[19] 中华人民共和国国家知识产权局

[51] Int. Cl[7]
G06F 9/46

[12] 发明专利说明书

[21] ZL 专利号 99804263.3

[45] 授权公告日 2003 年 12 月 10 日

[11] 授权公告号 CN 1130630C

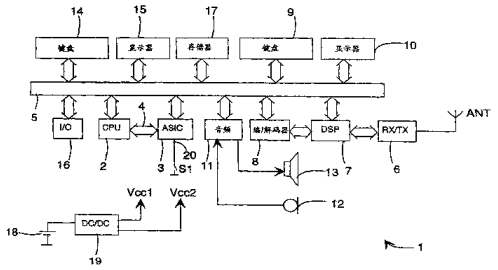
[22] 申请日 1999.1.21 [21] 申请号 99804263.3
[30] 优先权
[32] 1998. 1.21 [33] FI [31] 980135
[86] 国际申请 PCT/FI99/00039 1999.1.21
[87] 国际公布 WO99/38073 英 1999.7.29
[85] 进入国家阶段日期 2000.9.21
[71] 专利权人 诺基亚移动电话有限公司
地址 芬兰埃斯波
[72] 发明人 R·伦凯 V·萨里宁 J·坎托拉
J·莱斯克莱 K·莱姆皮宁
A·普尔霍宁
审查员 邓 茜
[74] 专利代理机构 中国专利代理(香港)有限公司
代理人 吴立明 张志醒

权利要求书 5 页 说明书 35 页 附图 14 页

[54] 发明名称 嵌入式系统

[57] 摘要

一种包含至少一个用于运行操作系统(OS_A、OS_B)的处理器(2)的嵌入式系统(1)。嵌入式系统(1)进一步包含：用于在处理器(2)中运行至少两个操作系统(OS_A、OS_B)的装置(17、401、412)；包含第一组线程(THA1、THA2、THA_IDLE)的第一操作系统(OS_A)；包含第二组线程(THB1、THB2、THB_IDLE)的第二操作系统(OS_B)；用于向处理器(2)生成中断(FIQ、IRQ、SWI)的装置(nFIQ、nIRQ、nSWI)；用于检查已经到达处理器(2)的中断(FIQ、IRQ、SWI)影响哪个线程(THA1、THA2、THB1、THB2、THA_IDLE、THB_IDLE)的执行的装置(401、603、617)；用于将中断数据传输给与影响处理器(2)所接收中断(FIQ、IRQ、SWI)的线程(THA1、THA2、THB1、THB2、THA_IDLE、THB_IDLE)有关的操作系统(OS_A、OS_B)的装置(401、412、603、609、617)。



ISSN1008-4274

知识产权出版社出版

1. 一种包含至少一个用于运行操作系统(OS_A、OS_B)的处理器(2)的嵌入式系统(1)，特征在于，嵌入式系统(1)进一步包含:

-用于在处理器(2)中运行至少两个操作系统(OS_A、OS_B)的装置(17、401、412);

-包含第一组线程(THA1、THA2、THA_IDLE)的第一操作系统(OS_A);

-包含第二组线程(THB1、THB2、THB_IDLE)的第二操作系统(OS_B);

-用于向处理器(2)生成中断(FIQ、IRQ、SWI)的装置(nFIQ、nIRQ、nSWI);

-检查装置(401、603、617)，用于检查已经到达处理器(2)的中断(FIQ、IRQ、SWI)影响哪个线程(THA1、THA2、THB1、THB2、THA_IDLE、THB_IDLE)的执行;

-传输装置(401、412、603、609、617)，用于将中断数据传输给与影响处理器(2)所接收中断(FIQ、IRQ、SWI)的线程(THA1、THA2、THB1、THB2、THA_IDLE、THB_IDLE)有关的操作系统(OS_A、OS_B)。

2. 按照权利要求 1 的嵌入式系统(1)，特征在于，所述检查装置(401、603、617)包含至少一个、至少部分为所述至少两个操作系统(OS_A、OS_B)所公用的中断处理程序(603、607)。

3. 按照权利要求 2 的嵌入式系统(1)，特征在于，它包含移动站功能和数据处理功能，所述至少两个操作系统(OS_A、OS_B)中，第一操作系统(OS_A)涉及移动站功能的运行，第二操作系统(OS_B)涉及数据处理功能的运行。

4. 按照权利要求 3 的嵌入式系统(1)，特征在于，包含至少一个至少部分涉及移动站功能的用户接口(UI1)和至少一个至少部分涉及数据处理功能的用户接口(UI2)。

5. 按照权利要求 4 的嵌入式系统(1)，特征在于，涉及移动站功能的用户接口(UI1)和涉及数据处理功能的用户接口(UI2)包含至少部分公用的显示器(10、15)。

6. 按照权利要求 4 的嵌入式系统(1)，特征在于，涉及移动站功能的用户接口(UI1)和涉及数据处理功能的用户接口(UI2)包含至少部分公用的数据提供装置(9、14)。

7. 按照权利要求 1 的嵌入式系统(1)，特征在于，它包含用于在没有第一操作系统(OS_A)的线程(THA1、THA2、THA_IDLE)正在运行时将第一操作系统(OS_A)的执行转到第二操作系统(OS_B)的执行的装置(2，THA_IDLE)。

8. 按照权利要求 1 的嵌入式系统(1)，特征在于，它包含用于在对处理器(2)的中断(FIQ、IRQ、SWI)影响第一操作系统(OS_A)下的至少一个线程(THA1、THA2、THA_IDLE)的运行时从执行第二操作系统(OS_B)转而运行第一操作系统(OS_A)的装置(2，SCH_A)。

9. 按照权利要求 1 的嵌入式系统(1)，特征在于，至少第一操作系统(OS_A)是实时操作系统。

10. 按照权利要求 1 的嵌入式系统(1)，特征在于，处理器(2)至少包含下列方式:

-用户方式(USER 方式);

-特权方式(SVC 方式);

-不定方式(UND 方式);

--一个或多个中断方式(FIQ 方式、IRQ 方式、SWI 方式);

并且安排第一操作系统(OS_A)至少部分地在不定方式(UND 方式)中运行，安排第二操作系统(OS_B)至少部分地在用户方式(USER 方式)中运行，安排中断处理程序(603、617)在某个中断方式(FIQ 方式、IRQ 方式、SWI 方式)中运行。

11. 按照权利要求 1 的嵌入式系统(1)，特征在于，第一组线程的线程中的一个线程(THA1、THA2、THA_IDLE)包含第二操作系统(OS_B)。

12. 按照权利要求 2 的嵌入式系统(1)，特征在于，它包含至少两个用于至少一个中断(FIQ、IRQ、SWI)的中断服务例程，并包含用于从这至少两个中断服务例程中选择一个与中断(FIQ、IRQ、SWI)一起使用的装置(2)。

13. 一种用于在嵌入式系统(1)的处理器(2)中执行操作系统(OS_A、OS_B)的方法，特征在于，在该方法中:

-在所述处理器(2)中执行至少两个操作系统(OS_A、OS_B);

-涉及第一操作系统(OS_A)时，执行第一组线程(THA1、THA2、THA_IDLE);

-涉及第二操作系统(OS_B)时，执行第二组线程(THB1、THB2、THB_IDLE);

-向处理器(2)生成中断(FIQ、IRQ、SWI);

-检查已经到达处理器(2)的中断影响哪个线程(THA1、THA2、THB1、THB2、THA_IDLE、THB_IDLE)的执行;

-将处理器(2)所接收中断(FIQ、IRQ、SWI)的信息传输给与影响处理器(2)所接收中断(FIQ、IRQ、SWI)的线程(THA1、THA2、THB1、THB2、THA_IDLE、THB_IDLE)有关的操作系统(OS_A、OS_B)。

14. 按照权利要求 13 的方法，特征在于，在检查处理器(2)所接收中断(FIQ、IRQ、SWI)时，至少使用一个至少部分为至少两个操作系统(OS_A、OS_B)所公用的中断处理程序。

15. 按照权利要求 13 的方法，特征在于，在嵌入式系统(1)中，执行移动站功能和数据处理功能，所述至少两个操作系统(OS_A、OS_B)中，第一操作系统(OS_A)涉及移动站功能的执行，第二操作系统(OS_B)涉及数据处理功能的执行。

16. 按照权利要求 15 的方法，特征在于，移动站功能由第一用户接口(UI1)使用，数据处理功能由第二用户接口(UI2)使用。

17. 按照权利要求 16 的方法，特征在于，移动站功能和数据处理功能在至少部分公用的显示器(10、15)上显示。

18. 按照权利要求 16 的方法，特征在于，为使用移动站功能和数据处理功能，通过至少部分公用的装置(9、14)来提供数据。

19. 按照权利要求 13 的方法，特征在于，在没有第一操作系统(OS_A)的线程(THA1、THA2、THA_IDLE)正在运行时，进行从第一操作系统(OS_A)的执行到执行第二操作系统(OS_B)的转换。

20. 按照权利要求 13 的方法，特征在于，在处理器(2)所接收的中断影响第一操作系统(OS_A)下的至少一个线程(THA1、THA2、THA_IDLE)的执行时，进行从第二操作系统(OS_B)的执行到执行第一操作系统(OS_A)的转换。

21. 按照权利要求 13 的方法，特征在于，至少第一操作系统

(OS_A)是实时操作系统。

22. 按照权利要求13的方法，特征在于，在第一组线程的一个线程(THA1、THA2、THA_IDLE)中执行第二操作系统(OS_B)。

23. 按照权利要求22的方法，特征在于，为第一组线程的每个线程(THA1、THA2、THA_IDLE)确定优先级，为了增加第二操作系统(OS_B)的执行时间，可以提高其中执行第二操作系统(OS_B)的线程的优先级。

24. 一种包含至少一个用于运行操作系统(OS_A、OS_B)的处理器的通信设备，特征在于，通信设备进一步包含:

-用于在该处理器中运行至少两个操作系统(OS_A、OS_B)的装置(17、401、412);

-包含第一组线程(THA1、THA2、THA_IDLE)的第一操作系统(OS_A);

-包含第二组线程(THB1、THB2、THB_IDLE)的第二操作系统(OS_B);

-用于向该处理器生成中断(FIQ、IRQ、SWI)的装置(nFIQ、nIRQ、nSWI);

-检查装置(401、603、617)，用于检查已经到达该处理器的中断(FIQ、IRQ、SWI)影响哪个线程(THA1、THA2、THB1、THB2、THA_IDLE、THB_IDLE)的执行;

-传输装置(401、412、603、609、617)，用于将中断数据传输给与影响该处理器所接收中断(FIQ、IRQ、SWI)的线程(THA1、THA2、THB1、THB2、THA_IDLE、THB_IDLE)有关的操作系统(OS_A、OS_B)。

25. 按照权利要求24的通信设备，特征在于，检查装置(401、603、617)包含至少一个、至少部分为至少两个操作系统(OS_A、OS_B)所公用的中断处理程序(603、607)。

26. 按照权利要求25的通信设备，特征在于，它包含移动站功能和数据处理功能，这至少两个操作系统(OS_A、OS_B)中，第一操作系统(OS_A)涉及移动站功能的运行，第二操作系统(OS_B)涉及数据处理功能的运行。

27. 按照权利要求26的通信设备，特征在于，包含至少一个

至少部分涉及移动站功能的用户接口(UI1)和至少一个至少部分涉及数据处理功能的用户接口(UI2)。

28. 按照权利要求27的通信设备，特征在于，涉及移动站功能的用户接口(UI1)和涉及数据处理功能的用户接口(UI2)包含至少部分公用的显示器(10、15)。

29. 按照权利要求27的通信设备，特征在于，涉及移动站功能的用户接口(UI1)和涉及数据处理功能的用户接口(UI2)包含至少部分公用的数据提供装置(9、14)。

30. 按照权利要求24的通信设备，特征在于，它包含用于在没有第一操作系统(OS_A)的线程(THA1、THA2、THA_IDLE)正在运行时将第一操作系统(OS_A)的执行转到第二操作系统(OS_B)的执行的装置(2，THA_IDLE)。

31. 按照权利要求24的通信设备，特征在于，它包含用于在对该处理器的中断(FIQ、IRQ、SWI)影响第一操作系统(OS_A)下的至少一个线程(THA1、THA2、THA_IDLE)的运行时从执行第二操作系统(OS_B)转而运行第一操作系统(OS_A)的装置(2，TH1_SCH_A)。

32. 按照权利要求24的通信设备，特征在于，至少第一操作系统(OS_A)是实时操作系统。

33. 按照权利要求24的通信设备，特征在于，该处理器至少包含下列方式:

-用户方式(USER方式);

-特权方式(SVC方式);

-不定方式(UND方式);

-一个或多个中断方式(FIQ方式、IRQ方式、SWI方式);

并且安排第一操作系统(OS_A)至少部分地在不定方式(UND方式)中运行，安排第二操作系统(OS_B)至少部分地在用户方式(USER方式)中运行，安排中断处理程序(603、617)在某个中断方式(FIQ方式、IRQ方式、SWI方式)中运行。

34. 按照权利要求24的通信设备，特征在于，第一组线程的线程中的一个线程(THA1、THA2、THA_IDLE)包含第二操作系统(OS_B)。

嵌入式系统

技术领域

本发明涉及一种嵌入式系统、方法以及通信设备。

背景技术

在电子产业中，已经设计了许多有某种微处理器或相应的处理器作为基本功能部件的设备。这种处理器例如可以与微控制器定位在一起，其中它还有一些外设被设置在同一个微芯片中。这些设备也称嵌入式系统，其例子有移动站—如蜂窝网络移动站CMT(蜂窝式移动电话)、PC(个人电脑)、PDA(个人数字助理)等等。要控制这种设备的操作，经常要设计设备专用的操作系统，其基本任务是看管系统的定时、资源控制、不同程序块之间消息的处理和传输。这种操作系统通常被称为实时操作系统(RTOS)。这种实时操作系统的典型特征是，例如事实上它能被用于预测对诸如按键的外部中断的响应时间或定时、存储器资源的最小化使用，此外，实时操作系统对控制任务非常有效。此外，在实时操作系统中—例如在便携式设备中为了延长电池的使用时间—通常设计了一些用于控制负载量的属性。这种实时操作系统的大小，一般占用在 10 至 100kB 的程序代码和信息存储器(随机存取存储器RAM)。

非实时操作系统与实时操作系统不同，例如运行不同操作的响应时间比实时操作系统中的长；另一方面，在非实时操作系统中也不能对响应时间进行预测。在多任务非实时操作系统中，每个执行过程都延缓相同或更低优先级的过程的执行，其中也可能延长响应时间。此外，非实时操作系统没有在操作系统的内核中确定的响应时间，这就对预测这种操作系统的执行速度造成了困难。

移动站的存储器容量和其它系统资源有限。此外，在移动站中，经常不存在对外设—诸如可写的海量存储器(硬盘)或诸如快闪存储器的内存卡—的系统支持。在移动站中，一般不可能以动态方式将程序装入系统—就是说，移动站的软件是在制造阶段装入移动站的，也不能在有些情况下—例如由于维护—对程序进行

更新。此外，许多移动站不含存储器管理单元(MMU)，因为移动站软件通常都是由移动站制造商制造和测试的。由于这些原因，特定于设备的实时操作系统也不包含对上述属性的支持。

制造个人电脑和个人数字助理的公司已经开发了有对上述属性支持的操作系统，但是这些操作系统不是照这样实时地运行的，这样它们就不能预测运行某些操作的最大响应时间。这种操作系统的大小通常在 200kB 到数 MB，这样，要让它们适应诸如移动站的便携式机器，未必容易。

实时操作系统，特别是为数据处理设备开发的操作系统，有部分对立的要求，其中难于完成一个有可用于该操作系统的相同编程界面、同时具有所有必要属性—诸如较少的存储器占用和实时特性—的操作系统。

近来，也开发了有数据处理器与电信设备组合的电子设备。这种设备在本说明书中称为通信设备。这种设备的一个例子是诺基亚 9000 通信器(Nokia 9000 Communicator)，它是一个体积较小的便携式设备，兼有数据处理操作和移动站操作二者的作用。然而，这种组合设备的一个目的是尽可能地使用以前开发的产品，其中一个目的是例如保持用于这种新设备中的不同设备的软件的兼容。实现这种目的的方法是，为其中的数据处理部分提供其自己的处理器和其自己的操作系统，并且移动站实施例也同样有它们自己的处理器和它们自己的操作系统。这样，就有可能尽可能有效地采用以前的产品开发的结果，加速设备的推出。然而，分立的处理器通常消耗比单一处理器解决方案所可能达到的更多的电量，其中这种便携式设备需要更高效的电池，否则其执行时间就不能保持得如分立的设备可能达到的那么长。

现有技术水平的解决方案中实现在单一处理器运行两个操作系统的方法是，通过采用两个操作系统的特点来实现各操作系统。这种类型的组合存在的问题是，例如当各操作系统有非常不同的属性时，不可能执行要被组合的操作系统的所有属性。这导致的不利情况是，例如为这些操作系统开发的所有应用程序都不能使用、或者使用时需要改动应用程序。此外，当开发了新的应用时，必须对根据一个操作系统编制的应用程序按照另一个操作系统作

改动。这就加大了所需的工作量和出错的概率。

此外，专利US-5,278,973披露了一种其中能在一个处理器中使用几个操作系统的数据处理器。然而，每次只能使用这些操作系统中的其中之一。如果要变换操作系统，则要求必须取消使用中的操作系统的执行，并且要重新启动数据处理器。

