(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 104460646 B
(45)授权公告日 2018.11.20

(21)申请号 201410468167.0

(22)申请日 2014.09.15

(65)同一申请的已公布的文献号
申请公布号 CN 104460646 A

(43)申请公布日 2015.03.25

(30)优先权数据
13184920.0 2013.09.18 EP

(73)专利权人 帝斯贝思数字信号处理和控制工程有限公司
地址 德国帕德博恩

(72)发明人 R·雷恩费尔纳 T·克斯坦

(74)专利代理机构 中国国际贸易促进委员会专利商标事务所 11038
代理人 邓斐

(51)Int.Cl.
G05B 23/02(2006.01)

(56)对比文件
US 2008319730 A1,2008.01.31,说明书第0072、0080-0081、0087、0100、0110、0159、0166、0176、0195、0239、0294、0378段.
CN 103098032 A,2013.05.08,说明书第0080-0083段.
US 2008229165 A1,2008.09.18,说明书第0052、0063、0068段.
US 20040221273 A1,2004.05.19,说明书第0039-0041、0026、0045、0052段,摘要.
JP 2001101031 A,2001.04.13,全文.
JP 2013084163 A,2013.05.09,全文.
JP 2000259445 A,2000.09.22,全文.
CN 1981250 A,2004.06.13,全文.
JP 2008262318 A,2008.10.30,全文.

审查员 李丽兰

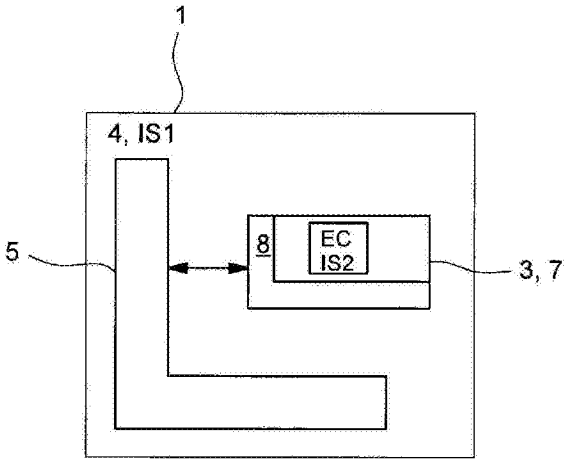
权利要求书2页 说明书7页 附图5页

(54)发明名称

用于对虚拟控制器进行实时测试的测试装置

(57)摘要

本发明涉及一种用于对带控制器代码(EC)的虚拟控制器(3)的至少一个部分进行实时测试的测试装置(1),其具有带第一指令集(IS1)的至少一个第一类型计算核心(4)和用于对虚拟控制器的环境进行实时模拟的至少一个模拟环境(5),利用至少一个第一类型计算核心对模拟环境和控制器代码进行计算。控制器代码能够在带第二指令集(IS2)的第二类型计算核心(6)上运行,第二类型计算核心的第二指令集不同于第一类型计算核心的第一指令集,对带控制器代码的虚拟控制器的测试如此实现:第一类型计算核心执行用于模拟第二类型计算核心的仿真程序并且仿真程序执行控制器代码,并且仿真程序具有用于与模拟环境交换数据和/或事件的模拟环境接口。



CN 104460646 B

1.用于对带有控制器代码(EC)的虚拟控制器(3)的至少一个部分进行实时测试的测试装置(1),该测试装置(1)具有带第一指令集(IS1)的至少一个第一类型计算核心(4)和用于对虚拟控制器(3)的环境进行实时模拟的至少一个模拟环境(5),其中,利用至少一个第一类型计算核心(4)对所述模拟环境(5)和所述控制器代码(EC)进行计算,

其特征在于:

控制器代码(EC)能够在带第二指令集(IS2)的第二类型计算核心(6)上运行,所述第二类型计算核心(6)的第二指令集(IS2)不同于所述第一类型计算核心(4)的第一指令集(IS1);

所述第一类型计算核心(4)执行用于模拟第二类型计算核心(6)的仿真程序(7)并且仿真程序(7)执行控制器代码(EC);

仿真程序(7)具有用于与模拟环境(5)交换数据和/或事件的模拟环境接口(8),

在仿真程序(7)上在时间方面最佳化地执行控制器代码(EC),由此在每个探测间隔内发生的计算在所述探测间隔内不中断地结束并且不会由于由仿真所决定地出现的限制而违背实时要求。

2.如权利要求1所述的测试装置(1),其特征在于:模拟环境(5)包括至少一个模拟程序(9)和一个过程模型(10),所述模拟程序(9)能够经由模拟环境接口(8)通过仿真程序(7)调出,和/或所述仿真程序(7)能够经由模拟环境接口(8)通过模拟程序(9)调出。

3.如权利要求2所述的测试装置(1),其特征在于:数据能够经由模拟环境接口(8)而在仿真程序(7)与过程模型(10)之间交换。

4.如权利要求2所述的测试装置(1),其特征在于:在仿真程序(7)借助模拟环境接口(8)通过模拟程序(9)调出时,仿真程序(7)对控制器代码(EC)的执行环境进行检测,该仿真程序(7)执行控制器代码(EC)的被调出的部分,并且该仿真程序(7)实时更新控制器代码(EC)的执行环境。

5.如权利要求3所述的测试装置(1),其特征在于:在仿真程序(7)借助模拟环境接口(8)通过模拟程序(9)调出时,仿真程序(7)对控制器代码(EC)的执行环境进行检测,该仿真程序(7)执行控制器代码(EC)的被调出的部分,并且该仿真程序(7)实时更新控制器代码(EC)的执行环境。

