(19)中华人民共和国国家知识产权局

(12)发明专利

(10)授权公告号 CN 107229543 B
(45)授权公告日 2022.04.22

(21)申请号 201710171555.6

(22)申请日 2017.03.22

(65)同一申请的已公布的文献号
申请公布号 CN 107229543 A

(43)申请公布日 2017.10.03

(30)优先权数据
2016-058614 2016.03.23 JP

(73)专利权人 瑞萨电子株式会社
地址 日本东京

(72)发明人 铃木慎一

(74)专利代理机构 中国贸促会专利商标事务所有限公司 11038
代理人 欧阳帆

(51)Int.Cl.
G06F 11/22(2006.01)
G04D 7/00(2006.01)

(56)对比文件
JP 2004199323 A,2004.07.15
JP H1131022 A,1999.02.02
US 2007280299 A1,2007.12.06
CN 103423011 A,2013.12.04
JP H10124465 A,1998.05.15
JP H01227524 A,1989.09.11

审查员 刘锦英

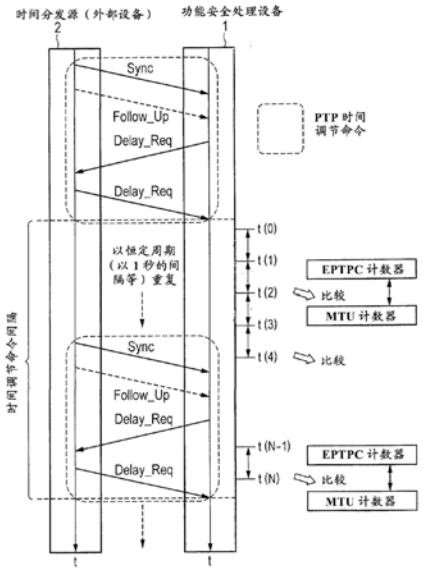
权利要求书2页 说明书16页 附图20页

(54)发明名称

半导体装置

(57)摘要

本发明涉及半导体装置。半导体装置包括包含第一计数器的第一计时器、包含第二计数器的第二计时器以及包含CPU的控制器以提供用于有效地诊断半导体装置(诸如,微控制器等)中内置的计时器的故障的技术。第一计时器执行与布置在半导体装置外部的外部设备的时间的时间同步。控制器将第一计数器的计数值与第二计数器的计数值进行比较以及基于比较结果检测第二计时器的故障。



CN 107229543 B

1.一种半导体装置,包括:

第一计时器,包括第一计数器以及执行与布置在所述半导体装置外部的外部设备的时间的时间同步;

第二计时器,包括第二计数器;以及

控制器,包括CPU,以及将所述第一计数器的计数值与所述第二计数器的计数值进行比较并且基于比较结果检测所述第二计时器的失灵,

其中所述第一计时器根据网络上的时间同步协议PTP执行时间同步并且将所述第一计数器的所述计数值调节至所述外部设备的时间,以及

其中所述控制器利用所述第一计时器基于所述时间同步生成的中断读出所述第一计数器的所述计数值和所述第二计数器的所述计数值,基于所读出的计数值得到所述第一计数器与所述第二计数器之间的计数器差,以及当所述计数器差大于第一预定值时判定所述第二计时器发生失灵。

2.根据权利要求1所述的半导体装置,

其中当所述第一计数器的所述计数值已经与第二预定值匹配时,所述第一计时器生成所述中断,以及

其中所述第二计数器基于所述中断开始计数以及基于在先前生成的中断之后生成的中断停止计数。

3.根据权利要求1所述的半导体装置,还包括:

I/O端口,

其中所述第一计时器基于所述第一计数器的所述计数值生成脉冲宽度调制信号,

其中所述I/O端口将所述脉冲宽度调制信号输出至所述半导体装置的外部,以及

其中所述第一计时器从所述外部设备接收同步命令并且在所述第一计时器与所述外部设备之间的时间差已经被更新时生成所述中断。

4.根据权利要求1所述的半导体装置,

其中,根据PTP选择在一件设备或者多件设备中的具有最精确时钟信号的设备的时间作为布置在所述半导体装置外部的所述外部设备的时间。

5.根据权利要求1所述的半导体装置,

其中所述第一预定值包括固定误差。

6.根据权利要求5所述的半导体装置,

其中基于充当所述第二计时器的振荡源的振荡器的精度和所述CPU的中断处理周期来设定所述第一预定值。

7.根据权利要求1所述的半导体装置,

其中当所述第二计数器溢出时,将溢出值与所读出的所述第二计数器的计数值相加。

8.根据权利要求1所述的半导体装置,

其中所述第二计时器包括多个计数器,通过将所述计数器级联地耦合在一起配置一个计数器以及由此增大溢出值。

9.一种半导体装置,包括:

第一计时器,包括第一计数器和脉冲输出计时器以及执行与布置在所述半导体装置外部的外部设备的时间的时间同步;

第二计时器,包括第二计数器、第三计数器和第四计数器;

第一I/O端口,输出由所述脉冲输出计时器生成的第一脉冲;

第二I/O端口,输出基于所述第二计数器的计数生成的第二脉冲;

第三I/O端口,输入从所述第一I/O端口输出的所述第一脉冲;

第四I/O端口,输入从所述第二I/O端口输出的所述第二脉冲;以及

控制器,包括CPU,

其中所述第三计数器对从所述第三I/O端口输入的所述第一脉冲的脉冲宽度进行计数,

其中所述第四计数器对从所述第四I/O端口输入的所述第二脉冲的脉冲宽度进行计数,以及

其中所述控制器将所述第三计数器的计数值与所述第四计数器的计数值进行比较并且基于比较结果检测所述第二计时器的失灵。

10.根据权利要求9所述的半导体装置,

其中所述第一计时器根据网络上的时间同步协议PTP执行时间同步以及将所述第一计数器指示的时间调节至所述外部设备的时间。

11.根据权利要求10所述的半导体装置,

其中所述脉冲输出计时器与所述第一计数器的所述计数值每次达到第二预定值同步地生成所述第一脉冲,

其中所述第二计时器与所述第一计数器的所述计数值每次达到第二预定值同步地生成所述第二脉冲,以及

其中所述控制器利用所述第一计时器生成的第一中断读出所述第三计数器的所述计数值,利用所述第二计数器生成的第二中断读出所述第四计数器的所述计数值,基于所读出的计数值得到所述第三计数器与所述第四计数器之间的计数器差,以及当所述计数器差大于第一预定值时判定所述第二计时器发生失灵。

12.根据权利要求11所述的半导体装置,

其中当所述脉冲输出计时器的所述脉冲上升时所述第一计时器生成所述第一中断,以及

其中当所述第二计数器的所述计数值与第三预定值匹配时,所述第二计时器生成所述第二中断。

13.根据权利要求10所述的半导体装置,

其中,根据PTP选择在一件设备或者多件设备中的具有最精确时钟信号的设备的时间作为布置在所述半导体装置外部的所述外部设备的时间。

14.根据权利要求11所述的半导体装置,

其中所述第一预定值包括固定误差。

15.根据权利要求14所述的半导体装置,

其中基于充当所述第二计时器的振荡源的振荡器的精度和根据所述PTP的时间同步误差来设定所述第一预定值。

半导体装置

[0001] 相关申请的交叉引用

[0002] 包括说明书、附图和摘要的2016年3月23日提交的日本专利申请No.2016-058614的公开的全部内容通过引用并入本文中。

技术领域

[0003] 本公开涉及半导体装置，以及适用于例如其中内置了计时器的类型的半导体装置。

背景技术

[0004] 微控制器被结合进了每件设备中，诸如家用电器、视听设备、蜂窝电话、汽车、工业机械等。微控制器是一种通过根据存储在存储器中的程序执行处理来控制每件设备的操作的半导体装置。要求包括要被结合进上面提到的(一件或多件)设备中的微控制器的组件依赖于其应用而具有可靠性。因此，除了通过要被控制的诊断传感器、致动器等来检测上述的传感器、致动器等中出现的故障以外，还要求微控制器检测微控制器自身中出现的故障。

[0005] 例如，在美国未经审查的专利申请公开No.2013/20978中公开了上面提到的半导体装置的一个示例。

发明内容

[0006] 本公开的目的是提供用于有效地诊断半导体装置中内置的计时器的故障的技术。

[0007] 本公开的其它主题和新颖特性将通过本说明书的描述和附图而变得清楚。

[0008] 如下将对本公开的代表性示例进行简单描述。

[0009] 即，根据本公开的一个实施例，提供有一种半导体装置，该半导体装置包括：第一计时器，包括第一计数器并且执行与布置在半导体装置外部的外部设备的时间的时间同步；第二计时器，包括第二计数器；以及控制器，包括CPU，以及将第一计数器的计数值与第二计数器的计数值进行比较并且基于比较结果检测第二计时器的失灵(和/或故障等)。

[0010] 根据本公开的一个实施例的半导体装置，可以有效地诊断内置计时器。

附图说明

[0011] 图1是例示根据本公开的第一实施例的微控制器的一个示例的框视图。

[0012] 图2是例示图1中所例示的EPTPC的一个示例的框图。

[0013] 图3A是例示时间同步的一个示例的图。

[0014] 图3B是例示EPTPC计数器与MTU计数器之间的关系的一个示例的时序图。

[0015] 图4是例示在诊断计时器的计数器的故障时执行的稳定处理操作的一个示例的流程图。

[0016] 图5A是例示在诊断计时器的计数器的故障时执行的时间匹配事件中断处理操作的一个示例的流程图。

[0017] 图5B是例示在诊断计时器的计数器的故障时执行的溢出中断处理操作的一个示例的流程图。

[0018] 图6是例示根据第二实施例的面向功能安全的工业电动机控制系统的一个示例的框图。

[0019] 图7是例示用于多个工业电动机的工业电动机控制系统的配置的一个示例的框图。

[0020] 图8是例示面向功能安全的工业电动机控制系统中的逻辑组成的一个示例的框图。

[0021] 图9是例示在诊断关于计时器生成PWM波形信号的情形的计数器的故障时执行的稳定处理操作的一个示例的流程图。

[0022] 图10A是例示在诊断关于计时器生成PWM波形信号的情形的计数器的故障时执行的PTP命令接收中断处理操作的一个示例的流程图。

[0023] 图10B是例示在诊断关于计时器生成PWM波形信号的情形的计数器的故障时执行的比较匹配中断处理操作的一个示例的流程图。

[0024] 图11是例示在计时器生成PWM波形信号的情形下的计数器的操作的一个示例的时序图。

[0025] 图12是例示根据第三实施例的工业电动机系统的一个示例的框图。

[0026] 图13是例示在PWM波形比较中执行的示意性操作的一个示例的时序图。

[0027] 图14是例示在PWM波形比较中执行的稳定处理操作的一个示例的流程图。

[0028] 图15A是例示在PWM波形比较中执行的上升沿中断处理操作的一个示例的流程图。

[0029] 图15B是例示在PWM波形比较中执行的比较匹配中断处理操作的一个示例的流程图。

[0030] 图16是例示根据第四实施例的32位计数器的操作的一个示例的时序图。

具体实施方式

[0031] 在下文中，将参考附图对本公开的实施例和实际示例进行描述。然而，在下列描述中，存在为相同的构成元件指定相同的附图标记并且省略其重复描述的情况。

[0032] 在要被装载在面向功能安全的设备(诸如工业设备、汽车相关产品等)上的微控制器中，除了检测CPU、内置存储器等的故障以外，检测外围模块(诸如多功能计时器、中断控制器、AD转换器等)中发生的故障也至关重要。特别地，多功能计时器的计数器的电路规模是大的，因此在多功能计时器中容易发生失灵。另外，多功能计时器的设计是复杂的，以便应对溢出、跳跃时间等。在故障中存在永久不可能恢复的硬错误以及暂时出现并且可以恢复的软错误。例如，装载在微控制器上的MTU(多功能计时器脉冲单元)的软错误发生概率比其它外围模块高，以及因此对多功能计时器的有效故障诊断是亟待解决的问题。

[0033] 本申请的发明人和其他人已经关于内置计时器的诊断对下列方法进行了检查。

[0034] (1)使用验证程序的诊断

[0035] 通过实际应用在设计内置计时器时所使用的验证程序来确认诸如开始、递增计数、停止等的基本操作。

[0036] (2)使用另一个内置计时器的诊断

[0037] 为了验证操作与要验证的内置计时器不同的另一个内置计时器并且将这些内置计时器中所包括的计数器的计数值彼此比较以便得到计数器差。

[0038] 然而,在方法(1)中,只可以确认逻辑开/关操作和单个操作,而难以确认例如将计数器等的定量值的检测、PWM(脉冲宽度调制)波形信号的生成等组合在一起的组合操作。在方法(2)中,当充当要被验证的内置计时器和另一个内置计时器的计数操作的基础的振荡器在这些内置计时器之间共用时,在内置计时器两者都发生失灵时可能不会检测到故障的可能性是高的。另外,在方法(1)和(2)中,要求通过暂时停止执行用户程序以进行诊断,因此降低了用户程序的处理效率。

[0039] 在一个实施例中,同步以太网控制器以特定的时间间隔来操作和停止多功能计时器的计数器,并且通过将多功能计时器的计数器的计数值与其时间已经根据同步以太网被调节的计数器的计数值进行比较来诊断多功能计时器的故障。

[0040] 根据一个实施例,由于使用了外部时间,因此提高了多功能计时器的故障检测率。另外,可以与时间同步序列的执行并行地执行多功能计时器故障诊断以及消除用户应用程序的执行的暂时停止。

[0041] [第一实施例]

[0042] 将参照图1对根据第一实施例的微控制器进行描述。图1是例示根据第一实施例的微控制器的配置的一个示例的框图。根据第一实施例的微控制器10包括只读存储器(ROM)11、随机存取存储器(RAM)12、中央处理单元(CPU)13等。另外,微控制器10还包括IEEE1588控制器(以下称为EPTPC)14、多功能计时器(MTU)15、ICU(中断控制器单元)16、ELC(事件链路控制器)17、I/O端口18等。微控制器10是由一个半导体芯片(半导体衬底)形成的半导体装置。ROM 11和CPU 13也称作控制器。附带地,RAM 12和ICU 16也可以包括在控制器中。将基于IEEE1588标准(参见IEEE1588-2008Ver2.0(IEEE1588同步以太网标准))的时间同步协议(以下称为PTP(精确时间协议))用于MTU 15的故障诊断。

[0043] ROM 11将用于检测MTU 15的故障的程序存储在自身中。ROM 11由非易失性存储器(诸如,例如,闪存存储器等)构成。RAM 12将用于执行程序的工作数据存储在自身中。RAM 11由易失性存储器(诸如,例如,SRAM等)构成。除了根据已经从ROM 12读出的程序执行用于所涉及的设备的控制操作、通信处理等的用户应用的处理以外,CPU 13还执行用于检测MTU 15的故障的处理(故障判定)和用于恢复MTU 15的处理。EPTPC 14通过使用以太网线路以及根据PTP来执行与外部设备(时间分发源)2的时间的时间同步。MTU 15是多功能计时器,该多功能计时器包括多个通道并且其中每个通道都具有多个功能,诸如,除了通过对值进行递增计数以及对值进行递减计数所执行的计数操作以外,还有生成和输出PWM波形信号的功能、输入捕获功能、通过使用PWM波形生成和输出功能以及输入捕获功能来控制工业电动机的功能等。ICU 16向CPU 13通知来自EPTPC 14和MTU 15的中断请求。ELC 17通过耦合事件信号来减小由软件干预引起的延迟,使得每当EPTPC 14的计数器达到特定的时间时就开始操作MTU 15。

[0044] 接着,将通过使用图2对EPTPC 14进行描述。图2是例示图1中的EPTPC 14以及与EPTPC 14有关的硬件的配置的一个示例的框图。

[0045] EPTPC 14包括第0通道PTP协议处理单元(CH0)141、第一通道PTP协议处理单元(CH1)142、分组中继单元143、时钟(时间)校正单元144等。第0通道PTP协议处理单元(CH0)

141和第一通道PTP协议处理单元(CH1)142执行协议处理,诸如PTP帧传输-接收处理、命令序列处理等。分组中继单元143对已经由第0通道PTP协议处理单元(CH0)141和第一通道PTP协议处理单元(CH1)142处理的以太网帧(以下称为帧)执行中继和校正处理。时钟(时间)校正单元144包括充当EPTPC 14的计数器的本地时钟计数器1441,并且具有校正时间以便被调节至时间分发源的时间的功能、在特定时间经由ICU 16向CPU 13通知中断的接收的功能。