发明内容

根据本发明的一个方面，提供一种包含至少一个用于运行操作系统的处理器的嵌入式系统，特征在于，嵌入式系统进一步包含：

-用于在处理器中运行至少两个操作系统的装置；

-包含第一组线程的第一操作系统；

-包含第二组线程的第二操作系统；

-用于向处理器生成中断的装置；

-检查装置，用于检查已经到达处理器的中断影响哪个线程的执行；

-传输装置，用于将中断数据传输给与影响处理器所接收中断的线程有关的操作系统。

根据本发明的另一个一个方面，提供一种用于在嵌入式系统的处理器中执行操作系统的方法，特征在于，在该方法中：

-在所述处理器中执行至少两个操作系统；

-涉及第一操作系统时，执行第一组线程；

-涉及第二操作系统时，执行第二组线程；

-向处理器生成中断；

-检查已经到达处理器的中断影响哪个线程的执行；

-将处理器所接收中断的信息传输给与影响处理器所接收中断的线程有关的操作系统。

根据本发明的另一个一个方面，提供一种包含至少一个用于运行操作系统的处理器的通信设备，特征在于，通信设备进一步包含：

-用于在该处理器中运行至少两个操作系统的装置；

-包含第一组线程的第一操作系统；

-包含第二组线程的第二操作系统；

-用于向该处理器生成中断的装置;

-检查装置，用于检查已经到达该处理器的中断影响哪个线程的执行;

-传输装置，用于将中断数据传输给与影响该处理器所接收中断的线程有关的操作系统。

本发明的一个目的是实现一种嵌入式系统，其中至少两个操作系统能被一个处理器之类执行。本发明根据的思想是，建立至少一个至少部分公用的中断处理程序，用于在各线程上检查每个中断影响哪个操作系统，其中要将中断数据传送给各操作系统。按照本发明的嵌入式系统的特征将在权利要求 1 的特征部分中所介绍的内容中表示。按照本发明的方法的特征将在权利要求 13 的特征部分中所介绍的内容中表示。按照本发明的通信设备的特征将在权利要求 24 的说明部分中所介绍的内容中表示。

本发明比现有技术水平的解决方案有显著的优点。由一个处理器耗费的负载量通常比假若将分立的处理器用于每个操作系统时的低。此外，因为能使用更多的诸如存储器的公用部件，所以减少了对空间的需求。按照本发明的嵌入式系统，使得已开发的应用程序无需作修改就有可能在其中使用，并且无论为哪个操作系统开发新应用程序，都无需首先对根据一个操作系统开发的应用程序按照另一个操作系统进行改变。与将两个操作系统组合起来使用任何一个操作系统的功能的解决方案以及每个操作系统有供自己支配的处理器的解决方案相比，本发明进一步减少了工作量和可能的出错量。此外，本发明的解决方案不减少操作系统的任何功能，它们仍然能在一定的应用或应用领域中一例如为了移动站功能的实现一被优化。

此外，由于耗电量的减少，在便携式设备中，一次充电能维持更长的操作时间。

附图说明

下面将结合各附图来更详细地说明本发明。附图简介:

图 1 表示一个操作系统的层次结构;

图 2 表示过程的一个状态模型;

图 3 表示一例过程元素(process element);

图 4a 表示按照本发明的有利实施例的嵌入式系统的简化框图;

图 4b 表示一个处理器的简化框图;

图 5 表示一个与本发明的嵌入式系统一起使用的操作系统的简化示意图;

图 6a 至 6i 表示本发明最佳实施例的嵌入式系统中处理中断的不同情形;

图 7a 和 7b 表示消息传输中的机构的简化示意图;

图 8 表示本发明最佳实施例的嵌入式系统中处理中断的一个情形中的简化有向图;

图 9 表示本发明最佳实施例的嵌入式系统中一种交叉调度选择的定时图;

图 10 表示本发明最佳实施例的嵌入式系统中一种交叉调度选择的功能图。

具体实施方式

为了便于更好地理解本发明，首先结合图 1 至 3 来说明现有技术水平的处理器和操作系统的各种解决方案。

处理器可按指令集划分成两类：复杂指令集计算机(CISC)和精简指令集计算机(RISC)。在精简指令集计算机中，去掉了一部分不常用的指令，例如，寻址方式(addressing forms)的数量可比复杂指令集计算机中的少。然而，在普通应用程序中，用非常少量的指令就能实现应用程序，其中一般就有可能用精简指令集计算机来实现必要的计算机程序。精简指令集计算机与复杂指令集计算机相比的优点是，例如能提高执行速度，另一方面，程序代码所需的存储空间也比使用复杂指令集计算机时的小。特别是，当程序存储器的容量减少时，对设备的负载量的需求也减少，另一方面，也能减少设备的体积。

在实际的实施例中经常使用一个微控制器，它包含实在的处理器和与处理器相连的辅助部件，诸如随机存取存储器(RAM)、只读存储器(ROM)、输入/输出设备(I/O)、定时器、模-数转换器(A/D转换器)、数-模转换器(D/A 转换器)等等。所以，可以将这些不同单元之间的接口线路做得尽可能地小，在这点上就能进一步减

小设备的尺寸，另一方面，还能加快这些不同单元之间信号传输的速度。微控制器制造商生成若干不同类型的微控制器，它们可能有相同的处理器，但是外围部件的数量和类型会不同，这样，如果需要的话，就能为每种应用配备最适合的微控制器。

在本说明书的前文介绍过，要为这种采用微控制器或相应处理器的设备开发操作系统，作为所谓的基本软件来运行。这个操作系统负责控制设备的不同单元、数据的读入等等，并使应用程序能在这种设备中一诸如个人电脑中使用。当计算机被开启时，操作系统被装入程序存储器中，执行某些初始化操作。随后用户就能启动所希望的应用程序，其中操作系统接收用户给出的命令，检索应用程序，将其加载到程序存储器中，然后启动应用程序。在多重处理系统中，有可能表面上象是同时地使用若干个这种应用程序，其中操作系统负责提供执行时间和这些不同应用程序之间的调度。此外，操作系统的任务是在这些不同应用程序之间传输信号，如果需要的话，还要在应用程序与外围设备之间安排信令(signalling)。

操作系统能控制进程一也即所谓的线程一的执行。一个线程可包含与一个或几个进程的执行有关的函数，另一方面，一个进程也能被划分成一个或多个线程。线程的执行是由操作系统调度程序控制的。换言之，线程是由操作系统给予进程的一种支持。

在已知操作系统中，通常能发现下述属性：命令解释器，它解释用户给出的命令一例如读取命令中的应用程序的名称并识别用户可能给出的要由应用程序传输的参数；调度程序，它照管对不同进程的资源的分配、照管中断请求服务一其任务是对中断作出反应并执行中断所要求的措施，例如向操作系统传输中断请求。此外，操作系统还有各种功能块，例如用于读键盘、写显示器、及控制可能的外部接口等等。资源例如包括主存储器、外设和处理器时间的使用。操作系统例如为每个待执行的进程保留一片存储区，进程能在其中读数据，进程也在其中读数据。如果有进程试图在为该进程保留的这个存储器区以外写数据，存储器控制单元一般阻止这种写操作并且通常会停止这种进程的运行。

许多处理器仍然使用不同的方式，诸如用户方式和特权方式

(也叫监管方式或内核方式)。由用户启动的应用程序和其它进程一般被设置成按照用户方式运行。相反，操作系统及其启动的部分进程通常以特权方式运行。用户方式的操作可能有更多的限制，例如处理器的一部分命令只以特权方式运行，在用户方式中则不能使用。在有些操作系统中，特权操作方式有若干级，例如三级。该操作方式的内核在特权方式的第一级中实现，具有的权力也最大。第二级例如有控制例程等等(可执行程序)。第三级例如有命令解释器。

处理器 2 一般包含使用中断的可能性。来自处理器 2 以外的中断请求能在一个或几个中断服务线 nIRQ、nFIQ(图 4b)上传输，其方式的变化(例如由逻辑方式 1 变为逻辑方式 0)能引起一个对处理器 2 的中断。对处理器 2 的中断请求，指令处理器 2 去转而运行相应的中断服务例程，当没有更高优先级的中断请求在等待时，这种运行几乎没有延迟。在中断服务例程中，至少要执行一部分中断处理操作。这些处理操作例如与相应的应用程序及中断的原因有关。由于运行中断服务例程会降低运行进程和操作系统的速度，所以一尤其在实时操作系统中，要使中断服务例程长度尽可能短，速度尽可能快。所以，在中断服务例程中可能要为操作系统设置例如一个状态变量一状态变量通知中断，通知有一部分中断处理措施正在等待执行的信息，其中操作系统例如按照其调度程序控制这些操作的执行。对中断服务例程的传输也可能受各种中断封锁和屏蔽的影响。当中断被禁止时，中断请求就被搁置，等待中断封锁的取消，之后，各中断请求按它们的优先级被有利地服务。在多数处理器 2 也能有选择地阻止中断，其中每个中断在中断屏蔽寄存器之类中配备以选通或禁止状态。

中断请求例如是根据外部设备给出的信号而生成的，例如，由通过串行总线接收的信息引起的串行总线中断请求，由按压键盘的按键引起的中断请求，由定时器引起的中断请求或者在某种执行进程中形成的所谓程序中断请求。能确定中断的优先级，其中较低优先级的中断服务例程的执行能被较高优先级的中断服务例程中断。相反，较高优先级的中断服务例程一般不会被较低优先级的中断服务例程中断。为了实现对操作系统的基本定时，通

常使用一个定时器，定时器以一定的间隔向处理器生成中断请求，处理器转而执行定时器的中断服务例程。所以要给定时器的中断服务以尽可能高的优先级。相反，例如显示的刷新通常则被提供以较低的优先级。

专利 US-5,515,538 描述了一种在数据处理器的多重处理操作系统中处理中断的方法。在该方法中，中断处理程序被安排成形成其自己的线程，在中断的情况下执行被引导到该线程。然而，该公开说明书中介绍的该方法只是为有一个操作系统的装置考虑的—该装置能同时包含几个用于运行这个操作系统的进程的处理器。

下面将简要地说明一种多重操作系统的操作。操作系统的调度程序每隔一定时间就检查此时应执行哪些应用程序，即进程。调度程序例如通过定时器中断来启动每个应用程序可进一步包含一个或几个线程，其中每个线程在不同的时间在某单一处理器系统中运行。一个线程例如正在读某外设接口(诸如键盘)上的数据，第二个进程可能正在处理该数据，第三个进程例如可能正在某外设(例如显示器)上写该处理过的数据。当这个应用程序被启动时，调度程序已经定义了每个线程每次的执行时间，以及运行这些进程的重复间隔时间。这样，调度程序启动线程的执行，执行时间过后，将线程转为等待，需要的话，将该线程的数据存储到存储装置中，然后启动执行新的线程。在调度程序执行了每个线程之后，调度程序就开始新一轮的执行，或者，如果达到了没有线程的间隔，就执行一个所谓的空闲线程。然而，在上述线程的执行期间，可能会发生中断请求，此时，操作系统可根据这些中断请求的优先级转而去执行中断服务例程，或者，调度程序为进行中断服务的确定一个适当时间，然后继续执行被中断的线程。例如，在一个由键盘接口实现的中断情形中，键盘中断程序读取被按键的代码并将其存储入临时存储单元中。中断服务例程结束之后，如果确定该键盘中断的优先级不超过被中断程序线程的优先级，操作系统的调度程序就为按键的检查程序确定一个执行时间，然后返回到被中断的程序线程。

附图 1 表示一个操作系统的层次结构的例子。操作系统的最

低层是内核，紧接的一层包含有关存储器控制的部分，第三层包含文件系统部分。位于这三层之上的例如是命令解释器、进程(未予示出)、接口程序、调度程序函数等等。

下面来具体说明一个操作系统的内核的结构。内核的操作的主要部分是中断处理、进程的控制和外围设备的控制。中断处理包括存储被中断程序的数据和将执行控制到正确的中断服务例程。进程的控制照管例如进程的创建、确定执行时间(调度)、结束进程的运行，定时等等。外围设备的控制功能例如包括启动数据传输处理与每个外围设备有关的一个或多个中断。在操作系统看来，进程总是处于三种状态之一中：就绪状态、运行状态或等待状态。附图 2 中表示了这种状态划分。运行状态包含此时在执行的进程，在这个状态中，含有一个处理器的系统一次最多只有一个进程。那些等待在运行状态中被执行的进程处于就绪状态。调度程序的任务是选择就绪状态的一个进程去在运行状态中执行。等待状态包含的那些正在等待某种操作(诸如按键、某种资源的重新分配等等)的进程。

调度程序将优先级最高的进程从就绪状态转变为运行状态。在运行状态中，进程通常是通过其自己的请求—例如请求其实现包括等待的某种操作系统服务—而转移到等待状态的。当这种等待结束时，发生从等待状态到就绪状态的转移。在调度程序改变其它某个为结束中断处理而要执行的进程时，该进程可从运行状态转变回就绪状态。

内核的实现一般要使用多个动态信息结构—通常是双向链表，其元素是从位于操作系统的区域中的自由存储空间分配的。例如，最好将每个进程的数据保存在进程控制块(PCB)中。进程元素已经被收集在进程表上，进程表的大小设定了系统中进程数量的上限。附图 3 中表示的本例中，例如在基本元素中包含下列信息:

-进程状态：等待、运行准备就绪或运行;

-进程的名称;

-进程的优先级;

-被中断进程的信息(环境)的存储(堆栈)的状态;

-被分配资源的数据：存储器、设备、打开文件等等；

-与跟踪进程分配相关的字段；

-与授权相关的字段。

应当指出，这只作为进程元素的实现的一个例子，在实际的实施例中，这些信息可以有很大程度的变化，进程元素中的数据可以有很大的不同。

进程符号是进程的名称，例如序列号。操作系统根据这个进程的名称来区分不同的进程。如果进程含有一个以上线程，则进程符号或进程元素的其它字段中会包含关于线程的这个号的信息。为了通知进程的状态，使用一个状态字段。这样，操作系统就在进程的状态改变时该变这个状态字段的值。调度程序用优先级字段的值来确定进程的执行顺序，其中例如该字段所含的数字值越大则该进程的优先级就越高。这样，调度程序通过比较处于等待状态的各进程的优先级字段的值，选择优先级值最大的进程作为下一个要执行的进程。

下面说明新进程的创建。在本例中，进程被存储成设备的存储装置(例如可写的海量存储器)中的一个文件。这个程序是按文件名来识别的。操作系统的内核中包含一个启动进程的服务例程，作为参数向这个服务例程给出的名称最好是这个文件的名称。该启动服务例程在进程表中为这个进程建立一个进程元素并向该进程分配一片主存区。将这个程序文件读入为其保留的存储区，将进程元素的各字段初始化以适当的值。之后，进程就被建立，等待执行。用于进程的进程符号例如可以是以后自由进程符号。进程的状态最好被设置为就绪值。操作系统的调度程序控制这些进程按照其调度程序执行。

在进程按线程执行的操作系统中，有可能将上述的进程创建(过程)分别改编成启动服务例程由进程建立一个或几个线程并在进程表上为各线程建立进程元素。

此外，操作系统还使用一种用来控制外围设备的基本功能，即所谓的设备驱动程序。设备驱动程序包含用于控制诸如显示器、键盘、编-解码器(codec)之类的设备的例程和信息结构。所需的例程例如是，设备的初始化操作，读/写、控制例程一诸如设置终

端的属性、以及处理与设备有关的中断的例程。与设备有关的变量和信息结构可以在例如所谓的设备描述符上收集。这类的信息例如可以是下列的:

-工作队列，向设备排队的传输请求的参数块在其中链接。如果同时的值和写是可能的，则有两个工作队列。参数块包含传输请求的参数例如与数据缓冲器的位置有关的信息、要传输的块的数量，等等;

-设备驱动程序例程的地址;

-传输控制中所要求的设备的变量和模式数据;

-特定于设备的参数。

在有些系统中，设备驱动程序是操作系统的进程。这些进程在一种不停的循环中运行，在这种循环中，它们首先等待传输请求消息。接到这种消息后，设备驱动程序进程启动传输并等待中断消息。操作系统的中断处理程序然后照管中断消息向设备驱动程序进程的传输。在实际的设备中，设备驱动程序进程通常有较高的优先级，操作系统的内核实时地运行。

当从可写的海量存储器读数据时，读操作按例如以下方式运行。首先从设备表搜索各设备描述符的地址。然后，从设备描述符提取读例程的地址，该地址被请求。读例程由传输请求的参数建立参数块。如果传输目前是通过该设备发生的，即该设备被另一个进程占用，该参数块就被链接在工作队列的最后，在队列中等待。如果该设备空闲，其中工作队列是空的，该参数块就被链接在工作队列的首位。随后，将块的数量变成片表面、磁道和扇区的数量，然后例如通过修改设备寄存器来启动传输。此后，等待传输的结束。传输结束之后，发生设备中断。操作系统的中断处理程序按照设备号从设备表中搜索设备描述符的地址，并从其中进一步搜索中断处理例程的地址，然后转而去执行中断处理程序的程序代码。中断处理程序最好检查核实没有错误发生。在成功地结束传输后，将该参数块脱离与工作队列的链接，唤醒一直在等待传输的进程。如果工作队列中存在排队的传输请求，则下一个传输操作几乎没有延迟地就被启动。

根据具体类型的外围设备，具体操作在某种程度上会与上述

说明的有所不同。就打印设备(诸如显示器和打印机)来说，用含有待打印字符的队列就能替代工作队列。

许多操作系统都使用报文传输机构，各进程通过它们就能互相传输报文。当使用报文队列时，进程涉及到一个或多个报文队列，向该进程传输的报文可以写入其中。进程之间的通信通常涉及缓冲技术，其中报文的发送方不必等待接收方收到报文。这样，在报文的写阶段就建立通知有报文正在等待进程的信息(状态变量)，其中进程在适当阶段读报文。