6.如权利要求4所述的测试装置(1),其特征在于:控制器代码(EC)是完整的,因此执行环境包括虚拟控制器(3)的完整的物理环境。

7.如权利要求6所述的测试装置(1),其特征在于:所述控制器代码(EC)包括操作系统、驱动程序和控制器的所有软件组件。

8.如权利要求4所述的测试装置(1),其特征在于:控制器代码(EC)是不完整的,因此执行环境仅包括虚拟控制器(3)的软件组件。

9.如权利要求8所述的测试装置(1),其特征在于:所述控制器代码(EC)不包括操作系统和驱动程序。

10.如权利要求5所述的测试装置(1),其特征在于:控制器代码(EC)是完整的,因此执行环境包括虚拟控制器(3)的完整的物理环境。

11.如权利要求10所述的测试装置(1),其特征在于:所述控制器代码(EC)包括操作系统、驱动程序和控制器的所有软件组件。

12.如权利要求5所述的测试装置(1),其特征在于:控制器代码(EC)是不完整的,因此执行环境仅包括虚拟控制器(3)的软件组件。

13.如权利要求12所述的测试装置(1),其特征在于:所述控制器代码(EC)不包括操作系统和驱动程序。

14.如权利要求3、5和10至13之任一项所述的测试装置(1),其特征在于:经由模拟环境接口(8)数据驱动地在仿真程序(7)与过程模型(10)之间由仿真程序(7)对控制器代码(EC)进行计算。

15.如权利要求2、4和6至9之任一项所述的测试装置(1),其特征在于:过程模型(10)的计算和通过仿真程序(7)对控制器代码(EC)的执行被模拟程序(9)中枢地控制。

16.如权利要求15所述的测试装置(1),其特征在于:所述过程模型(10)的计算(903)与控制器代码(EC)的执行(702)在时间上的协调。

17.如权利要求1至13之任一项所述的测试装置(1),其特征在于:仿真程序(7)对每个在运行时间被调出的第二指令集(IS2)内的控制器代码(EC)的指令进行解释并且在功能上与第一指令集(IS1)的指令一致地并且在第一类型计算核心(4)上执行。

18.如权利要求1至13之任一项所述的测试装置(1),其特征在于:仿真程序(7)在即将执行之前将控制器代码(EC)完全地或部分地转化到第一指令集(IS1)中。

19.如权利要求18所述的测试装置(1),其特征在于:在转化之前,为转化到第一指令集(IS1)中的控制器代码(EC′)分派与在虚拟控制器(3)上实际上可供使用的存储器一样多的存储器。

用于对虚拟控制器进行实时测试的测试装置

技术领域

[0001] 本发明涉及一种用于对具有控制器代码(Steuergeraetecode)的虚拟控制器的至少一个部分进行实时测试的测试装置，该测试装置具有带有第一指令集的至少一个第一类型计算核心和用于实时模拟虚拟控制器环境的至少一个模拟环境，其中，利用至少一个第一类型计算核心对所述模拟环境和所述控制器代码进行计算。

背景技术

[0002] 上述测试装置在控制器开发领域的现有技术中是众所周知的，它们在研究和开发中得到应用，其中经常的应用领域在汽车领域和在航空航天技术中。

[0003] 用于对虚拟控制器进行实时测试的测试装置在设计上以用于对真实的控制器进行测试的测试装置为依据。这样的测试装置用于对真实的控制器或者真实的控制器的组合实施所谓的硬件在环测试(HIL-Test)，这些控制器的互相配合需要被检查。真实的控制器当前为可实时的小型计算机，它们用于对技术过程施加影响。因此真实的控制器具有计算核心，在该计算核心上执行可实时的操作系统，在它的帮助下调整的和/或控制的探测系统如它们几十年来在系统学中众所周知的那样得以实施。真实的控制器经由I/O接口(Input/Output)而对需影响的过程产生影响或者经由所述I/O接口以测量技术的方式得到关于过程的信息。借助HIL模拟程序对真实的控制器进行测试涉及的是：真实的控制器在其实际使用环境中被使用之前的测试的最后步骤。真实的控制器例如可以是内燃机的马达控制器。

[0004] 为了对真实的控制器进行实时测试，真实的控制器经由其I/O接口而与测试装置相连。在配置有一个第一类型计算核心或多个第一类型计算核心的测试装置上运行模拟环境，该模拟环境同样可以实时模拟真实的控制器的环境。在马达控制器的上述实例的情况中，则例如涉及的是内燃机的实时模拟。由上述内容表明：测试装置的可实时性对于测试来说是至关重要的，因为借助该测试应该最终查清控制器在实时条件下是否是可行的，也就是说例如执行的算法是否能够以所要求的探测间隔来运算等。

[0005] 实际控制器的HIL模拟明显地具有优点：在模拟的环境中能够无危险地测试真实的控制器的性能。然而缺点同样是明显的：为了能够实施测试，真实的控制器无论如何必须作为这样的控制器存在。所以这一点是不利的，因为应该在批量应用中使用的控制器的开发伴随着巨大的费用和成本。如果在对真实的控制器进行测试的情况下才证实它满足不了特定的要求，那么并不少见的是：需要巨大的、远远超出简单的“修改”的努力，在最糟糕的情况中必须完全重新仔细考虑控制器的设计方案，这样实际上需要重新开发真实的控制器；结果是在时间上和经济上不能遵守计划目标。

[0006] 为了应对这个问题，也就是说为了尽可能早地将需检查的控制器的已知的组成部分列入功能测试中，已经想出所谓的虚拟控制器的设计方案(“Virtual Validation with dSPACE:Early PC-Based Validation of ECU Software”,Produktinformation dSPACE,2013)。此处尽管控制器不必物理性地存在，然而控制器代码必须存于与硬件无关的高级语言(例如C/C++,Assembler)中。在这种情况中，控制器代码被转化到第一类型计算核心的第

一指令集中，这样可以在具有第一类型计算核心的测试装置上执行所述控制器代码。一般来说，测试装置的与需测试的控制器的处理器执行程序/计算核心执行程序(Prozessor-/Rechenkernimplementierung)不同。在微机基础上的测试装置中，例如使用英特尔处理器作为第一类型计算核心，这些英特尔处理器具有相应的指令集(例如Intel IA32)，而控制器大多数情况下以微控制器为基础，也就是说具有与第一类型计算核心不同的第二类型计算核心，该第二类型计算核心具有与第一指令集不同的第二指令集(例如C166微控制器家族的C166指令集)。