[0046] 通常,EPTPC 14通过将第0通道ETHERC(以太网控制器)(CH0)21和第一通道ETHERC(CH1)22、第0通道EDMAC(用于以太网控制器的DMA控制器)(CH0)23和第一通道EDMAC(CH1)24以及PTPEDMAC 25组合在一起来使用它们。第0通道ETHERC(CH0)21和第一通道ETHERC(CH1)22执行与MAC层相对应的处理。第0通道EDMAC(CH0)23和第一通道EDMAC(CH1)24每个都适于有效地处理和管理标准帧并且充当与CPU 13的接口。PTPEDMAC 25处理PTP的帧。EPTPC 14经由执行数字信号到模拟信号的转换的物理层(PHY)(未例示)和以太网连接器(通常,RJ-45)7相对于以太网线缆5输入/输出标准/PTP帧。另外,EPTPC 14经由ELC 17耦合至MTU 15和I/O端口18并且具有在已经根据PTP与外部设备的时间同步的时间处开始对MTU 15的计时器计数器的值进行计数的功能以及稍后描述的脉冲输出功能(脉冲输出计时器功能)。尽管,在图2中,例示了将第0通道PTP协议处理单元(CH0)141和第一通道PTP协议处理单元(CH1)142、第0通道ETHERC(CH0)21和第一通道ETHERC(CH1)22以及第0通道EDMAC(CH0)23和第一通道EDMAC(CH1)24装载在微控制器10上的双通道配置,但是微控制器10可以具有单通道配置。

[0047] 接着,将使用图3A和图3B对用于故障诊断的MTU 15的配置的一个示例进行描述。图3A是例示在面向功能安全的设备与作为时间分发源的外部设备之间的时间同步的一个示例的图形。图3B是例示EPTPC计数器与MTU计数器之间的关系的一个示例的时序图。

[0048] 图3A和图3B中用于故障诊断的配置涉及根据PTP执行与外部设备2的时间的时间同步并且通过MTU 15对计数器的值进行计数的情况,该外部设备2是在装载了微控制器10的面向功能安全的设备1上的时间分发源。

[0049] (11)面向功能安全的设备1的微控制器10基于PTP将故障检测的时间间隔设定为时间匹配事件的间隔。

[0050] (12)微控制器10以用于根据PTP的固定时间调节发出的命令的间隔(此处,将间隔设定为一秒)开始与作为时间分发源的外部设备2的时间的时间同步。用于基于PTP的时间调节的命令为Sync、Follow_Up、Delay_Req、Delay_Resp等。

[0051] (13)微控制器10在EPTPC 14的本地时钟计数器1441的时间t(1)处开始MTU的计数器的递增计数。

[0052] (14)每当在(11)中设定的基于PTP的时间匹配事件发生时(t(2)、t(4)、……t(N)),微控制器10就将已经与外部设备同步的EPTPC计数器的增益程度与MTU计数器的增益程度进行比较。

[0053] (15)当计数器差的绝对值超过在上面的(14)中已经预先限定的阈值的范围时,微控制器10将其检测为计数器的故障。另外,当绝对值在阈值的范围内时,微控制器10继续在上面的(14)中的每当时间同步时对计数器的增益程度进行比较。

[0054] 图3B中的虚线指示EPTPC计数器与MTU计数器之间的计数值没有差(计数器差=0)

的情况，以及实线指示MTU计数器领先于EPTPC计数器(存在计数器差)的情况。在图3B中的示例中，尽管时间t(2)处的计数器差(D(2))、时间t(4)处的计数器差(D(4))……等在阈值范围内，但是时间t(N)处的计数器差(D(N))在阈值范围之外以及检测到故障。

[0055] 接着，将使用图4、图5A和图5B对用于计时器计数器的故障诊断的软件操作流程进行描述。图4是例示在诊断计时器计数器的故障时执行的稳定处理操作的一个示例的流程图。图5A是例示在诊断计时器计数器的故障时执行的时间匹配事件中断处理操作的一个示例的流程图。图5B是例示在诊断计时器计数器的故障时执行的溢出中断处理操作的一个示例的流程图。此处，使用从EPTPC 14生成的时间匹配事件中断和从MTU 15生成的溢出中断作为中断(即，中断信号)。当EPTPC 14的本地时钟计数器1441的值已经匹配了特定设定值时，从EPTPC 14生成时间匹配事件中断并且与中断的生成同时地开始对MTU 15的计时器计数器的值的计数。当计时器计数器(例如，16位计数器)的值已经溢出时，从MTU 15生成溢出中断。

[0056] <稳定处理>

[0057] 步骤S11：在已经开始用户应用之后，CPU 13开始执行用于计时器故障诊断的软件。随后，CPU 13执行对MTU 15的预定标器(提供至MTU 15的时钟信号与MTU 15的计时器计数器的输出的频率比)、计数操作模式(自由运行模式、比较匹配模式、PWM波形输出模式等)和MTU计数器的中断、初始化(mtu_count←0)等的设定。

[0058] 步骤S12：CPU 13执行EPTPC 14的同步模式、时间匹配事件、时间匹配事件发生时间等的设置。

[0059] 步骤13：CPU 13设定ELC 17以便将EPTPC 14的时间匹配事件耦合至MTU 15的启动因素。

[0060] 步骤S14：CPU 13设定ICU 16以便向CPU 13通知从MTU 15生成的溢出中断和从EPTPC 14生成的时间匹配事件中断。另外，CPU 13对时间匹配事件标志进行初始化(ptp_flag←假)。

[0061] 步骤S15：CPU 13将用于故障检测判定的MTU计数器(mtu_count)与EPTPC计数器(ptp_count)之间的计数值的差的绝对值的上限值设定为阈值。通过将由于振荡器的存在将出现的MTU 15的计时器计数器的误差、由中断处理引起的延迟、PTP的协议操作上的误差等考虑在内来设定阈值(thresh)。将通过稍后描述的公式(2)表明阈值的一个示例。此处，MTU计数器(mtu_count)和EPTPC计数器(ptp_count)是用于故障检测中的判定的软件元件而不是硬件元件，即，不是硬件的MTU 15的计时器计数器和EPTPC 14的本地时钟计数器1441。

[0062] 步骤S16：CPU 13通过使用EPTPC 14根据PTP开始与作为时间分发源的外部设备2的时间的时间同步。此处，根据PTP从一件设备或者多件设备中选择具有最准确的时钟信号的一件设备作为外部设备2(时间分发源)。

[0063] 步骤S17：CPU 13确认时间匹配事件是否发生。使用时间匹配事件标志(ptp_flag)判定时间匹配事件是否发生。在是的情况下(时间匹配事件已经发生)(ptp_flag＝真)，CPU 13进行到步骤S18中的处理，以及在否的情况下，CPU 13进行到步骤S1B中的处理。附带地，在用于稍后描述的中断处理的步骤S28中设定时间匹配事件标志(ptp_flag)。

[0064] 步骤S18：CPU 13确认计数器差的绝对值是否超过阈值。计数器差是通过从MTU计

数器(mtu_count)减去已经以MTU 15的操作频率(f_{mtu})与EPTPC 14的操作频率(f_{ptp})的比率加权的EPTPC计数器(ptp_count)得到的差。

[0065] 计数器差＝|mtu_count-(f_{mtu}/f_{ptp})ptp_count|

[0066] 当计数器差超过阈值(是)时,CPU 13将其检测为故障并且以错误终止处理。当计数器差没有超过阈值(否)时,CPU 13进行到步骤S19中的处理。

[0067] 步骤S19:CPU 13清空时间匹配事件标志(ptp_flag←假)。

[0068] 步骤S1A:CPU 13清空MTU计数器(mtu_count←0)。

[0069] 步骤S1B:CPU 13判定用户应用是否终止。当用户应用终止(是)时,CPU 13终止处理。当用户应用未终止(否)时,CPU 13回到步骤S17以及继续执行处理。

[0070] <中断处理>

[0071] 以下,将对时间匹配事件中断处理进行描述。

[0072] 步骤S21:当已经从EPTPC 14生成时间匹配事件中断时(t(0)、t(1)、t(3)、t(4)、……t(N-1)和T(N)),CPU 13通过将指示时间匹配事件中断处理发生频率的中断计数器的值增加一(N←N+1)来执行更新。

[0073] 步骤S22:CPU 13判定中断是否为奇数中断(t(1)、t(3)、……t(2n-1))。该判定通过参照中断计数器(N)来进行。在奇数中断(是)的情况下,CPU 13进行到步骤S23中的处理。在偶数中断(否)的情况下,CPU 13进行到步骤S25中的处理。

[0074] 步骤S23:MTU 15的计时器计数器开始递增计数操作。此处,尽管通过由ELC 17耦合事件信号来开始MTU 15的计时器计数器的递增计数操作并且因此不要求通过CPU 13执行软件处理,但是为了便于说明操作将对软件处理进行描述。附带地,当不使用ELC 17时,要求通过CPU 13执行软件处理。

[0075] 步骤S24:CPU 13读出EPTPC 14的本地时钟计数器1441的计数值(LCCV)并且将所读出的计数值设定为EPTPC计数器的起始值(ptp_start←LCCV)。随后,CPU 13进行到步骤S29中的处理。

[0076] 步骤S25:CPU 13终止MTU 15的计时器计数器的递增计数操作。

[0077] 步骤S26:CPU 13读出MTU 15的计时器计数器的计数值(TCNT)并且使用所读出的计数值更新MTU计数器的值。所更新的值是通过将MTU 15的计时器计数器的计数值与MTU计数器的当前值相加而得到的值(mtu_count←mtu_count+TCNT)。在某些情况下,用在步骤S31中从MTU 15生成的溢出中断来更新MTU计数器的值。

[0078] 步骤S27:CPU 13读出EPTPC 14的本地时钟计数器1441的计数值并且通过使用所读出的计数值来更新EPTPC计数器的值。所更新的值是通过从EPTPC 14的本地时钟计数器1441的计数值减去步骤S24中设定的EPTPC计数器的起始值而得到的值(ptp_count←LCCV-ptp_start)。

[0079] 步骤S28:CPU 13设定时间匹配事件标志(ptp_flag←真)。

[0080] 步骤S29:CPU 13将要生成下一个时间匹配事件的时间设定到EPTPC 14。附带地,该时间可以是在时间间隔方面与前一时间相同的时间,以及用于生成下一个时间匹配事件的时间的时间间隔可以相对于前一时间的时间间隔有所改变。

[0081] 接着,以下将对溢出中断进行描述。

[0082] 步骤S31:当已经从MTU 15生成溢出中断时,CPU 13将MTU计数器的值更新为通过

将溢出值(在16位计数器的情况下,2^{16}=65,536)与MTU计数器的当前值相加而得到的值(mtu_count←mtu_count+溢出值)。

[0083] <阈值的一个示例>

[0084] 将对阈值(thresh)的一个示例进行描述。当阈值被分成由因为振荡器的存在而导致的MTU 15的计时器计数器的误差引起的“$thresh_{osc}$”、由PTP的协议操作上的误差引起的“$thresh_{ptp}$”、由中断处理引起的“$thresh_{int}$”以及由将依赖于CPU 13的流水线和总线的状态而出现的处理中的波动引起的“$thresh_{flu}$”时,确立公式(1)。

[0085] $thresh=thresh_{osc}+thresh_{ptp}+thresh_{int}+thresh_{flu}$……(1)

[0086] 此处,忽略诸如温度特性等的其它因素的贡献,因为这充分小于上面提到的值的贡献。

[0087] 当MTU 15的操作频率为f_{mtu},为MTU 15提供时钟信号的振荡器的精度为x_{osc}并且时间匹配事件间隔为t_{pl}时,通过MTU计数器的计数数值的公式(1-1)表示由因为振荡器的存在而导致的MTU 15的计时器计数器的误差引起的“$thresh_{osc}$”。

[0088] $thresh_{osc}=f_{mtu}*x_{osc}*t_{pl}$……(1-1)

[0089] 例如,当f_{cpu}=120MHz,通过预定标器将MTU 15的频率除以16,$f_{mtu}=f_{cpu}/16$=7.5MHz,x_{osc}=100ppm,以及t_{pl}=100ms时,得到$thresh_{osc}$=75个周期。

[0090] PTP的协议操作上的误差包括与MTU 15的误差类似的向EPTPC 14提供时钟信号的振荡器的误差、用于时间调节的命令间隔、将取决于以太网上的总线状态而出现的命令延迟等。当已经将EPTPC 14并入微控制器10中时,通过由硬件元件执行协议操作并且由时钟(时间)校正单元144校正时间以便将时间调节至作为时间分发源的外部设备2的时间,将时间误差降低至1μs至100ns变为可能。在这种情况下,“$thresh_{ptp}$”相当于MTU计数器的计数数值中的10个周期到1个周期,并且这意味着$thresh_{ptp}<thresh_{osc}$。

[0091] 当CPU 13的操作频率为f_{cpu},中断处理出现频率为x_{int}并且中断处理的周期数为n_{int}时,由MTU计数器的计数数值的公式(1-2)表示由中断处理引起的“$thresh_{int}$”。

[0092] $thresh_{int}=(f_{mtu}/f_{cpu})*(x_{int}*n_{int})$……(1-2)

[0093] 此处,使得时间匹配事件中断处理与溢出中断处理在CPU 13的操作频率处的周期数近似相同。

[0094] 例如,假设f_{cpu}=120MHz并且$f_{mtu}=f_{cpu}/16$=7.5MHz。在中断处理中,通过另外加上中断响应时间与恢复时间、将中断响应时间设定为10个周期、将中断恢复时间设定为10个周期并且将中断处理设定为平均80个周期而得到n_{int}=100个周期。16位计数器的溢出每8.7ms(=65,536/7.5*10^6s)发生一次。因此,当执行了11次(=100/8.7)溢出中断处理时,通过加上已经执行了两次的时间匹配事件中断处理而确立x_{int}=13次以及得到$thresh_{int}$=81个周期。

[0095] “$thresh_{flu}$”是微控制器10的CPU 13执行的处理中的波动的上限值并且小于“$thresh_{int}$”,以及因此$thresh_{flu}<thresh_{int}$。

[0096] “$thresh_{osc}$”是误差的上限值,而“$thresh_{int}$”是中断处理的平均值。尽管“$thresh_{int}$”的贡献大于“$thresh_{osc}$”的贡献,但是将其应用为故障检测中的最差值。由上述可知,公式(1)近似为公式(2)。

[0097] $Thresh=thresh_{osc}+thresh_{ptp}+thresh_{int}+thresh_{flu}<thresh_{osc}+thresh_{osc}+$

$thresh_{int}+thresh_{int}=(f_{mtu}/f_{cpu})*(2*f_{cpu}*x_{osc}*t_{pl}+2*x_{int}*n_{int})$ ……(2)

[0098] 当将上面提到的数值应用于公式(2)时，得到thresh＝312个周期。

[0099] 在第一实施例中，根据EPTPC 14(第一计时器)的本地时钟计数器(第一计数器)1441的值对MTU 15(第二计时器)的计时器计数器(第二计数器)的故障进行诊断，其中EPTPC 14的本地时钟计数器1441的时间已经被调节至作为时间分发源的外部设备2的时间。

[0100] 根据第一实施例，由于可以根据PTP将多个外部时间用于诊断故障，因此不存在由故障诊断中共用的振荡器的异常所引起的失灵并且可以基于高度准确的时间诊断计时器的故障。另外，由于可以与基于PTP的时间同步操作的执行并行地执行计时器的故障诊断，因此，相较于通过停止执行用户程序而执行计时器的故障诊断的情况，可以更为抑制用户程序处理效率的降低。

[0101] [第二实施例]

[0102] 将使用图6对将微控制器应用于面向功能安全的工业设备的情况进行描述。图6是例示根据第二实施例的工业电动机控制系统的配置的一个示例的框图。

[0103] 微控制器10A安装在面向功能安全的设备1A上、控制面向功能安全的设备1A的操作、控制工业电动机3的操作以及与在网络上与以太网线缆5耦合的其它件工业设备4(诸如机器人、AC伺服电动机、机械工具等)进行通信。附带地，在功能安全标准(IEC61508第7部分)等中，要求根据符合度水平相互监控以及存在在面向功能安全的设备1A上安装两个微控制器(微控制器10A和微控制器20A)的情况。