按照本发明最佳实施例的嵌入式系统中，即在(图 4a 的)通信设备 1 中，使用两个操作系统。第一个操作系统在本说明书中标注为 OS_A，主要用于实现移动站功能；第二个操作系统在本说明书中标注为 OS_B，主要用来实现数据处理功能。数据处理功能主要指个人电脑中已知的功能等，例如文件的读取、可写式海量存储器的写、数据的打印、执行应用程序—诸如所谓的组织功能(笔记本、通讯数据的保存等等)等。由于共同使用一个处理器 2 与通信设备 1 是有益的，所以有必要能将不同操作设备 OS_A、OS_B 的这些功能结合起来，使得两个操作系统 OS_A、OS_B 的特点能同时保留。此外，一个标准是需要时各功能的实时性。下面举例来说明这些不同的操作系统 OS_A、OS_B 是如何能与同一个处理器结合的。

图 5 表示第二个操作系统 OS_B 的体系结构的一个例子。每个方框表示某个进程、线程、连接接口等。各方框进一步按该方框中是用用户方式 USER 还是用特权方式 SVC 的信息而划分。其进一步分隔出来的部分是方框 501 表示的各中断方式，各中断方式在图 6 中解释。各中断方式使用快速中断方式 FIQ、正常中断方式 IRQ 或软件中断方式 SWI。第二个操作系统 OS_B 的内核(方框 502)被设置成以特权方式 SVC(监管方式)运行。相反，第二个操作系统的调度程序 SCH_B(方框 503)、连接接口(方框 504)、(诸如数据处理器的日历应用程序之类的)用户进程的各线程(506、507 和 508)、以及设备驱动程序的连接接口(方框 505)，被设置成以用户方式 USER 运行，其中它们不能使用例如操作系统的堆栈(未予示出)或其它关键的存储器部件。特权方式进一步包含设备驱动程

序(方框 509)、空闲线程(510)以及所谓的可能的超级线程(方框511)。空闲线程 510 是操作系统在没有其它进程被执行时执行的某种等待进程。超级线程 511 在本说明书中指的是响应时间与正常线程 506、507 和 508 的响应时间相比非常短的线程。超级线程 511 的响应时间可能是例如数百微秒，而正常线程 506、507 和 508 的响应时间例如是数十毫秒。本发明也能被改造成与没有超级线程 511 的操作系统共同使用。

操作系统的内核 502 具有最高优先级。由于上述的响应时间要求，超级线程 511 具有比正常线程 506、507、508 的更高的优先级。这里，响应时间指的是从事件引起的中断请求到达到线程被启动所经历的时间。这个响应时间是个期望值，但实际上正常操作中的响应时间小于—最多等于—这个期望值。在超负荷的情形中，响应时间有时会期望值。响应时间也包含中断服务例程所经历的时间。现实中的目标是将中断服务例程设计的尽可能短，这样它们执行起来就更快。按照本发明最佳实施例的一个嵌入式系统在附图 4a 中由通信设备 1 表示。它包含例如处理器 2，处理器 2 例如是个微处理器或微控制器的一部分，其中图 4a 中框图的至少一部分方框可以由微控制器的功能块构成。除了处理器 2，这个实施例中还通过所谓的专用集成电路 3(ASIC)来实现一部分控制功能。处理器 2 与专用集成电路 3 之间可以有单独的数据传输总线 4，其中数据传输速度可以提高，另一方面，第二个数据传输总线 5 没有负载。为了获得移动站功能，该通信设备配备有发送器/接收器单元 6、发送/接收天线 ANT、数字信号处理单元(DSP)、用于声音信号的编码/解码的编码解码器 8、用于使用移动站功能的第一键盘 9 以及用于向用户显示信息的第一显示器 10。此外，通信设备 1 还包含音频部件 11，在音频部件中，对麦克风 12 产生的模拟信号进行必要的模-数转换，对发往扬声器 13 的信号进行必要的数-摸转换。为了有数据处理的性能，通信设备 1 配备有第二键盘 14、第二显示器 15 和接口部件 16，接口部件有例如将打印机与通信设备连接到一起的装置。键盘 9、14 和显示器 10、15 普遍使用的共同名称是用户接口(UI)，通过用户接口，用户与通信设备 1 之间至少能进行部分通信。在有些应用中，用

户接口也可包含音频装置，诸如麦克风 12 和扬声器 13。图 1 的通信设备 1 表示了两组用户接口 UI1、UI2，第一用户接口 UI1 包含第一键盘 9 和第一显示器 10，第二用户接口 UI2 包含第二键盘 14 和第二显示器 15。第一用户接口 UI1 主要用于移动站操作，另一用户接口 UI2 主要用于数据处理操作，但是这两组用户接口 UI1、UI2 必要的话都既能用于移动站操作也能用于数据处理操作。用户接口 UI1、UI2 不必是分立的，而是也能以一个用户接口的形式实现，例如有一个显示器和/或一个键盘。

用户接口 UI1、UI2 的键盘也能以使用所谓的触摸屏的方式来实现，其中，最好在显示器顶部安置有触摸感应装置，这个装置能对例如用户手指的触摸作出反应。触摸感应装置的操作原理本身是已知的，根据的是例如电容变化或电阻变化。

通信设备 1 进一步包含存储器 17，最好包括用于存储例如自举程序代码的只读存储器 ROM、用于装载待执行的应用程序和用于在使用期间存储数据的随机存取存储器 RAM，应用程序和数据也可能来自可写式海量存储器、快闪存储器和/或非易失性随机存取存储器 NVRAM。通信设备 1 的通信部件由第二数据传输总线相连，该总线包含例如地址总线、数据总线和控制总线。然而，为简明起见，没有将这些这些单独表示出来，因为对该领域的专家来说这是现行技术中已知的。电源来自电池 18，电池与电压生成部件 19 相连，如果需要的话，电压生成部件要尽可能包含稳压器、电压转换器之类的电路，以产生不同的工作电压 V_{cc1}、V_{cc2}。电压生成部件能进一步产生一个信号，用于把电池 18 的电量降低到使得通信设备 1 能停止操作的低水平。这样，要么必须对电池重新充电，要么必须关闭工作电压。在实际实施例中，这个电量状态信息引起一个对处理器 2 的中断请求，由于这个中断服务例程而产生一个数据存储请求，其中操作系统 OS_A、OS_B 最好向显示器 10、15 生成一个消息，表示对蓄电池 18 放电并开始向存储器 17 存储数据。此外，还能自动地进行对关闭工作电压的控制。

通信设备 1 也能在它们的一些外围设备中含其它处理器，但这对本发明来说并不重要—这是为了其它目的而不是为了实现操作系统的功能。

本示例性实施例中所用的处理器 2 是个属于 ARM7 系列的处理器，由 Advanced RISC Machines 公司制造。该处理器是一个所谓的精简指令集处理器。然而，本发明并不仅限于这类处理器或精简指令集处理器，本发明也能适合与其它类型的处理器共同使用。附图 4b 表示了一个 ARM7 系列的处理器的简化框图。就处理器 2 的运行来看，中心部件是命令解释器和控制部件 401。其作用是解释程序命令并控制这些命令所要求采取的措施，响应通过快速中断线 nFIQ 和正常中断线 nIRQ 到来的中断请求，产生软件中断请求，为处理器的操作而生成外部时钟(未予示出)、定时信号之类的信号，生成控制信号，用于例如在存储器 17 与处理器 2 之间的读数据和写数据，以及用于控制此时地址总线 402 所需的地址数据的形成。

数据总线读寄存器 403 将来自数据总线 404 的程序命令放置到数据总线读寄存器 403 的内部命令队列(未予示出)，将数据放置到数据寄存器(未予示出)。命令解释器与控制部件 401 读取内部命令队列中下一个要执行的程序命令并解释它。如果需要，命令解释器与控制部件 401 从数据寄存器中读取数据并将其传送到第二内部数据总线 405(B 总线)，数据可由此经桶形位移器(barrel shifter)406 被传输到算术逻辑单元(ALU)407，第一内部数据总线 409(A 总线)也已经被定向到这里。命令解释器与控制部件 401 以相应的方式，控制将数据写到数据总线写寄存器 408，数据可由此被传输到数据总线 404。

在算术逻辑单元 407 中，例如执行加法、减法及逻辑运算，其方式是已知的。数据可以从算术逻辑单元 407 沿着第三内部数据总线 410(ALU 总线)被传输到地址寄存器 411 以及有内部数据寄存器(寄存器组)的数据寄存器块 412。数据寄存器块 412 包含用于存储数据的寄存器、状态寄存器和例如程序计数器(PC)，程序计数器用于从通信设备的存储器 17 的程序存储区显示程序命令。为此，已经将内部地址总线 413 从数据寄存器块 412 定向到地址寄存器 411。

地址寄存器 411 与地址计数器部件 414 相连，其中，程序计数器的值一般递增 1，指向程序代码中以后的位置，该位置就是

程序代码中下一个要执行的程序命令通常位于的位置。在有些情况中，被执行的程序命令—诸如子程序请求或中断请求，导致向程序代码中另一个位置的转移，此时命令计数器与控制部件将这个地址设置到程序计数器并把它传输到地址寄存器 411，成为地址寄存器的值。地址改变总线 415 从地址计数器部件 414 指向地址寄存器 411 和数据寄存器部件 412。

处理器 2 进一步包含一个乘法部件 416，用于执行乘法和除法。乘法部件 416 与第一内部数据总线 409 和第二内部数据总线 405 相连。上述处理器 2 也包含上述以外的其它连接线。不过没有必要在这个上下文中详细说明。

处理器 2 的数据寄存器部件 412 中的一部分数据寄存器能在处理器的所有操作方式中使用，此外，为不同的操作方式保留了这种在其它操作方式中关闭的数据寄存器。通过这种安排，就有可能减少特别是与中断情况相联系的对存储数据的需要。此外，还为不同操作方式提供专用堆栈指针，其中每种操作方式—如果需要的话—都能支配存储器 17 中为堆栈保留的自己的存储区。

处理器 2 最好能被设置成至少三种操作方式：用户方式，这种方式中，资源的使用是成区划分的；特权方式(监管方式)，其中能使用处理器的所有资源；不定方式，它是特权方式的一种特殊形式，一般用于处理器 2 在程序代码中检测到不明命令时的情况中。在这个最佳实施例中，特地将这种不定方式用作将在本说明书中以后作说明的一种方式。在这个处理器 2 中，不定方式为此是有用的。需要的话，可用其它某种特权方式作为相应的方式。此外，对处理器 2 来说，可使用三种类型的中断：快速中断(FIQ，快速中断请求)、普通中断(IRQ，中断请求)和程序中断(SWI，软件中断请求)。快速中断 FIQ 比普通中断 IRQ 的优先级高，它们分别比程序中断的优先级更高。所以，就响应时间而言，最好安排重要的中断使用快速中断 FIQ。

当启动通信设备 1 时，在处理器 2 的控制下，将一段激活的程序代码最好从只读存储器或非易失性随机存取存储器存储到存储器 17 中，例如执行存储器检查的程序命令在激活的程序代码中。与开始这种启动相联系，还要加载文件的控制例程，其中如

果需要的话，可通过文件的控制例程来将操作系统 OS_A、OS_B 的程序代码加载到存储器 17 中。这是许多数据处理设备和操作系统中的已知技术。操作系统 OS_A、OS_B 的程序代码也能存储到例如只读存储器 ROM、非易失性存储器 NVRAM 或电可擦可编程只读存储器 EEPROM 中。所以，直接从存储位置使用操作系统 OS_A、OS_B 是有可能的(XIP 技术、在原位执行)。

加载操作系统 OS_A、OS_B 之后，启动它们。所以，第一步骤例如是执行对第二操作系统 OS_B 的启动操作，其中，操作系统的进程表、存储区、中断服务例程、报文队列和其它相应的描述符等等被创建，设备驱动程序和数据类型被加载，中断被允许。在下一个步骤中，启动执行相应的对第一操作系统 OS_A 的启动操作。当对第二操作系统 OS_B 的启动操作已经首先执行后，在对第一操作系统 OS_A 的启动操作中，例如操作系统 OS_A 的中断函数—中断的开始和中断的结束—被动态地连接到中断处理程序(例如，功能的开始地址被设置在存储器 17 中)。此外，形成一个连接中断处理程序的接口，用于调度第一操作系统 OS_A 的线程。

设备驱动程序执行对外围设备的方式的初始设置等操作。最好为操作系统的数据存储器 RAM 保留一个固定的区域(静态分配)，与普通操作中按需分配存储区域(动态分配)的选择相比，静态分配能更好地保证有充分的执行速度。

启动操作之后，就启动空闲线程的执行，可能启动某个应用程序的加载和执行。例如，有可能启动移动站功能，其中通信设备 1 准备好接收呼叫、报文、传真等，而用户如果希望的话，能发出呼叫、发送报文或传真—如果通信设备 1 中设置了这些功能的话。

如果本发明的通信设备 1 中发生中断，处理器就去执行一个中断解释程序或调度程序，存储被中断线程的状态。然后，调度程序启动中断服务例程。启动哪个中断服务例程，通常与中断请求的原因有关。这是能够推断出来的，其方式例如是，处理器 2 有个状态寄存器，其内容传达中断请求的原因。例如，由于键盘中断，状态寄存器的对应位转变成另一逻辑状态(例如从逻辑 0 状态到逻辑 1 状态)，处理器 2 检查该寄存器不同的各位，并启动相

应的中断服务例程。在有若干中断请求有效的情况下，优先级最高的中断服务例程最先启动。

处理器 2 也是已知的，它有若干条中断线 nFIQ、nIRQ 可供使用，其中中断线 nFIQ、nIRQ 中的状态变化产生中断请求。可以为每个中断线 nFIQ、nIRQ 定义一个特定的中断服务例程。此外，将这些中断服务例程的地址可以是表格形式的，其中处理器 2 按中断从表中提取中断服务例程的起始地址并将其设定为其程序计数器的值，下一个要执行的命令就在这个地址中。

在本发明中，有可能改变为中断 FIQ、IRQ、SWI 定义的部分、甚至全部中断服务例程。例如，有些应用程序可以为中断 FIQ、IRQ、SWI 定义要与其相联系使用的新的中断服务例程。当执行该应用程序时，处理器 2 修改存储器 17 中的中断服务例程。在本发明的一些有益的实施例中，处理器 2 也可能只修改这种要使用另一个中断服务例程的中断的起始地址。该中断服务例程的实际代码存储在存储器 17 中。这意味着存在中断服务例程的起始地址的某种地址表。也可能存在与上述不同的其它方案，来定义对中断 FIQ、IRQ、SWI 应当使用哪个中断服务例程。

中断服务例程中所要求的必要操作被执行，例如读键盘缓冲区和在存储器中存储数据。此外，中断服务能生成要发送到某线程或操作系统的信号。本说明书以后的部分将更详细地说明按本发明最佳实施例的通信设备 1 的操作。

中断服务响应时间通常甚至小于上述超级线程的相应时间，例如约 100 微秒。影响中断服务这种响应时间的因素例如包括，中断在某部分程序代码中已经被禁止了多久。中断取消可能涉及所有中断，或者涉及优先级低于特定水平的中断；中断也能通过中断的屏蔽寄存器等被暂时取消。这样，中断请求停留子存储器中，当这个中断请求的中断取消被撤销并且没有更高优先级的线程被执行时，就执行该中断服务例程。

当没有第一操作系统 OS_A 的线程正在执行时，会发生从第一操作系统 OS_A 到第二操作系统 OS_B 的转移。这样，从第一操作系统 OS_A 看来，执行发生在空闲线程中。

在通信设备 1 运行期间，可能也需要在不同操作系统 OS_A、

OS_B 中运行的应用程序之间传输信息。这在两个操作系统 OS_A、OS_B 至少使用部分公用资源时特别需要。例如，键盘 9、14 可包含公用的键，或者可以不同单独的键盘 9、14，而是使用一个公用的键盘。另一方面，用一个其中已经安排有不同操作系统 OS_A、OS_B 的连接接口的驱动程序来照管对这种资源的控制，通常是敏感的。这也便于对有一个以上的操作系统 OS_A、OS_B 试图同时使用相同资源的那种情况进行控制。

以下就不同操作系统 OS_A、OS_B 来说明本发明的通信设备的操作。在本发明这个最佳实施例中，第一操作系统 OS_A 是所谓的实时操作系统，其中为其确定了某些执行时间要求。第二操作系统 OS_B 的执行时间不是那么关键，但是考虑到用户的舒适性，第二操作系统 OS_B 也必须在合理时间内执行分配给它的操作。

第一操作系统 OS_A 的线程具有例如在嵌入式系统相同的设计阶段就确定了的优先级。采用优先级就有可能影响例如关键线程的执行顺序和响应时间。第一操作系统的调度程序 SCH_A 此时将线程设置为在一轮执行中最先运行的最高优先级。在实际系统中这意味着本发明最佳实施例的通信设备 1 的处理器 2 执行第一操作系统 OS_A 的调度功能，该调度功能已经由处理器 2 的程序命令实现。当运行线程时，处理器 2 按线程中编制的程序命令来执行措施。

执行了在第一操作系统 OS_A 中运行的各进程的所有线程 THA1、THA2 后，处理器 2 转而执行第二操作系统 OS_B 的操作，例如延迟的功能调用 DSR(被延迟服务例程，delayed service routine)或正在等待的线程 THB1、THB2。第一操作系统 OS_A 的线程 THA1、THA2，除中断处理期间的短时停顿外，总是有权运行的。本说明书后文提供对此的说明。在本发明这个最佳实施例中，在第一操作系统 OS_A 看来，第二操作系统 OS_B 的执行，相当于执行第一操作系统 OS_A 的第一个线程，诸如空闲线程 THA_IDLE 或超级线程。为了实现这一点，第一操作系统 OS_A 配备有代表空闲线程 THA_IDLE 的信息结构(未予示出)，不过在本实施例的实现中，第一操作系统 OS_A 不含有适当的空闲线程。