[0007] 上面概略叙述的运行方式带来缺点：并不是本来能够为第二类型计算核心运行的原控制器代码被测试装置测试，而仅仅是将控制器代码转化到测试装置的第一类型计算核心的第一指令集上。另一个困难在于：控制器代码为了实现概略叙述的运行方式无论如何必须存在在一种高级语言中，这样它可以被转化到一种确定的指令集上。然而，这个前提在多数使用情况中没有被满足或不能得到满足，例如出于对执行细节保密的原因。

发明内容

[0008] 因此本发明的目的是，提供一种用于对具有控制器代码的虚拟控制器进行实时测试的测试装置，利用该测试装置避免了作为本发明的出发点的实时测试装置的前述缺点。

[0009] 前面导出和表明的目的在文首述及的测试装置中通过如下方式得以实现：控制器代码能够在具有第二指令集的第二类型计算核心上运行，其中，第二类型计算核心的第二指令集不同于第一类型计算核心的第一指令集。另外，第一类型计算核心执行一个用于模拟第二计算核心的仿真程序，该仿真程序执行控制器代码。另外，通过如下方式得以实现：仿真程序具有用于与模拟环境交换数据和/或事件的模拟环境接口。

[0010] 通过根据本发明的测试装置可以用测试装置以控制器代码的原形式来执行该控制器代码，不需要将控制器代码从高级语言转化到第一计算核心的第一指令集中，其是测试装置的组成部分。只能在具有第二指令集的第二类型计算核心上运行的控制器代码的可执行性通过使用用于模拟第二计算核心的仿真程序成为可能，其中，所述仿真程序本身在第一类型计算核心上执行。因此仿真程序是第二类型计算核心、即需测试的控制器的计算核心与测试装置的第一类型计算核心之间的连接部分。

[0011] 仿真程序的模拟环境接口具有重要意义，经由该模拟环境接口可以与模拟环境交换数据和/或事件，也就是说例如控制器的I/O接口的管脚状态和例如针对中断线路(auf Interrupt-Leitung)的触发信号等，所述管脚状态不仅从被模拟的控制器出发作为模拟环境的输入变量，而且从模拟环境出发作为被模拟的控制器的输入变量。模拟环境接口具有特别的意义，因为它首先能够实现在仿真程序上对控制器和控制器代码执行模拟的同步和可同步性，并且其次能够实现对模拟环境的执行并且由此能够实现对虚拟控制器的环境执行实时模拟。因此，模拟环境接口还是通过仿真程序在第一类型计算核心上实时执行控制器代码的前提条件。

[0012] 完全可能的是：测试装置具有多个第一类型计算核心，这样例如在一个第一类型计算核心上执行模拟环境以及在另一第一类型计算核心上执行仿真程序。

[0013] 在本发明的一种优选的设计方案中规定：模拟环境包括至少一个模拟程序和一个过程模型，所述模拟程序能够经由模拟环境接口通过仿真程序调出，和/或反过来，所述仿

真程序能够经由模拟环境接口通过模拟程序调出。由此能够实现的是:仿真程序或者控制器代码的模拟可以对事件作出反应,例如出自虚拟控制器的环境的实时模拟的事件。模拟环境与控制器代码之间的这种连接例如可以借助被模拟的控制器的物理环境或借助接口程序(Schnittstellen-Routine)得以实现,其中,物理环境描述的是被模拟的控制器的寄存器、诸如中断线路和物理接头。在使用接口程序的情况下,在一个事件开始时由仿真程序执行控制器代码内的预定的功能。这个功能的确定可以借助符号表和一个关于在控制器内具有哪些接口程序的说明自动得到实现。

[0014] 如果模拟环境包括模拟程序和过程模型,那么另外在一种优选的设计方案中规定:数据和/或事件能够经由模拟环境接口而在仿真程序与过程模型之间交换。换言之,根据本发明的测试装置优选构造如下:在仿真程序借助模拟环境接口被模拟程序调出时仿真程序对控制器代码的执行环境进行检测,该仿真程序执行控制器代码的被调出的部分,并且该仿真程序实时更新控制器代码的执行环境。在这种情况下,执行环境描述的是所有限制条件,在仿真程序上的控制器代码的执行与这些限制条件相关联,除了内部状态参数之外还有描述被模拟的控制器的接口的和在必要时还被模拟环境影响的那些状态参数。

[0015] 优选测试装置构造如下,即,控制器代码是完整的,它特别是包括操作系统、驱动程序和控制器的所有软件组件,因而前述的执行环境描述虚拟控制器的完整的物理环境。在这个前提条件下,对虚拟控制器的测试实际上包括相应于虚拟控制器的真实控制器的所有方面。然而备选地,根据本发明的测试装置也可以包括这样的应用情况,在这些应用情况中控制器代码是不完整的,特别是不包括控制器的驱动程序和操作系统,因此执行环境仅仅包括虚拟控制器的软件组件。

[0016] 在根据本发明的测试装置的一种特别优选的变型中规定:经由模拟环境接口,数据驱动地在仿真程序与模拟环境之间、特别是在仿真程序与过程模型之间由仿真程序对控制器代码进行计算。通过这个作用原理通过仿真程序对控制器代码的执行还能够与模拟环境的计算或者模拟环境的过程模型的计算同步。

[0017] 在测试装置的一种对此备选的设计方案中规定:模拟环境包括至少一个模拟程序和一个过程模型,其中模拟程序可以被过程模型调出,和/或过程模型可以被模拟程序调出。在这种情况中优选测试装置构造如下,即,过程模型的计算和通过仿真程序对控制器代码的执行被模拟程序中枢地控制,从而以这种方式实现了所期望的同步执行、特别是过程模型的计算与控制器代码的执行在时间上的协调。