[0104] 要求由存储器管理单元(MMU)、存储器保护单元(MPU)等通过将布置了ROM 11和RAM 12上的安全数据的区域与布置非安全数据的区域分开以应付功能安全来保护该区域。因此，除根据第一实施例的微控制器10的配置以外，还将未例示的存储器管理单元(MMU)和/或存储器保护单元(MPU)装载在微控制器10A上。微控制器10A和工业电动机3经由电动机驱动器6耦合在一起。随后，通过根据由MTU 15生成并且从I/O端口18输出的PWM波形信号控制电动机驱动器6来控制工业电动机3。

[0105] 将使用图7对将图6中面向功能安全的设备并入用于多个工业电动机的工业电动机控制系统中的情况进行描述。图7是例示在用于多个工业电动机的工业电动机控制系统的配置的一个示例的框图。作为多件面向功能安全的设备的从设备B_1(1A1)、从设备B_2(1A2)、……以及从设备B_{N-1}(1AN-1)根据PTP与作为时间分发源的主设备A(2A)时间同步以便在特定的时间处A同时开始N(N＝2、3、……)个工业电动机3A、3A1、3A2、……、3AN-1的操作并且同时诊断每件面向功能安全的设备上装载的MTU 15。

[0106] 图8是在将第二实施例应用于面向功能安全的设备时的逻辑组成的一个示例的图。尽管在第一实施例中的图2中例示了与EPTPC相关的硬件具有双通道配置的情况，但是此处，将对单通道配置进行描述。

[0107] 通信应用和控制应用处理安全数据和非安全数据，并且将要布置在硬件中的处理细节、总线和存储器相互分开。因此，准备并且使用安全数据所特有的一个软件和非安全数据所特有的另一个软件。另外，具有时间同步功能和计时器诊断功能的应用(以下称为同步计时器诊断应用)处理安全数据，并且处理非安全数据的时间同步应用也装载在系统上。

[0108] 通信应用是用于工业以太网的通信应用，以及以太网/IP、PROFINET、EtherCAT等

可用作代表性的工业以太网标准。控制应用适于控制已经被并入AC伺服电动机、机器人等中的工业电动机的操作。TCP和UDP/IP各自执行逻辑通信耦合管理、基于TCP或者基于UDP的分组传输-接收处理、耦合设备信息的管理等。另外，由于在PTP中仅处理基于UDP的分组或者帧，因此可以使用仅安装了UDP/IP的中间件和/或可以不安装中间件。以太网驱动器通过使用ETHERC 21和EDMAC 23执行帧传输-接收处理、以太网线缆5的耦合和检测等。ERPTC驱动器通过使用EPTPC 14执行基于PTP的时间同步。MTU驱动器通过使用MTU 15基于MTU 15的定时信号生成PWM波形信号，该PWM波形信号是用于工业电动机3的控制信号。随后，MTU驱动器经由I/O端口18输出PWM波形信号。由于在第一实施例中已经对ETHERC 21和EDMAC 23进行了描述，因此此处省略其描述。

[0109] 将使用图9、图10A、图10B和图11对在根据第二实施例的时间同步计时器诊断应用中将PWM操作模式1用作MTU 15的操作模式时执行的操作流程进行描述。图9是例示在诊断关于计时器生成PWM波形信号的情况的计数器的故障时执行的稳定处理操作的一个示例的流程图。图10A是例示在诊断关于计时器生成PWM波形信号的情况的计数器的故障时执行的PTP命令接收中断处理操作的一个示例的流程图。图10B是例示在诊断关于计时器生成PWM波形信号的情况的计数器的故障时执行的比较匹配中断处理操作的一个示例的流程图。图11是例示在计时器生成PWM波形信号时MTU的计时器计数器的操作的一个示例的时序图。此处，使用从EPTPC 14生成的PTP命令接收中断和从MTU 15生成的比较匹配中断作为中断。当EPTPC 14已经从时间分发源接收了基于PTP的同步命令并且已经更新了计数器与时间分发源之间的时间差(offsetFromMaster)时，生成PTP命令接收中断。当在如图11所例示的PWM波形信号的周期处清空了包括在MTU 15中并且生成PWM波形信号的计时器计数器时，从MTU 15生成比较匹配中断。

[0110] <稳定处理>

[0111] 步骤S41：在启动用户应用之后，CPU 13启动时间同步计时器诊断应用。随后，CPU 13执行MTU 15的预定标器的设定、计数操作模式(PWM操作模式1)的设定、PWM波形信号的周期的设定、中断的设定、MTU计数器的初始化(mtu_count←0)等。

[0112] 步骤S42：CPU 13执行EPTPC 14的同步模式的设定、PTP命令接收中断的设定等。

[0113] 步骤S43：CPU 13设定ICU 6以便向CPU 13通知从MTU 15接收比较匹配中断以及从EPTPC 14接收PTP命令接收中断。另外，CPU 13对PTP命令接收标志进行初始化(ptp_flag←假)。

[0114] 步骤S44：CPU 13将用于故障检测判定的MTU计数器(mtu_count)与EPTPC计数器(ptp_count)之间的值的差的绝对值的上限值设定为阈值。通过将由振荡器的存在引起的MTU 15的计时器计数器的误差、由中断处理引起的延迟、PTP的协议操作上的误差等考虑在内来设定阈值(thresh)。将通过稍后描述的公式(4)指示阈值的一个示例。

[0115] 步骤S45：CPU 13通过使用EPTPC 14以及根据PTP开始与作为时间分发源的外部设备2的时间的时间同步。此处，根据PTP从一件或多件设备中选择具有最准确的时钟信号的一件设备作为时间分发源。

[0116] 步骤S46：CPU 13读出EPTPC 14的本地时钟计数器1441的计数值并且将所读出的计数值设定为EPTPC计数器的起始值(ptp_start←LCCV)。

[0117] 步骤S47：CPU 13指示开始MTU 15的PWM波形信号的输出。另外，如在第一实施例中

的步骤S23中描述的，还可以通过由ELC 17耦合事件信号来开始MTU 15的PWM波形信号的输出。此处，将使用图11对MTU 15的操作进行描述。MTU 15的计时器计数器对值进行递增计数并且在当计数值(TCNT)达到TGRD时，将来自I/O端口18的输出从低反相至高。计时器计数器继续对值进行递增计数，并且在当计数值达到TGRC时，将来自I/O端口18的输出从高反相至低并且向CPU 13通知比较匹配中断的生成。计时器计数器通过重复地执行上面提到的操作输出来自I/O端口18的PWM波形信号。在图11中，例示了周期400μs、脉冲宽度200μs以及占空比50％的PWM波形信号的一个示例。

[0118] 步骤S48:CPU 13确认PTP命令接收是否发生。用PTP命令接收标志(ptp_flag)来判定PTP命令接收是否发生。在PTP命令接收发生(是)的情况下，CPU 13进行到步骤S49中的处理。在PTP命令接收没有发生(否)的情况下，CPU 13进行到步骤S4C中的处理。

[0119] 步骤S49:CPU 13确认计数器差的绝对值是否超过阈值。计数器差是通过从MTU计数器(mtu_count)减去已经以MTU 15的操作频率(f_{mtu})与EPTPC 14的操作频率(f_{ptp})的比率加权的EPTPC计数器(ptp_count)得到的差。

[0120] 计数器差＝|mtu_count-(f_{mtu}/f_{ptp})ptp_count|

[0121] 当计数器差超过阈值(是)时，CPU 13将其检测为故障并且以错误终止处理。当计数器差没有超过阈值(否)时，CPU 13进行到步骤S4A中的处理。

[0122] 步骤S4A:CPU 13清空PTP命令接收标志(ptp_flag←假)。

[0123] 步骤S4B:CPU 13清空MTU计数器(mtu_count←0)。

[0124] 步骤S4C:CPU 13判定用户应用是否终止。当用户应用终止(是)时，CPU 13终止处理。当用户应用未终止(否)时，CPU 13回到步骤S48以及继续执行处理。

[0125] 〈中断处理〉

[0126] 以下，将对PTP命令接收中断进行描述。

[0127] 步骤S51:当已经从EPTPC 14生成了PTP命令接收中断时，CPU 13读出EPTPC 14的本地时钟计数器1441的计数值并且使用所读出的计数值更新EPTPC计数器的值。所更新的值是通过从EPTPC 14的本地时钟计数器1441的计数值(LCCV)减去已经在先前的PTP命令接收中断处理或者稳定处理中设定的PTP计数器的起始值而得到的值(ptp_count←LCCV-ptp_start)。

[0128] 步骤S52:CPU 13读出MTU 15的计时器计数器的计数值(TCNT)并且使用所读出的计数值更新MTU计数器的值。所更新的值是通过将MTU 15的计时器计数器的计数值(TCNT)与MTU计数器的当前值相加得到的值(mtu_count←mtu_count+TCNT)。

[0129] 步骤S53:CPU 13读出EPTPC 14的本地时钟计数器1441的计数值并且将所读出的计数值设定为EPTPC计数器的起始值(ptp_start←LCCV)。

[0130] 步骤S54:CPU 13设定PTP命令接收标志(ptp_flag←真)。

[0131] 接着，以下将对比较匹配中断进行描述。

[0132] 步骤S61:当已经从MTU 15生成了比较匹配中断时，CPU 13将MTU计数器的值更新至通过将与PWM波形信号的周期相对应的计数值(PWMP)与MTU计数器的当前值相加得到的值(mtu_count←mtu_count+PWMP)。

[0133] 〈阈值的一个示例〉

[0134] 将对阈值(thresh)的一个示例进行描述。与第一实施例类似，当阈值被分成由因

为振荡器的存在而导致的MTU 15的计时器计数器的误差引起的“$thresh_{osc}$”、由PTP的协议操作上的误差引起的“$thresh_{ptp}$”、由中断处理引起的“$thresh_{int}$”以及由将依赖于CPU 13的流水线和总线的状态而出现的处理中的波动引起的“$thresh_{flu}$”时，得到公式(3)。

[0135] $thresh = thresh_{osc} + thresh_{ptp} + thresh_{int} + thresh_{flu}$ ……(3)

[0136] 当CPU 13的操作频率为f_{cpu}，MTU 15的操作频率为$f_{mtu} = f_{cpu}/16$，向MTU 15提供时钟信号的振荡器的精度为x_{osc}并且PTP命令接收间隔为tp2时，由MTU计数器的计数数值的公式(3-1)表示由因为振荡器的存在而导致的MTU 15的计时器计数器的误差引起的“$thresh_{osc}$”。

[0137] $thresh_{osc} = f_{mtu} * x_{osc} * t_{p2}$ ……(3-1)

[0138] 例如，当$f_{cpu} = 120MHz$，$f_{mtu} = f_{cpu}/16 = 7.5MHz$，$x_{osc} = 100ppm$，并且$t_{p2} = 1s$时，得到$thresh_{osc} = 750$个周期。

[0139] 与第一实施例类似，PTP的协议操作上的误差近似为$thresh_{ptp} < thresh_{osc}$。

[0140] 当CPU 13的操作频率为f_{cpu}，中断处理出现频率为y_{int}并且中断处理的周期数为n_{int}时，由MTU计数器的计数数值的公式(3-2)表示由中断处理引起的“$thresh_{int}$”。

[0141] $thresh_{int} = (f_{mtu}/f_{cpu}) * (y_{int} * n_{int})$ ……(3-2)

[0142] 此处，使得PTP命令接收中断处理与比较匹配中断处理在CPU 13的操作频率处周期数近似相同。

[0143] 例如，假设$f_{cpu} = 120MHz$以及$f_{mtu} = f_{cpu}/16 = 7.5MHz$。在中断处理中，通过另外加上中断响应时间和恢复时间、将中断响应时间设定为10个周期、将中断恢复时间设定为10个周期并且将中断处理设定为平均80个周期以得到$n_{int} = 100$个周期。当因为每400μs从16位计数器生成一次比较匹配中断所以$y_{int} = 2500$次时，得到$thresh_{int} = 15625$个周期。

[0144] 由上述可知，确立$thresh_{int} >> thresh_{osc} > thresh_{ptp}$。“$thresh_{flu}$”是微控制器10A的CPU 13的处理中的波动的上限值，以及与第一实施例类似地，“$thresh_{flu}$”小于“threshint”并且公式(3)近似为公式(4)。

[0145] $thresh = thresh_{int} + thresh_{flu} < thresh_{int} + thresh_{int} = 2 * (f_{mtu}/f_{cpu}) * (y_{int} * n_{int})$ …(4)。

[0146] 当将上面提到的数值应用于公式(4)时，得到thresh＝31250个周期。

[0147] 在第二实施例中，通过由使用MTU 15的计时器计数器输出PWM波形信号来控制工业电动机，以及通过使用EPTPC 14的本地时钟计数器1441与上面提到的操作的执行并行地诊断MTU 15的计时器计数器。

[0148] 根据第二实施例，可以对计时器生成PWM波形信号时将出现的计数器的失灵进行诊断。另外，可以将根据第二实施例的系统应用于工业电动机控制设备以及面向功能安全的设备(符合诸如IEC 61508等的标准)。

[0149] [第三实施例]

[0150] 图12是例示根据第三实施例的工业电动机控制系统的配置的一个示例的框图。图13是例示在PWM波形比较中EPTPC 14的脉冲输出计时器和MTU 15的计时器计数器的操作的一个示例的时序图。

[0151] 已经基于两种类型的计时器(EPTPC 14和MTU 15)生成的第一脉冲和第二脉冲从第一端口(PORT1)18_1和第二端口(PORT2)18_2分别地输出并且分别地经由电动机驱动器

6B1和6B2分别地耦合至第一工业电动机3B1和第二工业电动机3B2。另外，经由布置在微控制器10B外部的耦合电路8使第一脉冲和第二脉冲分支并且将第一脉冲和第二脉冲分别地输入到作为其它通用端口的第三端口(PORT3)18_3和第四端口(PORT4)18_4中。随后，通过对第一脉冲和第二脉冲的脉冲宽度进行测量并且相互比较来检测计时器的故障。可替换地，可以布置可从外部操作的物理开关，由此在控制电动机时可以将PWM波形信号输出到第一工业电动机3B1和第二工业电动机3B2以及在进行诊断时可以将PWM波形信号输入到第三端口18_3和第四端口18_4中，代替脉冲分支。此处，第一端口18_1、第二端口18_2、第三端口18_3以及第四端口18_4是构成I/O端口18的部分的I/O端口。

[0152] 第一脉冲通过使得EPTPC 14的脉冲输出计时器的上升沿经由开关输出以生成PWM波形信号。第二脉冲使用MTU 15的通道0(MTU0)的PWM波形生成功能(PWM操作模式1)生成PWM波形并且与EPTPC 14的时间匹配事件的第一脉冲的生成同步地输出PWM波形信号。

[0153] 附带地，当要装载两个微控制器用于互相监控时，可以由彼此独立的单独的微控制器执行PWM波形信号的输出和PWM波形信号的输入(以及其比较)，以及可以双重地执行PWM波形信号的输出和PWM波形信号的输入(以及其比较)。

[0154] 将使用图14、图15A和图15B对第三实施例的操作流程进行描述。图14是例示在PWM波形比较中执行的稳定处理操作的一个示例的流程图。图15A是例示在PWM波形比较中执行的上升沿中断处理操作的一个示例的流程图。图15B是例示在PWM波形比较中执行的比较匹配中断处理操作的一个示例的流程图。使用从EPTPC 14生成的脉冲输出上升沿中断和从通道0(MTU0)生成的比较匹配中断作为中断。在如图13所例示的，在从EPTPC 14的脉冲输出计时器输出的波形信号从低至高变化的上升时间处从EPTPC 14生成脉冲输出上升沿中断。当如图13所例示的，当在PWM波形信号的周期中清空生成MTU0的PWM波形信号的计时器计数器时，从通道0(MTU0)生成比较匹配中断。

[0155] <稳定处理>

[0156] 步骤S71:CPU在特定时间(限定为T_s)处开始彼此同步的EPTPC 14的脉冲输出计时器以及MTU0的计数器的操作。通过由ELC 17进行事件信号耦合来执行计时器和计数器的操作的同步开始。

[0157] 步骤S72:CPU 13通过ELC 17将从EPTPC 14输出的脉冲耦合至I/O端口18(第一端口18_1)的输出。随后，通过在切换的输出模式中从EPTPC 14生成脉冲来执行端口输出。

[0158] 步骤S73:CPU 13将EPTPC 14的脉冲输出计时器的计时器起始时间设定为同步起始时间(TMSTTR←T_s)，将周期设定为200μs(TMCYCR←200000)并且将脉冲宽度设定为100μs(TMPLSR←100000)。