在转而执行第二操作系统 OS_B 之后，处理器 2 根据第二操作

系统 OS_B 的运行线程 THB1、THB2 的优先级和调度惯例，执行第二操作系统 OS_B 的调度功能。首先执行被被延迟服务例程 DSR—如果它们有些正在等待的话，然后最好按照优先级顺序，启动线程 THB1、THB2 的的执行。在执行了要被执行的线程 THB1、THB2 之后，线程 THB1、THB2 都不处于运行方式，处理器 2 就转到第二操作系统 OS_B 的空闲线程 THB_IDLE，在此停留，直到需要新一轮的执行。

在第二操作系统 OS_B 的执行期间，几乎任何阶段都可能发生向第一操作系统 OS_A 的转变。这最好以这样的方式发生，即由于某中断请求，处理器 2 转而执行中断服务例程，其中第一操作系统 OS_A 被启动的方式以后叙述。当第一操作系统 OS_A 中的执行已经前进到空闲线程 THA_IDEL 时发生向第二操作系统 OS_B 的返回。这样，在第二操作系统 OS_B 看来，第一操作系统 OS_A 的执行是运行中断服务例程。

本发明的一个目标一直是实现操作系统 OS_A、OS_B 的这样一种方式的联系，这种方式要使不同操作系统 OS_A、OS_B 的程序代码中的变化尽可能地少，另一方面，操作系统 OS_A、OS_B 的属性在应用开发者看来没有显著变化。在按照本发明最佳实施例的通信设备中，这种联系建立在中断服务例程的层次上。当发生设备中断 FIQ、IRQ 或程序中断 SWI 时，处理器 2 转而执行中断服务例程。要么所有中断能共用这个程序—此时处理器 2 的命令解释器与控制部件 401 如果需要的话通过检查寄存器之类就能推断出中断的原因，要么可以有若干个对应不同设备中断 FIQ、IRQ 或程序中断 SWI 的不同中断服务例程。在中断服务例程的开始，向中断开始例程增加一个程序调用(功能调用)，该调用中改变操作系统 OS_A、OS_B 的内部状态，其中操作系统 OS_A、OS_B 中的数据通知中断服务正在运行。向中断服务例程的转移、子程序的执行、状态数据的研究以及其它相应操作都是在命令解释器与控制部件 401 的控制下进行的，本说明书中较早时说明过解释器与控制部件，另一方面，这也是本领域的技术人员已知的技术。

随后，执行中断处理程序的功能，这意味着命令解释器与控制部件 401 执行为已经到达处理器 2 的中断所确定的程序命令，

这些程序命令最好存储子存储器 17 中。在中断处理程序程序中，要研究中断的原因并推断中断处理要求执行第一操作系统 OS_A 还是第二操作系统 OS_B 的线程。如果中断处理要求执行第一操作系统 OS_A 的线程，中断处理程序就将这个信息设置到第一操作系统的调度程序 SCH_A。如果中断处理要求执行第二操作系统 OS_B 的线程，中断处理程序就生成一个要在第一操作系统 OS_A 的各线程执行之后执行的被延迟服务例程 DSR。这种被延迟服务例程 DSR 也能通过第一操作系统 OS_A 的运行的线程来生成。这个被延迟服务例程 DSR 能通知第二操作系统 OS_B 的调度程序 SCH_B 中断处理要求执行第二操作系统 OS_B 中的某些线程。在有些情况中，中断处理程序也可能不用被延迟服务例程 DSR 就将该数据设置到第二操作系统的调度程序 SCH_B。

在中断服务例程中也有可能执行其它操作，例如向缓冲器读入数据，其中数据是在应用程序中该数据所涉及的线程处于执行阶段时被读入的。

在中断程序的结尾增加了一个中断结束功能请求，其中设置例如中断服务例程已经结束的信息。此外，该功能还返回关于在中断服务中是否执行了第一操作系统 OS_A 中的、从第一操作系统的调度程序 SCH_A 要求给予这些线程执行时间的任何线程的信息。中断服务例程结束后，处理器 2 的方式和第一操作系统 OS_A 的状态改变。如果中断处理曾导致对执行第一操作系统 OS_A 的线程的需要，则下一个执行的程序就是第一操作系统 OS_A 的调度程序，其中，将新的状态(就绪)赋予中断导致过对其采取措施(例如读缓冲器中信息)的线程。第一操作系统 OS_A 的调度程序 SCH_A 按优先级顺序控制执行第一操作系统 OS_A 的那些正在等待执行的(处于就绪状态的)线程。

在第一操作系统 OS_A 的处于第一操作系统 OS_A 的中断服务中或运行中的线程中，在有些情况中有可能生成被延迟服务例程 DSR，其中第二操作系统 OS_B 的线程的状态被改变。被延迟服务例程 DSR 是在处理器 2 已经执行了第一操作系统 OS_A 的所有等待执行的线程之后被处理的。处理了被延迟服务例程之后，进一步执行第二操作系统 OS_B 的调度程序例程，以便最好按优先级顺序

来运行第二操作系统 OS_B 的那些正在等待执行的线程。

在上述措施的执行期间可能发生的新的中断中，操作再一次被转移到中断处理，以后的操作按上述原则发生。

在中断之后发生的操作，以运行具有当时最高有效优先级的操作的方式继续。这受到例如中断时刻处理器 2 运行的阶段的影响，也受中断原因的影响。附图 6a 至 6i 中展示了一些其它的情况，将在以后作更详细的说明。嵌入式系统 1 的一个例子是其中实现了移动站功能和数据处理功能二者的通信设备，诸如个人数字助理 PDA 的功能。该通信设备的处理器 2 中使用两个操作系统 OS_A、OS_B。第一操作系统 OS_A 主要与移动站的控制和使用有关。第二操作系统 OS_B 主要与数据处理功能的控制和使用有关。

在图 6a 至 6h 的示例性情形中，这些情形涉及所谓的普通中断 IRQ(中断请求)。在按照本发明最佳实施例的这个通信设备 1 中也能采用图 6i 的例子中所示的所谓快速中断 FIQ(快速中断请求)。快速中断的优先级最好高于普通中断 IRQ 的优先级。此外，也可采用软件中断 SWI，其优先级最好低于普通中断 IRQ 的优先级。附图 6a 至 6i 中所示的各阶段在快速中断 FIQ 和程序中断 SWI 中也能改编，所以本说明书中主要只讨论普通中断 IRQ。

被延迟服务例程是第二操作系统 OS_B 的一个特性，能用于例如不同线程之间的报文传输，需要的话，也能用于不同操作系统 OS_A、OS_B 之间的报文传输。被延迟服务例程 DSR 将第二操作系统 OS_B 的线程的状态改变为等待执行的状态(就绪状态)。在第二操作系统 OS_B 中，被延迟服务例程 DSR 的优先级高于各线程的优先级，这样它们就能在第二操作系统的各线程运行之前被处理。

本例中的处理器 2 包含以下方式:

-用户方式 USER

-特权方式 SVC

-不定方式 UND

-快速中断方式 FIQ

-普通中断方式 IRQ

-软件中断方式 SWI。

本发明的这个最佳实施例中将不定方式 UND 用作标准运行方

式，尽管其在本例中所用的处理器 2 中的缺省值的应用是不定方式。

图 6a 表示的情形中，由于普通中断，处理器 2 执行第二操作系统 OS_B 下的线程 THB1(方框 601)，它例如包含通信设备 1 的数据处理功能中的一个与由用户启动的应用程序有关的进程。处理器 2 处于用户方式 USER。该普通中断导致该方式转变为普通中断方式 IRQ，其中处理器 2 转而运行普通中断功能的开始符(方框 602)，此后运行普通中断的服务程序(方框 603)。该中断没有导致需要运行第一操作系统 OS_A 的线程，其中在中断后没有调用第一操作系统的调度程序 SCH_A。在该中断中也没有生成被延迟服务例程 DSR。在中断处理的结尾，调用普通中断功能的结束符(方框 604)，在其中设置中断已经结束的信息。在该普通中断的服务程序结束之后，将处理器 2 的方式变回用户方式 USER。执行从第二操作系统 OS_B 被中断的线程 THB1 继续(方框 605)。

中断的原因例如是打开通信设备 1 的机盖(未予示出)，其中开关 S1 指出机盖的位置改变了状态，即打开了，处理器 2 通过读取与该开关相连的标识线 20 的逻辑状态(0/1)就能检查这个开关 S1 的位置。在这个案例中机盖的打开没有立即导致任何行动。

图 6b 表示的情形中，由于普通中断，处理器 2 正在运行第一操作系统 OS_A 的空闲线程 THA_IDLE(方框 606)。在这种情形中，方式从不定方式 UND 转变为普通中断方式 IRQ，处理器 2 转而运行普通中断功能的开始符(方框 602)，然后运行普通中断的服务程序。该中断没有导致需要运行第一操作系统 OS_A 的线程，其中在中断后没有调用第一操作系统的调度程序 SCH_A。在该中断中也没有生成被延迟服务例程 DSR。在中断处理结束后，请求普通中断功能的结束符(方框 604)，在其中设置中断已经结束的信息。在该普通中断的服务程序结束之后，将处理器 2 的方式变回不定方式 UND。执行从空闲线程 THA_IDLE 继续(方框 605)。

中断 IRQ 的原因例如是，在通信设备 1 的机盖打开时用户按下数据处理功能的键盘 14 的一个键。中断处理程序检查在这种情况下按下相应的键对移动站功能的运行是否有任何影响，并证实没有影响，该中断没有导致在移动站功能中采取措施。

图 6c 表示的情形中，在发生一个普通中断时，处理器 2 正在运行第二操作系统 OS_B 的线程 THB1(方框 601)。该线程例如包含通信设备 1 的数据处理功能中与由用户启动的应用程序有关的进程。这种情形在图 8 中通过有向示意图也有表示。处理器 2 处于用户方式 USER。所以，方式转变为普通中断方式 IRQ，处理器 2 转而运行普通中断功能的开始符(方框 602)，然后运行普通中断的服务程序(图 6c 的方框 603，图 8 中的箭头 801 和 802)。该中断导致了需要运行第一操作系统 OS_A 的至少一个线程，但在中断中没有生成被延迟服务例程 DSR。在中断处理结束后(箭头 803)，调用请求普通中断功能的结束符(方框 604)，设置中断已经结束的信息。在该普通中断的服务程序结束之后，将处理器 2 的方式变回不定方式 UND，同时调用第一操作系统的调度程序 SCH_A，该调度程序如果需要的话为各线程设置新的状态(图 6c 的方框 608，图 8 中的箭头 804)。随后，第一操作系统 OS_A 的正在等待执行的线程(就绪方式的线程)被运行(箭头 805 至 808)。然后，当没有线程等待运行时，将处理器 2 的方式变回用户方式 USER，执行从第二操作系统 OS_B 被中断的线程 THB1 继续(图 6c 的方框 605，图 8 中箭头 809、810 和 811)。

在一个表示这种情形的例子中，通信设备 1 已经收到一个未应答的呼叫。所以，移动站功能的显示器 10 以及数据处理功能的显示器 15 显示一条表示有已接收、未应答的呼叫的消息，例如正文“1 missed call”。用户按下数据处理功能的回车键(未予示出)，期间就产生一个普通中断 IRQ。在中断处理中，能用来从移动站功能的显示器 10 删除该正文的第一操作系统 OS_A 的线程被设置为等待运行方式。中断结束后，处理器 2 转而运行第一操作系统的调度程序 SCH_A，其中，第一操作系统 OS_A 的线程能在已经到达运行方式后从移动站功能的显示器 10 删除该正文。

图 6d 表示的情形中，当发生普通中断时，处理器 2 正在运行第二操作系统 OS_B 下的线程 THB1(方框 601)并被设置在用户方式 USER。所以，方式被转变为普通中断方式 IRQ，处理器 2 转而运行普通中断功能的开始符(方框 602)，然后运行普通中断的服务程序(方框 603)。该中断导致了需要运行第一操作系统 OS_A 的至

少一个线程。此外，在中断中，以及/或者在中断之后运行的第一操作系统 OS_A 的某线程中，生成了一个被延迟服务例程 DSR。

在中断处理结束后，调用普通中断功能的结束符(方框 604)，设置中断已经结束的信息。在普通中断的服务程序结束之后，将处理器 2 的方式变回不定方式 UND，调用第一操作系统的调度程序 SCH_A，该调度程序如果需要的话为各线程设置新的状态，控制第一操作系统 OS_A 的正在等待执行的线程按照其调度过程执行(方框 608)。当没有第一操作系统 OS_A 线程还在等待运行后，第二操作系统 OS_B 的被延迟服务例程 DSR 被处理，然后，将处理器 2 的方式变回用户方式 USER，执行从第二操作系统 OS_B 的被中断线程 THB1 继续(方框 605)，这是因为，被该状态的被延迟服务例程改变了的第二操作系统 OS_B 的线程，优先级比第二操作系统 OS_B 的被中断线程 THB1 的低。

可以举一个例子，情况是用户已经用数据处理功能的键盘 14 写了一条短消息(SM)。按下传输键时导致一个普通中断 IRQ。在中断处理中，第一操作系统 OS_A 的一个或多个线程被设置为等待运行方式，以便从缓冲器读取该待传输的消息以及传输，而在数据处理功能的显示器 15 上则生成一个图标，由其向用户通知将有消息被传输。中断结束后，处理器 2 转而运行第一操作系统的调度程序 SCH_A，其中，第一操作系统 OS_A 的线程在运行方式中传输该消息，然后，生成一个用来删除该图标的被延迟服务例程 DSR。随后，第二操作系统的调度程序 SCH_B 将第二操作系统 OS_B 的被中断线程 THB1 设置成运行方式，因为其优先级在这个示例性情形中高于第二操作系统 OS_B 的图标删除线程。以后这个线程到达运行方式，从数据处理功能的显示器 15 删除该图标。

图 6e 表示的情形中，当普通中断到来时，处理器 2 正在运行第一操作系统 OS_A 的空闲线程 THA_IDLE(方框 606)并被设置为不定方式 UND。所以，方式转变为普通中断方式 IRQ，处理器 2 转而运行普通中断功能的开始符(方框 602)，然后运行普通中断的服务程序。该中断导致需要运行第一操作系统 OS_A 的至少一个线程。在该中断和/或第一操作系统 OS_A 的线程中生成了一个被延迟服务例程 DSR，这进一步导致第二操作系统 OS_B 的一个或多个

线程的状态转变成等待执行(就绪)。在中断结束后，调用普通中断功能的结束符(方框 604)，在其中设置中断已经结束的信息。在该普通中断的服务程序结束之后，改变处理器 2 的方式和第一操作系统 OS_A 的状态。处理器 2 的方式变成不定方式 UND，然后请求第一操作系统的调度程序 SCH_A(方框 608)，它将各线程依次设置成运行方式，即运行第一操作系统 OS_A 的正在等待执行的(处于就绪方式的)线程。在第一操作系统 OS_A 没有正在等待执行的线程后，就处理第二操作系统 OS_B 的被延迟服务例程(方框 609)，然后调用第二操作系统的调度程序 SCH_B(方框 610)，它按第二操作系统 OS_B 的运行顺序—最好按优先级顺序，将正在等待执行的第一个线程设置成运行方式。然后，处理器 2 的方式被设置回用户方式 USER，运行从第二操作系统 OS_B 的、此时具有最高优先级的线程 THB1 继续(方框 605)。第二操作系统的调度程序 SCH_B 按其调度原则控制各线程的运行。

在示例性情形中，一条短消息到达了通信设备 1，例证是移动站功能的显示器 10 上和数据处理功能的显示器 15 上的图标。用户按下移动站功能的键盘 9 上的阅读键(未予示出)，引起一个普通中断。在中断处理中，要把第一操作系统 OS_A 的一个或多个线程设置为等待运行方式，以便从移动站功能的显示器 10 删除图标并在数据处理功能的显示器 15 上显示收到的消息。此外，还生成一个被延迟服务例程 DSR，以便从数据处理功能的显示器 15 删除图标。处理器 2 在中断后运行第一操作系统 OS_A 的调度功能，处理被延迟服务例程 DSR，然后转而去运行第二操作系统 OS_B 的调度功能，其中第二操作系统的线程在到达执行之后从数据处理功能的显示器 15 删除图标。

图 6f 表示的情形中，当普通中断到来时，处理器 2 执行第二操作系统 OS_B 下的线程 THB1(方框 601)并被设置为用户方式 USER。所以，方式被转变为普通中断方式 IRQ，处理器 2 转而运行普通中断功能的开始符(方框 602)，然后转至普通中断的服务程序(方框 603)。该中断导致了需要运行第一操作系统 OS_A 的至少一个线程。此外，在中断中，以及/或者在中断之后运行的第一操作系统 OS_A 的某线程中，生成了一个被延迟服务例程 DSR，这

进一步导致对第二操作系统 OS_B 的各线程的执行时间的重新设置。在中断处理结束后，调用普通中断功能的结束符(方框 604)，设置中断已经结束的信息。在普通中断的服务程序结束之后，处理器 2 的方式和第一操作系统 OS_A 的状态被改变。处理器 2 的方式变成不定方式 UND，然后，调用第一操作系统的调度程序 SCH_A，该调度程序—如果需要的话—为线程设置新的状态(方框 608)。运行第一操作系统 OS_A 的正在等待执行的线程(处于就绪方式中的线程)。然后，当没有等待运行的第一操作系统 OS_A 线程剩下时，第二操作系统 OS_B 的被延迟服务例程 DSR 被处理(方框 609)，然后，调用第二操作系统 OS_B 的调度程序 SCH_B(方框 610)，它将第二操作系统 OS_B 正在等待的、排在运行顺序中第一位的线程设置成要在运行方式中执行。在本例中，中断处理导致被中断线程 THB1 还不是要前进到运行方式的线程—相反却是第二操作系统 OS_B 的另一个、其状态变成等待运行的线程 THB2。将处理器 2 的方式变回用户方式 USER，然后处理器 2 转而去运行被设置成运行方式的线程 THB2(方框 611)。被中断线程 THB1 依次到达运行方式(方框 605)。