[0018] 与按照哪一个前述方法使过程模型的实时计算与通过仿真程序对控制器代码的执行之间同步无关地,有益之处当然在于:在仿真程序上在时间方面最佳化地执行控制器代码,由此在每个探测间隔内发生的计算无论如何在这个间隔内还能够不中断地结束并且不会由于在此出现的限制而违背实时要求。为了使运行时间特性(Laufzeitverhalten)最佳化,可以使用所有已知的方法和技术。另外,在对控制器代码进行解释的情况中可以使得用于运行时间的存储器预留(Speicherreservierung)减少或完全避免。在控制器代码的二进制转化的情况中,适当的方法可以在于:所有的原子的代码块被先验鉴别并且在运行时间之前被转化。备选地,块缓存(Blockcache)可以选择得如此大,使得在初始化阶段之后所有模拟所需的代码块被转化,并且因此在块缓存中不再发生置换。另一种方法可以在于:对控制器代码内的链参考进行识别并加以解算。

[0019] 文首导出的目的还可以根据方法得以实现，为了实施本发明的方法，从用于对带有控制器代码的虚拟控制器的至少一个部分进行实时测试的测试装置出发，其中，所述测试装置具有带第一指令集的至少一个第一类型计算核心和用于模拟虚拟控制器的环境的至少一个模拟环境。本方法规定：利用至少一个第一类型计算核心对模拟环境和控制器代码进行计算。另外规定：控制编码被如此提供，即，它可以在带第二指令集的第二类型计算核心上执行，其中，第二类型计算核心通常为需测试的批量控制器的计算核心。就这点而言，第二类型计算核心的第二指令集不同于第一类型计算核心的第一指令集。于是根据本方法规定：在测试装置的第一类型计算核心上执行用于模拟第二计算核心的仿真程序并且该仿真程序自己执行控制器代码。另外，根据本发明规定：在仿真程序与模拟环境之间交换数据和/或事件，其中，信号可以向两个方向流动，也就是说数据和/或事件可以从仿真程序传向模拟环境的方向以及从模拟环境传向仿真程序的方向。一般来说，通过根据本发明的运行方式实现的可能性是：使对虚拟控制器的环境实时进行的模拟与控制器代码的模拟同步，从而通过仿真程序也实时执行控制器代码。

附图说明

[0020] 详细地具有大量提高和进一步发展根据本发明的测试装置的可能性。为此参照借助附图对实施例的说明。附图中：

[0021] 图1a示出由现有技术已知的具有需测试的真实控制器的测试装置；

[0022] 图1b示出由现有技术已知的具有需测试的虚拟控制器和真实控制器的测试装置；

[0023] 图2a示出根据本发明的具有虚拟控制器的测试装置的第一实施例；

[0024] 图2b示出图1a所示的实施例，其具有虚拟控制器和真实控制器的组合；

[0025] 图3示出根据本发明的测试装置的另一实施例；

[0026] 图4示出根据本发明的具有数据驱动的仿真程序的测试装置的实施例；

[0027] 图5示出根据本发明的具有由模拟程序控制的仿真程序的测试装置的另一实施例。

具体实施方式

[0028] 图1a示意性示出的是由现有技术已知的用于对真实的控制器2a、2b进行实时测试的测试装置1。与此相对，图1b示出的是由现有技术已知的测试装置1，该测试装置除了应测试的真实控制器2之外还具有两个需测试的虚拟控制器3a、3b。测试装置1具有第一类型计算核心4，该第一类型计算核心的出众之处在于第一指令集IS1。另外，在图1a和1b所示的测试装置1具有用于实时模拟需测试的控制器的环境的模拟环境5。真实的、即物理性存在的控制器2具有第二类型计算核心6，这个第二类型计算核心6以第二指令集IS2为特征。在示出的实例中，控制器的第二类型计算核心6的第二指令集IS2不同于测试装置1的第一类型计算核心4的第一指令集IS1。

[0029] 在图1a、1b中，测试装置1以微机为基础，第一指令集IS1在示出的实例中与Intel 32-Bit-x86-Architecture(IA-32)兼容。第二类型计算核心6是具有C166类型计算核心的微控制器，因而与此相关的第二指令集IS2肯定与测试装置1的第一类型计算核心4的第一指令集IS1不同。真实的控制器2经由它的物理接口而与测试装置1的对应的物理I/O接口相

连接。

[0030] 在图1a所示的真实控制器2的实时测试的情况中，控制器代码EC理所当然地可以实施成其原形式，即面向第二类型计算核心6的第二执行组IS2。但是对真实的控制器2的控制器代码EC进行测试的前提是：真实的控制器2实际上物理性地存在，这伴随产生已经说明的缺点。

[0031] 图1b所示的测试装置1还与虚拟控制器3a、3b共同运行。这些控制器物理上并不存在，更确切地说它们通过如下方式得以实现，即，控制器代码EC被转化到测试装置1的第一类型计算核心4的第一指令集IS1中，这样被转化的控制器代码EC′可以在测试装置1上运行。这虽然能够提早地测试控制器代码而无需真实的控制器存在，但是其缺点在于：并不是原控制器代码EC被测试，而是经转化的变型EC′。另外，控制器代码EC必须存在于一种高级语言中，从而可以转化到第一类型计算核心4的第一执行系统IS1上。

[0032] 图2a、2b现在示出的是根据本发明的测试装置1。在此测试装置1也具有带有第一指令集IS1的第一类型计算核心4，在此模拟环境5也用于实时模拟虚拟控制器3的环境。在此最终也借助第一类型计算核心4来计算虚拟控制器3的控制器代码EC，然而相对由现有技术已知的例如如图1a和1b的测试装置存在大的区别。虚拟控制器3的控制器代码EC能在带有第二指令集IS2的第二类型计算核心上运行，其中所述第二指令集IS2不同于测试装置1的第一类型计算核心4的第一指令集IS1。原本为第二类型计算核心设想的控制器代码EC仍然可以与测试装置1一起执行，因为测试装置1的第一类型计算核心4执行一个仿真程序7，这个仿真程序7模拟第二类型计算核心并且被模拟的第二类型计算核心执行控制器代码。通过这种措施不需要将能够在带有第二指令集IS2的第二类型计算核心上运行的控制器代码EC转化到第一指令集IS1上，原则上可以执行原控制器代码EC，该原控制器代码通常设立为用于与测试装置1的目标硬件不同的目标硬件。不再存在如下的需要：控制器代码EC必须存在于一种高级语言中，以便能够转换到测试装置的第一指令集IS1上。