[0159] 步骤S74:CPU 13将MTU0的操作模式设定为PWM操作模式1。随后，CPU 13将提供至MTU0的时钟信号的频率设定为120MHz(PCLKA←120MHz)，将预定标器设定为PCLKA/16，将用于低至高反相的计数器比较值设定为200μs(TGRD←0x5DC)以及将用于高至低反相的计数器比较值设定为400μs(TGRC←0xBB8)。另外，CPU 13将起始时间处的计数值设定为比用于从低至高反相的计数器比较值小一个计数的值以使得在启动计时器和计数器之后将计数值直接从低至高反相(TCNT←0x5DB(=TGRD-1))。

[0160] 步骤S75:CPU 13设定ICU 16以便向CPU 13通知来自EPTPC 14的脉冲输出上升沿中断的生成以及来自MTU0的计数值从高至低反相的计数器的比较匹配中断的生成。另外，

CPU 13对EPTPC 14的脉冲输出上升沿中断标志和MTU0的比较匹配中断标志进行初始化(ptp_flag←假,mtu0_flag←假)。

[0161] 步骤S76:CPU 13使用与用于PWM输出的通道不同的通道用于输入脉冲的脉冲宽度测量,以及为第一脉冲的脉冲宽度测量分配通道1(MTU1)并且为第二脉冲的脉冲宽度测量分配通道2(MTU2)。随后,CPU 13将提供至MTU1的时钟信号的频率设定为120MHz(PCLKA←120MHz)、将预定标器设定为PCLKA/1并且将操作模式设定为外部脉冲宽度测量功能模式以便测量高脉冲宽度。CPU 13还以相同的方式对MTU2执行设定。

[0162] 步骤S77:CPU 13将用于判定故障检测的脉冲宽度差的绝对值的上限值设定为阈值。通过将由振荡器的存在引起的MTU 15的计数器的误差、PTP的协议操作上的误差等考虑在内来设定阈值(thresh)。由通过稍后描述的公式(6)表示阈值的一个示例。

[0163] 步骤S78:CPU 13通过使用EPTPC 14根据PTP开始与另一件外部设备的时间的时间同步。此处,根据PTP从一件设备或者多件设备中选择具有最准确的时钟信号的设备作为时间分发源。

[0164] 步骤S79:当EPTPC 14的本地时钟计数器1441的计数值达到步骤S73中设置的T_s时,EPTPC 14已经生成的第一脉冲和MTU 15已经生成的第二脉冲从第一端口18_1和第二端口18_2分别地输出。附带地,硬件元件执行如下的这个步骤。

[0165] 首先,将使用图13对通过使用EPTPC 14的输出计时器执行的PWM波形信号输出操作进行描述。脉冲输出计时器与EPTPC 14的脉冲输出计时器的计时器起始时间(T_s)同步地从低至高上升并且输出了周期为200μs并且脉冲宽度为100μs的脉冲。第一脉冲通过脉冲输出计时器的上升而从低至高上升并且来自第一端口18_1的输出开始。第一脉冲通过脉冲输出计时器的下一个上升从高至低反相以及脉冲输出上升沿中断被生成。通过重复地执行该操作从第一端口18_1输出第一脉冲的PWM波形信号。该PWM波形信号的周期为400μs,脉冲宽度为200μs并且占空比为50%。

[0166] 接着,将通过使用图13对使用MTU0的计时器计数器的PWM波形信号输出操作进行描述。MTU0的计时器计数器对值进行递增计数并且在计数值(TCNT0)达到TGRD时将来自第二端口18_2的输出从低至高反相。该操作的计时与计时器起始时间T_s同步。计时器计数器继续对值进行递增计数并且在计数值达到TGRC时将来自第二端口18_2的输出从高至低反相以及从而生成比较匹配中断。计时器计数器通过重复地执行上面提到的操作从第二端口18_2输出第二脉冲的PWM波形信号。该PWM波形信号的周期为400μs,脉冲宽度为200μs并且占空比为50%。

[0167] 步骤S7A:将已经从第一端口18_1输出的第一脉冲输入到第三端口18_3中以及将已经从第二端口18_2输出的第二脉冲输入到第四端口18_4中。附带地,硬件元件也执行如下这个步骤。

[0168] 首先,将通过使用图13对通过使用MTU1的计时器计数器进行的第一脉冲的脉冲宽度的测量进行描述。在输入进第三端口18_3中的第一脉冲的上升处开始MTU1的计时器计数器的计数数值的计数以及在第一脉冲的下降处终止该计数。基于在那时获取的计时器计数器的计数值(TCNT1)测量高脉冲宽度。

[0169] 其次,将通过使用图13对通过使用MTU2的计时器计数器进行的第二脉冲的脉冲宽度的测量进行描述。在输入进第四端口18_4中的第二脉冲的上升处开始MTU2的计时器计数

器的计数数值的计数以及在第二脉冲的下降处终止该计数。基于在那时获取的计时器计数器的计数值(TCNT2)测量高脉冲宽度。

[0170] 步骤S7B:CPU 13判定脉冲宽度的获取是否已经完成。通过设定EPTPC 14的脉冲输出计时器上升沿中断标志(ptp_flag)以及MTU 15的比较匹配中断标志(mtu_flag)进行判断。当已经完成脉冲宽度的获取(是)时,CPU 13进行到步骤S7C中的处理以及当没有完成脉冲宽度的获取(否)时,CPU进行到步骤S7B中的处理。

[0171] 步骤S7C:CPU 13确认脉冲宽度之间的差的绝对值是否超过阈值。

[0172] 脉冲宽度差＝|mtu1_count-mtu2_count|

[0173] 当脉冲宽度差的绝对值超过阈值时,CPU将其检测为故障并且以错误而终止处理。

[0174] 步骤S7D:当在步骤S7C中脉冲宽度差的绝对值未超过阈值时,CPU 13清空EPTPC 14的脉冲输出计时器上升沿中断标志和MTU 15的比较匹配中断标志(ptp_flag←假,mtu_flag←假)。

[0175] 步骤S7E:CPU 13判定用户应用是否终止。当用户应用终止时,CPU 13终止处理。当用户应用并未终止时,CPU 13回到步骤S7B以及继续执行处理。

[0176] <中断处理>

[0177] 如上所述,MTU1的计时器计数器在已经从第三端口18_3输入的第一脉冲的高周期中执行递增计数操作并且在第一脉冲的低周期中停止递增计数操作。另外,MTU2的计时器计数器在已经从第四端口18_4输入的第二脉冲的高周期中执行递增计数操作并且在第二脉冲的低周期中停止递增计数操作。

[0178] 以下,将对脉冲输出计时器上升沿中断进行描述。

[0179] 步骤S81:当已经从EPTPC 14生成脉冲输出计时器上升沿中断时,CPU 13读出MTU1的计时器计数器的计数值(TCNT1)以及获取所读出的计数值作为第一脉冲的脉冲宽度(mtu1_count←TCNT1)。

[0180] 步骤S82:CPU 13清空MTU1的计时器计数器(TCNT1←0)。

[0181] 步骤S83:CPU 13设定EPTPC 14的脉冲输出计时器上升沿中断标志(ptp_flag←真)。

[0182] 以下,将对比较匹配中断进行描述。

[0183] 步骤S91:当在MTU0的其值从高至低反相的计数器中已经生成了比较匹配中断时,CPU 13读出MTU2的计时器计数器的计数值(TCNT2)以及获取所读出的计数值作为第二脉冲的脉冲宽度(mtu2_count←TCNT2)。

[0184] 步骤S92:CPU 13清空MTU2的计时器计数器(TCNT2←0)。

[0185] 步骤S93:CPU 13在MTU0的其值从高至低反相的计数器中设定比较匹配中断标志(mtu0_flag←真)。

[0186] <阈值的一个示例>

[0187] 将对阈值(thresh)的一个示例进行描述。当阈值除了分成由因为振荡器的存在所导致的MTU计数器的误差引起的“$thresh_{osc}$”和由PTP的协议操作上的误差引起的“$thresh_{ptp}$”以外,阈值还被分成用于经由布置在微控制器外部的耦合电路传播的脉冲的传播延迟的“$threshold_{ppg}$”时,得到公式(5)。……

[0188] $thresh＝thresh_{osc}+thresh_{ptp}+thresh_{ppg}$……(5)

[0189] 附带地，在第三实施例中，由于用于获取脉冲宽度的计时器计数器用硬件启动和停止，因此对中断处理中的延迟没有影响。

[0190] 当上升沿中断间隔为t_{p3}时，将“$thresh_{osc}$”表示为类似于第二实施例的公式的公式(5-1)。例如，当f_{cpu}＝120MHz、f_{mtu}＝f_{cpu}＝120MHz、x_{osc}＝100ppm并且t_{p3}＝400μs时，得到$thresh_{osc}$＝5个周期。

[0191] $thresh_{osc}=f_{mtu}*x_{osc}*t_{p3}$……(5-1)

[0192] 可以与第一实施例类似地将PTP的协议操作上的误差在时间上减小至1μs至100ns以及在MTU计数器的计数数值上减小至$thresh_{ptp}$＝8个至80个周期。

[0193] 例如，当将高速度设定为$3.0x10^{8}$m/s、将传播脉冲的导体的相对介电常数设定为10并且将布线距离设定为10cm时，脉冲的传播延迟变得不多于10ns并且近似为$thresh_{ppg}<thresh_{osc}$。

[0194] 由上述可知，公式(5)近似为公式(6)。

[0195] $thresh=thresh_{osc}+thresh_{ptp}+thresh_{ppg}<2*thresh_{osc}+thresh_{ptp}$……(6)

[0196] 当将上面提到的数值应用于公式(6)时，“thresh”总计为100个周期。

[0197] 在第三实施例中，由具有本地时钟计数器(第一计数器)的预定值的EPTPC 14的脉冲输出计时器(第一计时器)生成的PWM波形信号从第一I/O端口输出以及从第三I/O端口输入，并且其脉冲宽度由MTU 15(第二计时器)的计时器计数器(第三计数器)来测量。由具有本地时钟计数器(第一计数器)的预定值的MTU 15(第二计时器)的计时器计数器(第二计数器)生成的PWM波形信号从第二I/O端口输出以及从第四I/O端口输入，并且其脉冲宽度由MTU 15(第二计时器)的计时器计数器(第四计数器)来测量。从而，可以在不停止执行用户程序的情况下通过计时器诊断PWM波形信号的生成和输出功能。

[0198] [第四实施例]

[0199] 由于根据第一实施例的MTU 15包括多个通道，因此可以将两个通道的16位计数器级联地耦合在一起以用作一个32位计数器。在32位计数器中，在实际操作中不发生溢出。

[0200] 图16是例示根据第四实施例的32位计数器的操作的一个示例的时序图。根据第四实施例的32位计数器被配置为使得16个高阶位被分配至MTU 15的通道1(MTU1)的计数器并且16个低阶位被分配至MTU 15的通道2(MTU2)的计数器。对MTU1的计数器的计数值进行递增计数以及每当MTU2的计数器溢出时就清空MTU2的计数器并且再次从零开始递增计数，以及由此两个计数器作为一个32位计数器工作。

[0201] 因为除了使用MTU 15的通道1(MTU1)的计数器和通道2(MTU2)的计数器、通过将计数器级联地偶合在一起来执行对32位的操作设定以及基本上消除了溢出中断的生成以外，根据第四实施例的32位计数器的配置和操作与根据第一实施例的计数器的配置和操作相同，因此省略其描述。要被级联地耦合在一起的计数器不限于通道1和通道2的计数器。

[0202] 〈阈值〉

[0203] 与第一实施例类似，阈值(thresh)的一个示例近似为公式(7)。

[0204] $thresh=(f_{mtu}/f_{cpu})*(2*f_{cpu}*x_{osc}*t_{p1}+2*x_{int}*n_{int})$……(7)

[0205] 此处，CPU 13的操作频率为f_{cpu}，MTU 15的操作频率为f_{mtu}＝f_{cpu}/16，为MTU 15提供时钟信号的振荡器的精度为x_{osc}，时间匹配事件间隔为t_{p1}，中断处理出现频率为x_{int}以及中断处理的周期数量为n_{int}。

[0206] 由于消除了来自MTU 15的溢出中断的生成，因此与第一实施例中的出现频率相比，中断处理出现频率x_{int}从13次时间匹配事件中断处理降低至两次时间匹配事件中断处理。例如，当f_{cpu}＝120MHz，f_{mtu}＝(f_{cpu}/16)＝7.5MHz，x_{osc}＝100ppm，t_{pl}＝100ms，x_{int}＝2以及n_{int}＝100时，得到thresh＝175个周期。

[0207] 在第四实施例中，通过将两个16位计数器级联地耦合在一起来配置32位计数器。由此，可以通过降低中断处理出现频率以进行比第一实施例中的诊断更准确的诊断。另外，可以在将计时器用作32位计数器时对计时器进行诊断。

[0208] 尽管，如上面提到的，已经基于优选实施例和实际示例对已经由本申请的发明人和其他人提出的本公开进行了具体地描述，但是，不用说的是，本公开不限于前述实施例和实际示例并且可以在不偏离本公开的主旨的范围内以各种方式进行更改和修改本公开。