作为上述的一个例子的情形中，用户在用数据处理功能的键盘 14 写一条短消息并将充电器(未予示出)连接到通信设备 1。这导致一个普通中断 IRQ。在中断处理中，在移动站功能的显示器 10 上生成通知进行充电的图标的第一操作系统 OS_A 的线程，被设置为等待运行方式。此外，通过一个被延迟服务例程 DSR，在数据处理功能的显示器 15 上生成一个相应的图标。

图 6g 表示的情形中，普通中断到来时，处理器 2 在运行第二操作系统 OS_B 下的线程 THB1(方框 601)，它例如是通信设备 1 的数据处理功能中的建立连接的应用程序。处理器 2 处于不定方式 UND。所以，将方式转变为普通中断方式 IRQ，处理器 2 转而运行普通中断功能的开始符(方框 602)，此后运行普通中断的服务程序(方框 603)。中断没有导致需要运行第一操作系统 OS_A 的线程，其中在中断后没有调用第一操作系统的调度程序 SCH_A。在中断处理结束后，调用普通中断功能的结束符(方框 604)，在其中设置中断已经结束的信息。在普通中断结束之后，处理器 2 的方式

和第一操作系统 OS_A 的状态改变。处理器 2 的方式变回不定方式 UND。运行从被中断线程 THB1 继续(方框 613)。尽管在中断中生成过第二操作系统 OS_B 的被延迟服务例程 DSR，这些 DSR 都会在第一操作系统的所有运行线程被执行之前得到处理。

例如用户正在拨电话号码时，由于按键，产生一个普通中断 IRQ。在中断处理中，按键被读入存储器 17 并在移动站功能的显示器 10 上显示，然后中断返回，等待下一个按键。

图 6h 表示的情形中，普通中断到来时，处理器 2 在运行第二操作系统 OS_B 下的线程 THB1(方框 612)，它例如是通信设备的数据处理功能中的建立连接的应用程序。处理器 2 处于不定方式 UND。所以，将方式转变为普通中断方式 IRQ，处理器 2 转而运行普通中断功能的开始符(方框 602)，然后运行普通中断的服务程序(方框 603)。中断导致需要运行第一操作系统 OS_A 的至少一个线程。在中断结束后，调用普通中断功能的结束符(方框 604)，在其中设置中断已经结束的信息。在普通中断的服务程序结束之后，改变处理器 2 的方式和第一操作系统 OS_A 的状态。处理器 2 的方式变成不定方式 UND，然后请求第一操作系统的调度程序 SCH_A(方框 608)，如果需要，它为各线程设置新状态。随后，按照优先级顺序，运行第一操作系统 OS_A 的正在等待执行的(处于就绪方式的)线程。在本例中，中断导致第二线程 THA2 立即前去执行(方框 614)，被中断线程 THA1 可能要稍后才轮到去运行(方框 613)。尽管在中断中形成过第二操作系统 OS_B 的被延迟服务例程，它们不会在第一操作系统 OS_A 的运行线程被执行之前就被运行。

例如用户拨电话号码后，按下听筒提起(handset-up)键(未予示出)，引起一个普通中断。中断处理之后，第一操作系统的调度程序 SCH_A 将一个建立连接的线程设置成执行。

图 6i 进一步表示的情形中，当普通中断服务例程正在运行时(方框 615)发生一个快速中断 FIQ。处理器 2 处于普通中断方式 IRQ，于是转为快速中断方式 FIQ 并运行快速中断功能的开始符(方框 616)，然后运行快速中断的服务程序(方框 617)。在快速中断处理结束后，调用快速中断功能的结束符(方框 618)，设置快速

中断 FIQ 已经结束的信息。在快速中断的服务程序结束之后，将处理器 2 的方式变回普通中断状态 IRQ，然后，运行从被中断的位置继续(方框 619)。

作为这种情况的例子是，当处理器 2 正在运行键盘 9、14 的中断处理时，安装在通信设备 1 中的——例如专用集成电路 3 中的定时器(未予示出)生成一个快速中断 FIQ。处理器 2 读取定时器的值并将其存入存储器 17。然后，处理器 2 返回到键盘 9、14 的中断处理。

在处理器 2 的上述方式转换中，处理器 2 最好开始使用不同的寄存器，不必总要把数据转移到临时存储单元。然而在有些中断情况中，可能需要数据转移，这在某种程度上会拖延中断的处理。还有，有些用作处理器 2 的微处理器并不包含上述的所有方式，这些不同的方式可通过方式变量等指明。

有时也可能需要确定对第二操作系统 OS_B 的响应时间。所以，第一操作系统的调度程序 SCH_A 可以在为第二操作系统 OS_B 剩下的执行时间会小于响应时间所要求的时间的情况下提高第一操作系统 OS_A 的空闲线程 THA_IDLE 的优先级。这样，第一操作系统 OS_A 的空闲线程 THA_IDLE 在其它非关键线程之前轮到运行，导致转而去运行第二操作系统 OS_B。这种提高优先级的方法，例如是用一个被设置为在预定时间后中断的定时器来实现的。如果当定时器中断到达时处理器 2 尚未到达第一操作系统 OS_A 的空闲线程 THA_IDLE，就调用第一操作系统的调度程序 SCH_A，由它来提高空闲线程 THA_IDLE 的优先级。不过这里应当注意，这时最好在提高空闲线程 THA_IDLE 的优先级后只在较短的时间内保持有效，以便也定时地为第一操作系统 OS_A 的非关键线程提供执行时间。空闲线程 THA_IDLE 的优先级不能被提高到太高的水平，因为第一操作系统 OS_A 的关键线程必须得到执行。这种调度也可以被称作交替式调度。

附图 9 和 10 表示上述交替式调度选择方案的一个例子。图 9 中描述的是一例情况的时序图，这种情况是，要执行第一操作系统 OS_A 中运行的进程的运行线程 THA1、THA2、THA3 和第二操作系统 OS_B 的运行线程 THB1、THB2、THB3。图 10 中为进一步阐明

交替式调度方式而描述了第一操作系统 OS_A 的一个进程图。

在这个示意图中出现了在第一操作系统 OS_A 中运行的进程的三个线程 THA1、THA2、THA3 和在第二操作系统 OS_B 的运行的进程的三个线程 THB1、THB2、THB3，但是应当明白，可以有更多或更少的在操作系统 OS_A、OS_B 中运行的线程。

第一操作系统 OS_A 的第一线程 THA1 例如是电话应用程序，具有相当高的优先级。第一操作系统 OS_A 的第三线程 THA3 例如是为用户接口配备的。在第一操作系统 OS_A 中，第二线程 THA2 的优先级低于第一线程 THA1 的优先级，第三线程 THA3 的优先级低于第二线程 THA2 的优先级。

第二操作系统 OS_B 的第一线程 THB1 例如是用于通过接口部件 16 通信的程序。第二操作系统 OS_B 的第二线程 THB2 和第三线程 THB3 例如是一些 PDA 应用程序—例如一个记事本应用程序和一个日历应用程序。本例中，假设第二操作系统 OS_B 的第一线程 THB1 的优先级高于第二操作系统 OS_B 的第二线程 THB2 的优先级，第二操作系统 OS_B 的第三线程 THB3 在这三个线程中优先级最低。

图 9 中用较粗的线条表示线程的执行。例如，在时刻 T1，开始轮到第一操作系统 OS_A 的第二线程 THA2 运行。这由标注号 901 表示。在稍后的时刻 T2，开始轮到第一操作系统 OS_A 的第三线程 THA3 运行(902)。也假设第三线程 THA3 是第一操作系统 OS_A 的空闲线程 THA_IDLE 之前的最后一个线程。所以过了一会后，在时刻 T3，开始轮到第二操作系统 OS_A 的第一线程 THB1 运行(903)，检查是否需要通过接口部件 16 通信。在稍后的时刻 T4，开始轮到第二操作系统 OS_B 的第二线程 THB2 运行(904)。在时刻 T5 也为第二操作系统 OS_B 的第三线程 THB3 留下一些执行时间(905)。本例中，在时刻 T6 轮到运行的下一个线程(906)是第二操作系统 OS_B 的第三线程 THB1。

图 10 中也表示了从时刻 T7 开始的以上情况。线程 THA1、THA2、THA3、调度程序 SCH_A、中断服务例程 ISR、中断向量和第一操作系统的其它服务在图 10 中以方框表示。图 10 中排列了各线程的优先级，位置最高的线程有最高的优先级。第一操作系统 OS_A 的空闲线程 THA_IDLE 的优先级最低。在时刻 T7，第一操作

系统 OS_A 的调度程序 SCH_A 暂时改变第一操作系统 OS_A 的空闲线程 THA_IDLE 的优先级。这种情形在图 10 中时刻 T7 的右边表示。在时刻 T11，第一操作系统 OS_A 的调度程序 SCH_A 将第一操作系统 OS_A 的空闲线程 THA_IDLE 的优先级改变回最低优先级。第一操作系统 OS_A 的空闲线程 THA_IDLE 的优先级的改变例如可由定时器、由某个事件等处理。

用户例如按下第二接口 UI2 的某键。于是在时刻 8 又开始轮到第一操作系统 OS_A 的第三线程 THA3 运行(907)。调度程序 SCH_A 已经提高了第一操作系统 OS_A 的空闲线程 THA_IDLE 的优先级。第一操作系统 OS_A 的第三线程 THA3 正在运行时，有经接口部件 16 进行传输的需要。这就引起一个中断—因为空闲线程 THA_IDLE 的优先级在这个时刻 T9 高于第一操作系统 OS_A 的第三线程 THA3 的优先级。于是第一操作系统 OS_A 的第三线程 THA3 的执行被中断，开始轮到第二操作系统 OS_B 的第一线程 THB1 运行(908)。在第二操作系统 OS_B 的第一线程 THB1 的执行之后，在时刻 T10，继续第一操作系统 OS_A 的第三线程 THA3 的执行。

本例中，第一操作系统 OS_A 的调度程序 SCH_A 在时刻 T11 将第一操作系统 OS_A 的空闲线程 THA_IDLE 的优先级降回到最低水平。

第一操作系统 OS_A 的第三线程 THA3 的执行在时刻 T12 再次被中断，因为有通知说有更高优先级的线程要执行(910)。这个线程例如是第一操作系统 OS_A 的第二线程 THA2。

在时刻 T13，第一操作系统 OS_A 的第二线程 THA2 被中断，因为有通知说有更高优先级的线程—第一操作系统 OS_A 的第一线程 THA1—要执行(911)。

在时刻 T14，第一操作系统 OS_A 的第一线程 THA1 执行后，第一操作系统 OS_A 的第二线程 THA2 的执行继续(912)。

在时刻 T15，第一操作系统 OS_A 的第二线程 THA2 的执行完成。同时(时刻 T15)，第二操作系统 OS_B 的第一线程 THB1 也变得运行准备就绪。然而，第一操作系统 OS_A 的第三线程 THA3 的执行却在时刻 15 继续(913)。这与较早时在时刻 T9 的行为相反。

第一操作系统 OS_A 的第三线程 THA3 的执行在时刻 15 第一操

作系统 OS_A 的第二线程 THA2 已经执行后继续。

第二操作系统 OS_B 的第一线程 THB1 在时刻 T16 开始轮到运行(914)，服务于经接口部件 16 的传输。

表 1 进一步总结了与中断相联系的从一种方式向另一种方式的转换。第一列表示改变方式时处理器 2 所处的方式，第一行表示处理器 2 被转换到的方式。表的其它方框表示导致各方式的变化的情况。各缩写就是本说明书前文中所用的不同方式和中断的缩写。

	USER	UND	SVC	IRQ	FIQ
USER	----	----	OS_B 线程请求 SWI	IRQ 中断 OS_B 线程	FIQ 中断 OS_B 线程
UND	从最后一个 OS_A 线程或 DSR 返回	----	DSR 或空闲线程请求 SWI	IRQ 中断 OS_A 线程、DSR 或空闲线程	FIQ 中断 OS_A 线程、DSR 或空闲线程
SVC	从快速 SWI 返回	在慢 SWI 期间	----	IRQ 中断快速 SWI	FIQ 中断快速 SWI
IRQ	如果不存在活动的 OS_A 线程或 DSR，从 IRQ 返回	如果存在活动的 OS_A 线程或 DSR，从 IRQ 返回	----	----	FIQ 中断 IRQ
FIQ	如果没有活动的 OS_A 线程或 DSR，从 FIQ 返回	如果活动的 OS_A 线程或 DSR，从 FIQ 返回	----	如果 IRQ 被中断，从 FIQ 返回	----

表 1

下面的例子描述不同操作系统 OS_A、OS_B 中执行的线程之间消息的传输。图 7a 和 7b 以简化示意图表示这种消息传输机制。消息驱动器 MD(message driver)是操作系统 OS_A、OS_B 公用的，也是消息队列 MQ1、MQ2 公用的。第二操作系统 OS_B 的运行线程

THB1 调用消息驱动器 MD，将所传送消息(例如一条正文消息)和目标线程 THA1 的符号作为参数传输(图 7a 箭头 701)。消息驱动器 MD 将该消息转移到第一消息队列 MQ1(箭头 702)并最好生成一个软件中断(SWI)，这个中断激活一个基本如本说明书前文据图 6 所述的中断处理—只不过该中断是由软件中断 SWI 而不是普通中断 IRQ 引起的。离开中断处理程序导致激活第一操作系统的调度程序 SCH_A，目标线程在由优先级确定的时间被执行(箭头 703)。这个目标线程读取第一消息队列 MQ1 中的消息(箭头 704)并例如将该消息传输到一个无线电通道。

在第一操作系统的应用程序中，能执行一个被延迟服务例程 DSR，由此可能例如以下列方式(图 7b)将消息传输到第二操作系统 OS_B 的线程。在第二操作系统 OS_B 中，运行线程 THB2 已经生成了一个消息请求并一直等待着该消息(箭头 705)。该消息请求引起一个软件中断 SWI，中断处理程序检查中断的原因后断定第二操作系统 OS_B 的线程 THB2 正在带来来自第一操作系统 OS_A 的线程 THA2 的消息。中断处理后，调用第一操作系统 OS_A 的调度程序 SCH_A(706)。在轮到所希望的线程 THA2 运行后，该线程生成一个消息并将其添加到第二消息队列 MQ2(箭头 707)。此外，线程 THA2 还生成一个被延迟服务例程 DSR(箭头 708)。在运行了第一操作系统 OS_A 的各线程后，这个被延迟服务例程 DSR 紧接着就被处理，其中由线程 THA2 添加的该被延迟服务例程 DSR 将第二操作系统 OS_B 的线程的状态改变为等待执行(就绪)(箭头 709)。第二操作系统 OS_B 的调度程序 SCH_B 按照其调度原则控制各线程的执行。当发过传输请求的线程 THB2 运行时(710)，它通过消息驱动器 MD 的连接接口从第二消息队列 MQ2 读取消息。

在实际实施例中，消息队列 MQ1、MQ2 最好能按优先级来配置。这意味着，高优先级的接收线程的消息比较低优先级的消息更快经过队列。另一种可选方案是，采用比本例中的第一 MQ1 和第二消息队列 MQ2 更多的消息队列。

总而言之，可以注意到，本发明能实现在一个处理器下运行一个或多个操作系统 OS_A、OS_B。一个目标是使公用部件最少，也能在这种通信设备 1 中运行现行技术水平的应用程序。操作系

统 OS_A、OS_B 的公用部件主要是中断处理程序，中断处理程序中形成了与操作系统 OS_A、OS_B 的联系。在上面的例子中，消息驱动器 MD 也是公用的。

在实际的嵌入式相同 1 中，有可能为不同的操作系统 OS_A、OS_B 配置许多公用的中断处理程序(中断服务例程)。每次可以有对应一种中断的其中一个中断处理程序是活动的，即，例如对应快速中断 FIQ 的一个，对应普通中断 IRQ 的一个和对应软件中断 SWI 的一个。这样就有可能将为该中断配置的另一个中断处理程序变为活动的中断处理程序。

应当指出，两个操作系统 OS_A、OS_B 在处理器 2 的系统中运行也能以这样的方式实现，例如由第二操作系统 OS_B 的线程 THB1、THB2、THB_IDLE 来形成第一操作系统的线程，诸如空闲线程 THA_IDLE 或超级线程，其中与所有与第二操作系统 OS_B 有关的运行线程 THB1、THB2、THB_IDLE 都在这个线程中执行。以对应的方式，有可能由第一操作系统 OS_A 的线程 THA1、THA2、THA_IDLE 来形成第二操作系统的线程，诸如空闲线程 THB_IDLE 或超级线程 511，其中与所有与第一操作系统 OS_A 有关的运行线程 THA1、THA2、THA_IDLE 都在这个线程中执行。另一个可选方案是，第一操作系统 OS_A 的每个线程 THA1、THA2、THA_IDLE 以独立于第二操作系统 OS_B 的超级线程 511 的形式形成，第二操作系统 OS_B 把这些线程作为单独的超级线程(未予示出)来处理。如本说明书前文所述，这些超级线程的响应时间，通常比普通线程的响应时间短得多。响应时间的要求最高的任务仍然能得到运行，方法是为它们生成例如与定时器的中断处理程序相联系的中断处理，其中在转而执行这些任务时不必使用调度功能。

