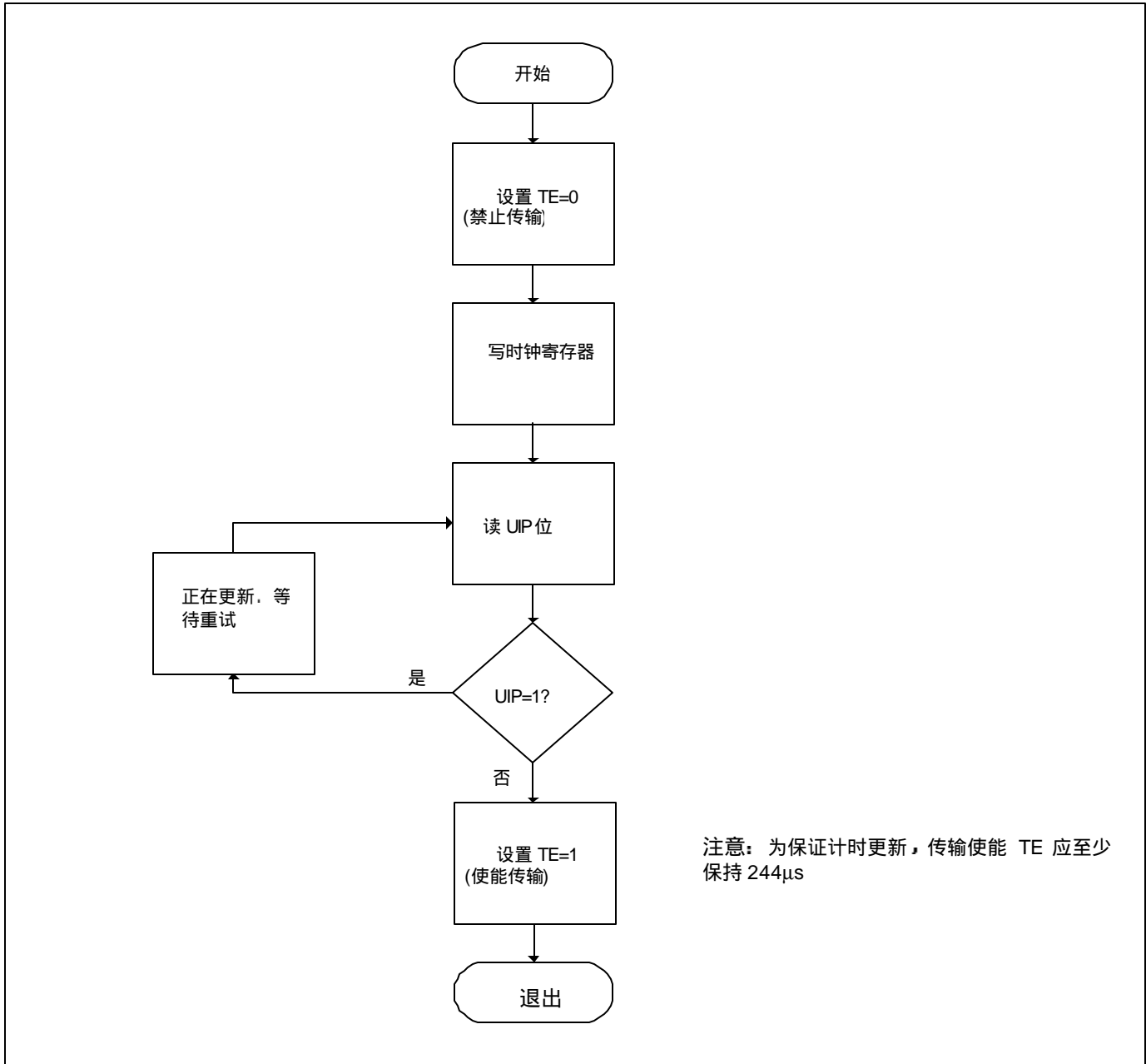


图 5. 使用 TE 和 UIP 位写时钟寄存器

当往计数寄存器写数据时, 可使用 TE。当 TE 被写入 1 时, 所有写入用户寄存器中的数据将传输到内部计数器。如果所有寄存器被写, 则无需特别担心数据冲突。如果仅改写部分寄存器, 相对于其余的寄存器, 将可能产生读寄存器和获取有效数据的问题。在这种情况下, 应当判断 UIP 位, 以避免冲突。