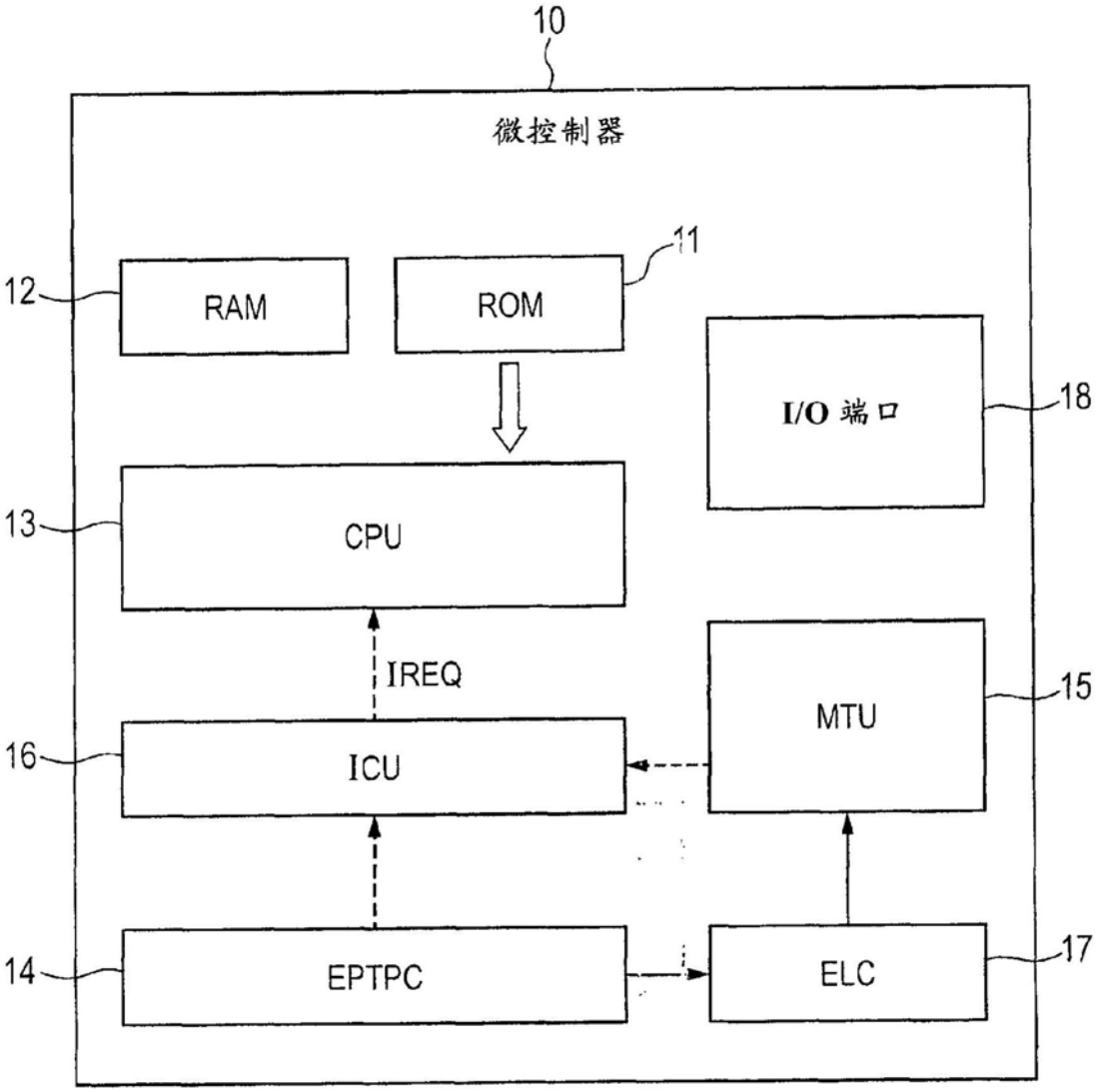

10
微控制器
11
12
RAM
ROM
I/O 端口
18
13
CPU
IREQ
MTU
15
16
ICU
14
EPTPC
ELC
17

图1

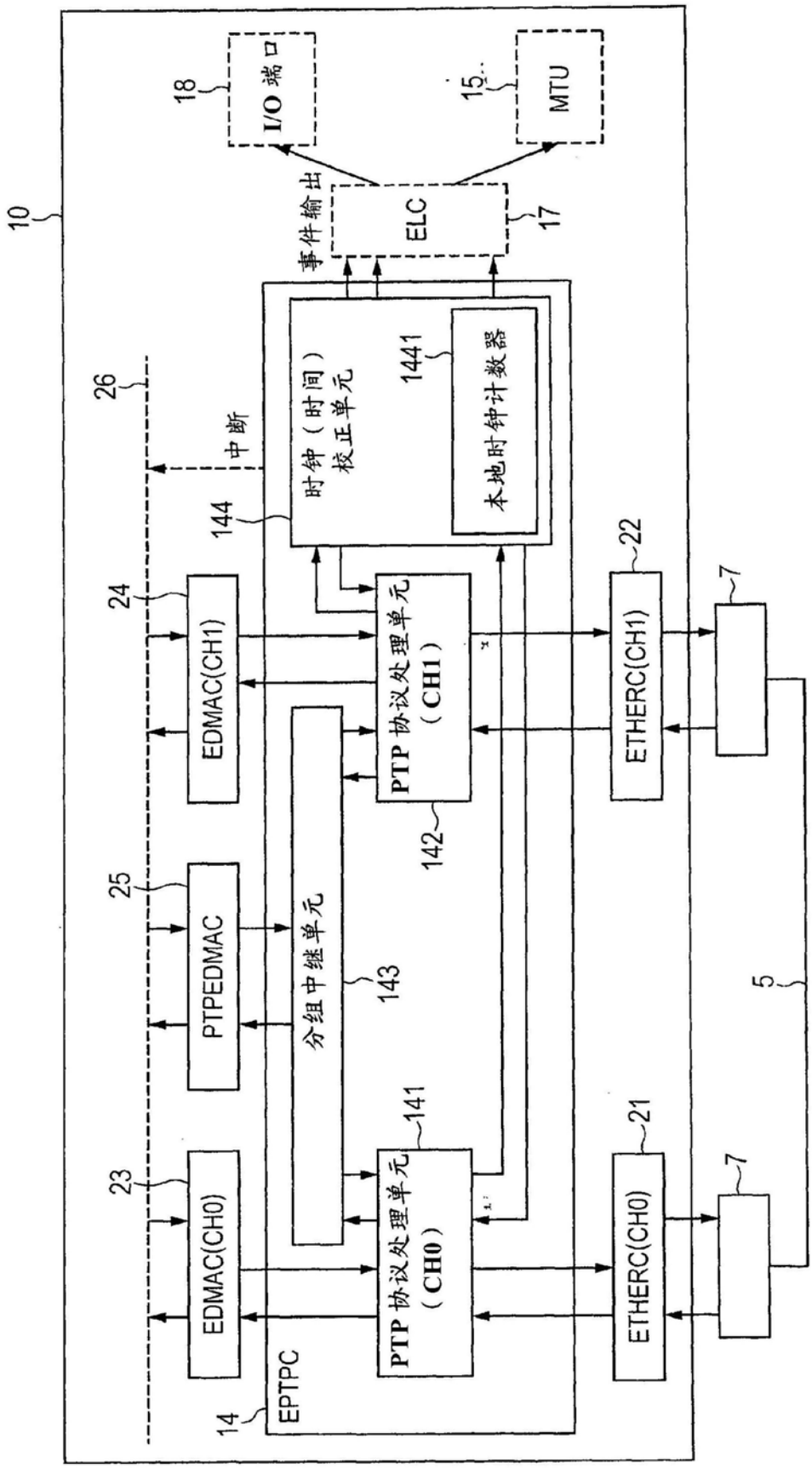

10
18
I/O 端口
15
MTU
事件输出
ELC
17
26
中断
144
时钟（时间）校正单元
1441
本地时钟计数器
24
EDMAC(CH1)
PTP 协议处理单元（CH1）
142
22
ETHERC(CH1)
7
25
PTPEDMAC
分组中继单元
143
5
23
EDMAC(CH0)
141
PTP 协议处理单元（CH0）
21
ETHERC(CH0)
7
14
EPTPC

图2

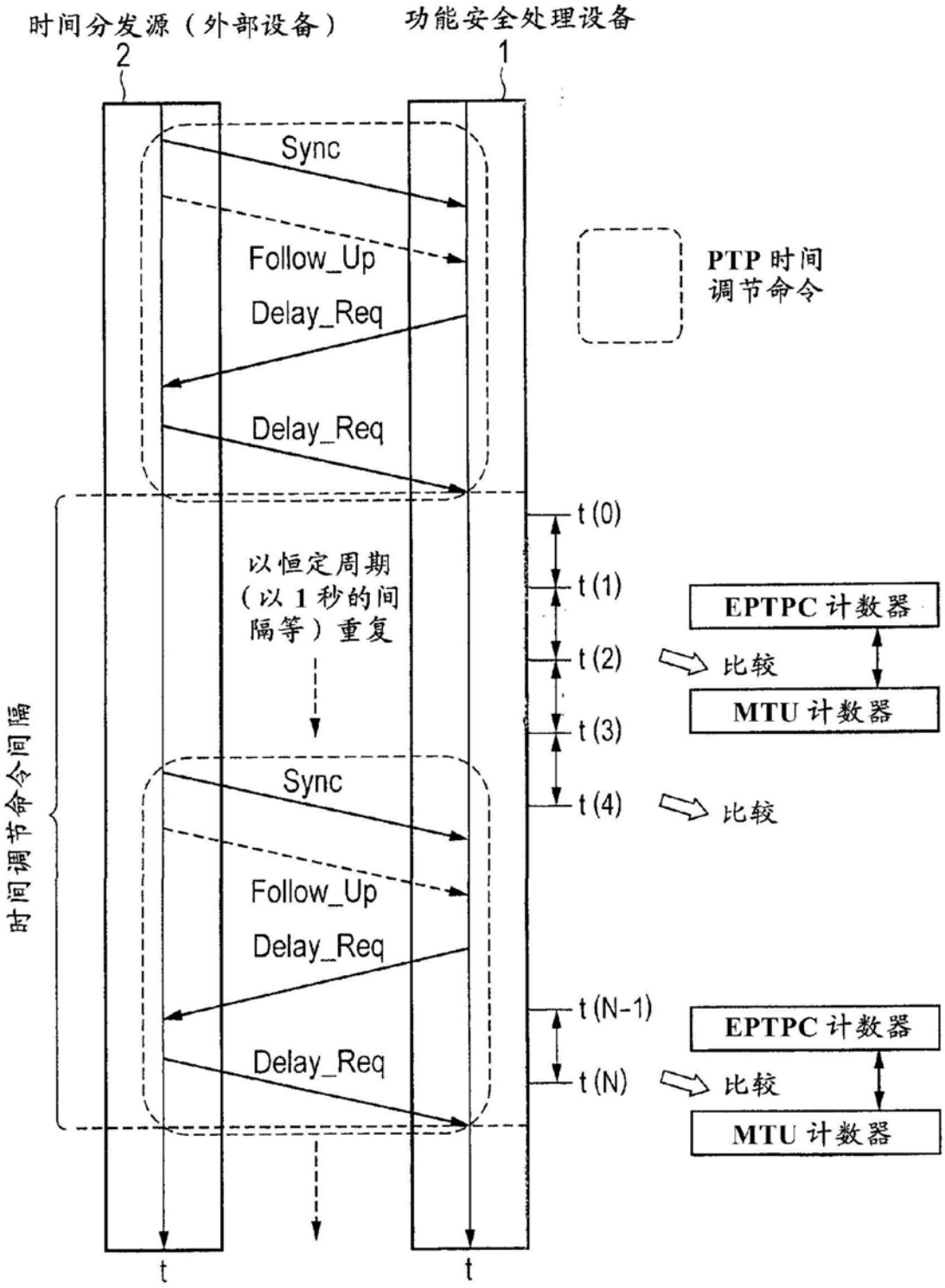

时间分发源（外部设备）
2
功能安全处理设备
1
Sync
Follow_Up
Delay_Req
Delay_Req
PTP 时间
调节命令
以恒定周期
（以 1 秒的间
隔等）重复
时间调节命令间隔
t (0)
t (1)
t (2)
t (3)
t (4)
EPTPC 计数器
比较
MTU 计数器
比较
Sync
Follow_Up
Delay_Req
Delay_Req
t (N-1)
t (N)
EPTPC 计数器
比较
MTU 计数器
t
t

图3A

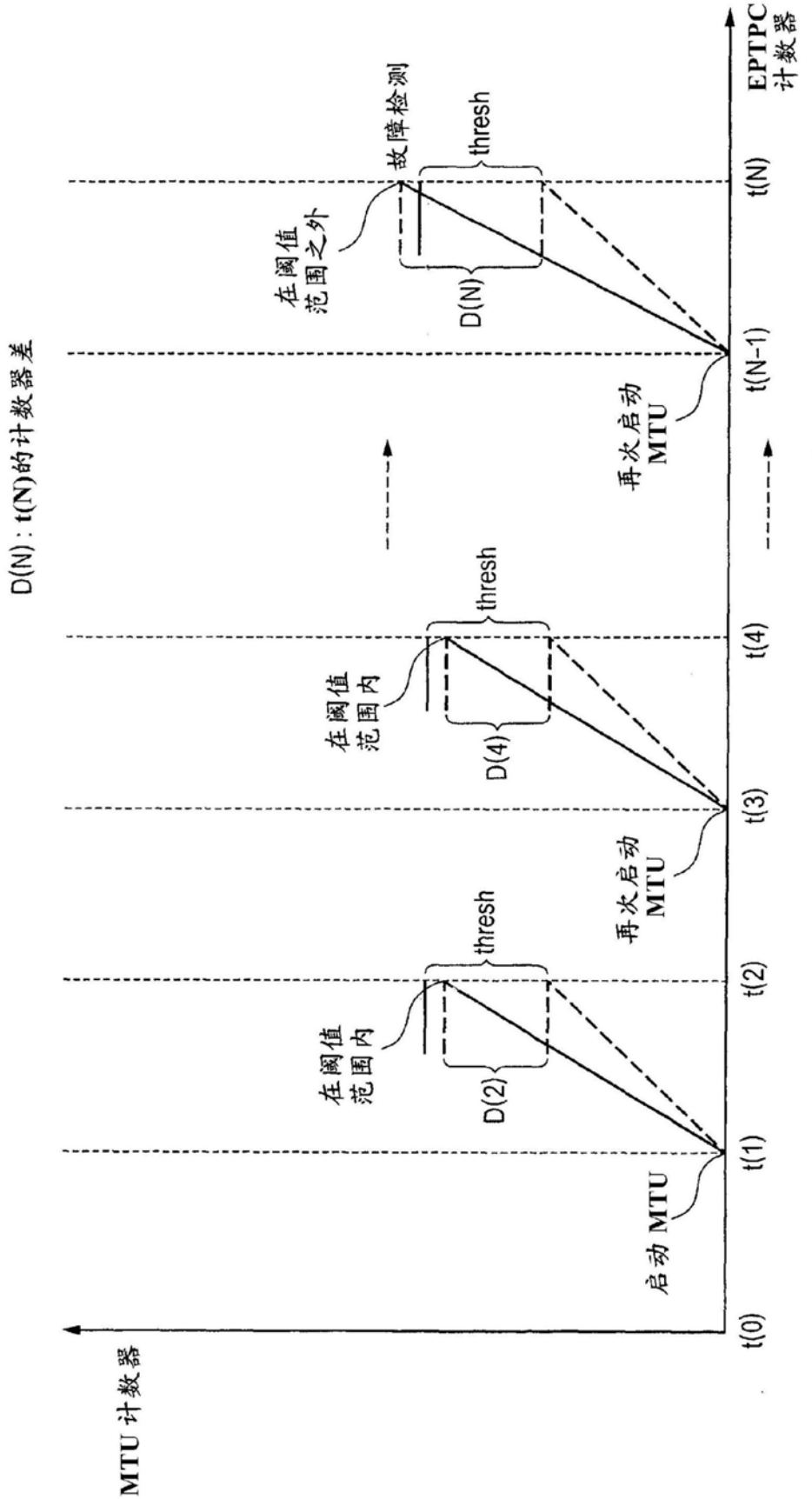

MTU 计数器
D(N) : t(N)的计数器差
EPTPC 计数器
t(0)
t(1)
t(2)
t(3)
t(4)
t(N-1)
t(N)
启动 MTU
再次启动 MTU
再次启动 MTU
在阈值范围内
在阈值范围内
在阈值范围之外
故障检测
thresh
thresh
thresh
D(2)
D(4)
D(N)

图3B

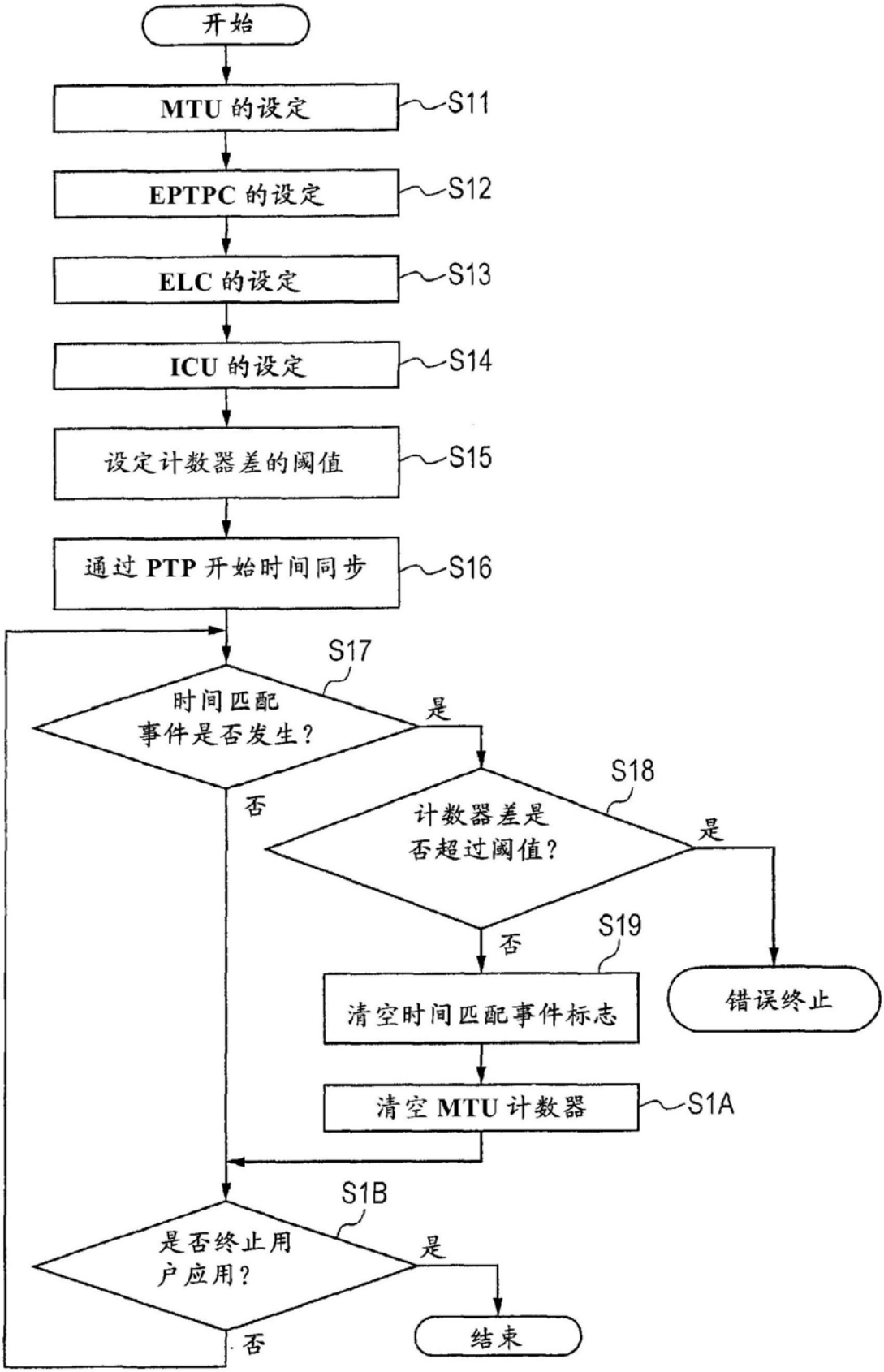

开始
MTU 的设定
S11
EPTPC 的设定
S12
ELC 的设定
S13
ICU 的设定
S14
设定计数器差的阈值
S15
通过 PTP 开始时间同步
S16
S17
时间匹配
事件是否发生？
是
否
S18
计数器差是
否超过阈值？
是
否
S19
清空时间匹配事件标志
错误终止
清空 MTU 计数器
S1A
S1B
是否终止用
户应用？
是
否
结束