[0033] 重要的是，仿真程序7具有模拟环境接口8，仿真程序7经由该模拟环境接口能够与模拟环境5交换数据和/或事件。因为模拟环境接口8通过仿真程序7能够使虚拟控制器3的环境的实时模拟与第二类型计算核心的模拟同步，所以模拟环境接口8以这个功能性对于仿真程序7在实时模拟的范围中的应用来说是至关重要的。

[0034] 图2b示出的是：根据本发明的测试装置1并不局限于对仅仅一个虚拟控制器3或多个虚拟控制器3进行测试，而更确切地说还可能的是：测试装置1也与外部的真实的控制器2运行。图2b所示的同时对虚拟控制器3和真实的控制器2或者对混合的控制器结合体的测试的实施例可以是所述测试装置1的一种常见的使用情况。

[0035] 图3示出的是测试装置1，其中模拟环境5包括模拟程序9和过程模型10。模拟程序9可以经由模拟环境接口8通过仿真程序7调出，以及反过来仿真程序7可以经由模拟环境接口8通过模拟程序9调出。模拟程序9总体上是可实时的平台，以便在考虑到实时条件的情况下执行过程模型10。除了实时环境之外，模拟程序为了能够计算过程模型10还具有合适的数值法。因此，模拟程序9在时间上对执行过程模型10的不同任务进行协调并且提供工具，以便还能够考虑模拟的范围内的非周期性的事件。

[0036] 在图3中另外可以看到，通过模拟环境接口8可以在仿真程序7与过程模型10之间交换数据。为此过程模型10认识被模拟的虚拟控制器3的接口，从而产生过程模型10与被模

拟的虚拟控制器3之间的数据方面的结合。

[0037] 图4和5示出的是根据本发明的测试装置1，其中借助方框图不仅应示出该测试装置1的结构，而且还应该能看出该测试装置1内的功能流程。在两个图示中，模拟环境5与具有模拟环境接口8的仿真程序7相应相对地设置，模拟环境5通过所述模拟环境接口而与仿真程序7相互作用。在模拟环境5中还分别示出带有过程模型10的模拟程序9。

[0038] 在图4中可以看到，通过模拟环境接口8而可以在仿真程序7与过程模型10之间交换数据。图4所示实施例的出众之处在于：经由模拟环境接口8由仿真程序7数据驱动地在仿真程序7与过程模型10之间执行控制器代码EC。因此经由模拟环境接口8识别出过程模型10的重要的状态参数和由此例如接口的确定通道上的信号是否发生变化，接着在仿真程序7中触发相应的动作。

[0039] 从模拟程序9的静止状态901出发，下一个用于过程模型10的计算步骤被激活。为此，首先由仿真程序7计算的有关的控制器数据被读入输入程序902中。

[0040] 仿真程序7上的计算在那里同样从静止状态701出发，其中在步骤702中执行仿真程序7。可以通过各种不同类型的触发来实现状态702。这样例如状态702可以通过从静止状态701出发的定时器中断得以达到或通过数据事件得以达到。在数据事件的情况中，首先进入状态802并且在与此相连的中断的情况中然后到达状态702。例如当一个Pin-Change-Interrupt与一个I/O-Pin相连时，是这种情况。如果中断没有与I/O-Pin相连，那么重新离开状态802并且在702中不发生其它行动。状态802则保障的是：用于仿真程序的数据处于被相应编码的状态，即在仿真程序的执行环境中。

[0041] 利用仿真程序还执行存在于第二指令集IS2内的控制器代码EC。模拟程序9和仿真程序7实际上平行运行。模拟只有如之前详细说明的那样在与事件相连的中断的情况中被中断，否则只有需相互交换的数据通过模拟环境接口8变成可用的并且被仿真程序7或者模拟程序9读取。

[0042] 在过程模型10的计算步骤903结束之后，过程模型10的实时状态参数在输出功能904中被提供给仿真程序使用。这一点在模拟环境接口8内在数据块802中要么简单地通过对仿真程序7的物理环境的存储要么通过调出控制器代码EC内的相应的接口功能得以实现。在这种情况下，“物理环境的存储”应该理解为：例如改变的I/O信号通过将该改变的I/O信号适当地存储在仿真程序的环境中变得可供仿真程序使用。与此相反，只有当这样的接口功能的利用得到相应的配置以及在控制器代码EC内的中断被触发时，才在步骤702中在控制器代码EC内调出接口功能。如果中断没有被连接起来，那么只有如前所述的那样信息作为这样的信息被提供使用。在这种情况下上后者意味着：信号在物理上贴近以及作为这样的信号通过适当的存储而成为可用并且可以通过相应的简单的读取过程被利用；但是在数据块702内未触发步骤。相应的内容适用于过程模型10内的计算903。

[0043] 在图5所示的实施例中，模拟环境5与仿真程序7之间的同步通过如下方式得以实现：过程模型10的计算与通过仿真程序7对控制器代码EC的执行被模拟程序9中枢地控制，这在当前主要涉及的是过程模型10的计算903与控制器代码EC的执行702在时间上的协调。在图5中可以看到：不仅过程模型10而且仿真程序7的执行是如何通过模拟环境接口8从模拟程序的静止状态901中被触发的，通过这种方式保障时间上的同步。当然这一点的前提是：可以通过模拟程序9调出过程模型10。在仿真程序7方面在步骤801中首先确定控制器的

必须被处理的功能性。接着在步骤811中所有实时的输入变量被读入，既有由过程模型10提供使用的输入变量也有来自可能的外部来源11的其它输入变量。在步骤812中由物理数据制成执行环境，这样信息通过简单地读取操作可以提供给仿真程序7使用。对于以接口程序作业的情况，直接在步骤702中通过仿真程序7对控制器代码EC的相应的功能进行加工处理。在计算完成之后在步骤813中，执行环境连同实时更新的数据被重新存储并且通过在步骤814中执行输出功能而写出相应的结果，一方面是涉及过程模型10的结果，另一方面是涉及外部数据接收装置(Datensenke)12的结果。以这种方式方法在仿真程序7上实时地对涉及过程模型10和涉及控制器代码EC的计算步骤相互同步地进行计算。