就第一操作系统 OS_A 的每个线程 THA1、THA2、THA_IDLE 以第二操作系统 OS_B 中独立的超级线程 511 的形式实现的情况来说，如果希望在第一操作系统 OS_A 的不同线程 THA1、THA2、THA_IDLE 之间传输数据，就要求按照第二操作系统 OS_B 的信号传输。相反，当第一操作系统 OS_A 的线程 THA1、THA2、THA_IDLE 是以一个超级线程 511 的形式实现的时，数据传输就能通过使用第一操作系统 OS_A 的消息传输方法而以更简单的方式实现。

本发明并不仅限于上述各实施例，可以在后附权利要求的范围内修改。

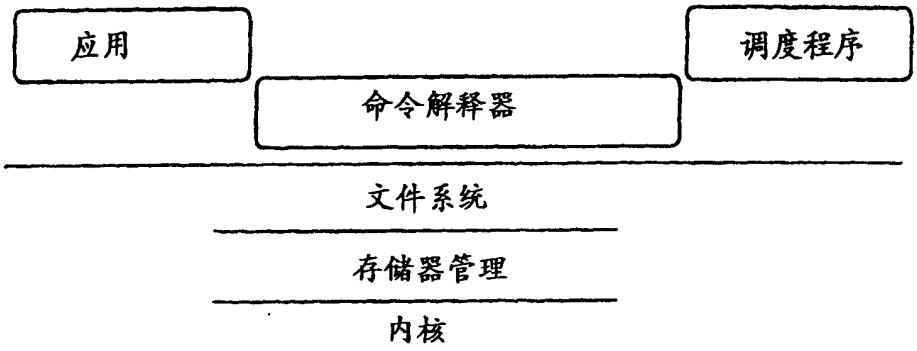


图 1

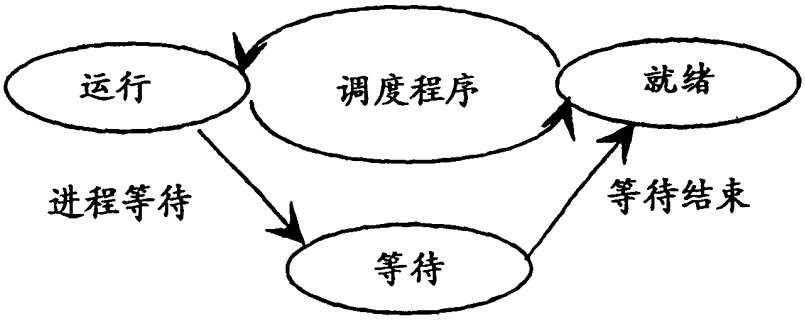


图 2

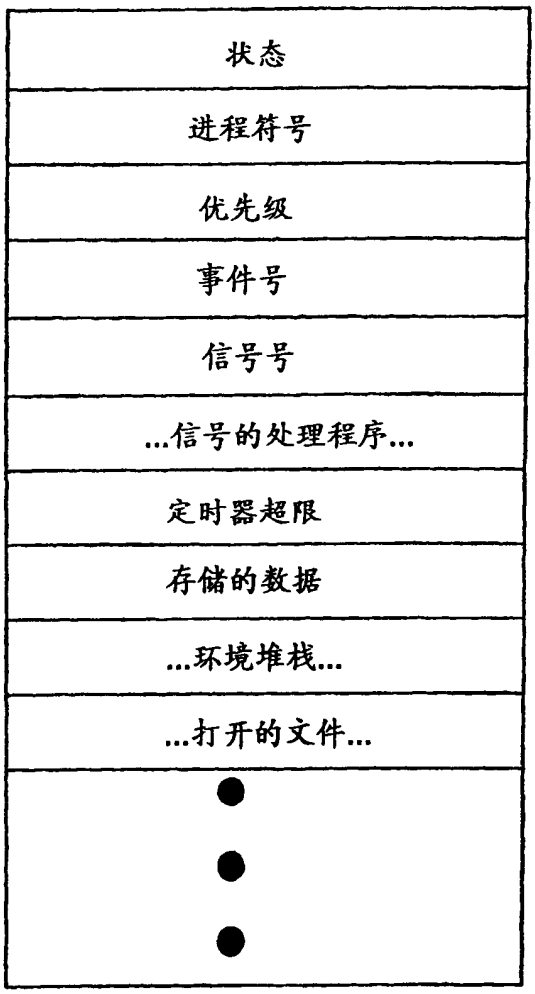


图 3

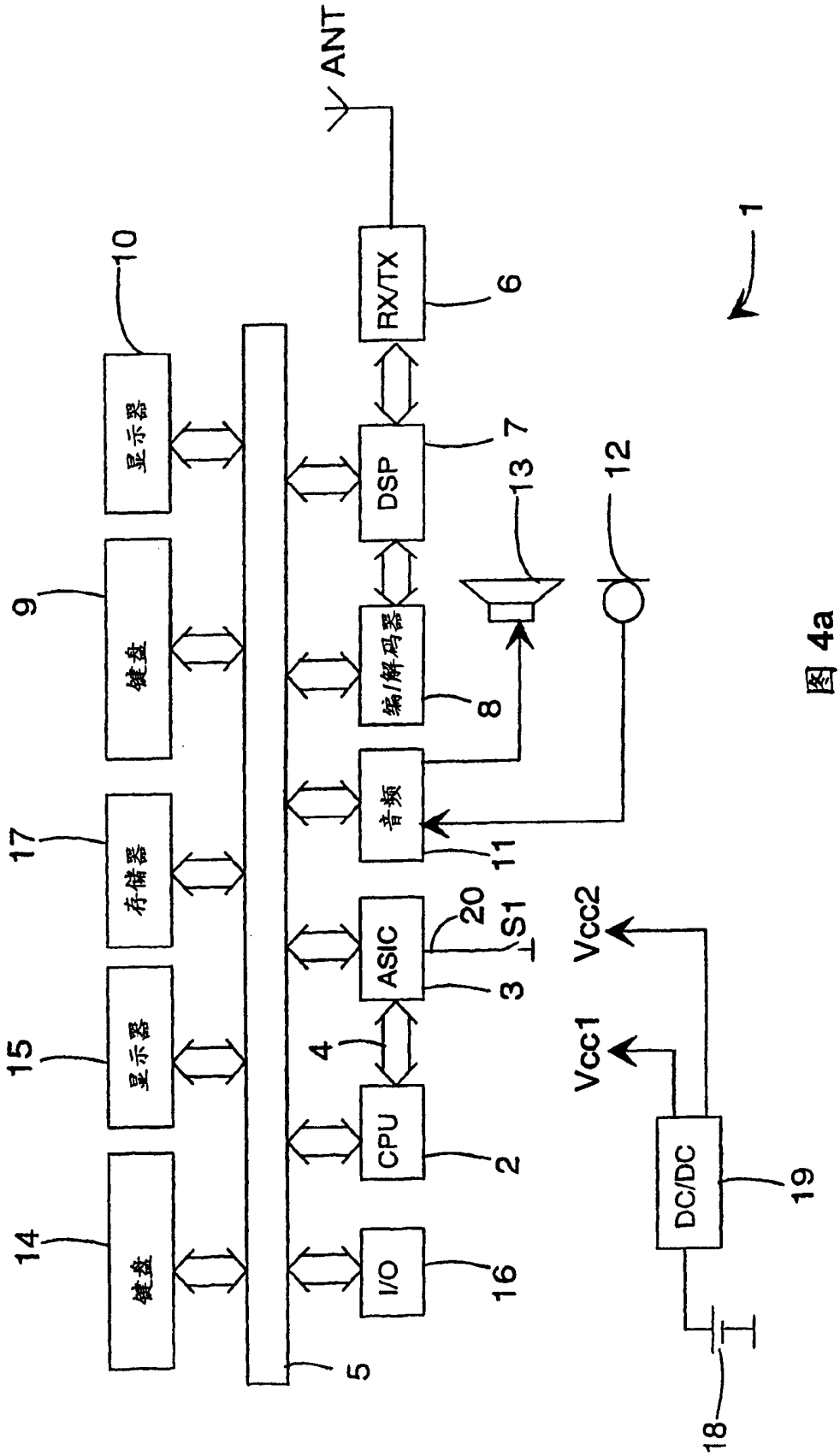

ANT
RX/TX
6
DSP
7
编/解码器
8
音频
11
ASIC
3
CPU
2
I/O
16
5
显示器
10
键盘
9
存储器
17
显示器
15
键盘
14
4
S1
20
13
12
1
Vcc2
Vcc1
DC/DC
19
18

图 4a

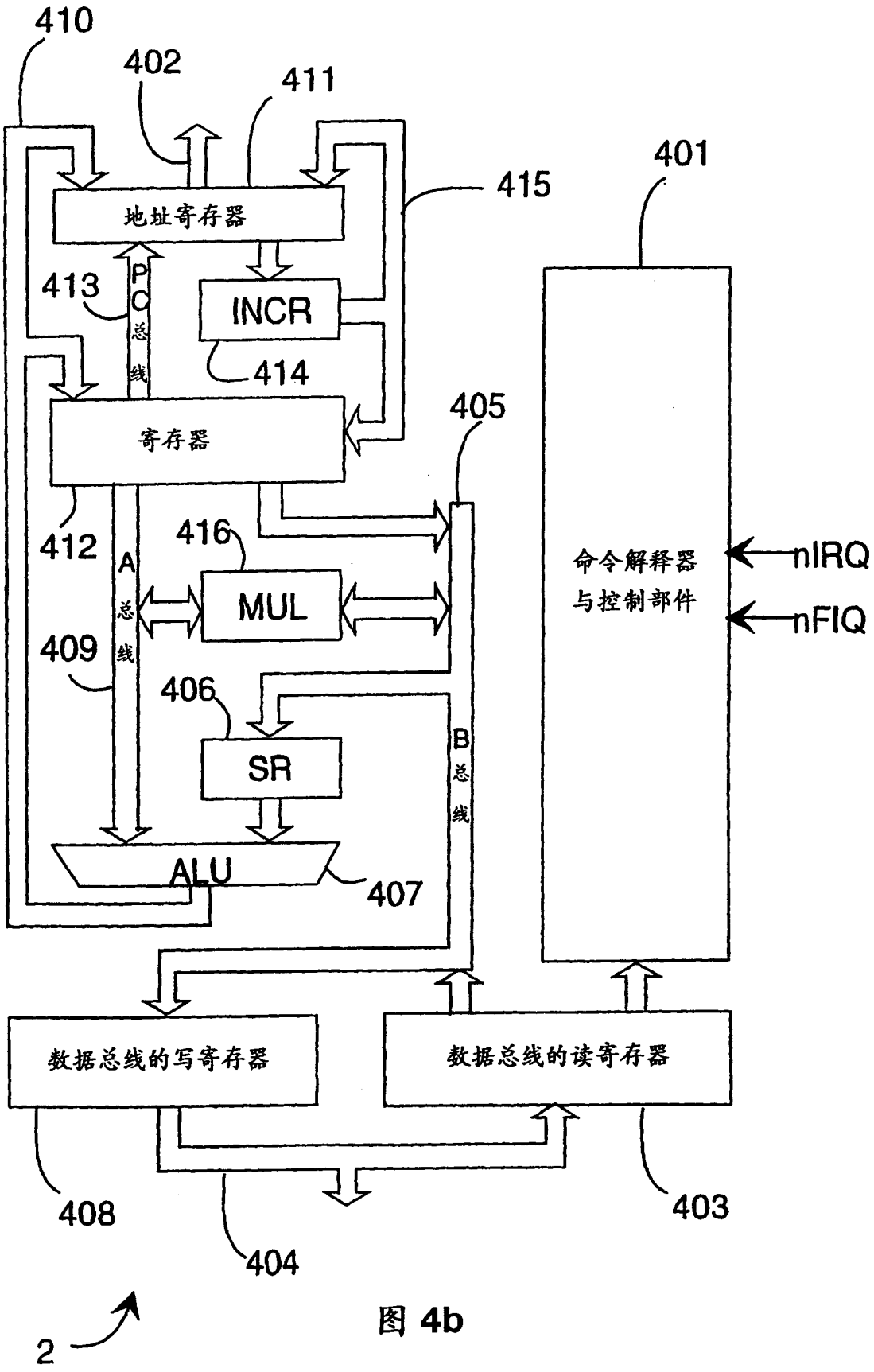

410
402
411
401
415
地址寄存器
413
PC总线
INCR
414
405
寄存器
412
416
A总线
MUL
命令解释器
与控制部件
nIRQ
nFIQ
409
406
SR
B总线
ALU
407
数据总线的写寄存器
数据总线的读寄存器
408
404
403
2

图 4b

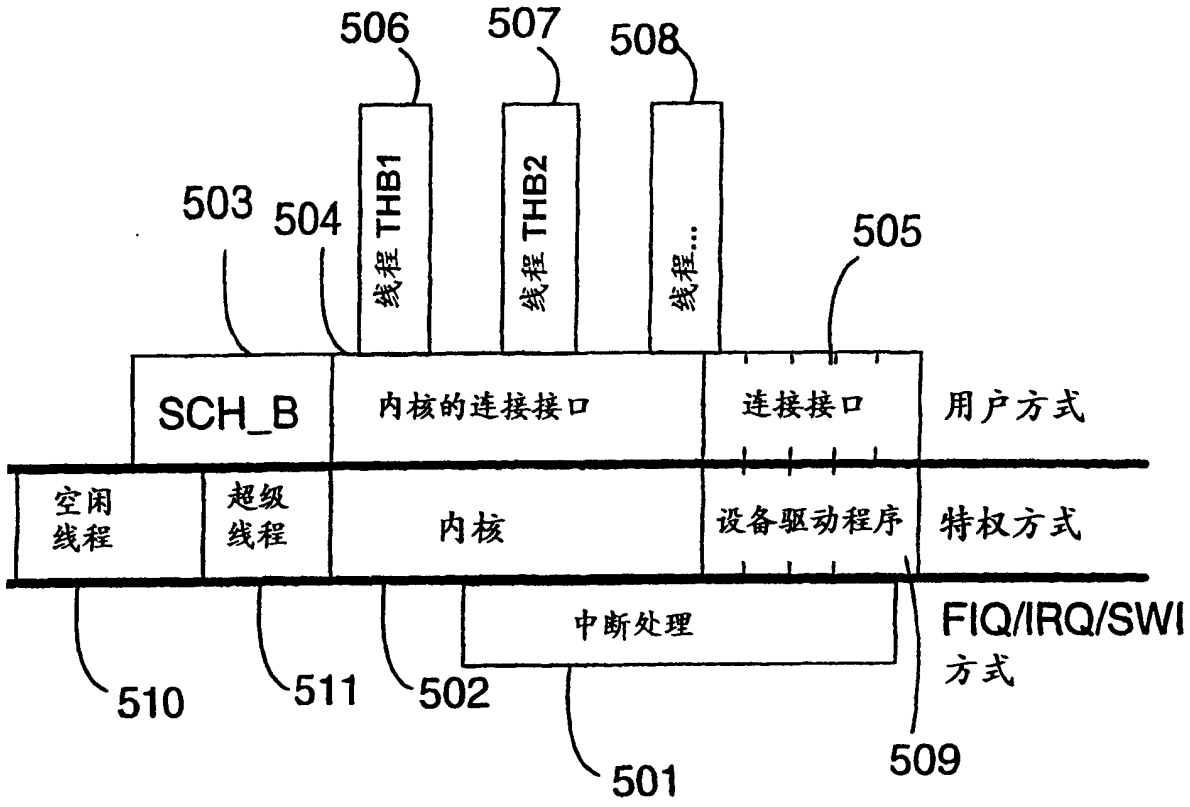

506
507
508
503
504
505
线程 THB1
线程 THB2
线程...
SCH_B
内核的连接接口
连接接口
用户方式
空闲
线程
超级
线程
内核
设备驱动程序
特权方式
中断处理
FIQ/IRQ/SWI
方式
510
511
502
501
509