图4

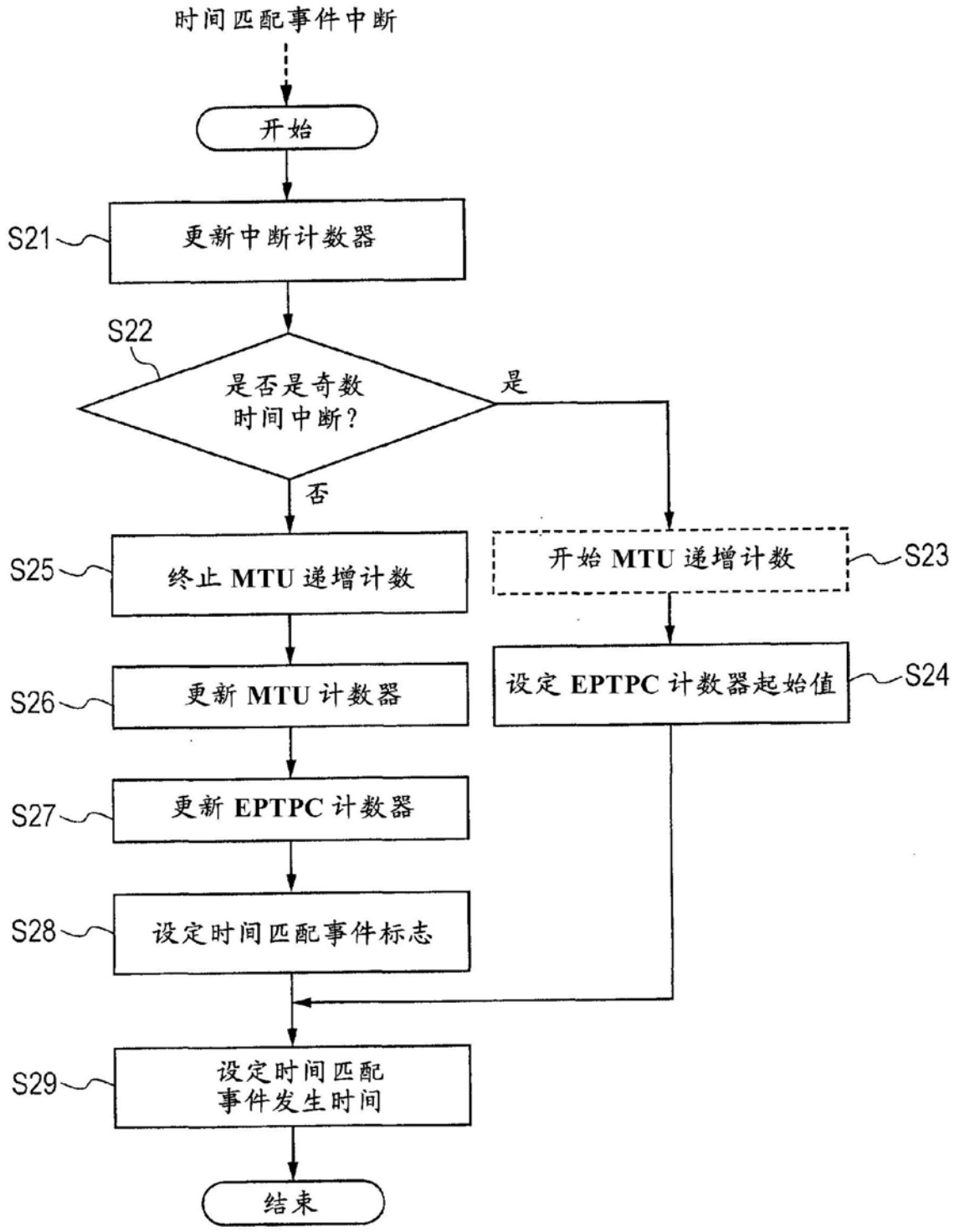

时间匹配事件中断
开始
S21
更新中断计数器
S22
是否是奇数
时间中断?
是
否
S25
终止 MTU 递增计数
S23
开始 MTU 递增计数
S26
更新 MTU 计数器
S24
设定 EPTPC 计数器起始值
S27
更新 EPTPC 计数器
S28
设定时间匹配事件标志
S29
设定时间匹配
事件发生时间
结束

图5A

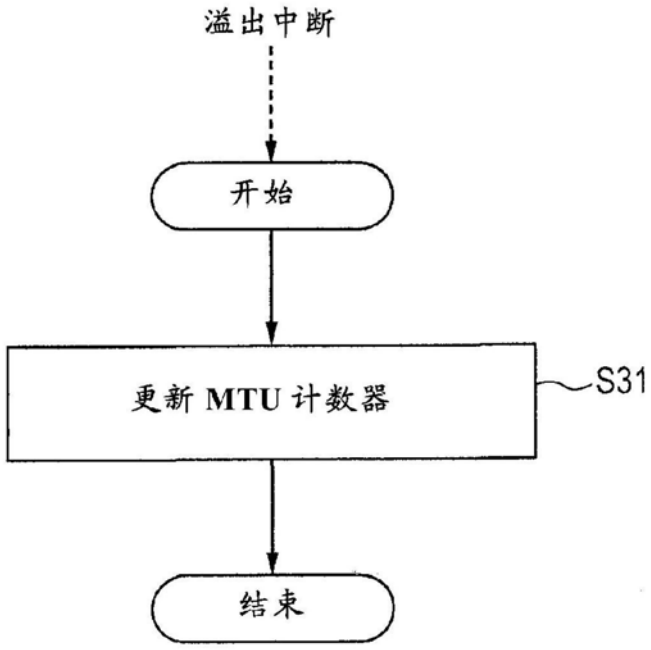


图5B

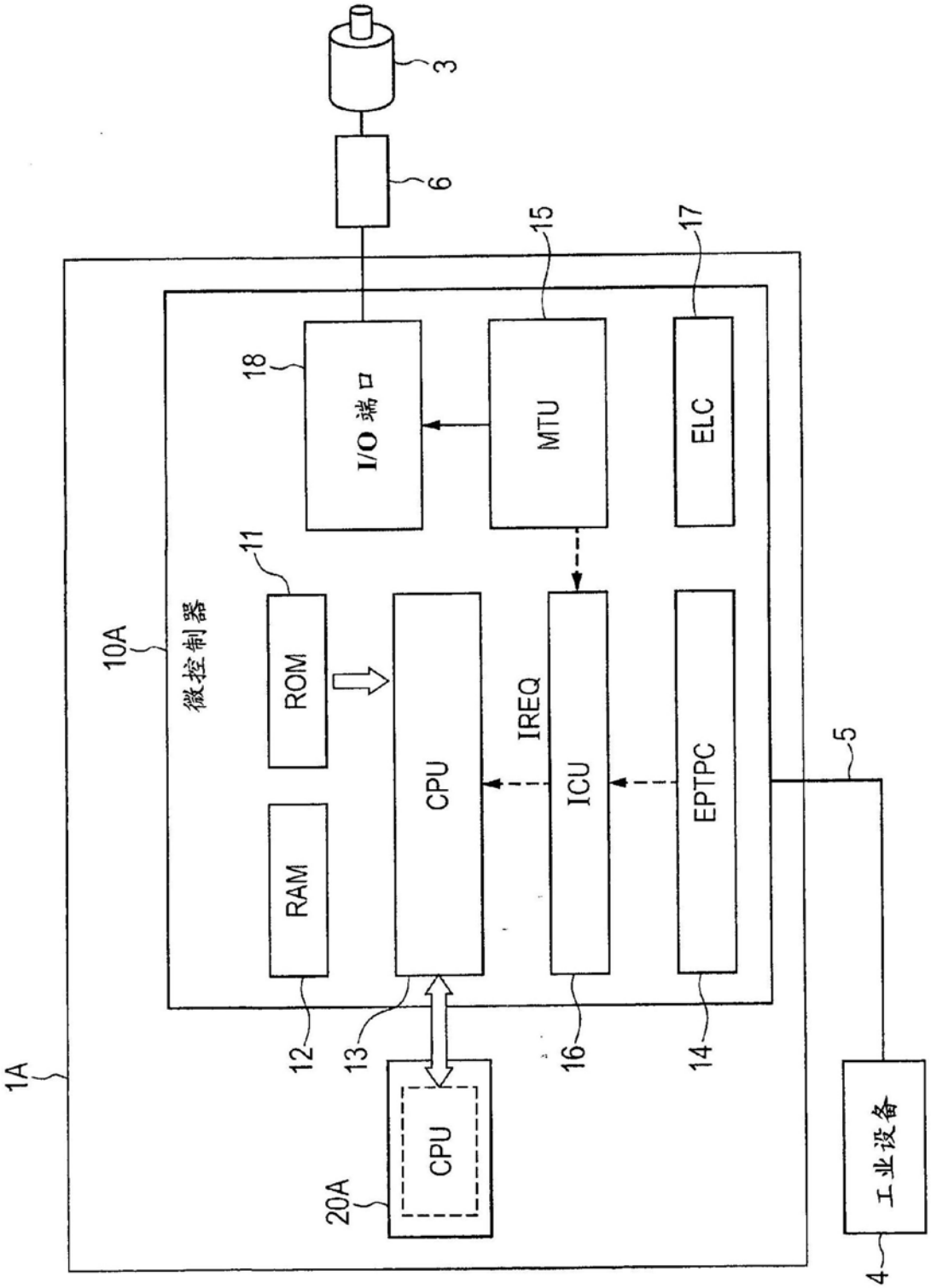

3
6
15
17
18
I/O 端口
MTU
ELC
11
10A
微控制器
ROM
CPU
IREQ
ICU
EPTPC
5
RAM
1A
12
13
16
14
CPU
20A
工业设备
4

图6

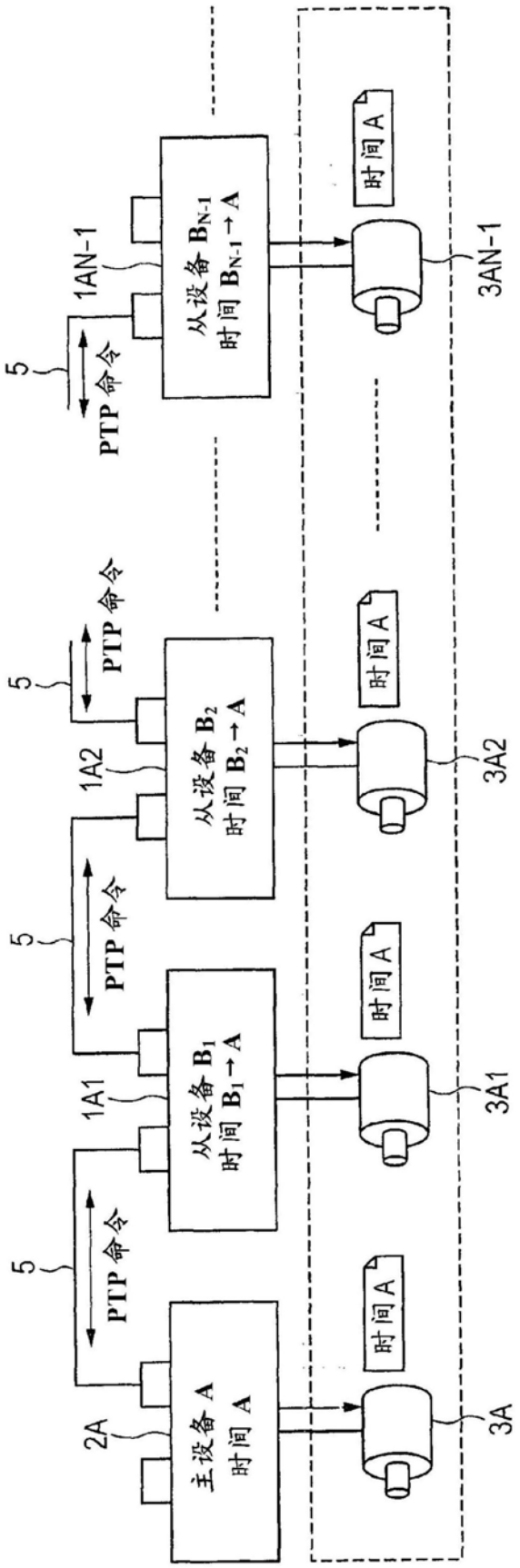

主设备 A
时间 A
2A
PTP 命令
5
从设备 B_1
时间 $B_1 \rightarrow A$
1A1
从设备 B_2
时间 $B_2 \rightarrow A$
1A2
从设备 B_{N-1}
时间 $B_{N-1} \rightarrow A$
1AN-1
时间 A
3A
3A1
3A2
3AN-1

图7

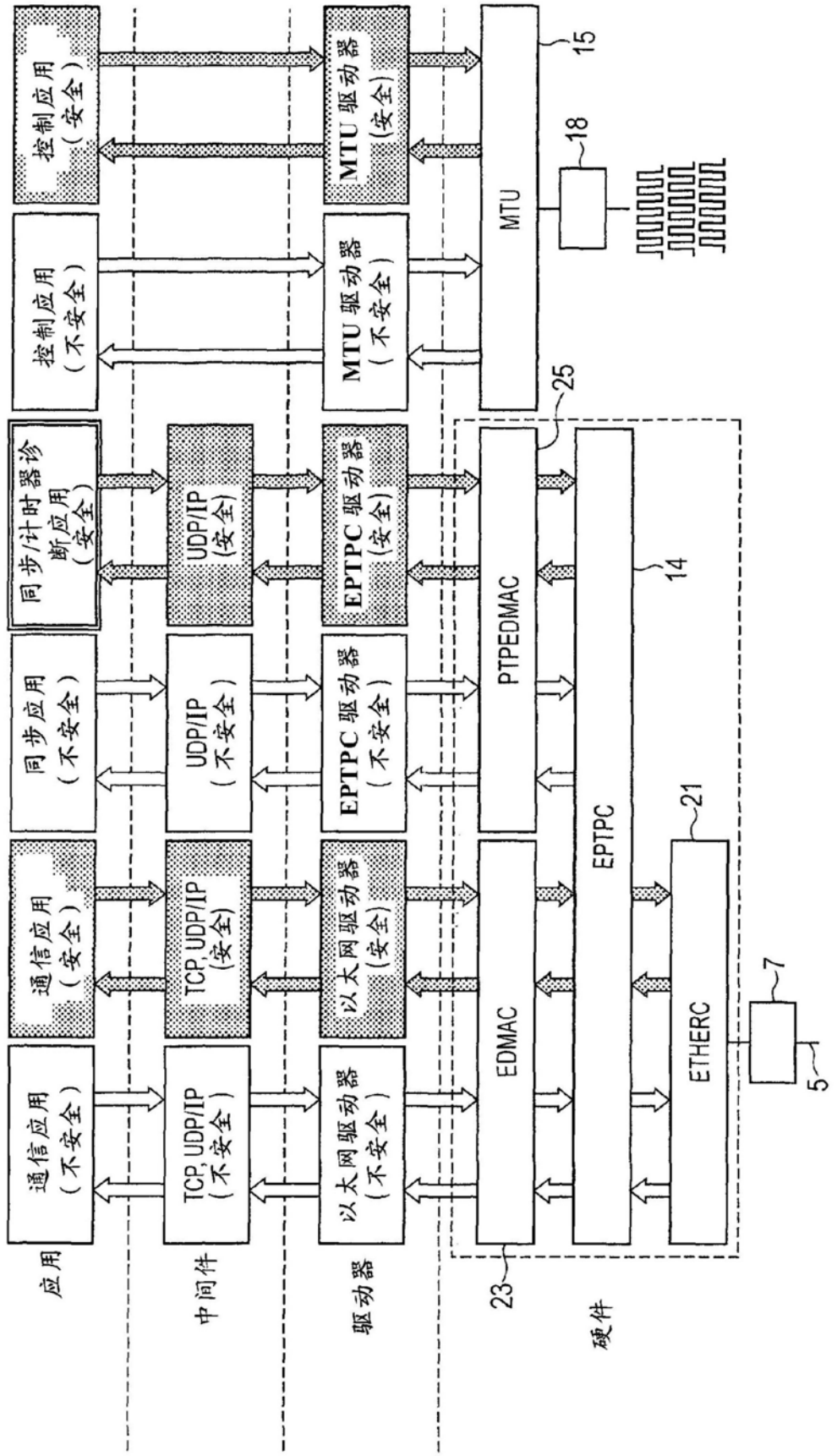

应用
中间件
驱动器
硬件
通信应用（不安全）
通信应用（安全）
同步应用（不安全）
同步/计时器诊断应用（安全）
控制应用（不安全）
控制应用（安全）
TCP, UDP/IP（不安全）
TCP, UDP/IP（安全）
UDP/IP（不安全）
UDP/IP（安全）
以太网驱动器（不安全）
以太网驱动器（安全）
EPTPC 驱动器（不安全）
EPTPC 驱动器（安全）
MTU 驱动器（不安全）
MTU 驱动器（安全）
EDMAC
PTPEDMAC
EPTPC
ETHERC
MTU
5
7
14
15
18
21
23
25