[0044] 在图4和5所示出的实施例中，仿真程序7对每个在运行时间被调出的第二指令集IS2内的控制器代码EC的指令进行解释并且然后在功能上与第一指令集IS1的相应的指令一致地在测试装置1的第一类型计算核心上执行这个被调出的指令。

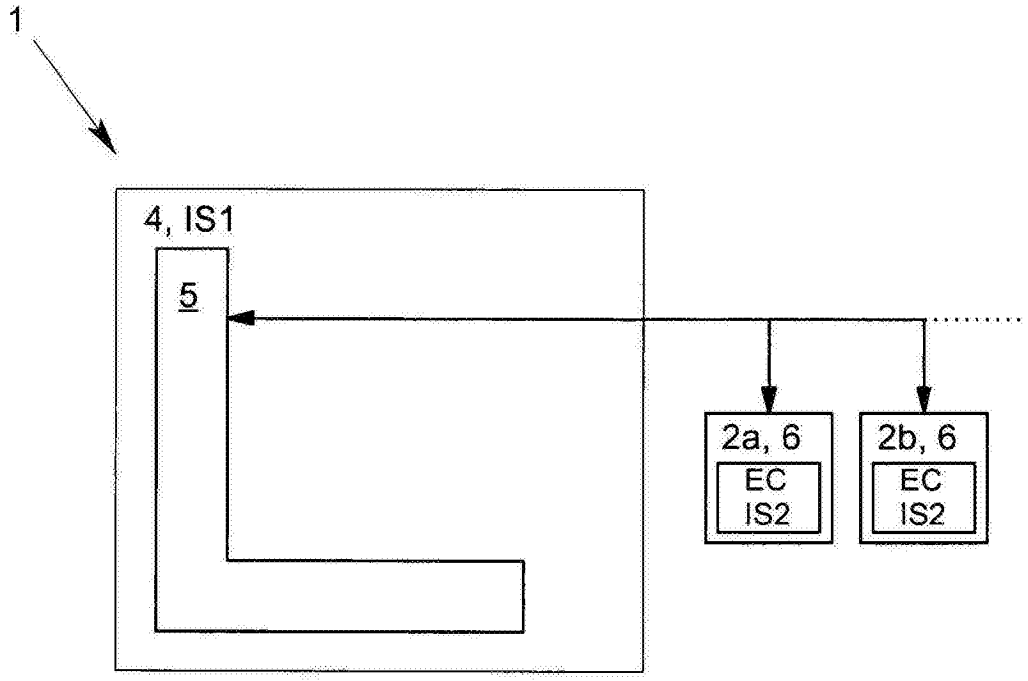