图 5

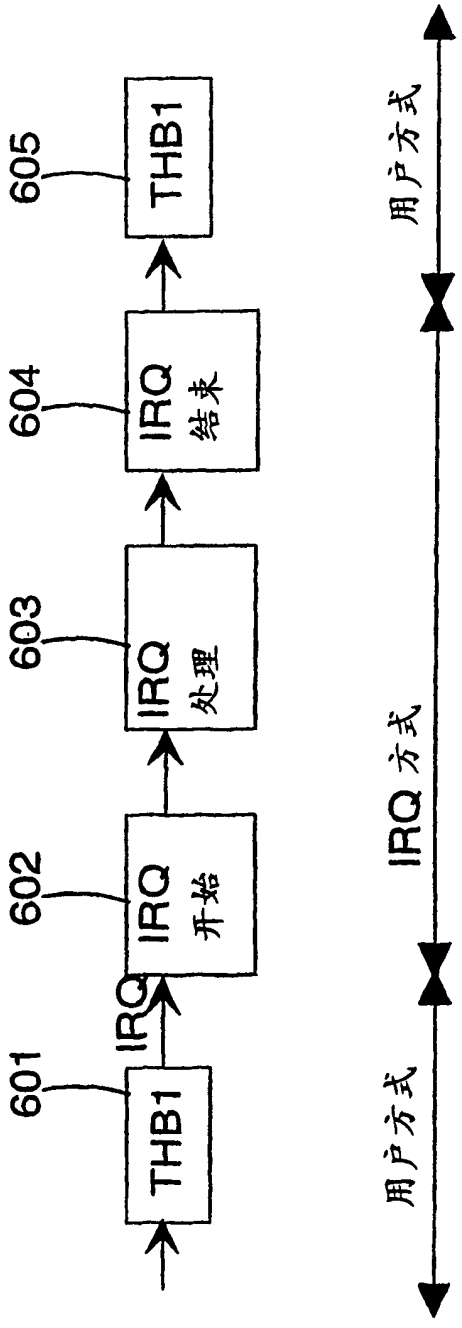

601
THB1
IRQ
602
IRQ
开始
603
IRQ
处理
604
IRQ
结束
605
THB1
用户方式
IRQ 方式
用户方式

图 6a

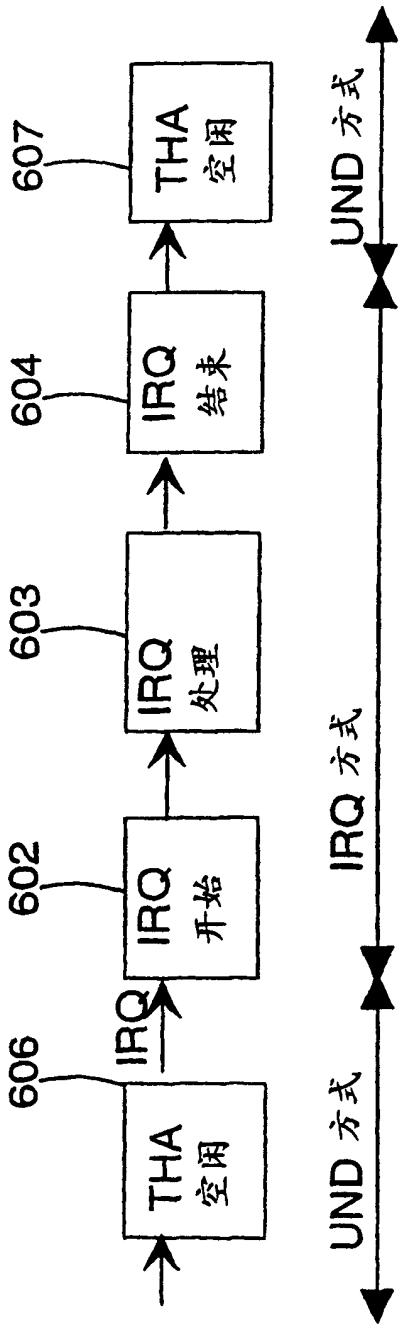

606
THA
空闲
IRQ
602
IRQ
开始
603
IRQ
处理
604
IRQ
结束
607
THA
空闲
UND 方式
IRQ 方式
UND 方式

图 6b

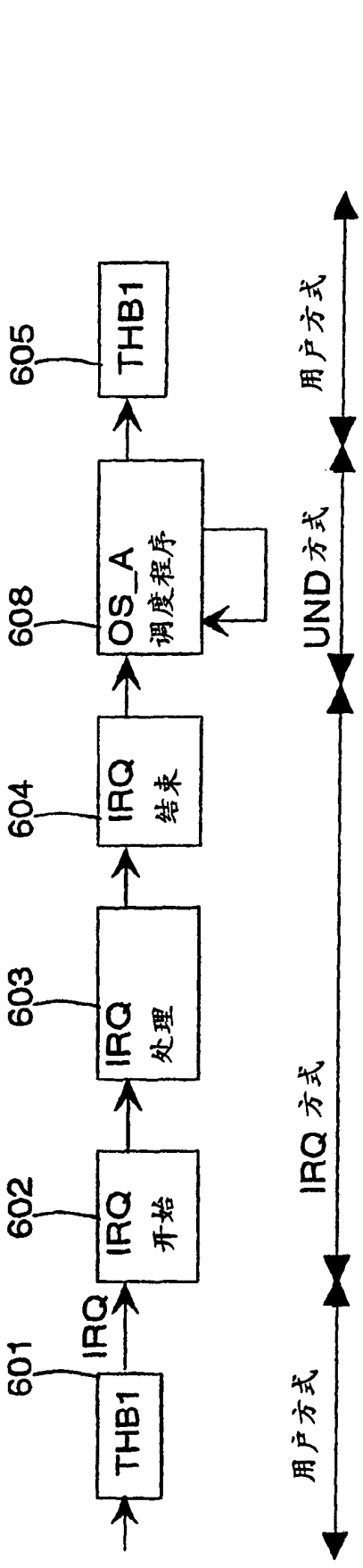

601
THB1
IRQ
602
IRQ
开始
603
IRQ
处理
604
IRQ
结束
608
OS_A
调度程序
605
THB1
用户方式
IRQ 方式
UND 方式
用户方式

图 6c

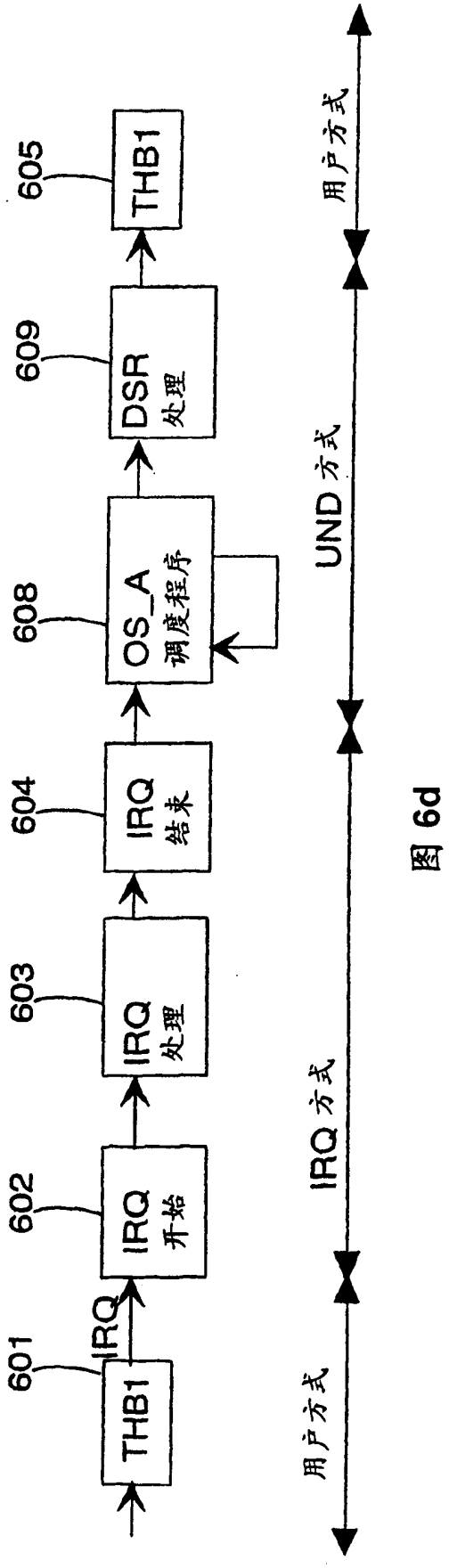

601
THB1
IRQ
602
IRQ
开始
603
IRQ
处理
604
IRQ
结束
608
OS_A
调度程序
609
DSR
处理
605
THB1
用户方式
IRQ 方式
UND 方式
用户方式

图 6d

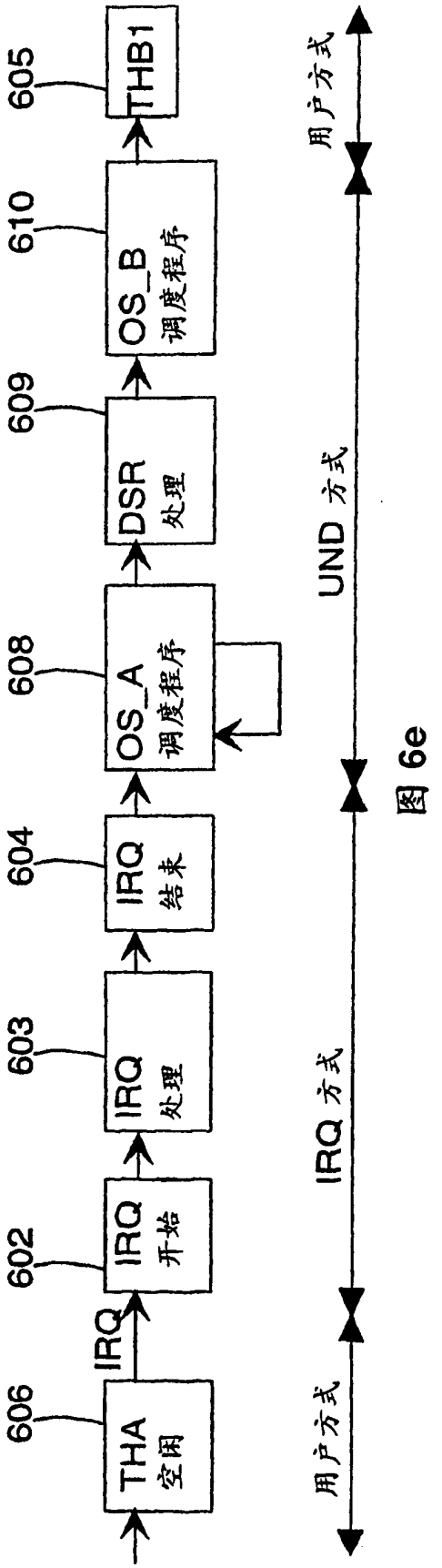

606
THA
空闲
IRQ
602
IRQ
开始
603
IRQ
处理
604
IRQ
结束
608
OS_A
调度程序
609
DSR
处理
610
OS_B
调度程序
605
THB1
用户方式
IRQ 方式
UND 方式
用户方式

图 6e

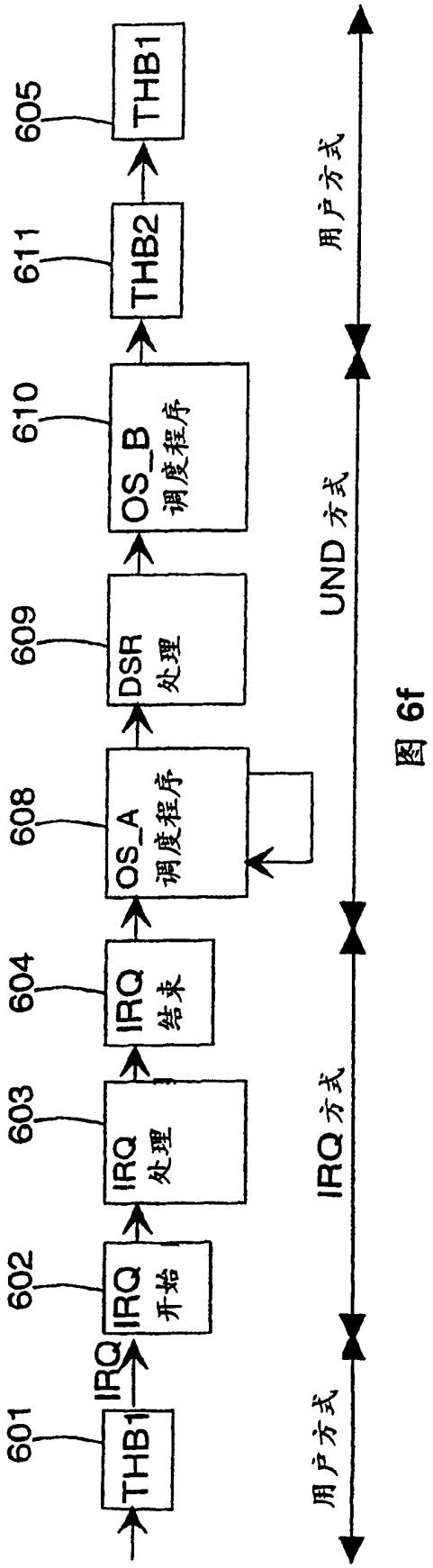

601
THB1
IRQ
602
IRQ
开始
603
IRQ
处理
604
IRQ
结束
608
OS_A
调度程序
609
DSR
处理
610
OS_B
调度程序
611
THB2
605
THB1
用户方式
IRQ 方式
UND 方式
用户方式