图8

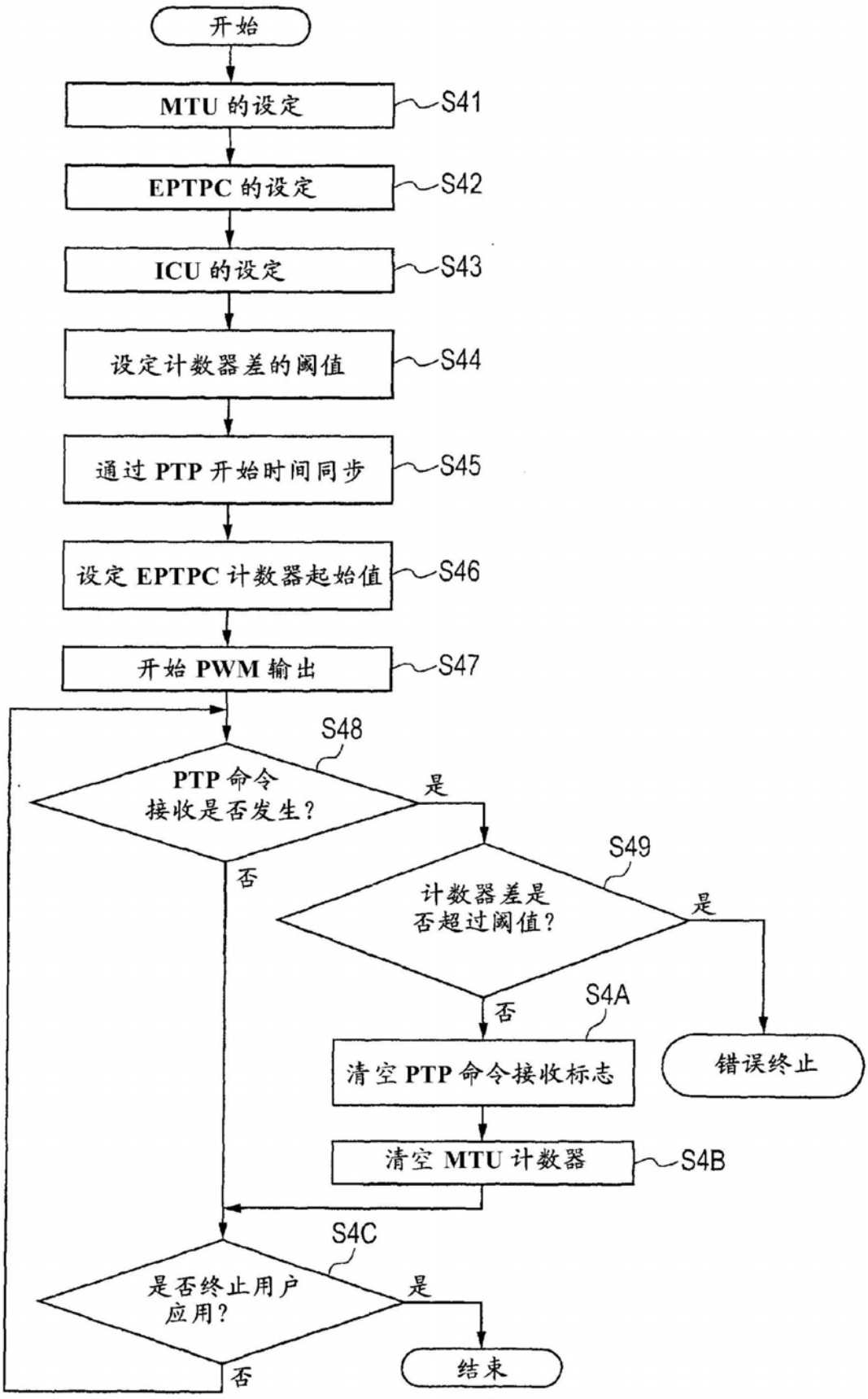

开始
MTU 的设定
S41
EPTPC 的设定
S42
ICU 的设定
S43
设定计数器差的阈值
S44
通过 PTP 开始时间同步
S45
设定 EPTPC 计数器起始值
S46
开始 PWM 输出
S47
S48
PTP 命令
接收是否发生?
是
否
S49
计数器差是
否超过阈值?
是
否
S4A
清空 PTP 命令接收标志
错误终止
清空 MTU 计数器
S4B
S4C
是否终止用户
应用?
是
否
结束

图9

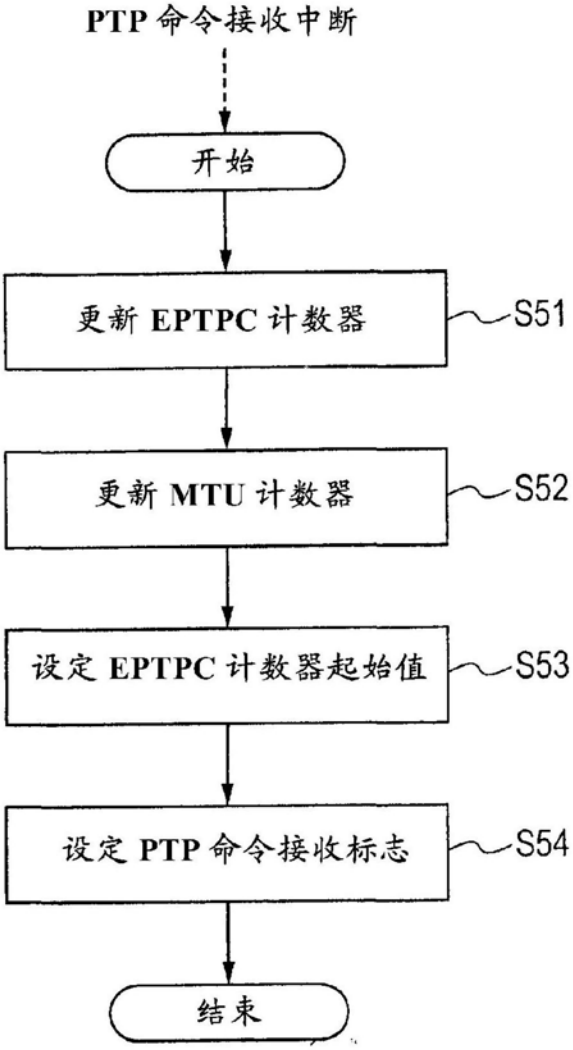

PTP 命令接收中断
开始
更新 EPTPC 计数器
S51
更新 MTU 计数器
S52
设定 EPTPC 计数器起始值
S53
设定 PTP 命令接收标志
S54
结束

图10A

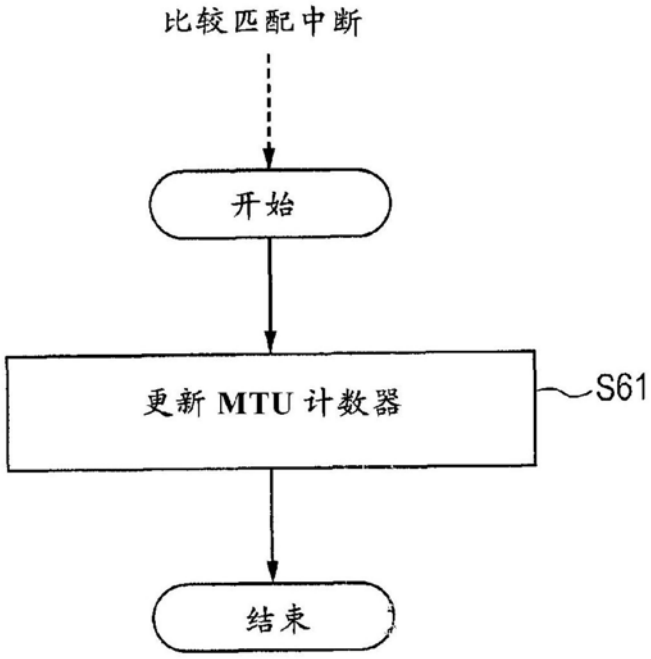


图10B

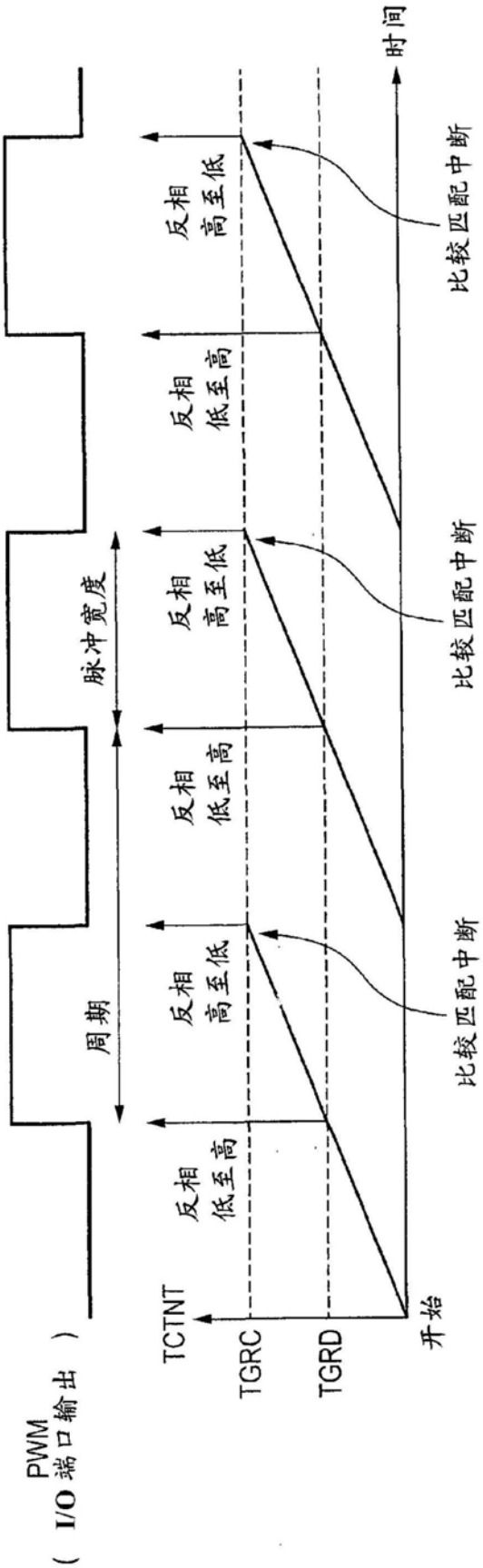

时间
比较匹配中断
比较匹配中断
比较匹配中断
反相高至低
反相低至高
反相高至低
反相低至高
反相高至低
反相低至高
脉冲宽度
周期
TCTNT
TGRC
TGRD
开始
PWM
（I/O端口输出）

图11

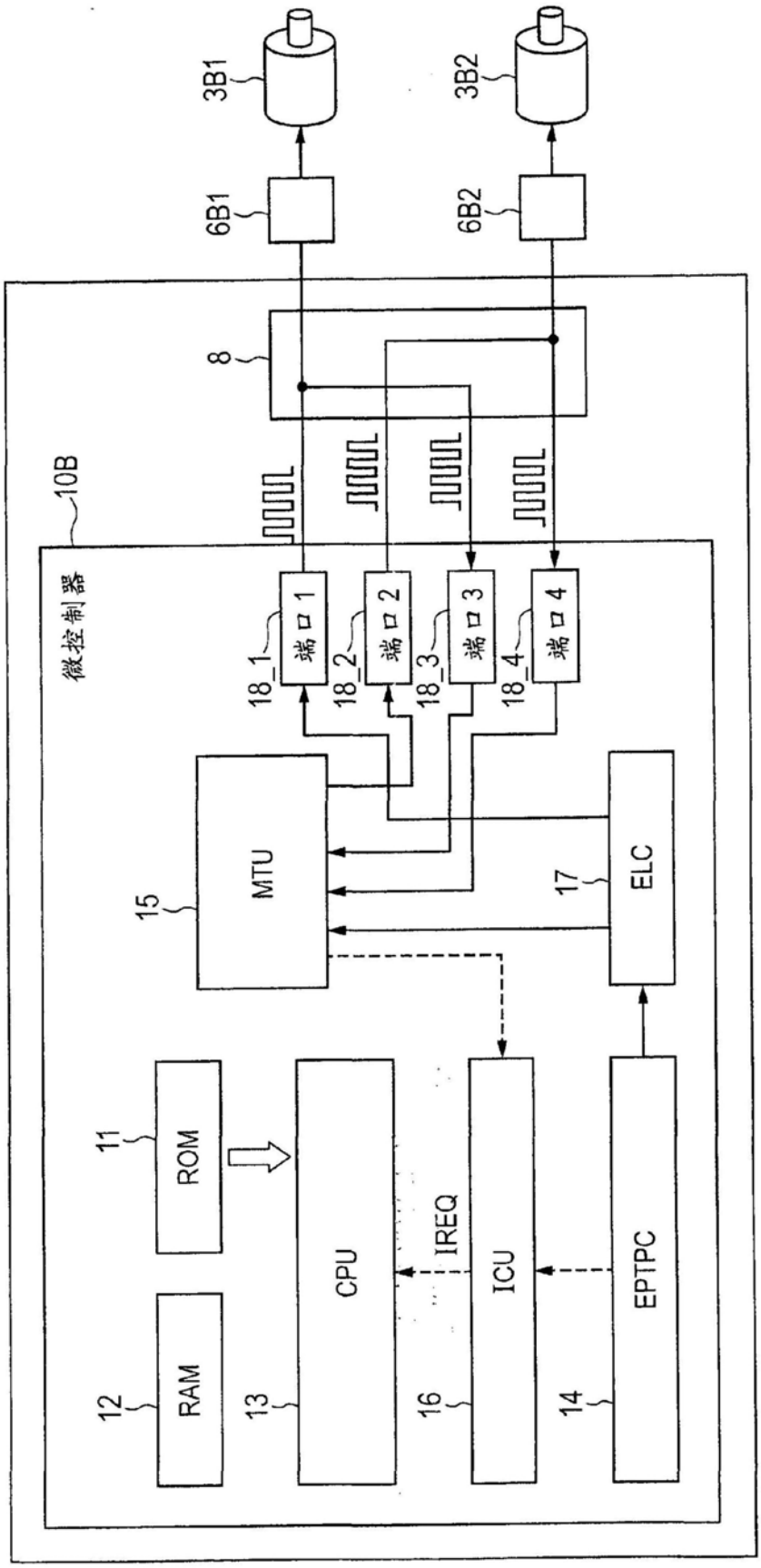

3B1
3B2
6B1
6B2
8
10B
微控制器
端口1
端口2
端口3
端口4
18_1
18_2
18_3
18_4
MTU
15
ELC
17
ROM
11
CPU
IREQ
ICU
EPTPC
RAM
12
13
16
14