图1a

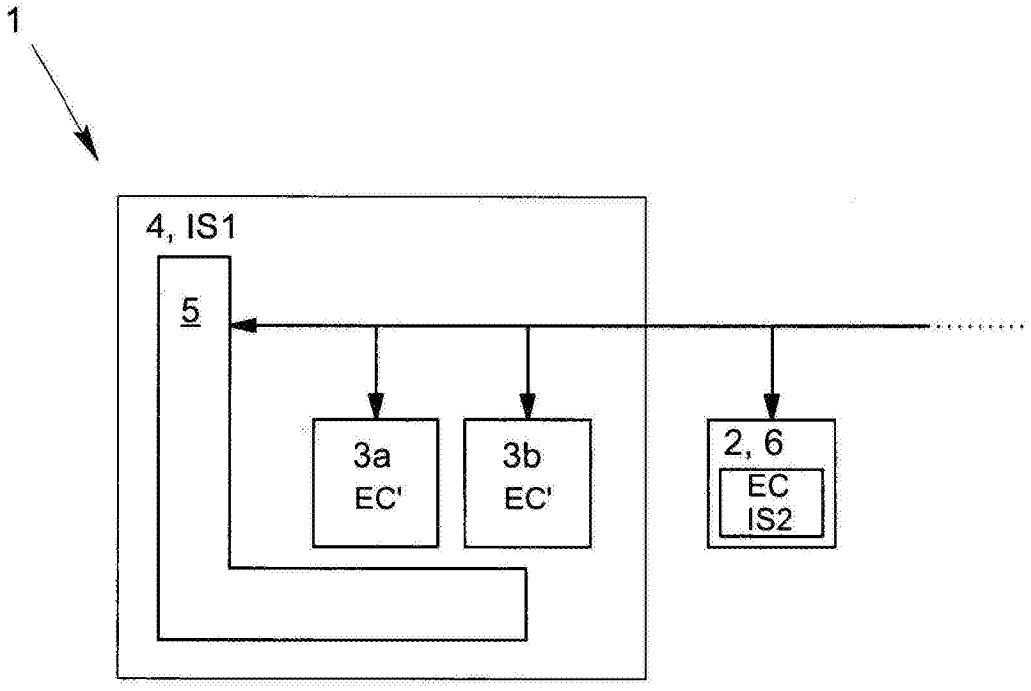


图1b

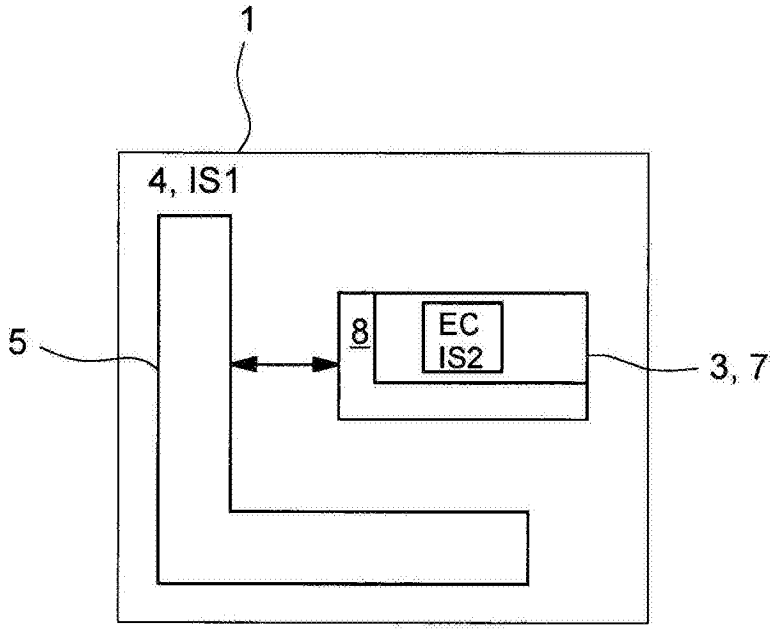


图2a

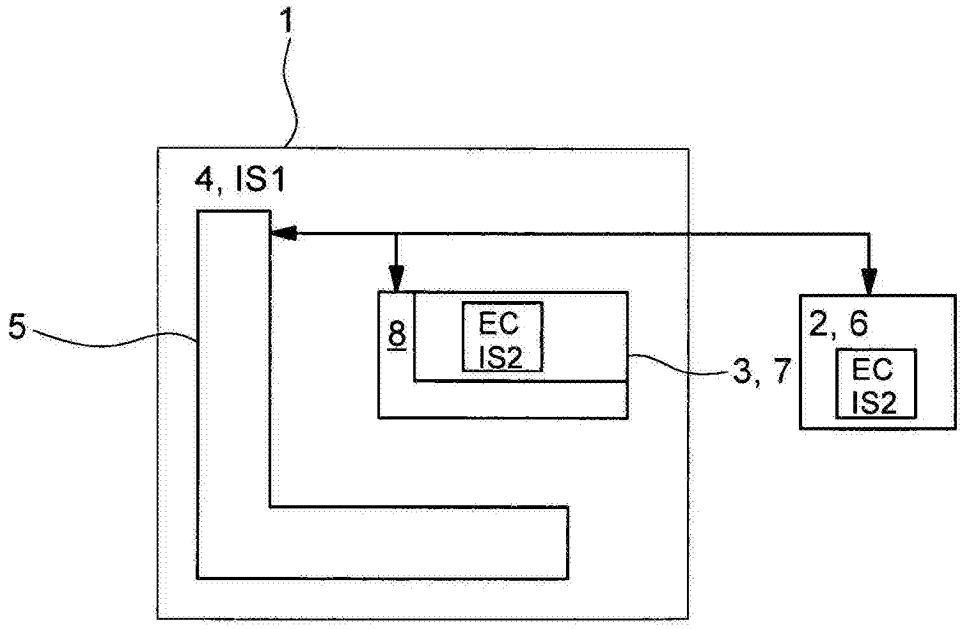