图 6f

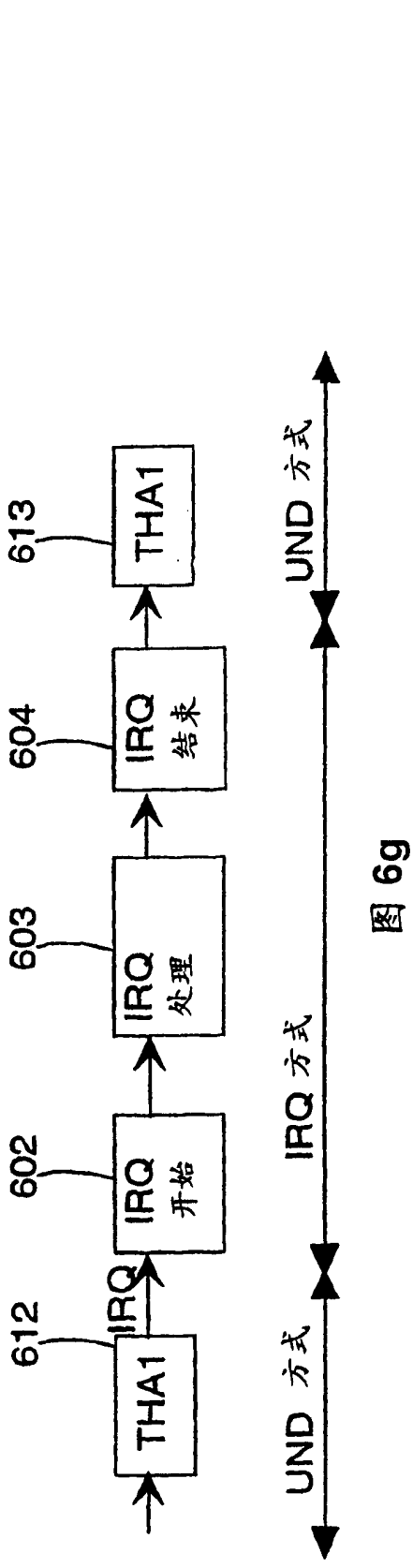

612
THA1
IRQ
602
IRQ
开始
603
IRQ
处理
604
IRQ
结束
613
THA1
UND 方式
IRQ 方式
UND 方式

图 6g

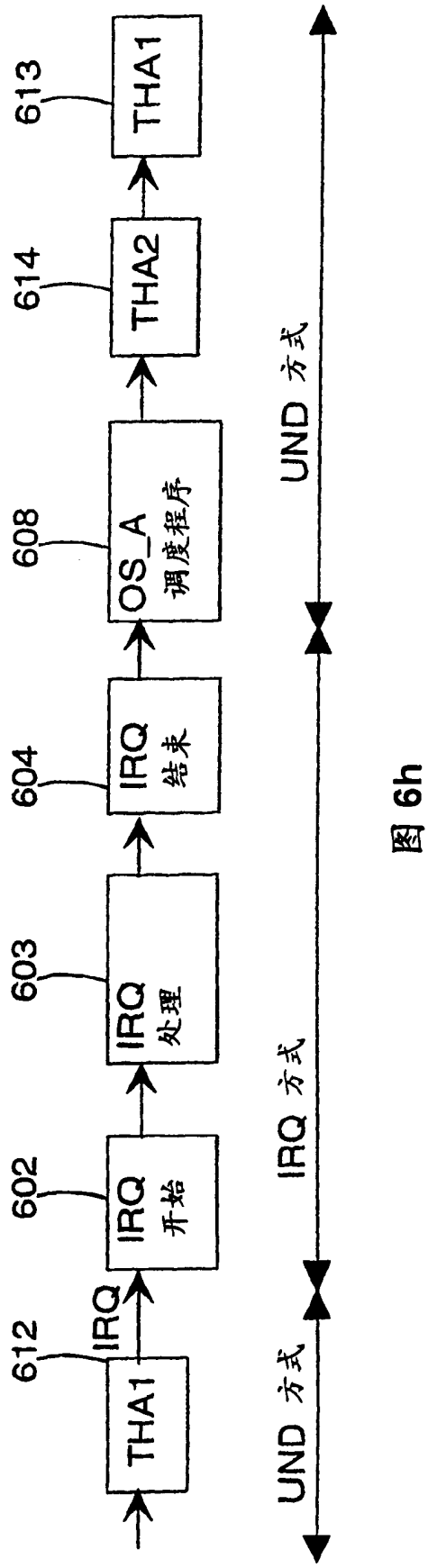

612
THA1
IRQ
602
IRQ
开始
603
IRQ
处理
604
IRQ
结束
608
OS_A
调度程序
614
THA2
613
THA1
UND 方式
IRQ 方式
UND 方式

图 6h

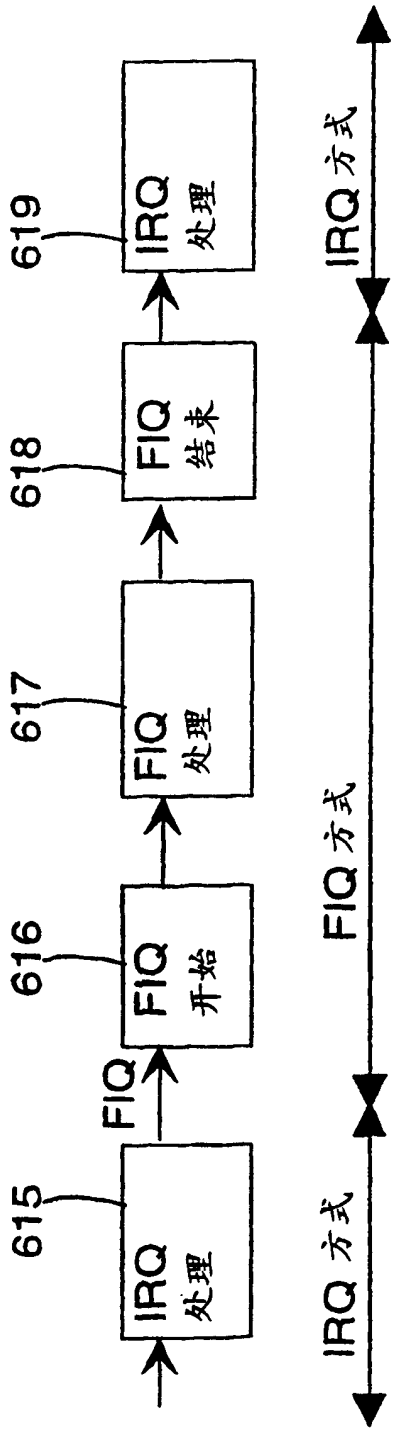

615
IRQ
处理
FIQ
616
FIQ
开始
617
FIQ
处理
618
FIQ
结束
619
IRQ
处理
IRQ 方式
FIQ 方式
IRQ 方式

图 6i

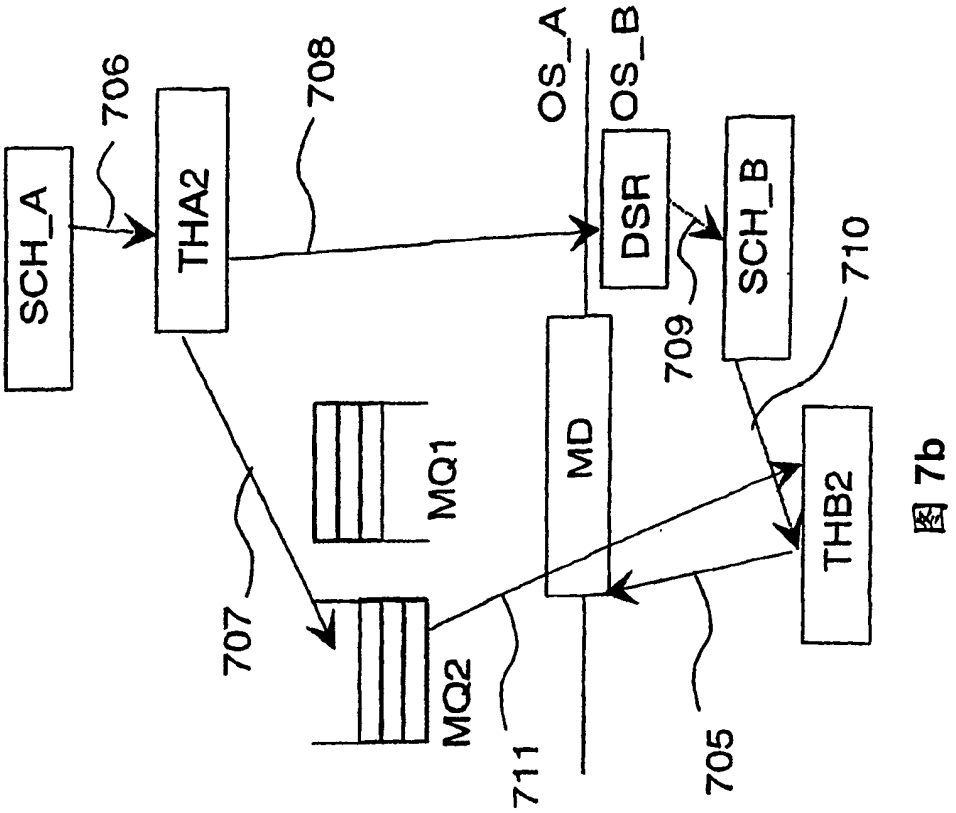

SCH_A
706
THA2
708
OS_A
OS_B
DSR
SCH_B
709
710
MD
MQ1
THB2
707
MQ2
711
705

图 7b

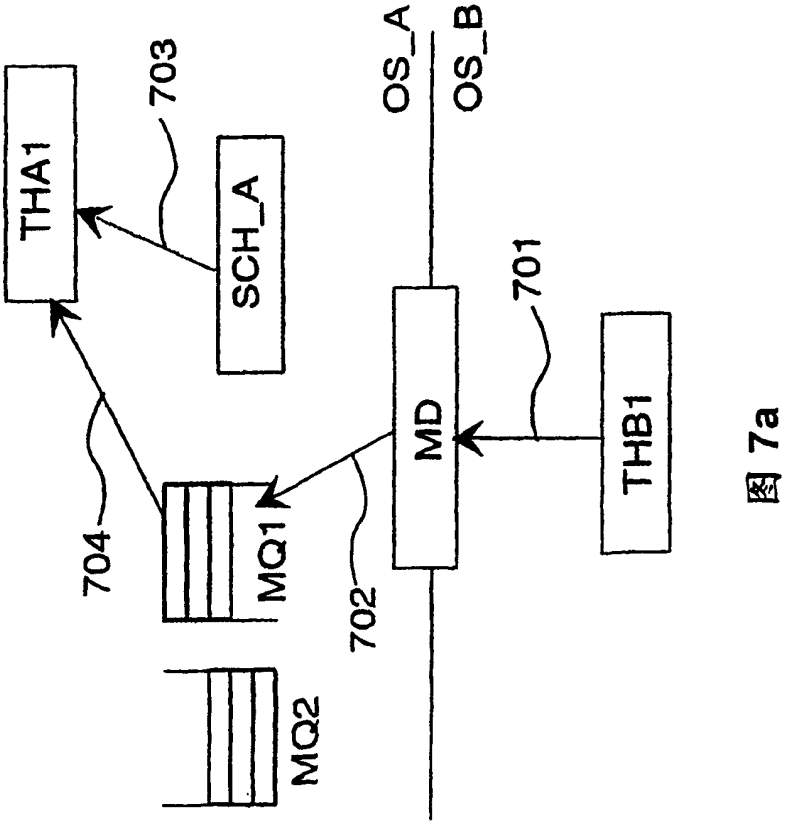

THA1
703
SCH_A
OS_A
OS_B
701
MD
THB1
704
MQ1
702
MQ2

图 7a

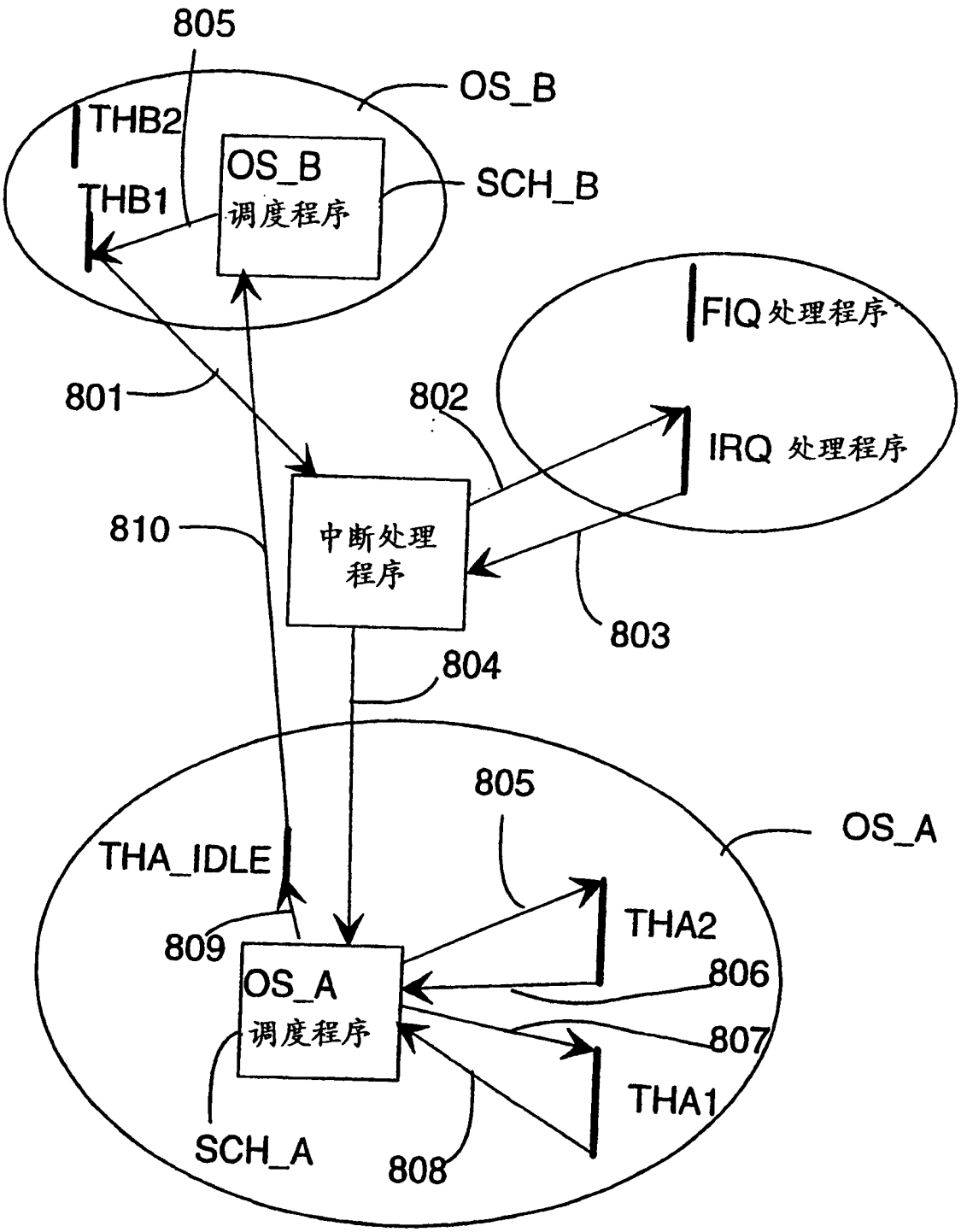

805
OS_B
THB2
OS_B
调度程序
SCH_B
THB1
FIQ 处理程序
801
802
IRQ 处理程序
中断处理
程序
810
803
804
805
OS_A
THA_IDLE
THA2
809
OS_A
调度程序
806
807
THA1
SCH_A
808

图 8

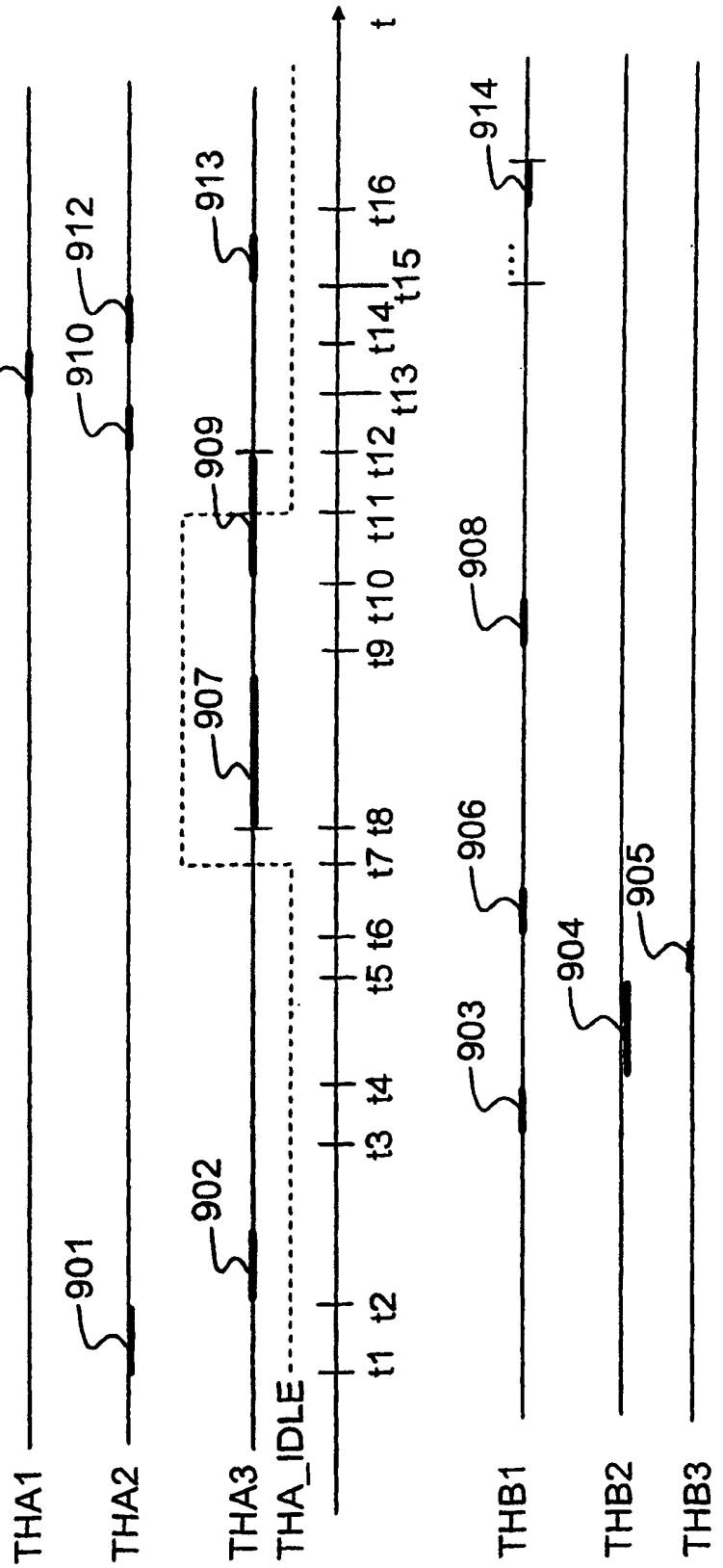

THA1
THA2
THA3
THA_IDLE
THB1
THB2
THB3
911
901
910
912
902
913
909
907
914
908
906
903
904
905
t1 t2 t3 t4 t5 t6 t7 t8 t9 t10 t11 t12 t13 t14 t15 t16 t

图 9

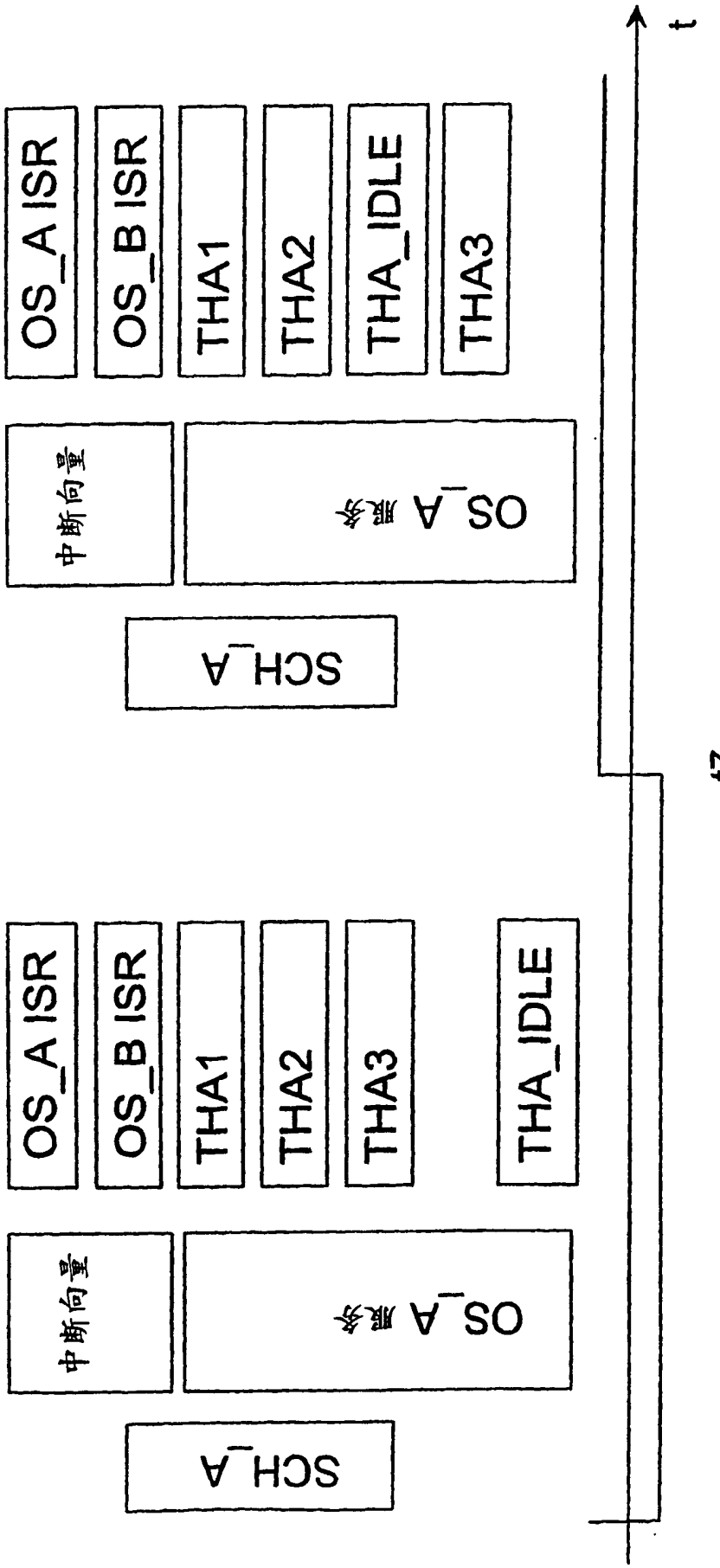

t
t7
SCH_A
中断向量
OS_A 服务
OS_A ISR
OS_B ISR
THA1
THA2
THA3
THA_IDLE
SCH_A
中断向量
OS_A 服务
OS_A ISR
OS_B ISR
THA1
THA2
THA_IDLE

图 10