图12

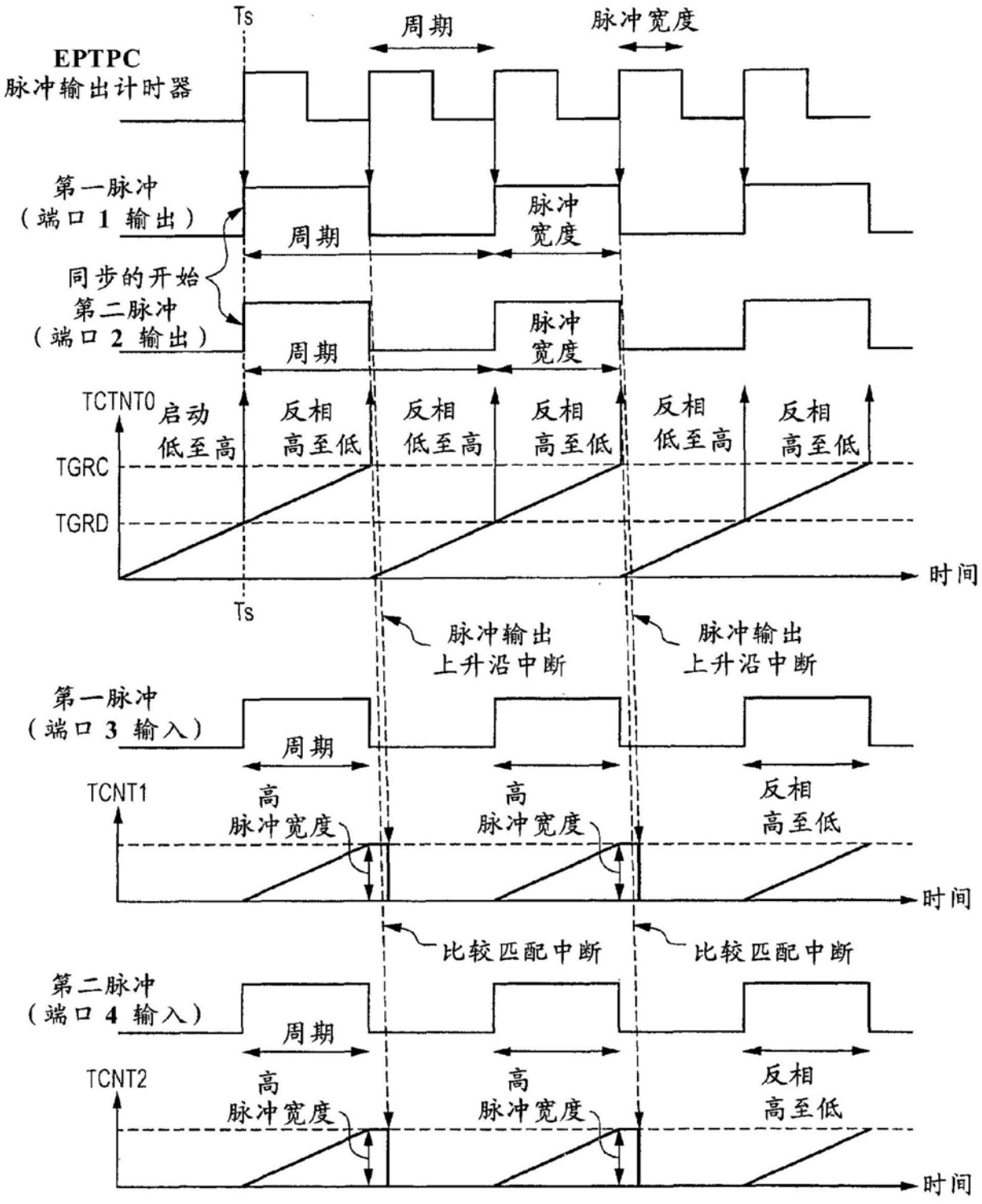

Ts
周期
脉冲宽度
EPTPC
脉冲输出计时器
第一脉冲
（端口 1 输出）
周期
脉冲
宽度
同步的开始
第二脉冲
（端口 2 输出）
周期
脉冲
宽度
TCTNT0
启动
低至高
反相
高至低
反相
低至高
反相
高至低
反相
低至高
反相
高至低
TGRC
TGRD
时间
Ts
脉冲输出
上升沿中断
脉冲输出
上升沿中断
第一脉冲
（端口 3 输入）
周期
TCNT1
高
脉冲宽度
高
脉冲宽度
反相
高至低
时间
比较匹配中断
比较匹配中断
第二脉冲
（端口 4 输入）
周期
TCNT2
高
脉冲宽度
高
脉冲宽度
反相
高至低
时间

图13

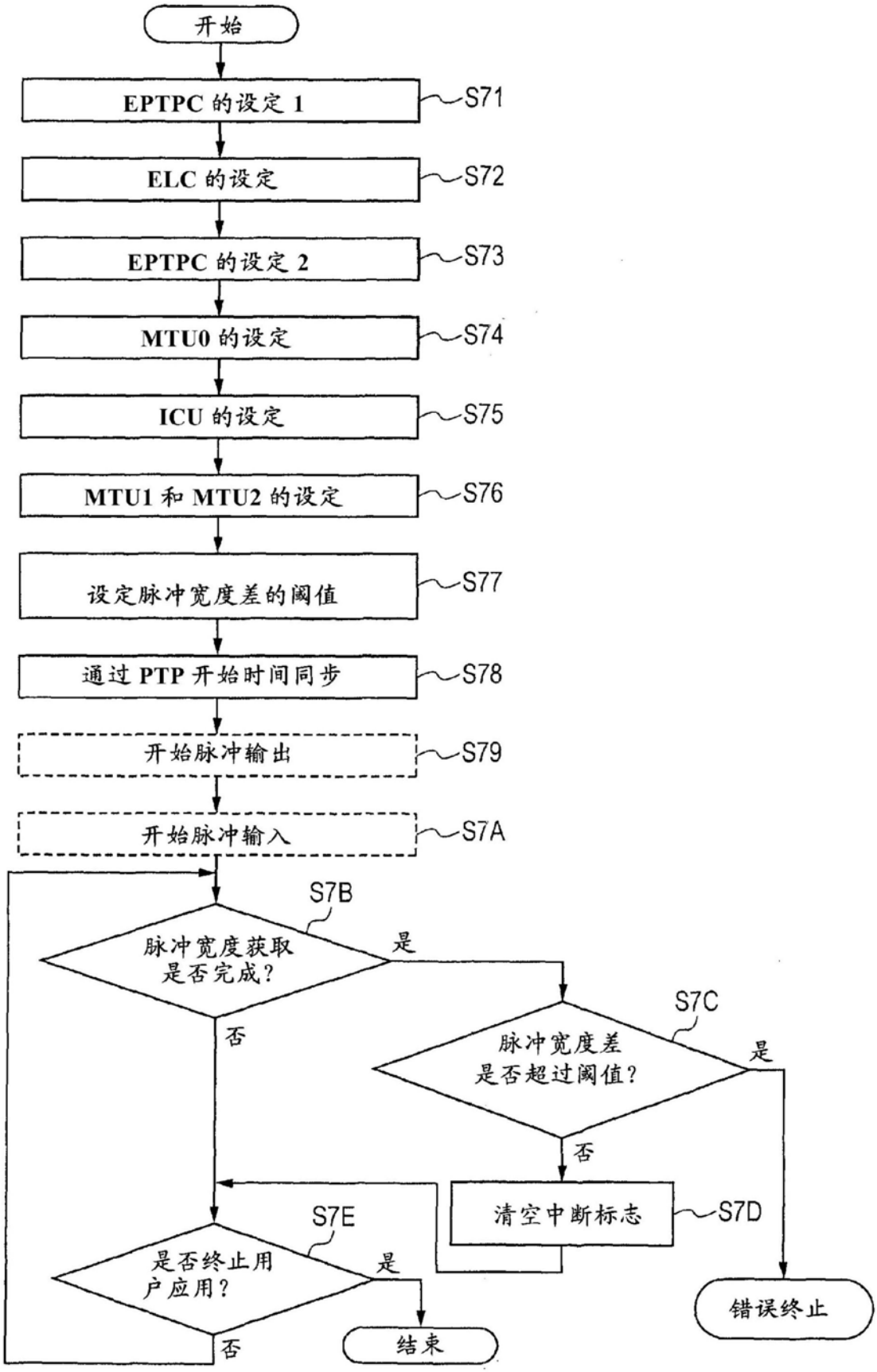

开始
EPTPC 的设定 1
S71
ELC 的设定
S72
EPTPC 的设定 2
S73
MTU0 的设定
S74
ICU 的设定
S75
MTU1 和 MTU2 的设定
S76
设定脉冲宽度差的阈值
S77
通过 PTP 开始时间同步
S78
开始脉冲输出
S79
开始脉冲输入
S7A
S7B
脉冲宽度获取
是否完成?
是
否
S7C
脉冲宽度差
是否超过阈值?
是
否
清空中断标志
S7D
S7E
是否终止用
户应用?
是
否
结束
错误终止

图14

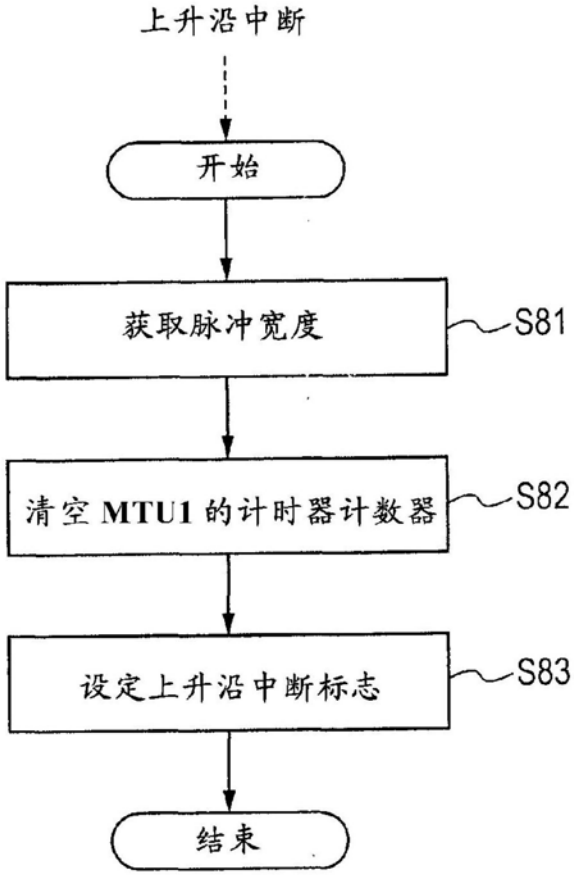


图15A

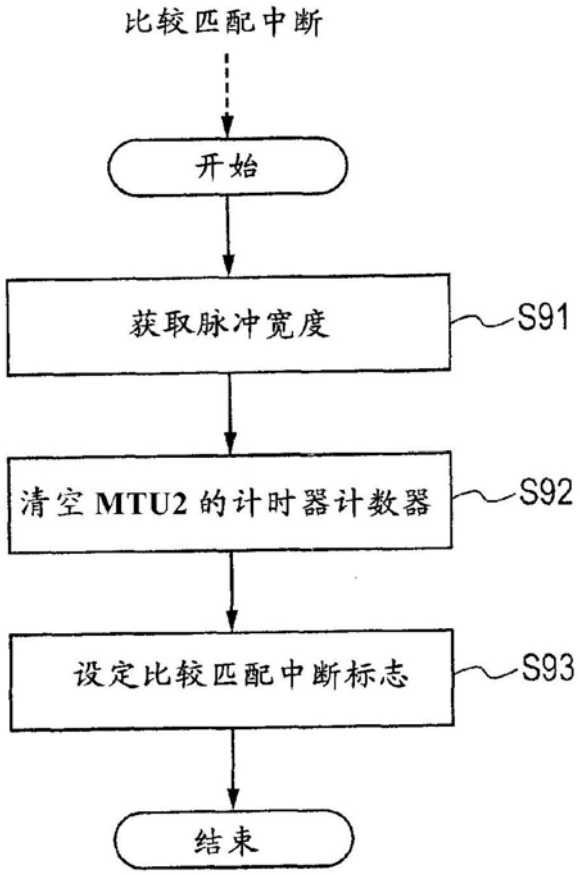


图15B

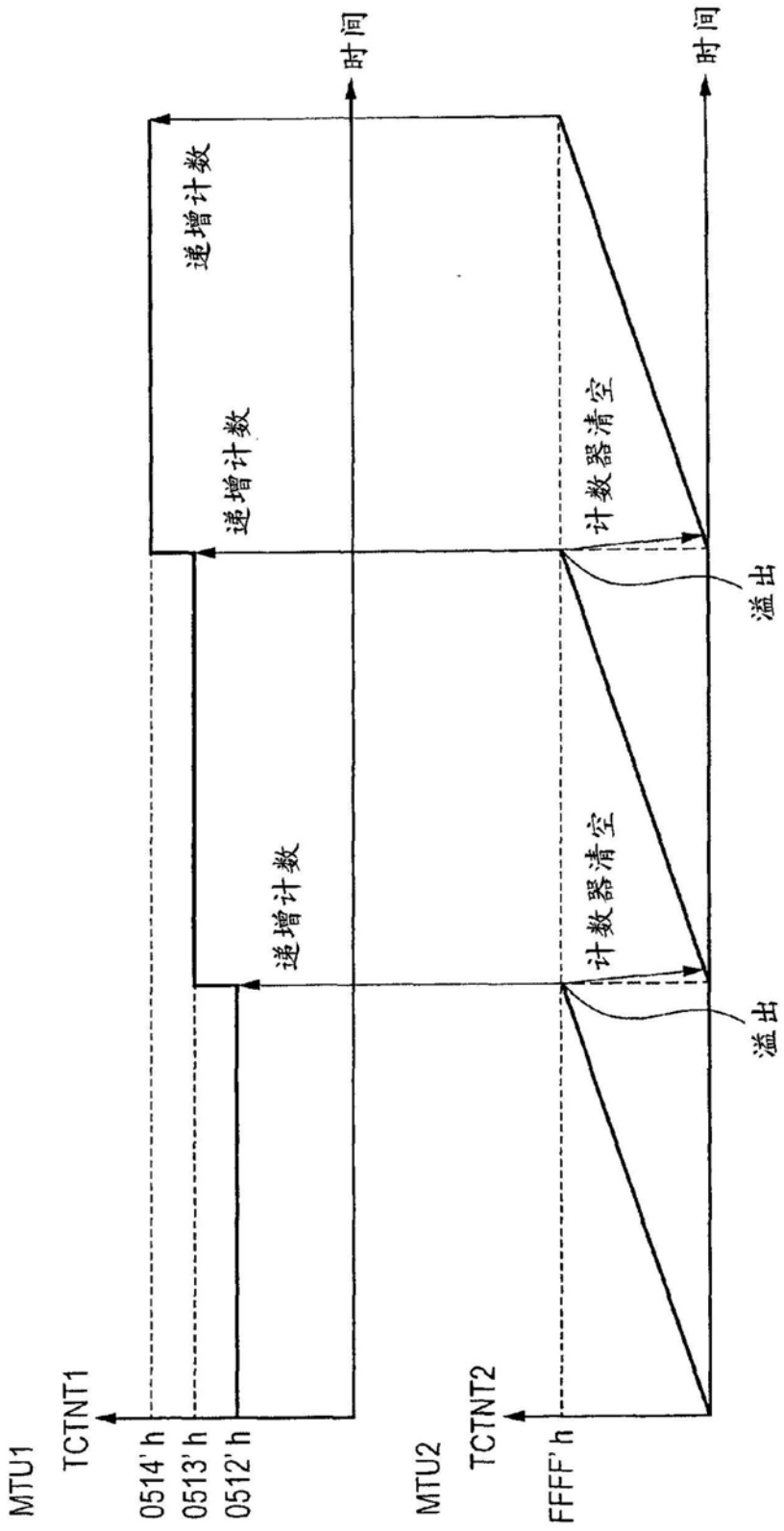

时间
递增计数
递增计数
计数器清空
溢出
时间
递增计数
计数器清空
溢出
MTU1
TCTNT1
0514' h
0513' h
0512' h
MTU2
TCTNT2
FFFF' h

图16