图2b

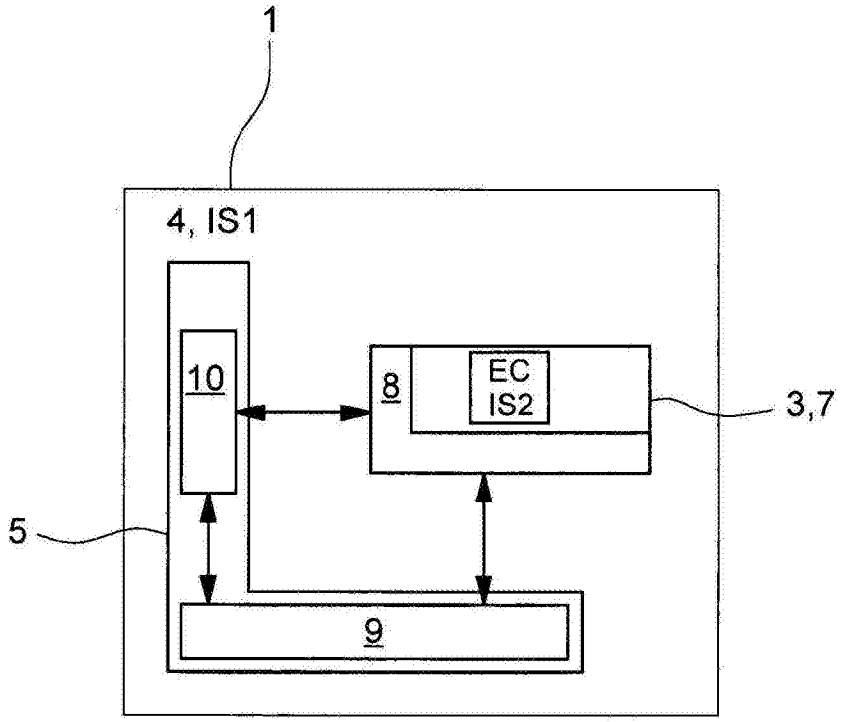


图3

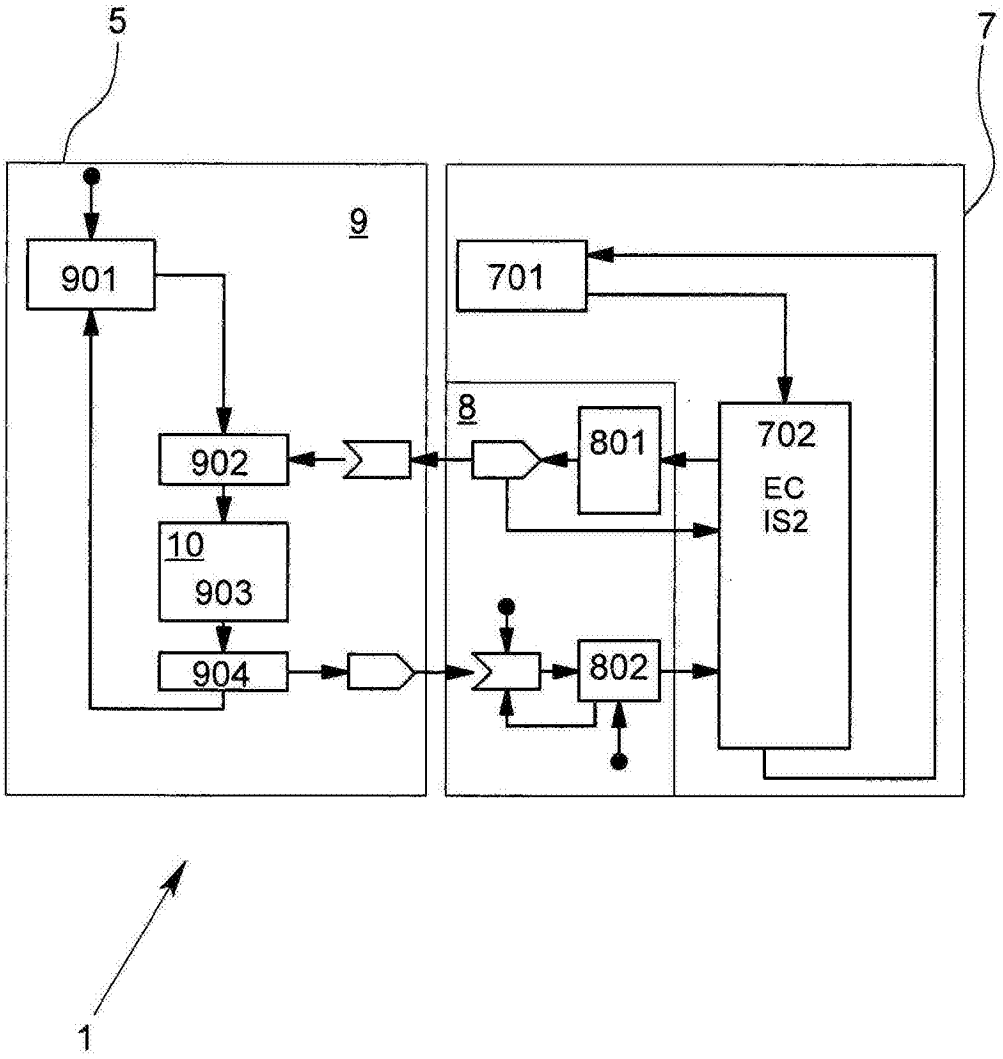


图4

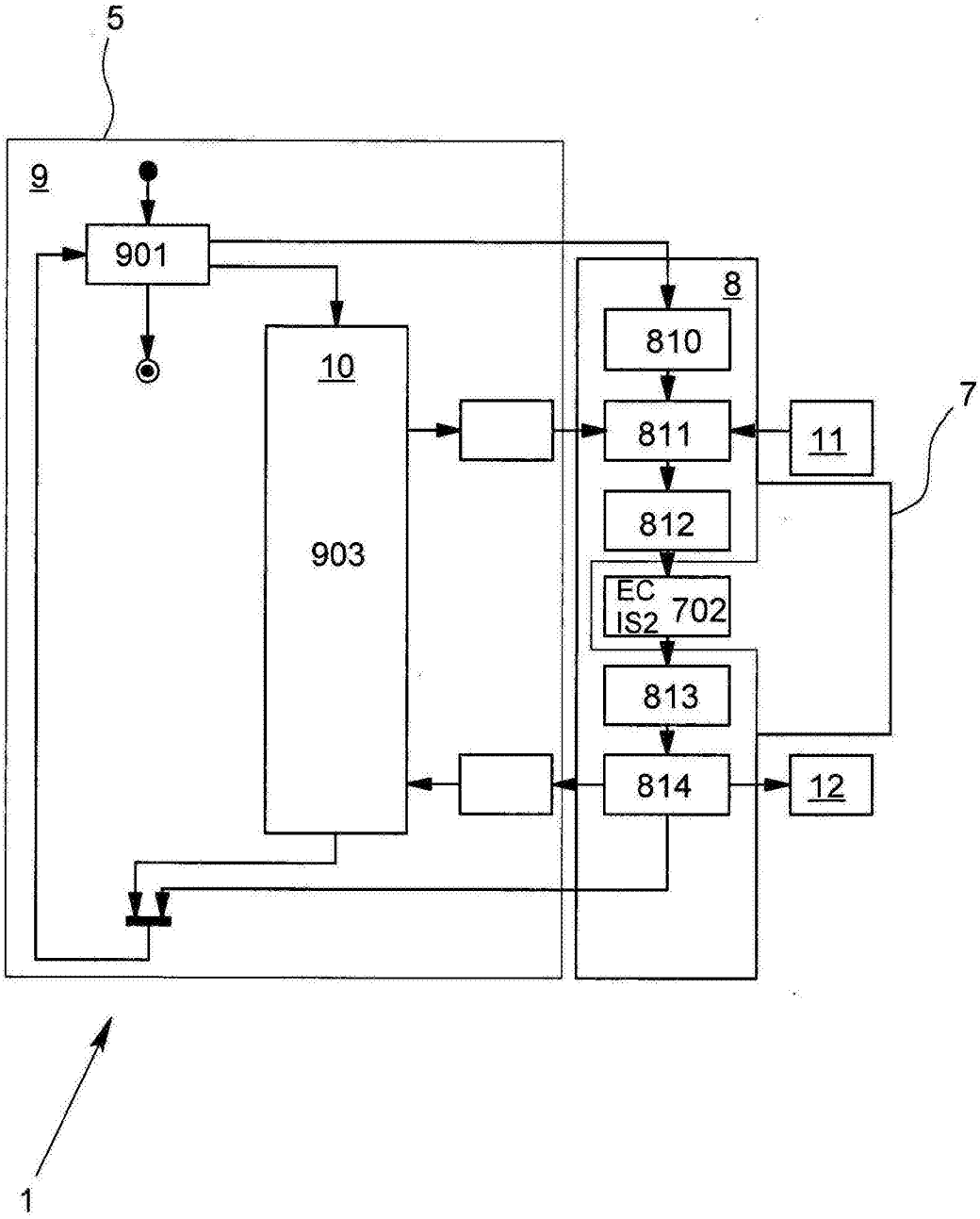


图5